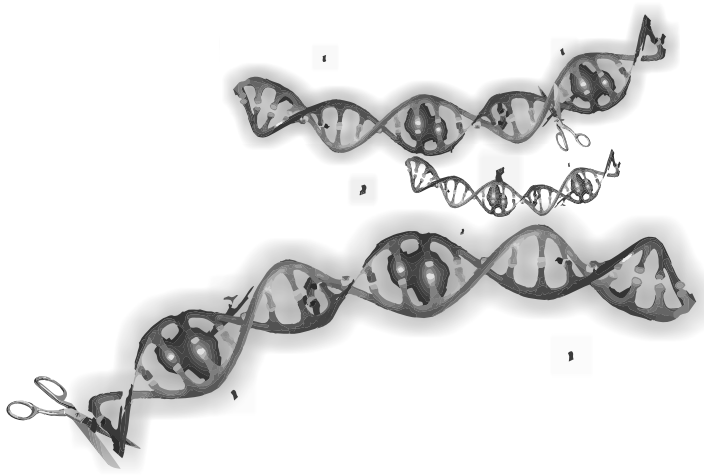
BLAISE PASCAL MAGAZINE 77



Playing Billiards
TMS Webcore 1.1 creating a webtop app
CalendarProject
What is CSS ? Examples and history
WebAssembly: Design, History and Video
JavaScript - AJAX - SingleWebPage - CORS explained
What is HTML (5)
PAS2JS
How to install kbmMW
REST easy with kbmMW #18 – Configuration #2
REST easy with KBMMW #19 – HTTP – Headers and Cookies
kbmMW Professional and Enterprise Edition v. 5.08.10 released!

BLAISE PASCAL MAGAZINE 77
Object Pascal / Internet / JavaScript / WebAssembly / Pas2Js / Databases
CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux

Blaise Pascal

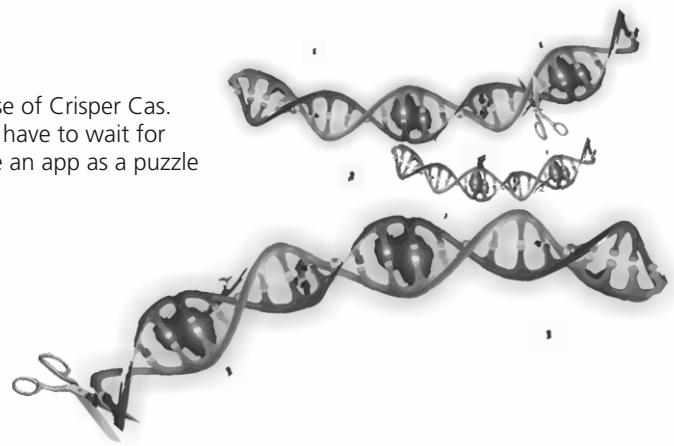
CONTENT

ARTICLES

Playing Billiards	Page 5
TMS Webcore 1.1 creating a webtop app	Page 9
CalendarProject	Page 15
What is CSS ? Examples and history	Page 32
WebAssembly: Design, History and Video	Page 43
JavaScript - AJAX - SingleWebPage - CORS explained	Page 45
What is HTML (5)	Page 53
PAS2JS	Page 58
How to install kbmMW	Page 61
REST easy with kbmMW #18 – Configuration #2	Page 72
REST easy with KBMMW #19 – HTTP – Headers and Cookies	Page 73
kbmMW Professional and Enterprise Edition v. 5.08.10 released!	Page 84

The Crisper Cas puzzle

We are developing a special article about the use of Crisper Cas. It took more time then we expected so you will have to wait for this article and puzzle. The intention is to create an app as a puzzle to explain the workings.



ADVERTISERS

Barnsten	Page 56 / 57
Lazarus handbook	Page 60
LibStick	Page 52
Mitov Software	Page 30
Components 4 Developers	Page 86

Publisher: PRO PASCAL FOUNDATION in collaboration with the Pascal User Group © Stichting Ondersteuning Programmeertaal Pascal

Pascal is an imperative and procedural programming language, which Niklaus Wirth designed in 1968–69 and published in 1970, as a small, efficient language intended to encourage good programming practices using structured programming and data structuring. A derivative known as Object Pascal designed for object-oriented programming was developed in 1985. The language name was chosen to honour the Mathematician, Inventor of the first calculator: Blaise Pascal (see top right).

Colofon

Stephen Ball
http://delphiaball.co.uk
@DelphiABall

Miguel Bebensee
mbebensee@ibexpert.biz
http://devstructor.com

Peter Bijlsma -Editor
peter @ blaisepascal.eu

Dmitry Boyarintsev
dmitry.living @ gmail.com

Michaël Van Canneyt,
michael @ freepascal.org

Marco Cantù
www.marcocantu.com
marco.cantu @ gmail.com

David Dirkse
www.davdata.nl
E-mail: David @ davdata.nl

Benno Evers
b.evers
@ everscustomtechnology.nl

Bruno Fierens
www.tmssoftware.com
bruno.fierens @ tmssoftware.com

Holger Flick
holger@flixments.com

Primož Gabrijelčič
www.primoz @ gabrijelcic.org

Mattias Gärtner
nc-gaertnma@netcologne.de

Peter Johnson
http://delphidabbler.com
delphidabbler@gmail.com

Max Kleiner
www.softwareschule.ch
max @ kleiner.com

John Kuiper
john_kuiper @ kpnmail.nl

Wagner R. Landgraf
wagner @ tmssoftware.com

Vsevolod Leonov
vsevolod.leonov@mail.ru

Andrea Magni www.andreamagni.eu
andrea.magni @ gmail.com
www.andreamagni.eu/wp

Paul Nauta PLM Solution Architect
CyberNautics
paul.nauta@cybernautics.nl

Kim Madsen
www.component4developers

Boian Mitov
mitov @ mitov.com

Jeremy North
jeremy.north @ gmail.com

Detlef Overbeek - Editor in Chief
www.blaisepascal.eu
editor @ blaisepascal.eu

Howard Page Clark
hdpc @ talktalk.net

Heiko Rompel
info@rompelsoft.de

Wim Van Ingen Schenau -Editor
wisone @ xs4all.nl

Peter van der Sman
sman @ prisman.nl

Rik Smit
rik @ blaisepascal.eu
www.romplesoft.de

Bob Swart
www.eBob42.com
Bob @ eBob42.com

B.J. Rao
contact@intricad.com

Daniele Teti
www.danieleteti.it
d.teti @ bittime.it

Anton Vogelaar
ajv @ vogelaar-electronics.com

Siegfried Zuhr
siegfried @ zuhr.nl

Editor - in - chief
Detlef D. Overbeek, Netherlands Tel.: +31 (0)30 890.66.44 / Mobile: +31 (0)6 21.23.62.68
News and Press Releases email only to **editor@blaisepascal.eu**

Editors
Peter Bijlsma, W. (Wim) van Ingen Schenau, Rik Smit

Correctors
Howard Page-Clark, Peter Bijlsma

Trademarks All trademarks used are acknowledged as the property of their respective owners.
Caveat Whilst we endeavour to ensure that what is published in the magazine is correct, we cannot accept responsibility for any errors or omissions. If you notice something which may be incorrect, please contact the Editor and we will publish a correction where relevant.

Subscriptions (2017 prices)

	Internat. excl. VAT	Internat. incl. VAT	Including Shipment
Printed Issue ±80 pages	€ 240	€ 261,60	€ 85,00
Electronic Download Issue 80 pages	€ 55	€ 66,55	———
Printed Issue Inside Holland(Netherlands) ±80 pages	———	€ 200	€ 40,00

Member and donator of WIKIPEDIA

Subscriptions can be taken out online at **www.blaisepascal.eu** or by written order, or by sending an email to **office@blaisepascal.eu**
Subscriptions can start at any date. All issues published in the calendar year of the subscription will be sent as well.
Subscriptions run 365 days. Subscriptions will not be prolonged without notice. Receipt of payment will be sent by email.
Subscriptions can be paid by sending the payment to:
ABN AMRO Bank Account no. 44 19 60 863 or by credit card or Paypal
Name: Pro Pascal Foundation-Foundation for Supporting the Pascal Programming Language (Stichting Ondersteuning Programeertaal Pascal)
IBAN: NL82 ABNA 0441960863 BIC ABNANL2A VAT no.: 81 42 54 147 (Stichting Programmeertaal Pascal)
Subscription department
Edelstenenbaan 21 / 3402 XA IJsselstein, The Netherlands
Mobile: + 31 (0) 6 21.23.62.68 **office@blaisepascal.eu**

Copyright notice

All material published in Blaise Pascal is copyright © SOPP Stichting Ondersteuning Programeertaal Pascal unless otherwise noted and may not be copied, distributed or republished without written permission. Authors agree that code associated with their articles will be made available to subscribers after publication by placing it on the website of the PGG for download, and that articles and code will be placed on distributable data storage media. Use of program listings by subscribers for research and study purposes is allowed, but not for commercial purposes. Commercial use of program listings and code is prohibited without the written permission of the author.

From the editor

The new year has already started, and I must say these are turbulent times. Even in the software paradise.

Idera bought another firm for its software - shop. Just yesterday I heard of it: TRAVIS CI. No explanation of how and what this means. I have the feeling that we as users are not taken seriously.
I think the shop owns now a lot of only vaguely matching sorts of software where the solutions they promise might be interesting.
But no working together or some kind of integration; not one of them works with our favourite product: Delphi, not even in the slightest way. Separately they might be useful - but that's it. I am very much worried about this.

When does Embarcadero start on the next important features we would expect it to do? I have the feeling that Delphi is not any more an important Program for Embarcadero, just one of a kind.
I recall that being said by Atanas Popov the leading figure of Embarcadero but I never expected it to go this far. He already suggested this.

This doesn't show any knowledge of how to build coherence in to your company.

What would be the new features we want so much?
I'd like to know by asking you - the users: what do you want for features?
Let's speak out ourselves and tell them.
So I suggest you send a list of what you think might be worth of integrating into Delphi: more integration with the internet? Making small devices accessible for programming in Delphi, like the Raspberry, other small boards for use like Arduino, how about AI (Artificial Intelligence) and one simple thing: make the IDE more stable? Finally decide to make a Linux Desktop Version...

How about creating a Real Language Engine- with capabilities of using for translation and handling your applications?

How about the development or integration of a special gaming console - very necessary to gain interest of the young people. Anything about Robotics? You as a user could have ideas we never heard of before...

Send your wishes to us - maybe we can convince Embarcadero to do something about these items?

Happily, people like Bruno Fierens are doing what is most needed:
inventing the future.
Making Delphi fit in to the Internet, creating WebTop (- *Not Desktop*-) -so called Progressive Web Apps. (*like the one in this issue, starting at page 9*).
The possibilities there are vast: I personally think that the Internet will become the only important Operating System for the future.

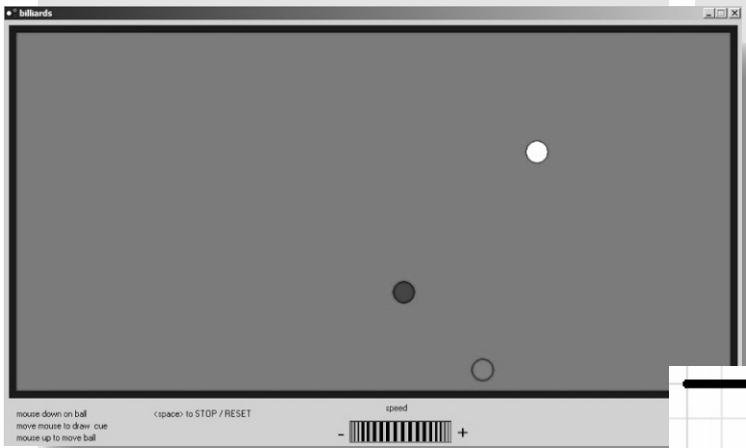
Imagine this: - and it is already happening - with webtop-apps you don't need Google Shop, you don't need anything like the Apple Shop, you don't need any shop at all. Maybe if it would mean no extra cost or some company that wants to supervise you. You should be and are master of your own app.

And don't forget the future: the Quantum Internet is coming very soon, it will be like the early days of the ARPANET -(the precursor of the later Internet). We of course will have to reserve time slots on the Quantum-machines. Like we did in earlier times: the ENIAC buying time. I promise to keep you informed...

For Our Readers in the Netherlands and Belgium:
on the 9th of March 2019
there will be a Meetup in Holland,
Wsselstein see our website

https://www.blaisepascalmagazine.eu/events/

starter expert DX



INTRODUCTION

This article describes a simple billiards game:
the fysics of bouncing balls and a Delphi programming language implementation.

HOW TO USE THE PROGRAM

TO START A BALL MOVEMENT:

1. position mousepointer over ball.
2. press left-mousebutton down and hold.
3. move mousepointer in opposite ball movement direction (draw cue).
4. release mousebutton to fire ball.

To stop motion:
Press mousebutton on billiards table or hit space bar
Pressing the spacebar twice resets the balls to their original position.

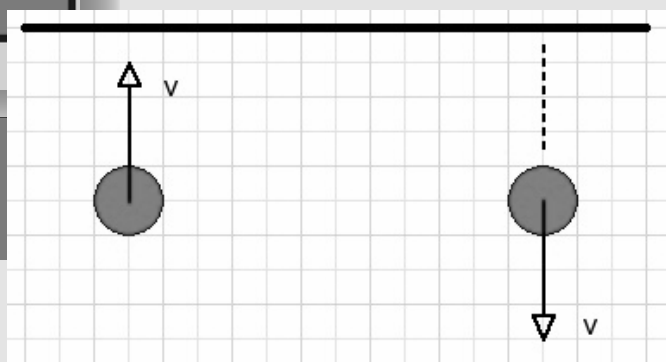
To change ball sped:
Press down mousebutton over rotation (speed) button and move mouse left or right.

This project was built to demonstrate the ball movement only, not to play billiards. Once started the balls move forever which is funny but not very realistic.
However, it should not be too difficult to make a real billards game:

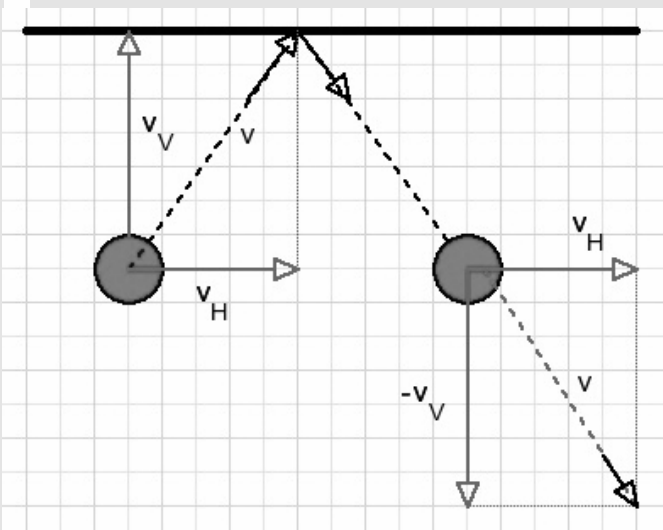
1. add the effect of friction, so ball speed decreases.
2. let the length of the cue define the ball speed.

The fysics of bouncing balls
In the following descriptions we assume "elastic" collisions. This means that the balls do not change shape.
No energy is lost by deformation or friction.

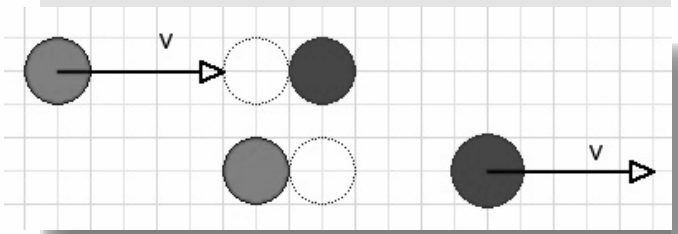
1. Bouncing on edges
When a ball hits an edge, it slightly compresses the point of impact which then
applies a force in the opposite direction of movement.
This makes the ball decellerate first, then accellerating to the original speed but in opposite direction.



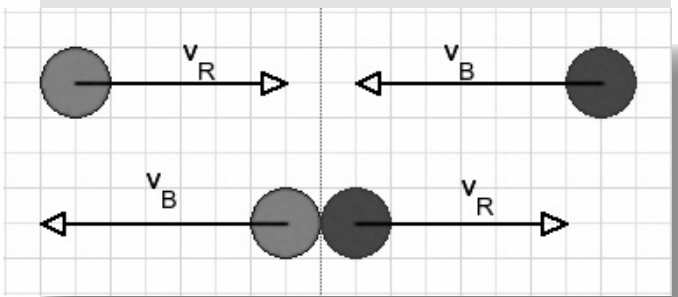
In picture above v is the speed of the ball.
Ball movement is perpendicular to the egde.
For other ball directions, the speed has to be decomposed in a horizontal and a vertical component.
After impact, the new direction is the (vector) sum of the horizontal movement and the reversed vertical movement.



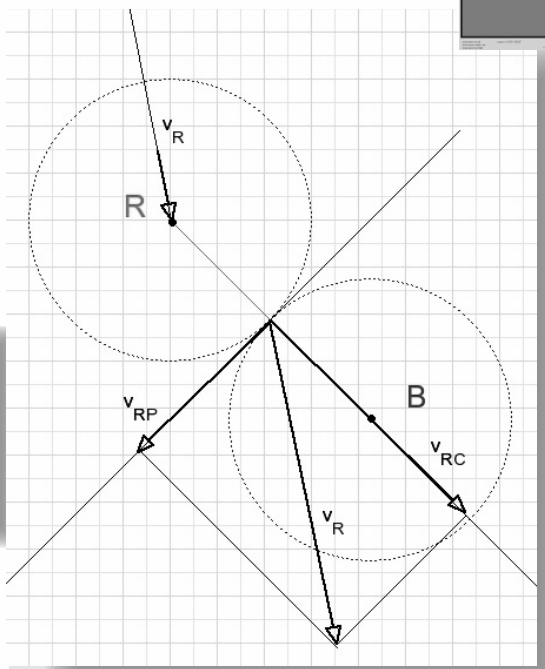
2. Bouncing between balls.
First the situation where only one ball is moving.
In picture below the red ball hits the blue one.
The direction is towards the blue ball center.
The red ball decellerates, the same force however accellerates the blue ball.
After impact, the red ball is in rest and the blue ball continues in the same direction
and with the same speed of the red ball.



Next the situation where both balls are moving towards each others center at different speeds.
Let this speeds be Vr and Vb where Vr = Vb + x.
If x = 0 then the balls bounce and then continue movement in opposite direction.
If the red ball hits the blue one with a speed of x, we saw before that this speed x is passed to the blue ball.
So, in this case after collision, the red ball has a speed of Vb and the blue ball has
a speed of (Vb+x).
A ball simply gets the speed of the other one.

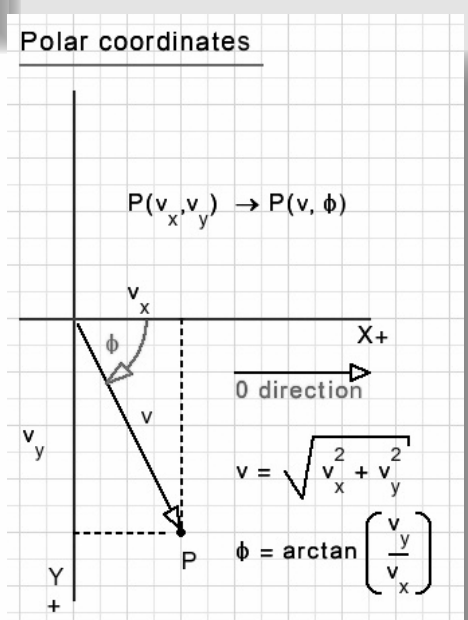


Balls moving in random directions
Ball speeds must be decomposed in speeds in the centerline direction and perpendicular to this centerline.
Again a ball gets the velocity of the centerline component of the other ball.

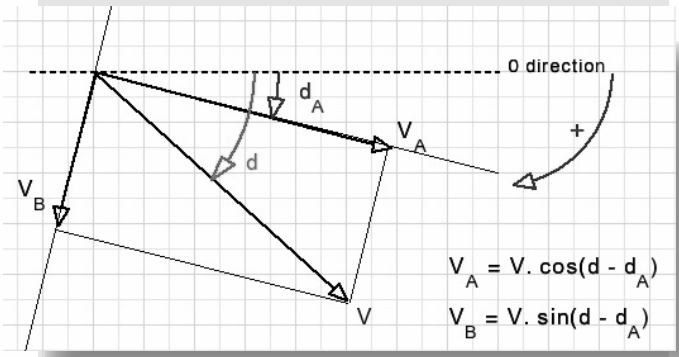


In the above picture only the red ball speed is decomposed for reasons of clarity. If the blue ball speed is decomposed similarly into components vBC and vBP then after collision the ball speeds are
red : vBC + vRP
blue : vRC + vBP

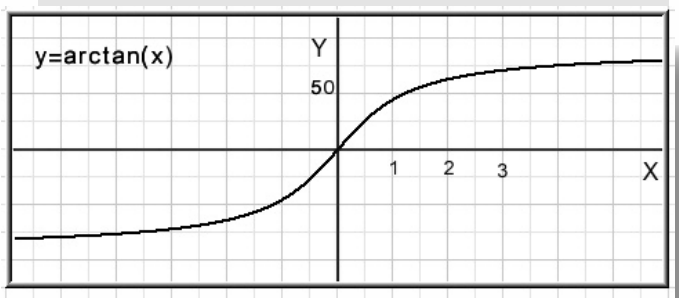
DIRECTION
The speed always is a positive number.
The direction is in radians so:
angles from -180...+180 degrees in radians are -pi .. +pi.
Also note, that in our coordinate system the positive y (vertical) direction is down. So, in the right bottom quadrant both x and y coordinates are positive.
We apply polar coordinates by defining velocities as a positive value together with the angle of the direction.



Picture above shows the conversion from a carthesian- to a polar coordinate system.
Decomposing a velocity into two other perpendicular directions is not a problem:



d is the ball movement direction in radians. Angles increase if clockwise rotation.
Other directions do not cause problems or conflicts because the sine and cosine
functions supply the right signs in all quadrants.
One problem remains: the arctan function does not differentiate between diagonally opposite quadrants.
It always returns an angle between -pi/2 to pi/2



Above, angles are measured in degrees, not radians.
Following function does the work:

```
function XYdirection(x,y : single) : single;
//direction (0,0) to (x,y) in radians (-pi .. + pi)
const pi05 = 0.5*pi;
begin
 if x = 0 then
 begin
  if y < 0 then result := -pi05 else result := pi05;
 end
 else begin
    result := arctan(y/x);
    if x < 0 then
     if y < 0 then result := result - pi
      else result := result + pi;
   end;
end;
```

THE PROGRAM
DATA FORMATS.

```
const balldiameter = 30;
   ballradius = balldiameter * 0.5;
   speed = 0.25; //pixels per step
   pi05 = pi*0.5;
   pi2  = pi*2;

type TBallcolor =
(Redball,whiteball,blueball,noBall);
   TBall = record
        xpos : single; //x coordinate of ball center
        ypos : single; //y ..
        vel  : single; //velocity
        dir  : single; //direction
       end;

var bm : TBitmap;
  ballmap : array[redBall..blueBall] of Tbitmap;
  ball : array[redBall..blueBall] of TBall;
  eraseBall : TBitmap;
  moving,drawing : boolean;
  x1,y1,x2,y2 : smallInt;
  selected : TballColor;
```

The bitmap (BM) size is 960*480 pixels, same as paintbox1 on the form.
Edges are drawn on the canvas of the form, around the paintbox.

Balls are separate transparent bitmaps which image is painted at creation time.
The eraseMap is a bitmap with green canvas to erase balls.

ACCURACY
A ball always starts with a speed of 0.25 (pixel per update).
This is the maximum value.
Ball positions are advanced step by step.
After each step checks are made for any overlap between a ball and edge or another ball.
Small steps will yield a higher accuracy, a more precise point of impact.
After 8 increments the screen is updated by

1. erasing the balls
2. painting the balls at new positions
3. copying the bitmap (bm) to the paintbox.

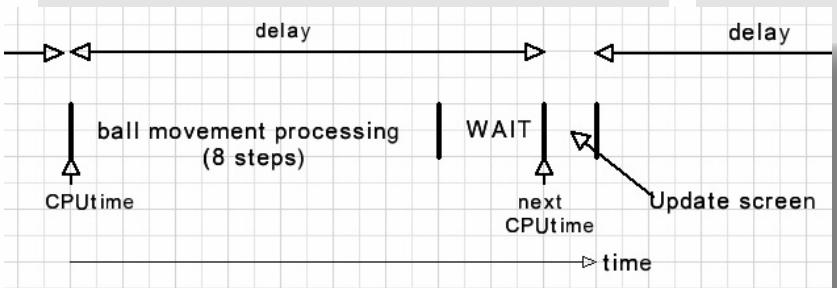
TIMING

The computer has a milliseconds counter: **`function gettickcount`** returns the number of milliseconds since the computer was switched on.
However, this clock value is updated only at about 16 milliseconds intervals which is too slow.

Another clock runs at CPU clock speed.
This is the one we use.
Processors have different clock speeds so a conversion is needed from CPU ticks to real time.
This is done by procedure GetCPUticks which returns the number of clock ticks per microsecond.
Function CPUtime returns the number of clock ticks in a 64 bit integer since the computer was switched on.

To allow for fast or slow ball movement a rotation button is created under the billiard table.
This button is a home brew component but to make things easy for my readers. I have added the code as a separate unit and I create the button at the start.
No need to install foreign components.

The position value of the rotation button runs from 0 to 100. This value is used to control ball speed.



Ball movement processing takes about 500microseconds for 8 steps. (if 3Ghz clock)
Delay values range from 1000 to 10000 secsonds.
Increments of the delay value are relative, not absolute.

$$delay = a.e^{px} \qquad x=0 \rightarrow delay = 1000$$

$$a = 1000$$

$$x=100 \rightarrow delay = 10000$$

$$1000.e^{100p} = 10000$$

$$e^{100p} = 10 = e^{2.3} \rightarrow p = 0.023$$

$$delay = 1000.e^{0.023(100 - ballspeedBtn.position)}$$

100 - buttonposition because a higher value must yield a shorter delay time.

```
variable nextCPUtime =
CPUtime + CPUticks * delay.
```

BALL MOVEMENT

At each step, the ball positions are advanced by the value of their velocity. After that a check is made to see if an edge was hit. Then checks are made for collision : red/white, red/blue, white/blue.
`Procedure collision(a,b : TBallColor)` checks and processes ball movement for collisions.

The Meaning of variables in
`procedure collision(a,b : TBallcolor)`

xA	ball a X coordinate
yA	ball a Y coordinate
xB,yB	ball b similar...
dA	ball a direction
vA	ball a velocity, always positive
dB,vB	ball b similar...
d	distance between centers of balls a,b
dCC	direction of line between centers of balls
vAC	ball a centerline component velocity (+ or -)
vAP	ball a velocity of component perpendicular to center line (+ or -)
vBC,vBP	ball b similar...

For details I refer to the theory above and the source code.

INTRODUCTION

If the terminology "progressive web application" (PWA) is new to you, it is important to first give the definition of what a "progressive web application" is. In the words of Google, a progressive web application is a web application, i.e. it runs in the browser and it implements a series of requirements to make the web application "reliable, fast and engaging". It can also be interpreted as a web application that comes close to or matches the behaviour and features of a native installed application. To be classified as a "progressive web application", it must adhere to following guidelines:

- Is accessible via HTTPS
- Can be cached and implements cache control
- Adapts to different screen sizes, resolutions, i.e. mobile and desktop
- Defines its application name, description, colour & icon
- Continues to work in offline scenarios
- Needs to load fast enough
- Gracefully handles situations where JavaScript is not available
- Should not have blocking code

When an application satisfies these requirements, modern mobile device browsers (Chrome on Android or Safari on iOS) and the Chrome desktop browser (on Windows, Chrome OS, macOS, Linux) will recognize the application as a "progressive web application" and offer the users to install the application on the home screen or desktop. When the application is installed this way, it offers:

- It is launched full screen on a mobile device browser, i.e. without address bar as if it is a native application
- It is launched from cache in an offline situation
- It can deal with push notifications
- It can access device hardware such as camera, GPS, microphone

REQUIREMENTS IN DETAIL

Let's go over the requirements and have a look in detail about what must be done for the web application to meet these requirements.

HTTPS

To make the application accessible over HTTPS means in most cases simply that it needs to be deployed on a HTTPS enabled webserver.

CACHING

The second requirement is somewhat more complex. The application needs cache control. This is code that will control how exactly your web application should be cached. This code should be in a separate service worker JavaScript file and the service worker needs to be registered when the web application starts. The service worker is responsible for redirecting to the cache in an offline situation and specify what to store in the browser cache.

ADAPTS TO DIFFERENT SCREEN SIZES

This requirement will highly depend on what kind of web application you want to create. The bigger the web application is, the more difficult it might be to make it look good on a mobile device screen as well as a desktop screen. This is typically achieved in the web world with responsive design. So here, the developer needs to make sure the application only uses the available screen estate and is workable on both a mobile device browser and desktop browser.

DEFINES APPLICATION NAME, DESCRIPTION, COLOUR & ICON

This is relatively easy to handle by providing a manifest file. This is a separate JSON file with predefined JSON key/value pairs specifying things like icons for different resolutions, an application title, name, description, start URL and theme colours.

HANDLE OFFLINE SCENARIOS

The web application should gracefully handle offline situation. This means that the application should not cause any error when there is suddenly no internet connection available. Handling this can vary from catching the offline state and show a screen where the user cannot use any function that requires an internet connection to more sophisticated techniques that cache also application data and transactions while being offline and synchronize this when the application state goes from offline to online. This way, the user wouldn't notice the application is in offline state and can continue to use the full application functionality.

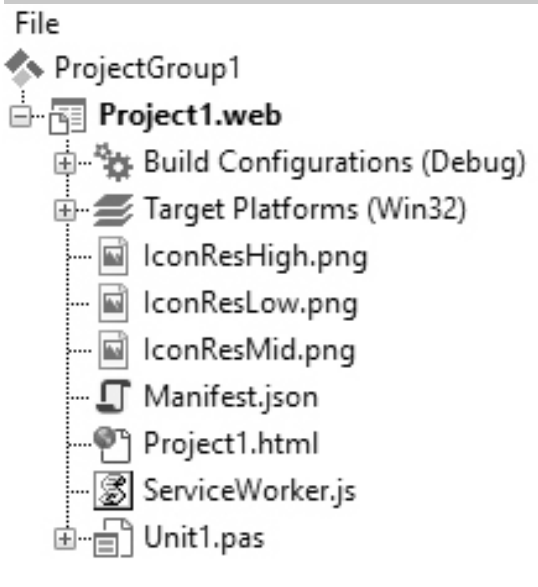
FAST LOADING
Making sure your application is not too big, is deployed on a fast server and doesn't start loading huge amounts of data in its loading phase is key here.

No JavaScript scenario
This is a simple requirement as it's sufficient to show a message that JavaScript is required instead of having the browser generate an error.

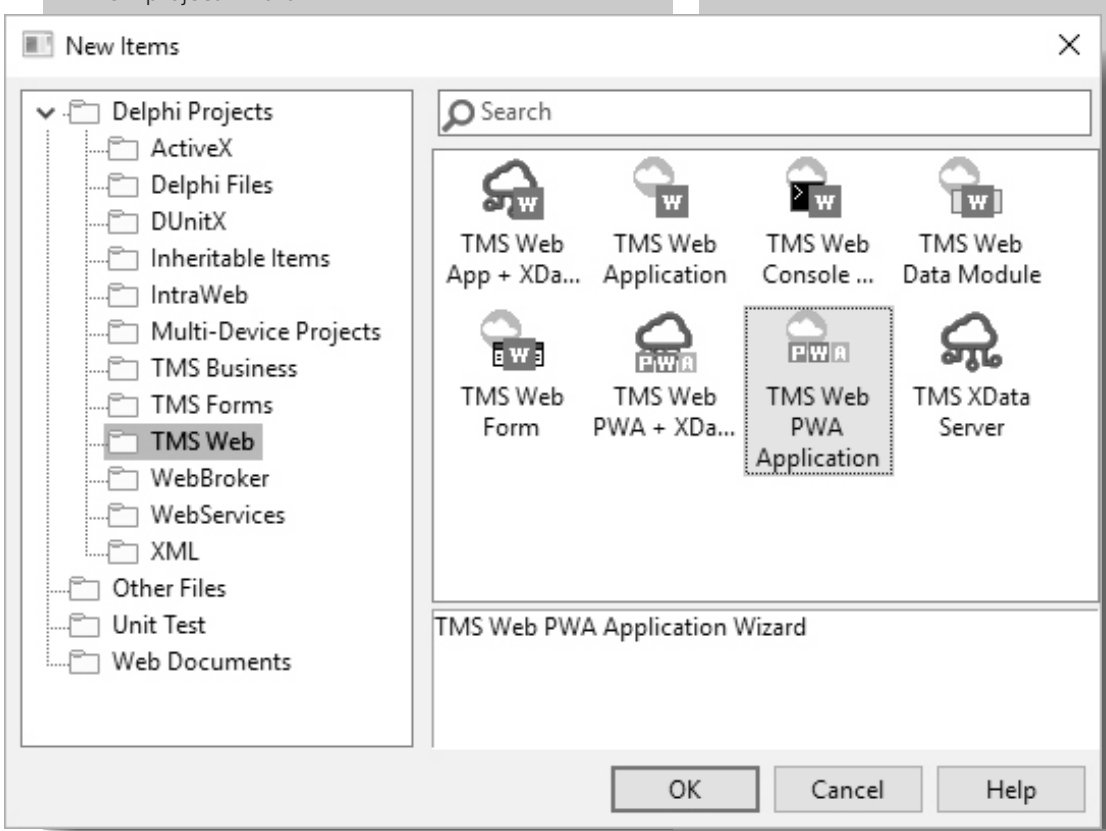
No blocking code
Regardless wether this is for a regular web application or progressive web application, using blocking code, like a synchronous HTTP(s) request, is a bad practice and should no longer be used.

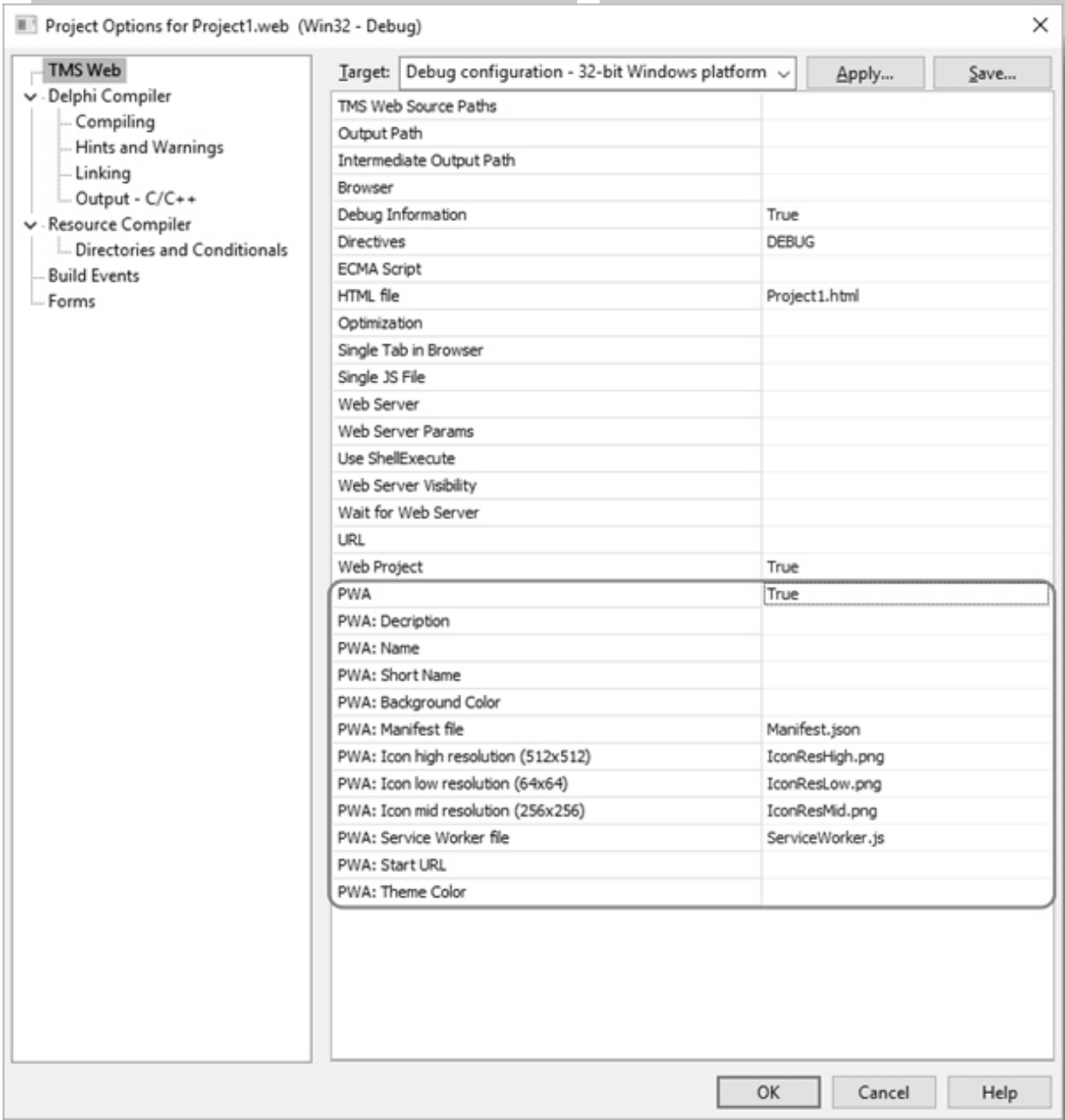
TMS WEB CORE PROGRESSIVE WEB APPLICATIONS
When creating a web application with TMS WEB Core, it has been made as easy as possible to make your web application a progressive web application. This means that the TMS WEB Core progressive web application template already generates the service worker JavaScript and manifest JSON file for you. The main application HTML file also handles the registration of the service worker, satisfies the requirements for setting the viewport correct on mobile and desktop browser and catches the no JavaScript scenario. The TApplication class also exposes the Applicaiton.IsOnline: boolean property as welll as an event handler Application.OnOnlineChange that is triggered when the browser goes from online to offline state or the other way around.
All this is already done for you when creating a new TMS WEB Core progressive web application from the Delphi IDE new project wizard:

The default project structure for a progressive web application is:



Access is provided to the application settings from the project property editor

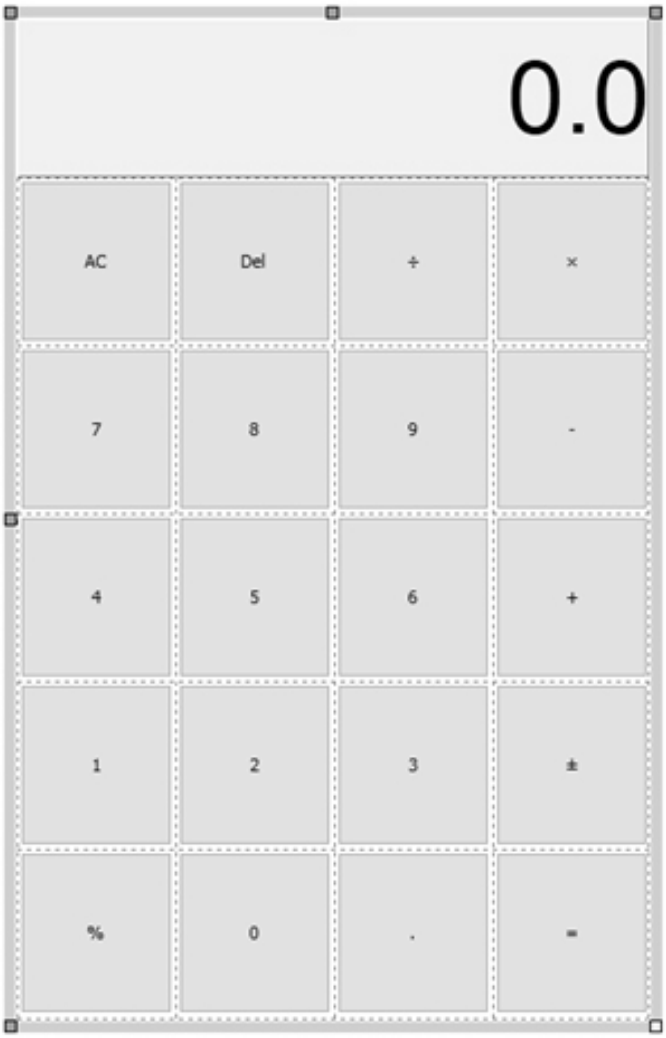




A first sample progressive web application
In TMS WEB Core, a sample progressive web application is included. It is a calculator application. As a calculator doesn't need server access, there is here no special code needed for handling online and offline scenarios. What is needed is designing the calculator this way that it looks good on a mobile device as well as on a desktop browser.
To do this, the calculator display is a client aligned label on a top aligned panel.

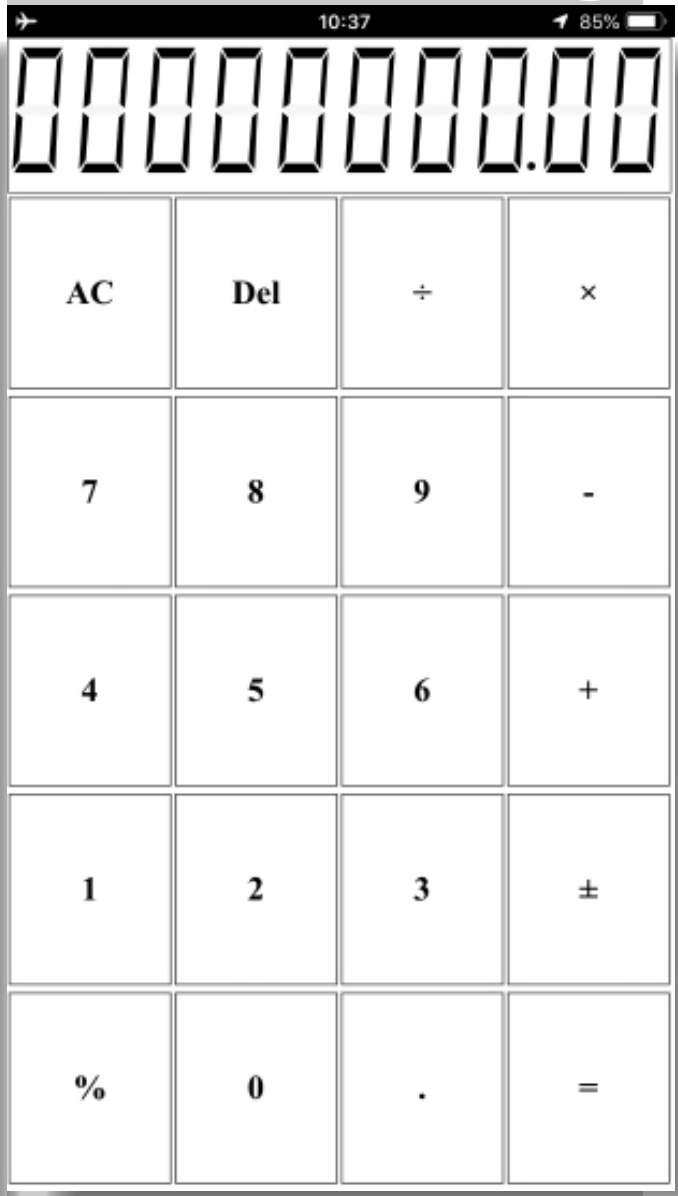
To make the calculator look good, we've used a 7 segment LED looking like TrueType font that is deployed along with the web application. This font resource is added to the project and having done this will let the service worker do the necessary to also cache this font.

Next, for the calculator buttons, we use a client-aligned TWebGridPanel with 4 columns and 5 rows. Each cell in the grid panel hosts a client aligned button for the calculator operations. Using the grid panel and alignment, this ensures that the calculator buttons are always displayed nicely, regardless of screen size.
In the IDE, it looks like:



Other than this, there is not much more to do to create our first TMS WEB Core progressive web application except to deploy it on a HTTPS enabled server. When accessing the URL on a smartphone, the experience becomes:

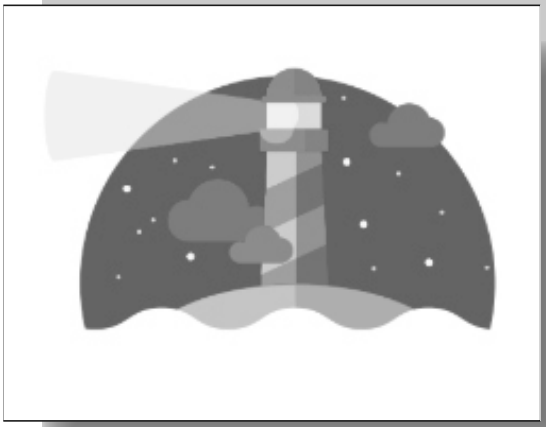
From the manifest, iOS found out the application name & icon and suggests to add it this way to the desktop:



And now comes the really nice thing, let's switch the iPhone to airplane mode (notice the airplane mode indicator in the top left corner of the screen), we can still start our TMS WEB Core application and use it:

TOOLS

When developing more complex progressive web applications, Google provides a very handy tool Lighthouse to do automated testing on your web application to check if it meets all requirements.



When it doesn't meet the requirements, you will see the details where exactly some is wrong that needs to be addressed to make it a progressive web application. The Google Lighthouse tool can be found in the Google Chrome store and can be installed as plugin in Chrome. When your web application is open in the browser, click the Lighthouse icon on the toolbar in Chrome and the report generation starts. When Lighthouse reports your web application meets the PWA requirements 100%, you can be sure that Chrome will allow to install the application on the device.

CONCLUSION

Progressive web applications will play in many ways a very important role and will be a crucial technology to make web applications behave nicely when switching between online / offline situations, switching between desktop and mobile devices and as such offer a better user experience. Moreover, it offers a mechanism to deploy applications to end-users without passing via the Apple, Google or Microsoft gateways and paywalls. We're researching and implementing the needed support in the TMS WEB Core framework to make developing such progressive web applications as easy as it can be.

LITERATURE

To learn more about progressive web applications, we recommend following resources:
https://en.wikipedia.org/wiki/Progressive_web_applications
https://developers.google.com/web/progressive-web-apps/

You can get started with TMS WEB Core using a fully functional trial version available at
https://www.tmssoftware.com/site/tmswebcore.asp

Made by a Win 10 standard control TCalenderView (CV) and linked to a free kbmmwMemTable and helped by a free Colorbutton built in Delphi Rio. Your Data will never go to Microsoft or to Google

Figure 1: overview of the different versions of the app

INTRODUCTION

This month-calender program was created because of - as so often - annoyance. First of all I wanted my appointments not to get into the hands of any third Party for obvious reasons. I wanted also to create a very simple small project that does several things for programmers. Usually the data components and especially Month-Components are not so easy to use; you almost never can do what you want in the way you want it and so you need to be very pragmatic.

To start I will show you how to prepare a program in its essence, I learned in my days to simply write down all the requirements, sketch the UI by words and designing. That is NOT a very difficult task and one should be able to make these outlines in maximum one hour.

After that write some pseudo - code:

```
procedure fizzbuzz
 For i := 1 to 100 do
  set print_number to true;
  If i is divisible by 3 then
   print "Fizz";
   set print_number to false;
  If i is divisible by 5 then
   print "Buzz";
   set print_number to false;
  If print_number, print i;
  print a newline;
end
```

I think especially for starters it still is a very useful way of organizing things. (I haven't yet done the Test Unit, that might be worth its own article). As usual I will also convert this one to Lazarus, first a trial and then solve possible differences. Or in the end if the conversion does not work, I can create a new program that does the same.

And the last problem to solve:

Let' s make a **Progressive Web App**. I call them WebTopApps because they are the internet equivalent of a DesktopApp. How to do that you can learn in Bruno's (TMSSoftware) article on page... where the basic principles are provided to do such thing. But the Lazarus version will be in a next edition I suppose, because we do not have all the time to do so in this issue.

And there is a lot to learn from this simple product; let' s start with the Requirements:

REQUIREMENTS:

1. Make it as small as possible, but flexible if needed
2. It should remember where you want it on your desktop In an intelligent way.
3. The month view is sometimes needed so it would be able to extend.
4. For the User interface it would be nice you can choose of a lot of extra functions:
   - Handle its segments by clicking on it.
   - Handle it by right-clicking.
   - Move it by altering the form size and border so it can be dragged to its final place by you.

   In case of the smallest version there needs to be a simple way of closing the program.
5. Its workings need to be as simple as possible so you can make your own alterations without to have to rewrite all of its code. The code part will be made by components that are standard available or that are free to use.
   Therefore you will have to install the **Client Dataset** - the MemTable (free) from **kbmMW** and my favourite component:
   the **ColorButton- code** also available for free. I chose to make the year/month calendar out of 16 elements because that allows you to have always a complete year overview and since I would love to be able to put in some appointments we need the Client Dataset. I chose kbmMWmemtable because it' s free and it has far more features than what is standard in Delphi (Not yet Lazarus - but will be soon). You have a much better and easier way of handling it: you can handle it with SQL. (Not only with filters, like the normal Client Datset).
   And its handling and preparing for starting it is normally a problem. Not so in the **kbmMWmemtable.**
   I advise you to use it and in an other article in this issue, page 43, I tell you how to install it and what you should know before starting.

USER-INTERFACE SKETCH
See the drawings. Let' s make fun so that people can change the colour of the form and choose a colourbutton that allows you to change its colour. This kind of a month/year/clock should hardly be visible in its smallest view, so I think changing colour adds to keep your desktop simple.

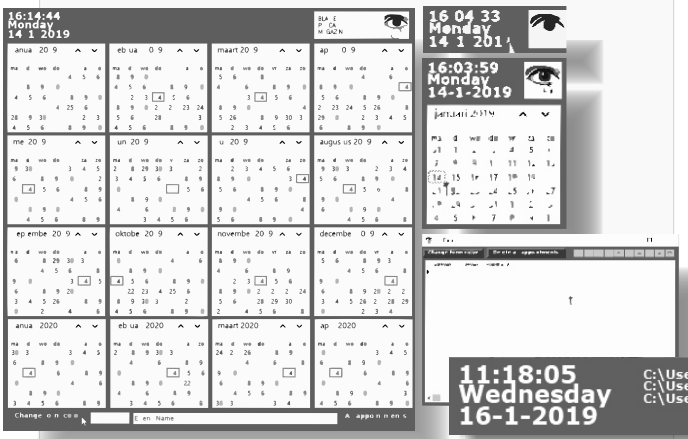


Figure 2: Sketching the forms

PSEUDO CODE AS A WORKING EXAMPLE
Remember that Pseudo code has one big advantage.
By writing down the pseudo code you have already an almost complete documentation of your code! I myself have a preference for as simple code as possible.
Way after you have written your code, someone will try to understand it again. So... On top of that the real Einstein's will write it in that way, showing of how compact your could be, becomes later a burden.

Figure 3: The iconic view:

To make things work we need to find the places where we can find the settings which are needed to know: there are two functions: **GetFolderPath** and **GetShellFolder**
GetFolderPath gets the path to the system special folder that is identified by the specified enumeration, and uses a specified option for accessing special folders.

```
function GetFolderPath(Wnd: HWnd; CSIDLFolder: Integer): string;
begin
  SetLength(Result, MAX_PATH);
  SHGetFolderPath(Wnd, CSIDLFolder, 0, 0, PChar(Result));
  SetLength(Result, StrLen(PChar(Result)));
  //specific for windows
  if (Result <> '') then Result := IncludeTrailingBackslash(Result);
end;
function GetShellFolder(CSIDLFolder : integer) : string;
begin
  SetLength(Result, MAX_PATH);
  SHGetSpecialFolderPath(0, PChar(Result), CSIDLFolder, false);
  SetLength(Result, StrLen(PChar(Result)));
  //specific for windows
  if (Result <> '') then Result := IncludeTrailingBackslash(Result);
end;
```

In the oncreate **procedure TCalendarFrm.FormCreate(Sender: TObject);** happens among other settings - which are not described because they are obvious - the following:
I have put the labels (3,4 and 5) that show these paths to be invisible - for trial or other purpose you can set them to visible again-, I needed them only when I started the very first application design. If you want to know some more about this : **https://docs.microsoft.com/en-us/dotnet/api/system.environment.getfolderpath?view=netframework-4.7.2**

GetShellFolder does almost the same: This function is a superset of SHGetSpecialFolderPath, included with earlier versions of the Shell. See the result in Figure 4. This might be interesting if you want to make an installer for this program. The syntax of this function is not very complicated and speaks for it self.

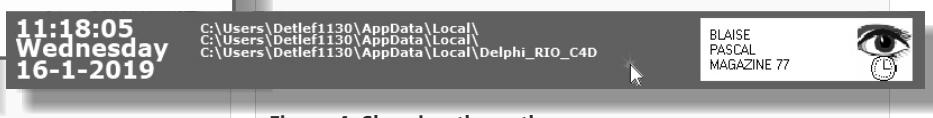


Figure 4: Showing the path

In the form-create we need to create an array because we need to read the exact translation of the day: is it Monday or is it Friday, I would like to see that at once:

```
Var day   : array[1..7] of string;
....
 day[1] := 'Monday';
 day[2] := 'Tuesday';
 day[3] := 'Wednesday';
 day[4] := 'Thursday';
 day[5] := 'Friday';
 day[6] := 'Saturday';
 day[7] := 'Sunday';

LblWeekday.Caption:= day[DayOfTheWeek(Now)];
```

DayOfTheWeek is ISO 8601 compliant, since it uses Monday as the start of the week. So because of this all we have now the ability to run the first smallest design. See Figure 3

There is also a very attractive group of special date functions, almost overwhelming: **`CV2.DATE := INCMONTH(DATE, 1);`**

```
function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime;
```

Description: Returns a date shifted by a specified number of months.

IncMonth returns the value of the Date parameter, incremented by **NumberOfMonths** months. **NumberOfMonths** can be negative, to return a date N months previous.

If the input day of month is greater than the last day of the resulting month, the day is set to the last day of the resulting month. The time of day specified by the Date parameter is copied to the result.

See Also:
IncAMonth
DecodeDate
EncodeDate
ReplaceDate
IncDay
IncHour
IncMilliSecond
IncMinute
IncSecond
IncWeek
IncYear
Date and Time Support
Category: API Documentation
This is a very nice category of API calls that are very helpful.

MANAGING THE KBMMEMTABLE

This part consists of 2 main elements, because there is some extra information involved.

ELEMENT 1.
Now the next very interesting part starts: integrating **kbmMW ClientDataSet**. As I said before: on page 61 in this issue I explain how to install the Memtable Dataset. So here I use it as being installed already. It is like all other datasets a matter of drag and drop and creating fields. Creating the fields and initiating them is very much different from the normal ClientDatset. Its initiation is far easier then in any other Dataset.
Yet because we are in the creation part of the app, we need to prepare some settings; after having put a `TDataSource` on the form followed by the `kbmMemTable1` and the `kbmBinaryStreamFormat1` we can start on explaining:
The `TDataSource` is needed to make a connection between a grid and the table. This grid will be shown in a second form that we prepare later on.

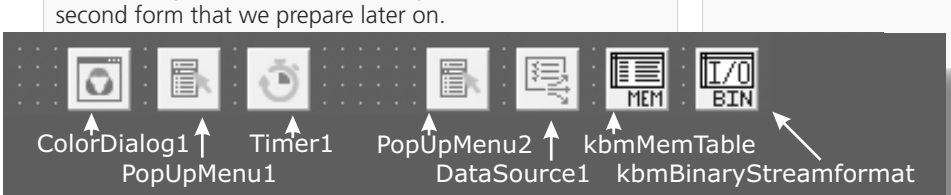


Figure 5: Sketching the forms

Now drop these items on the form:
`TDataSource / kbmMemTable1 / kbmBinaryStreamFormat1.`
The `TDataSource` has to be connected with the `kbmMemTable1`, do that in the **ObjectInspector** under `Dataset : kbmMemTable1`

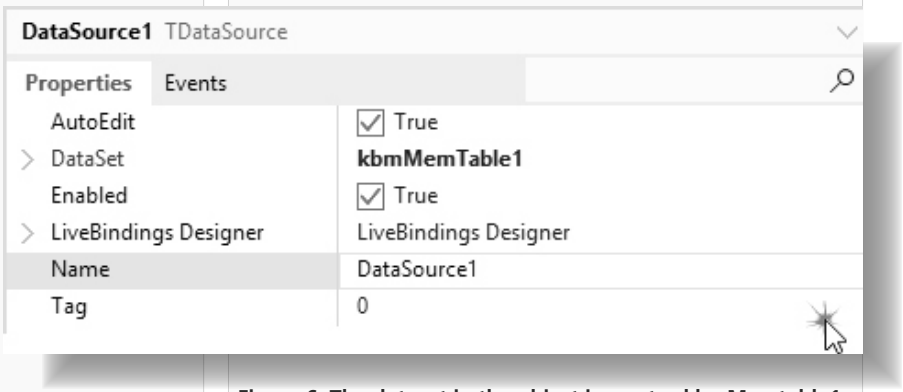


Figure 6: The dataset in the object inspector kbmMemtable1

Now you need to connect the `kbmMemTable1` to the `kbmBinaryStreamFormat1.`
Why? Because if you want to save the content you need to have a file that saves your appointments and other data.

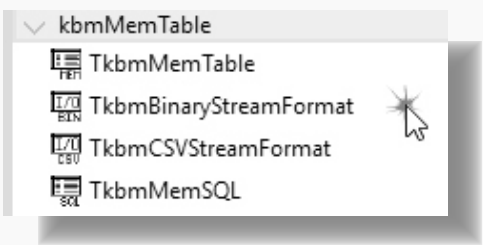


Figure 7: Listing the memtable components
I have chosen the Binary format because you can choose out of `kbmBinaryStreamFormat` and `kbmCSVstreamformat.`

A BINARY FILE
is a computer file that is not a text file. The term "binary file" is often used as a term meaning "non-text file". Many binary file formats contain parts that can be interpreted as text; for example, some computer document files containing formatted text, such as older Microsoft Word document files, contain the text of the document but also contain formatting information in binary form.

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

Figure 8: a binary file

A CSV File
In computing, a comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.
The CSV file format is not fully standardized. The basic idea of separating fields with a comma is clear, but that idea gets complicated when the field data may also contain commas or even embedded line-breaks. CSV implementations may not handle such field data, or they may use quotation marks to surround the field. Quotation does not solve everything: some fields may need embedded quotation marks, so a CSV implementation may include escape characters or escape sequences.
In addition, the term "CSV" also denotes some closely related delimiter-separated formats that use different field delimiters, for example, semicolons. These include tab-separated values and space-separated values. A delimiter that is not present in the field data (such as tab) keeps the format parsing simple. These alternate delimiter-separated files are often even given a .csv extension despite the use of a non-comma field separator. This loose terminology can cause problems in data exchange. Many applications that accept CSV files have options to select the delimiter character and the quotation character. Semicolons are often used in some European countries, such as Italy, instead of commas.

STREAMING:
just a continuous stream of data. I found a very special explanation series

streaming
1. *COMPUTING*
a method of transmitting or receiving data (especially video and audio material) over a computer network as a steady, continuous flow, allowing playback to start while the rest of the data is still being received.
2. *BRITISH*
the practice of putting schoolchildren in groups of the same age and ability to be taught together.
"streaming within secondary schools is common practice"

adjective
1. *BRITISH*
(of a cold) accompanied by copious running of the nose and eyes.
"she's got a streaming cold"

2. *COMPUTING*
relating to or making use of a form of tape transport, used mainly to provide backup storage, in which data may be transferred in bulk while the tape is in motion.

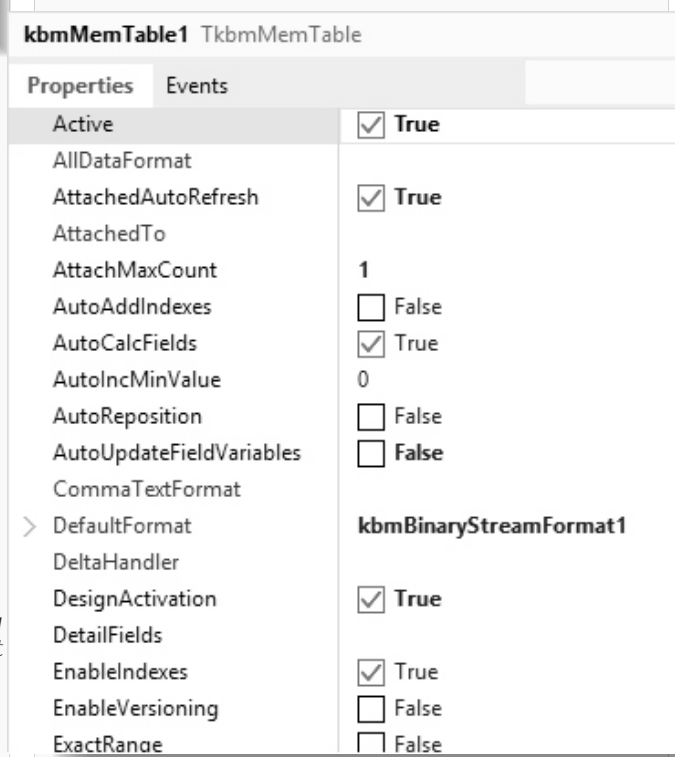


Figure 9: The default format must be set

Now we have to activate the database

```
kbmMemTable1.Open;
kbmMemTable1.Active;
```

and then set the name of the file where the data in is recorded:

```
kbmMemTable1.LoadFromFile(ExtractFilePath(Application.exename)
+'Calendar_kbmMW_Binary.detlef');
```

This will make sure you can choose whatever place you would like your project to be in registered if you keep your exe-name there as well. To create your Database:
Right click on the memtable: there will pop up a window

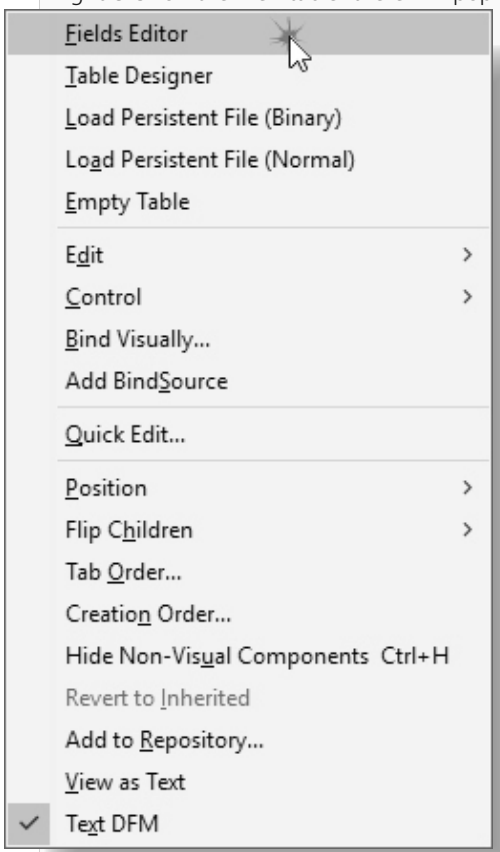


Figure 10: Right clicking on the memtable shows the pop up list

The Fields Editor choice means the Editor will show up. See figure 11

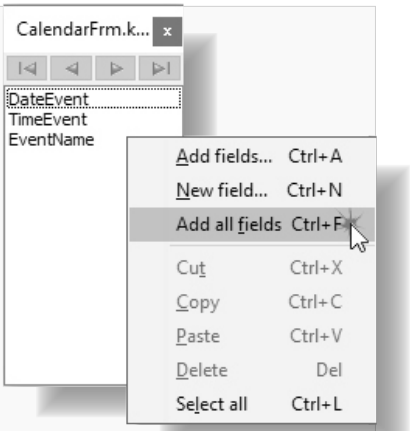


Figure 11: Right clicking on the memtable shows this pop up list

The Table Designer will appear. See figure 12. You can also load the persistent file (Binary) or load a normal file.
You can also clear the table.

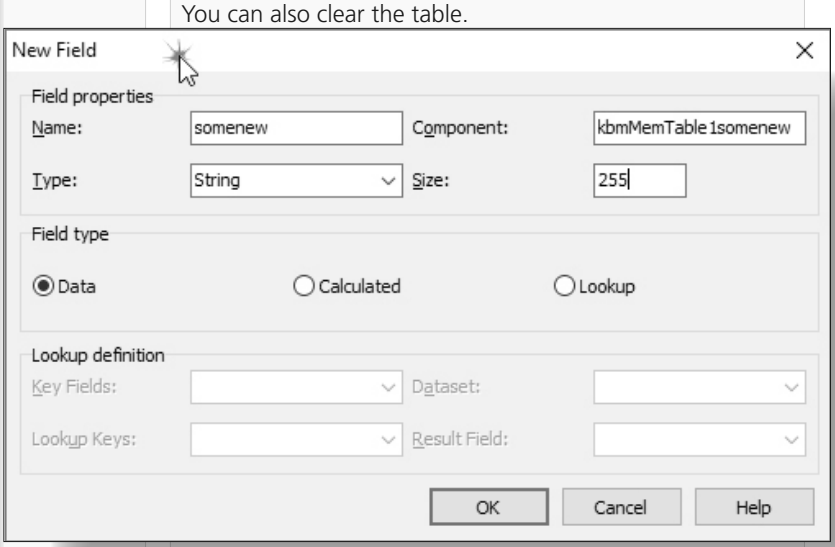


Figure 12: Table Designer

As soon as you chose to add a field, or create a new one you will be shown the **Field Editor.** At the top of the left side of the Editor is an item **Name** and a field.
Fill in a name ('somenew') and you will see the name coming back in the right Component field: **kbmMemTable1somenew** which is a merging of the name of the table and your fieldname.
You can of course give it your own name.

The type has a drop-down-list of field-values which you will probably know from building databases.
The size is dependent of the kind of values you select. Sometimes you can make choices, but often you can't because it's pre-defined.
If you are more experienced you can even choose for the kind of field: Data or Calculated or Lookup.
On page 20 in this article I show how to do this in a detailed simple project. You can of course find the field as I programmed as an example. I made it very simple: three fields: DateEvent which is of DateTime, TimeEvent which is Time and EventName is of type string and I gave it a reasonable length: 150.

ELEMENT 2.

TESTAPP
To explain how you connect and start the **kbmMemTable - ClientDataSet** I created a separate project for you to learn how to create the memtable:

Simply create a new VCL-app and then put five components on the form:
a **DBGrid,** a **DBNavigator, DataSource, kbmMemTable** and a **kbmBinaryStreamFormat.**
You probably will not yet have arranged for some necessities that occur if you install new components:

You need to handle the **path for kbmMemTable.**
These are two steps you will have to control and or alter:
In **Delphi** go to **Tools -> Options -> Environment Variables.**
In the lower part called **User System Overrides** you can add if necessary a new variable and a path to the correct Directory.

First step:

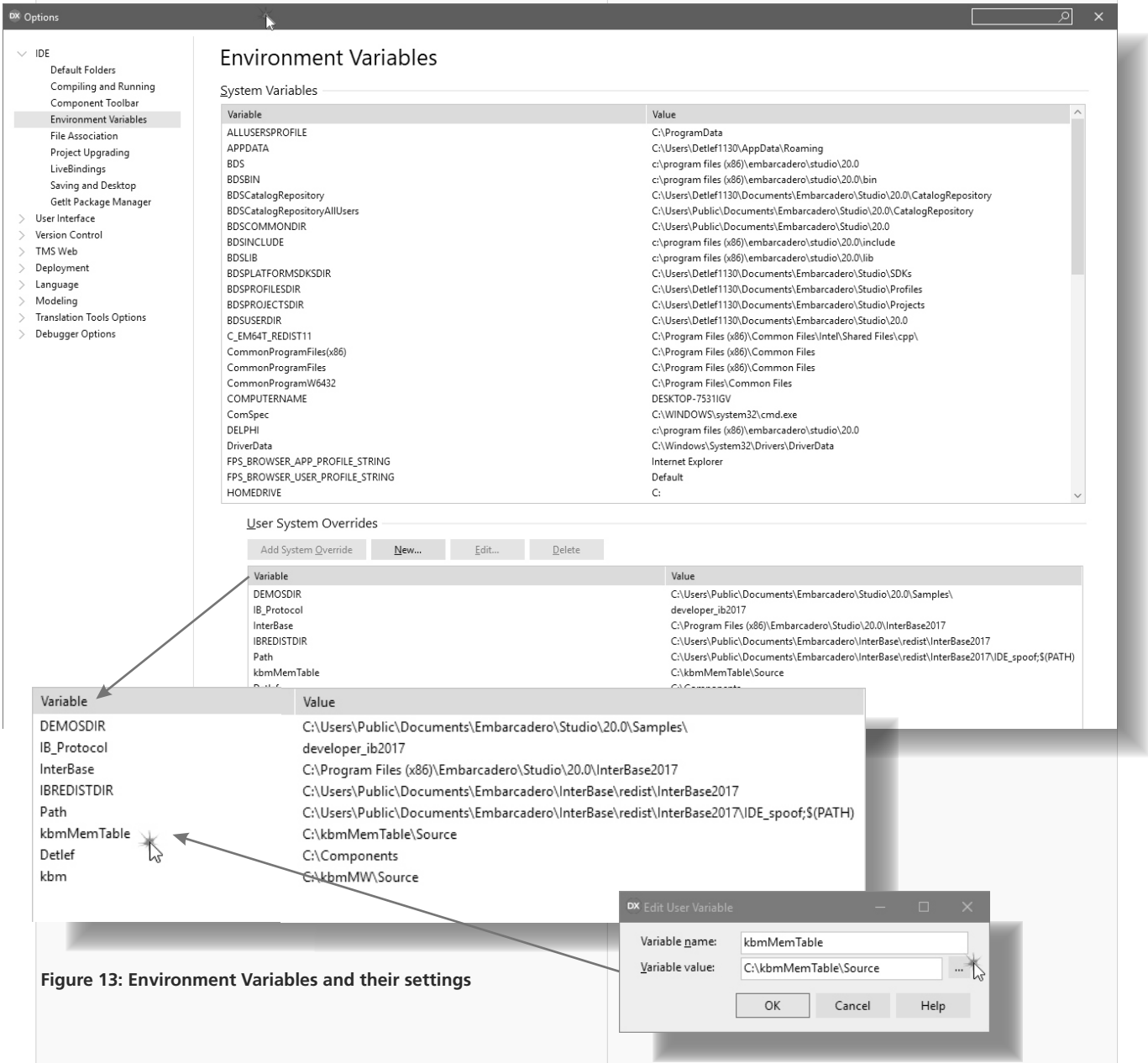


Figure 13: Environment Variables and their settings

The second step:
Go to **Project Options: -> Project -> Options:**
Go to Search path, click on the ellipsis-button at the end of the line ••• (three dots)

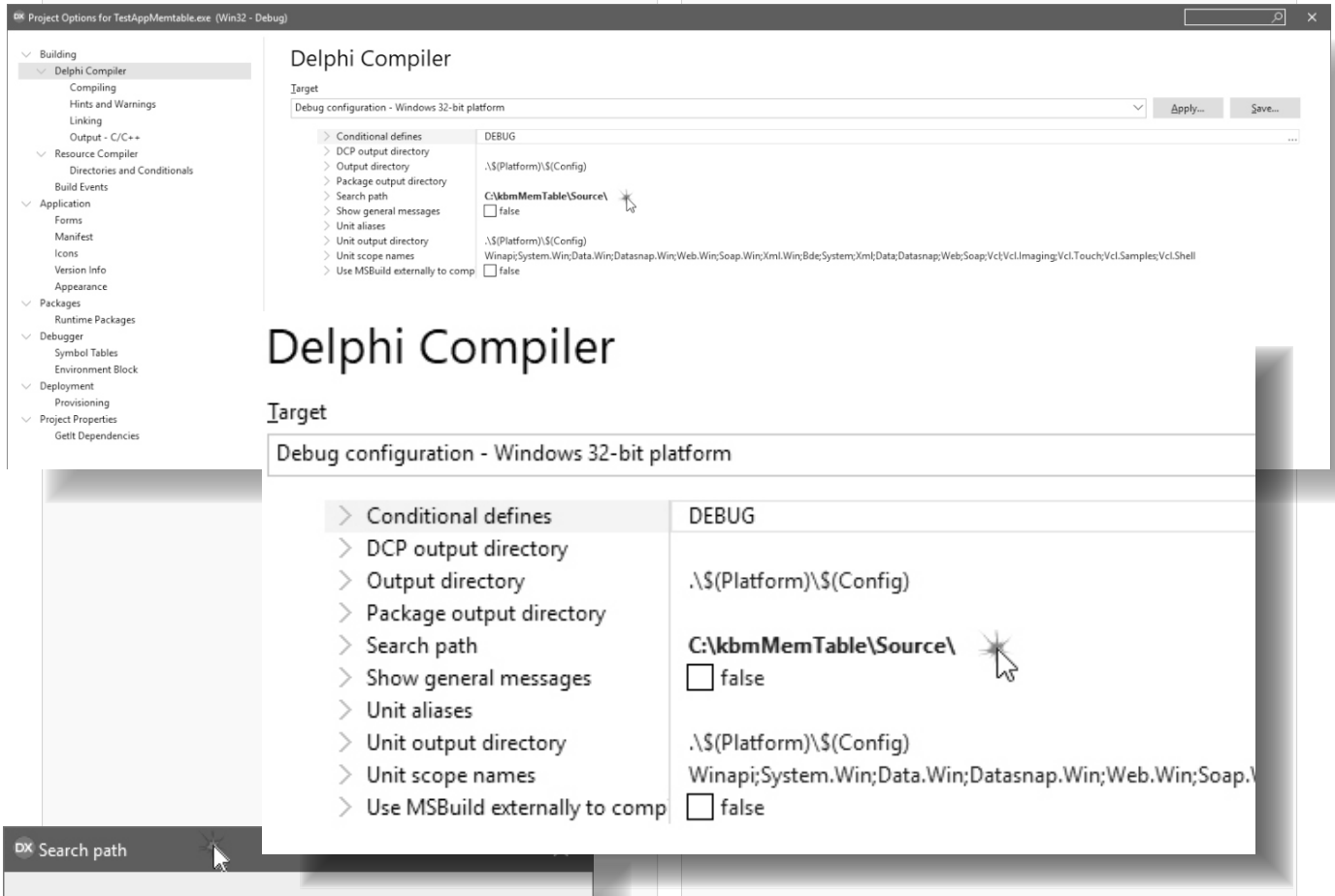


Figure 14: Project Options

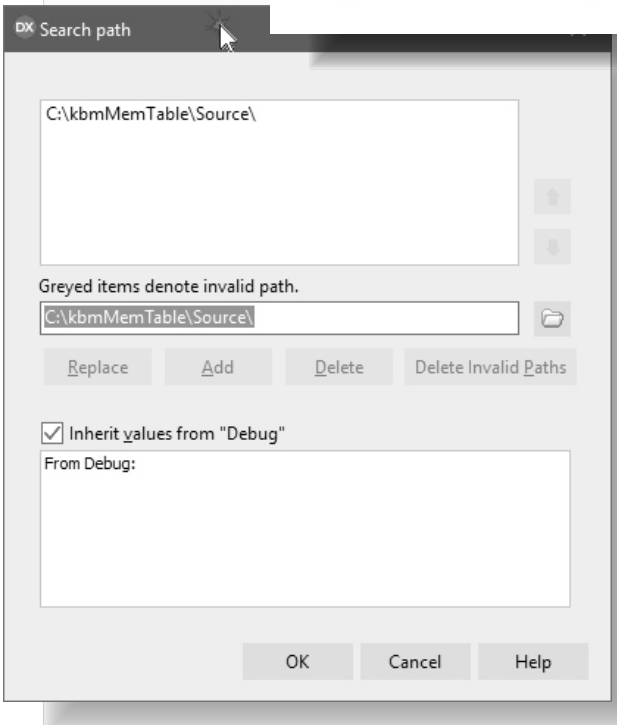


Figure 15: Project Options detail adding a path

The details window opens and here you can alter the path by **add / replace and remove.**
If after these two steps **Delphi** still complains about something, there is an error you need to solve your self.
If you do add the path to deep (Something like **`c:\kbmMemTable\Source`**`\D103\Win64\`) you might get into trouble as well.

Simply add something like this:
`c:\kbmMemTable\Source\` not
`c:\kbmMemTable\Source\D103\Win64\`
because it can't find it then.

Now let us put together the **test app:**
we connect the **DataSource** with the **kbmMemTable,** connect the **DBGrid** and the **DBNavigator** with the **DataSource** and finally the **`kbmMemTable`** (default format) with the **`kbmBinaryStreamFormat.`**

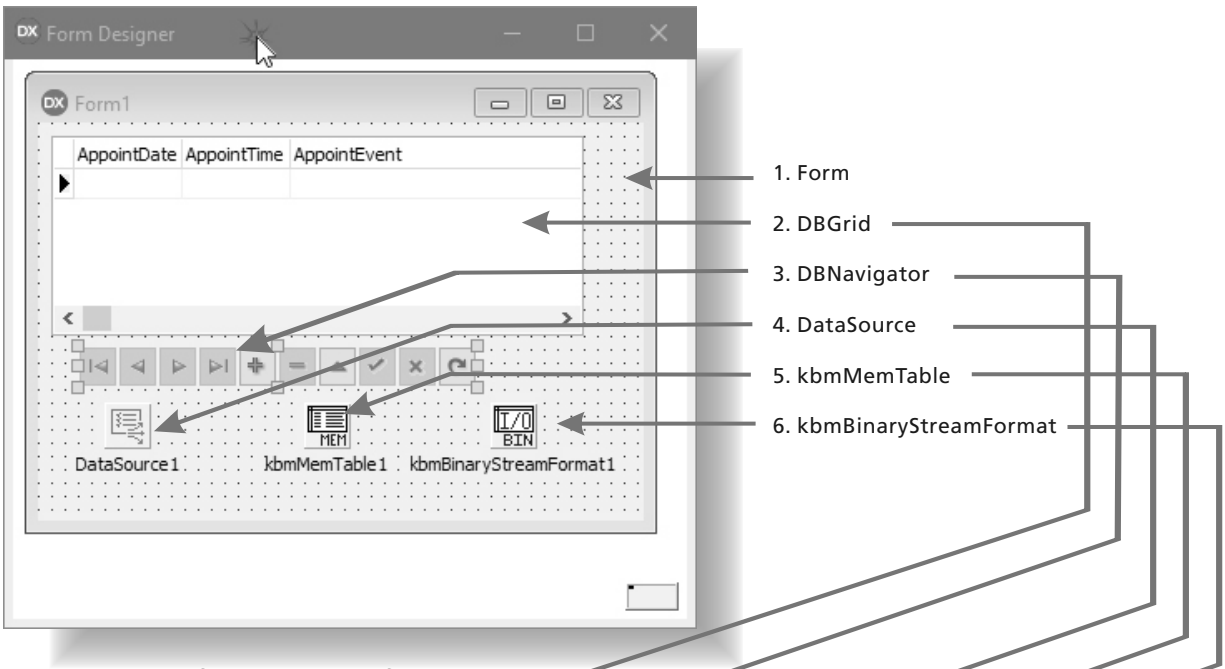


Figure 16: The form as example of all the items to be used

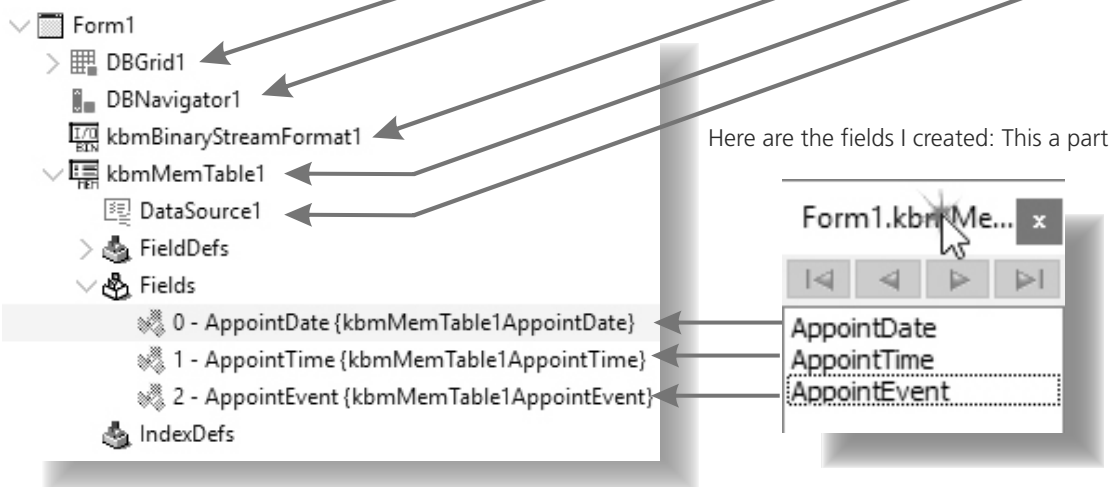


Figure 17: The IDE shows the items that are used

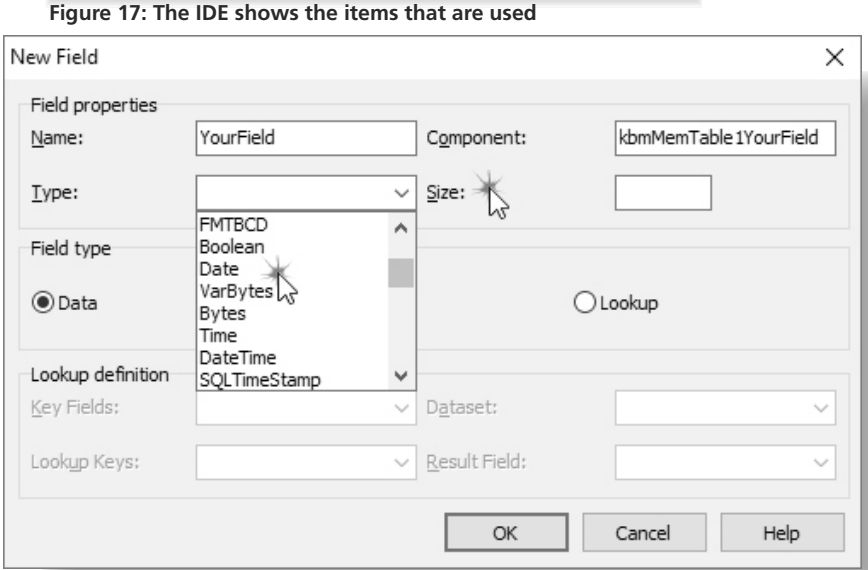


Figure 18: The field editor in detail

we of course need to have an on **OnCreate**;

```
procedure TForm1.FormCreate(Sender: TObject);
begin
// kbmMemTable1.LoadFromFile(ExtractFilePath(Application.exename)
// +'TestMemTable_kbmMW_Binary.MyOwnExtension');
 kbmMemTable1.LoadFromFile(ExtractFilePath(Application.exename)
 +'Calendar_kbmMW_Binary.detlef');
end;
```

This is Example-code, which is different from what is written in the program. I created two variables to make sure that we will have the right creation of Directories and Files: **FN** meant for the ini-file that has to be set in a directory in windows where you are allowed to alter a file. If you work under **Win10** you will notice that files in the directory for the program itself can not be changed without Administrator-rights. So I chose to put the ini-file aswel as the to be created or updated file of the Memtable **(Variable kbm**) in the AppData section of windows. In the final install file you will find the settings for all this.
All we need to do now is to become able to fill the grid with data and save that we need to add this to the **OnClose** action:

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  If (kbmMemTable1.State in [dsEdit, dsInsert]) then kbmMemTable1.Post;
 kbmMemTable1.saveToFile('Calendar_kbmMW_Binary.detlef');
 end;
```

BACK TO THE APP
At the first-time- compilation you could get a warning that the file does not yet exist. After the closing-action of the app it will exist, so that is all. No problem with handling this memtable. It's never been so easy.

After all this you should be able to understand the creation of a complete kbmMemTable test app.
Let us go back to the `OnCreate` event of the Calendar app and the part I did not yet explain:
there is a small helper that might be of interest:

ReportMemoryLeaksOnShutdown:=true;
It is part of the Delphi Ide. A complete version of this (fastMM) you can obtain from
http://delphibistro.com/?p=186

Reports memory leaks on shutdown.
***Warning:** ReportMemoryLeaksOnShutdown only works on Delphi applications, it has no effect in C++ applications and in packages.*
Set `ReportMemoryLeaksOnShutdown` *to report memory leaks on shutdown.*
The memory manager can report memory that was allocated but not freed by the time the memory manager shuts down. Such memory blocks are called memory leaks and are often the result of programming errors. When this global variable is set to **True**, *the Memory Manager will scan the memory pool when it shuts down and report all unregistered memory leaks in a message dialogue. The default value for* `ReportMemoryLeaksOnShutdown` *is* **False**.

All the settings for the placement of the form are recorded in C4D_RIO_appINI in the OnCreate.

```
 ...
// kbmMemTable1.saveToFile(ExtractFileName(Application.ExeName)+'Calendar_kbmMW_Binary.detlef');
 kbmMemTable1.saveToFile(kbm);
 C4D_RIO_appINI := TIniFile.Create(ChangeFileExt(FN,'.ini'));
 try
  Top  := C4D_RIO_appINI.ReadInteger('Placement','Top', Top);
  Left  := C4D_RIO_appINI.ReadInteger('Placement','Left', Left);
  Width := C4D_RIO_appINI.ReadInteger('Placement','Width', Width);
  Height := C4D_RIO_appINI.ReadInteger('Placement','Height', Height);
 finally
  C4D_RIO_appINI.Free;
 end;
```

In the **onClose** event:

```
procedure TCalendarFrm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  ...
  C4D_RIO_appINI := TIniFile.Create(ChangeFileExt(Fn,'.ini'));
  try
  with C4D_RIO_appINI, CalendarFrm do
  begin
    WriteInteger('Placement','Top', Top);
    WriteInteger('Placement','Left', Left);
    WriteInteger('Placement','Width', Width);
    WriteInteger('Placement','Height', Height);
   end;
 finally
 C4D_RIO_appINI.Free;
 end;
```

To get the selected Data from the (CV) CalendarView elements we have to build for each item a **DoubleClick** event.Double click because the Calnedarform will respond to onclick evebnts for selcting dates and or changing months and years tec.

```
procedure TCalendarFrm.CV2DblClick(Sender: TObject);
begin
 if Edit1.Text = '' then
 begin
  showmessage('add the time');
  Exit;
 end
 else
 begin
  kbmMemTable1.Edit;
  kbmMemTable1.append;
  kbmMemTable1.FieldByName('DateEvent').AsDateTime := (CV2.Date);
  kbmMemTable1.FieldByName('TimeEvent').AsDateTime := StrToDateTime(Edit1.Text);
 end;

 if EditEvent.Text = '' then
 begin
  showmessage('add the event name');
  EditEvent.Text;
  exit;
 end
 else
 begin
  kbmMemTable1.FieldByName('EventName').AsString  := EditEvent.Text;
  ColorButton5Click(sender); // Only for calendarview 1
 end;
```

In the **onShow** I have created the LoadDates Procedure

```
procedure TCalendarFrm.FormShow(Sender: TObject);
begin
  LoadDates
end;
```

The procedure it self consists of sections for each calendar:

```
procedure TCalendarFrm.LoadDates;
Var I,D: Integer; DA: Array of Tdate;
begin
 // CV1 //////////
 CV1.Visible := True;
 SetLength (DA, kbmMemTable1.RecordCount);  /// necessary for the array length
 With kbmMemTable1 Do
 Begin
  First;
  for D := 0 to RecordCount -1 do
  Begin
   DA[D] := trunc(FieldByName('DateEvent').asdatetime);
   /// The Truncate procedure truncates a file at the current file position.
   /// All data after the current file position is erased.

   Next /// Repetition
  End;
 End;
 /// Add the selected data - array
 CV1.AddToSelectedDates (DA);
```

Then a few last things

```
procedure TCalendarFrm.ColorButton1Click(Sender: TObject);
begin
 /// Changing the form color
 if ColorDialog1.execute then CalendarFrm.Color := ColorDialog1.Color;
end;

procedure TCalendarFrm.ColorButton5Click(Sender: TObject);
begin
 /// Creating the form
 If FrmData = Nil Then FrmData := TFrmData.Create(Self);
 FrmData.Show;
end;
```

Installing the ColorButton is easy itself no worries:
Go to Component/Install component

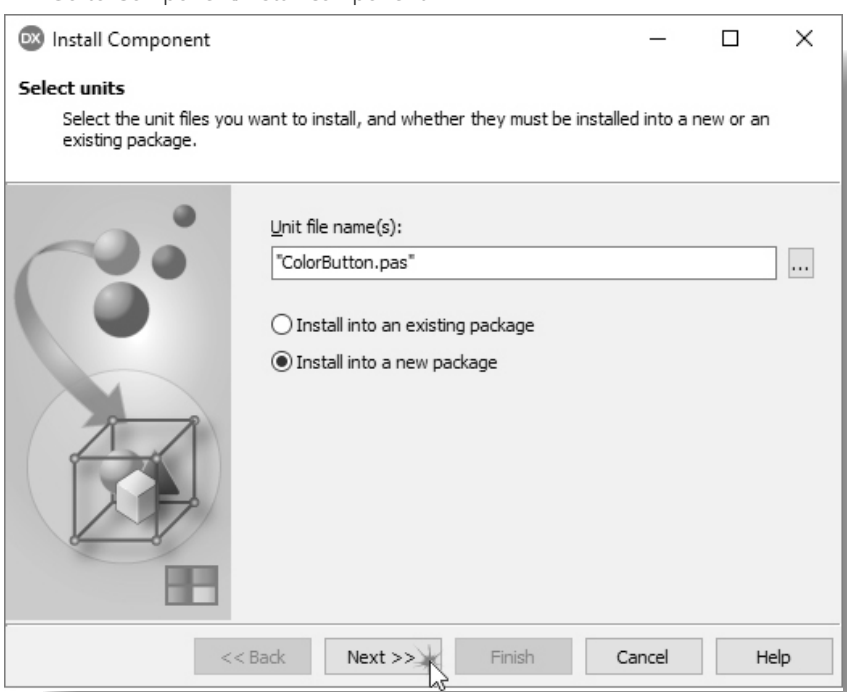


Figure 19: after selection of the file

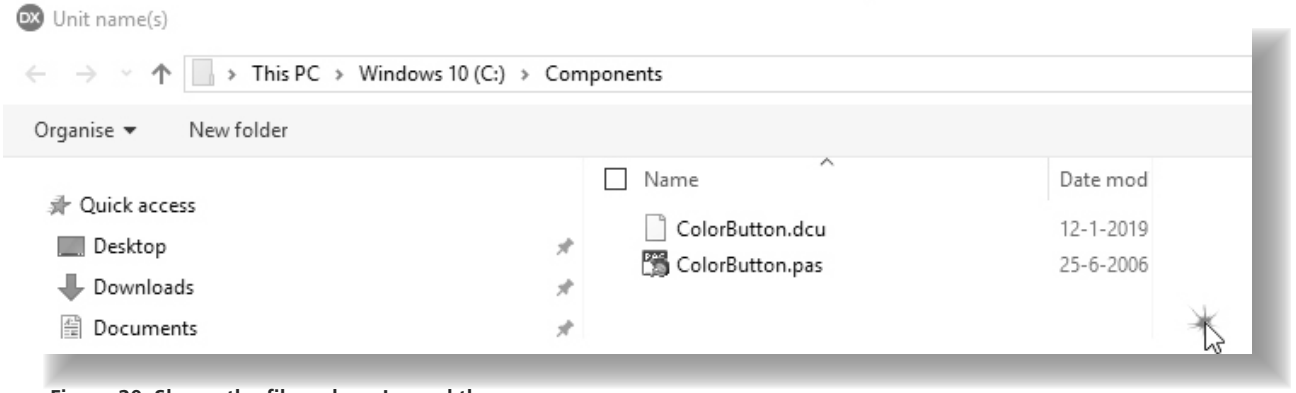


Figure 20: Shows the files where I saved them

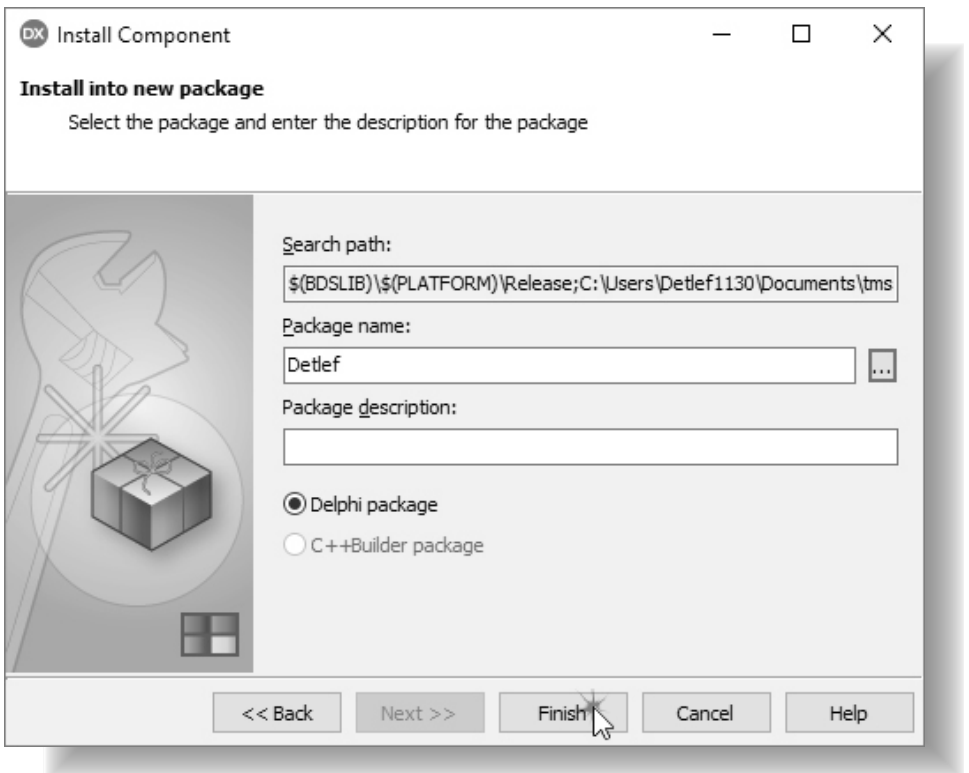


Figure 21: The Package Name is of course abitrary, the description could be an explanation like: Button with colours and Pictures

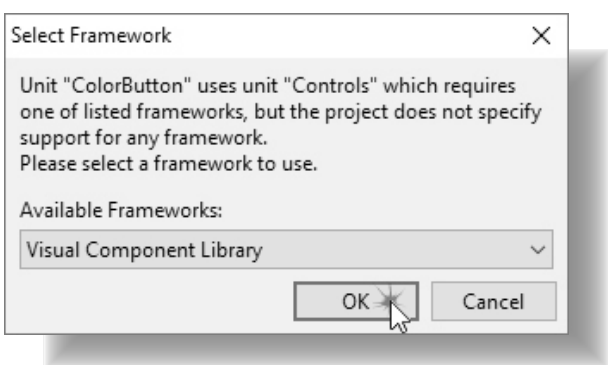


Figure 22: If you do not have yet an available FrameWork it's best to place it under the VCL

USING THE APP

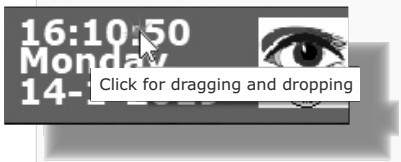


Figure 23: Click for exit and draging

If you click on the time label, the form view will change to a normal windows view where you can drag the Calendar clock to where you want it to appear next time as well. You can close on the normal windows cross at the right top

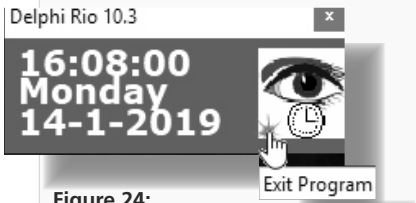


Figure 24:
Normal form view

Normal form view: Drag and drop enabled. You can also click on the icon and the the app will be closed.

Figure 25:
Right click for more options

By right clicking a popup will show up, with extra options

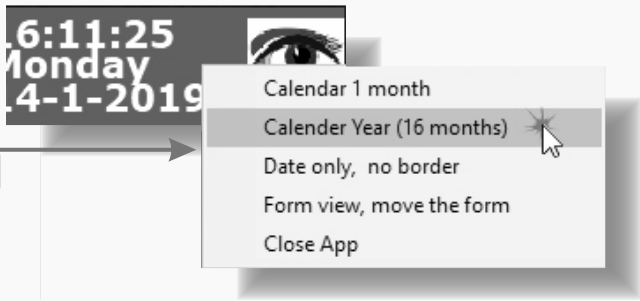


Figure 28: Opens the calendar

The year calendar will open. On the next page more extra's.
I chose to use 16 calendar so you will always have a whole years view.

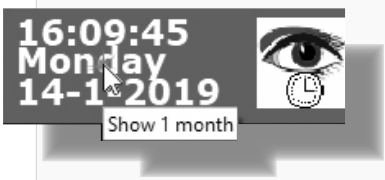


Figure 26: Click on the dayname

Figure 27: Opens the month view

If you click on the name of the day, the 1 month view is opened.

USING THE APP

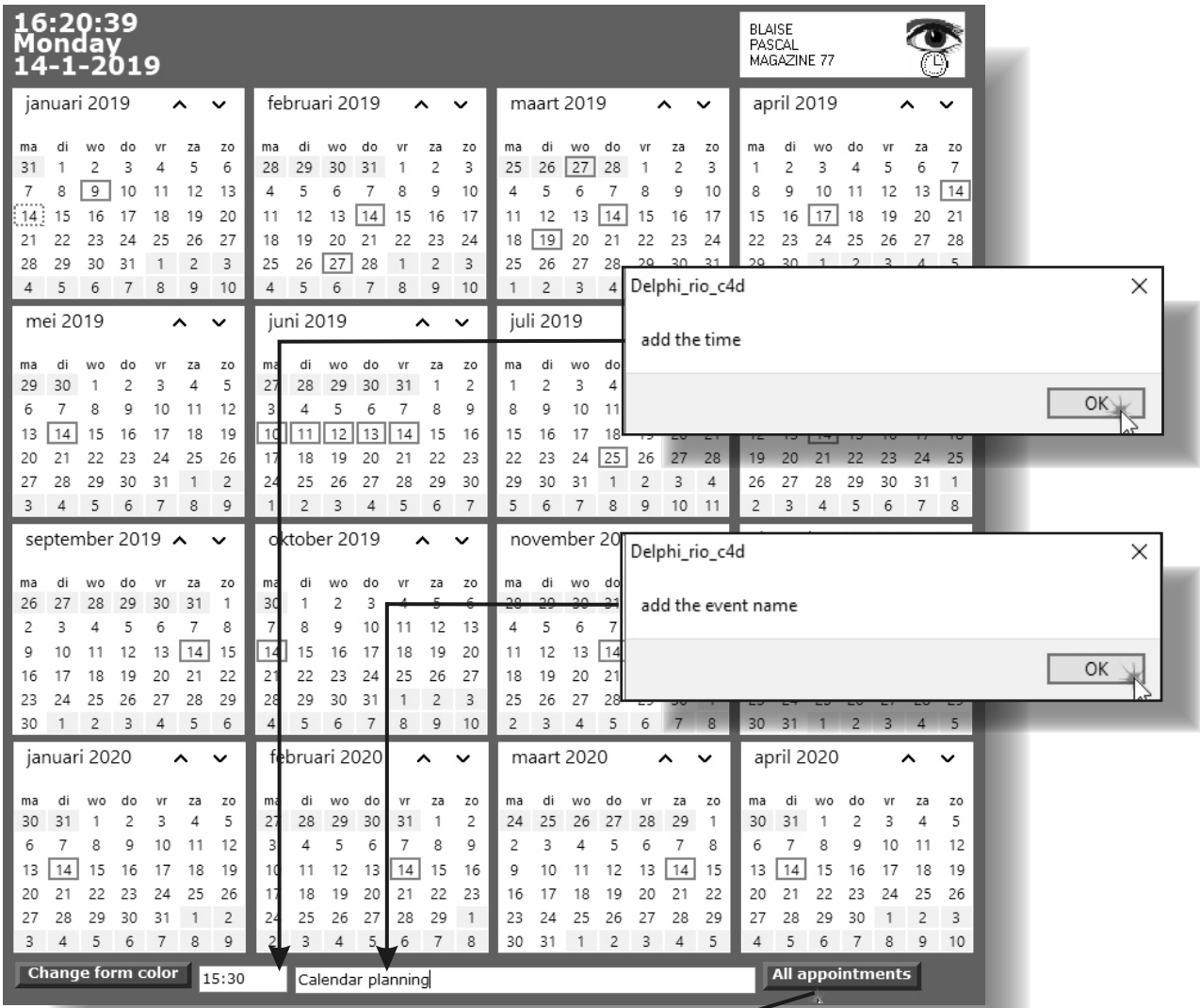
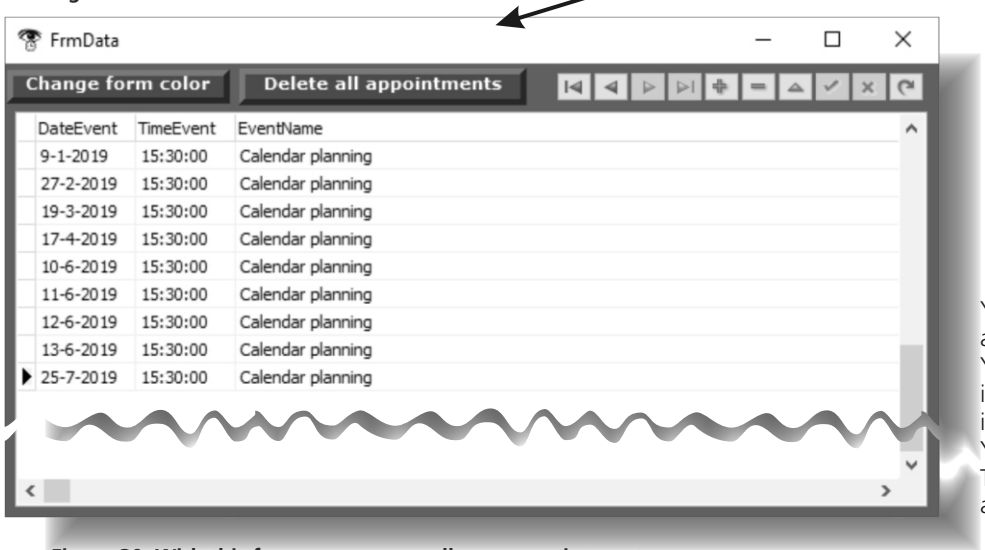


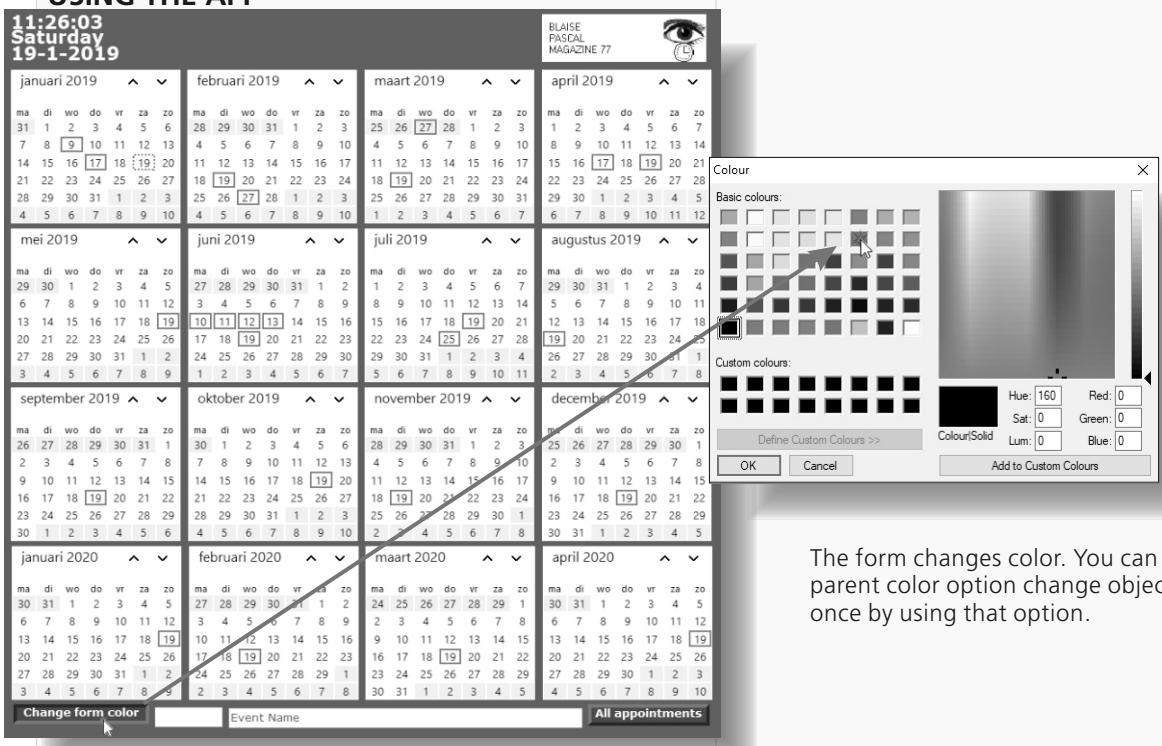
Figure 29: You need to fill the two fields



You can delete all appointments at once, or each one separately. You can make changes directly in the grid. As soon as you close it the Data will be updated. You can also add data here. The form color can be changed as well.

Figure 30: With this form you can see all your appointments

USING THE APP



The form changes color. You can by using the parent color option change object colours at once by using that option.

Figure 31:
Change the form color, - the buttoncolor will not change you can of course make it to

I will add some extras in the next issue and create a Lazarus version.
I will also create a version for the web: a real **WEB-TOP APP** (Progressive Web App) so you will not be anymore dependent on the google calendar. No Spies Allowed.(NSA) where did I hear that before?

This app is complete with an installer of course and available at your downloadpage after log in.

Please add suggestions and write us:
office@blaisepascal.eu

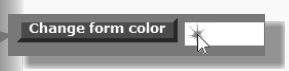
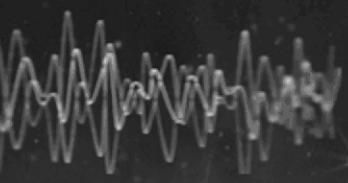


Figure 32: On hover above this part you will see the buttons appear.

MITOV SOFTWARE

WWW.MITOV.COM



Signal Processing

Arduino

Audio

Computer Vision

Video

Communication
Animation
AI
Visual
Instruments
Delphi
Components
Galaxy
Process Control
Mouse Click Away

```
h1 { color: white;
background: orange;
border: 1px solid black;
padding: 0 0 0 0;
font-weight: bold;
}
/* begin: seaside-theme */

body {
background-color:white;
color:black;
font-family:Arial,sans-serif;
margin: 0 4px 0 0;
border: 12px solid;
}
```

CSS

INTRODUCTION

Cascading Style Sheets (CSS) is the **STYLE SHEET LANGUAGE** used for describing the presentation of a document written in HTML. CSS is one of the most important techniques for the World Wide Web, together with HTML and JavaScript, see page 45 in this issue.

The **CSS** is created to enable the separation of presentation and content, including layout, colors, and fonts. Making this separation between design and content, improves the way to handle content. It makes sure you can always put a creative person (designer) or a specialist that needs not to understand anything of the techniques of programming and provide more flexibility and control in the specification of presentation design, enable multiple web pages to share the same formatting by specifying the relevant CSS in a separate .css file. It reduces complexity and avoides repetition of instructions in the content.

This separation from format and content also creates the opportunity to present the same html page in different styles for different rendering methods.

It can do that for:

1. on-screen,
2. in print,
3. by voice (via speech-based browser or screen reader),
4. on Braille-based tactile devices.
5. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name **CASCADING** comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The **CSS specifications** are maintained by the **World Wide Web Consortium (W3C)**.
Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998).
The W3C operates a free CSS validation service for CSS documents, so you can test its outcomes.

There are other markup languages than HTML or HTM(5), they support the use of CSS including XHTML, plain XML, SVG, and XUL.

SYNTAX

CSS has a simple syntax and uses a number of English keywords to specify the names of various style properties.
A style sheet consists of a list of **rules.** Each rule or rule-set consists of one or more **selectors,** and a **declaration block**.

SELECTOR

In CSS, **selectors** declare which part of the markup a style applies to by matching tags and attributes in the markup itself.

Selectors may apply to the following:

- all elements of a specific type, e.g. the second-level headers **`h2`**
- elements specified by attribute, in particular:
  - **`id:`**
    an identifier unique within the document
  - **`class:`**
    an identifier that can annotate multiple elements in a document
- elements depending on how they are placed relative to others in the document tree.

CLASSES AND IDS

`Classes` and **`IDs`** are case-sensitive, start with letters, and can include **alphanumeric characters**, **hyphens** and **underscores.** A class may apply to any number of instances of any elements.
An **`ID`** may only be applied to a single element.

PSEUDO-CLASSES

`Pseudo-classes` are used in CSS selectors to permit formatting based on information that is not contained in the document tree.
One example of a widely used pseudo-class is: **`hover,`** which identifies content only when the user "points to" the visible element.
Usually by holding the mouse cursor over it.
It is appended to a selector as in a **`:hover or #elementid:hover.`**
A **`pseudo-class`** classifies document elements, such as **`:lin`**k or :**`visited`**, whereas a **`pseudo-element`** makes a selection that may consist of partial elements, such as **`::first-line or ::first-letter.`**

Selectors may be combined in many ways to achieve great specificity and flexibility.
Multiple selectors may be joined in a spaced list to specify elements by **`location, element type,`**

For example, **`div .myClass {color: red;}`** applies to all elements of class **`myClass`** that are inside div elements, whereas **`.myClass div {color: red;}`** applies to all div elements that are in elements of **`class myClass`**. The following table provides a summary of selector syntax indicating usage and the version of CSS that introduced it.

Pattern	Matches	First defined in CSS level
`E`	an element of type E1	
`E :link`	an E element is the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited)	1
`E :active`	an E element during certain user actions	1
`E ::first-line`	the first formatted line of an E element	1
`E ::first-letter`	the first formatted letter of an E element	1
`. c`	all elements with class="c"	1
`# myid`	the element with id="myid"	1
`E .warning`	an E element whose class is "warning" (*the document language specifies how class is determined*)	1
`E #myid`	an E element with ID equal to "myid"	1
`E F`	an F element descendant of an E element1*any element	2
`E [foo]`	an E element with a "foo" attribute	2
`E [foo="bar"]`	an E element whose "foo" attribute value is exactly equal to "bar"	2
`E [foo~="bar"]`	an E element whose "foo" attribute value is a list of whitespace-separated values, one of which is exactly equal to "bar"	2
`E [foo|="en"]`	an E element whose "foo" attribute has a hyphen-separated list of values beginning (from the left) with "en"	2
`E :first-child`	an E element, first child of its parent	2
`E :lang(fr)`	an element of type E in language "fr" (*the document language specifies how language is determined*)	2
`E ::before`	generated content before an E element's content	2
`E ::after`	generated content after an E element's content	2
`E > F`	an F element child of an E element	2
`E + F`	an F element immediately preceded by an E element	2
`E [foo^="bar"]`	an E element whose "foo" attribute value begins exactly with the string "bar"	3
`E [foo$="bar"]`	an E element whose "foo" attribute value ends exactly with the string "bar"	3
`E [foo*="bar"]`	an E element whose "foo" attribute value contains the substring "bar"	3
`E :root`	an E element, root of the document	3
`E :nth-child(n)`	an E element, the n-th child of its parent	3
`E :nth-last-child(n)`	an E element, the n-th child of its parent, counting from the last one	3
`E :nth-of-type(n)`	an E element, the n-th sibling of its type	3
`E :nth-last-of-type(n)`	an E element, the n-th sibling of its type, counting from the last one	3
`E :last-child`	an E element, last child of its parent	3
`E :first-of-type`	an E element, first sibling of its type	3
`E :last-of-type`	an E element, last sibling of its type	3
`E :only-child`	an E element, only child of its parent	3
`E :only-of-type`	an E element, only sibling of its type	3
`E :empty`	an E element that has no children (*including text nodes*)	3
`E :target`	an E element being the target of the referring URI	3
`E :enabled`	a user interface element E that is enabled	3
`E :disabled`	a user interface element E that is disabled	3
`E :checked`	a user interface element E that is checked (*for instance a radio-button or checkbox*)	3
`E :not(s)`	an E element that does not match simple selector s	3
`E ~ F`	an F element preceded by an E element	3

DECLARATION BLOCK
A **declaration block** consists of a list of declarations in braces. Each declaration itself consists of a property, a colon **(:)**, and a **value**. If there are multiple declarations in a block, a semi-colon (;) must be inserted to separate each declaration.

Properties are specified in the CSS standard.
Each property has a set of possible values.
Some properties can affect any type of element, and others apply only to particular groups of elements.

Values may be keywords, such as **"center"** or **"inherit",** or **numerical values**, such as 200px (200 pixels), 50vw (50 percent of the viewport width) or 80% (80 percent of the window width).
Color values can be specified with keywords (e.g. **"red")**, hexadecimal values (e.g. **#FF0000**, also abbreviated as **#F00**), RGB values on a **0 to 255** scale (e.g. **rgb(255, 0, 0)**), RGBA values that specify both color and alpha transparency (e.g. **rgba(255, 0, 0, 0.8)**), or **HSL** or **HSLA** values (e.g. **hsl(000, 100%, 50%), hsla(000, 100%, 50%, 80%)**).

USE
Before CSS, nearly all presentational attributes of HTML documents were contained within the HTML markup. All font colors, background styles, element alignments, borders and sizes had to be explicitly described, often repeatedly, within the HTML. CSS lets authors move much of that information to another file, **the style sheet**, resulting in considerably simpler HTML.

For example, **headings (h1 elements), sub-headings (h2), sub-sub-headings (h3)**, etc., are defined structurally using HTML. In print and on the screen, choice of font, size, color and emphasis for these elements is presentational.

Before CSS, document authors who wanted to assign such typographic characteristics to, say, all h2 headings had to repeat HTML presentational markup for each occurrence of that heading type.
This made documents more complex, larger, and more error-prone and difficult to maintain.
CSS allows the separation of presentation from structure.

CSS can define color, font, text alignment, size, borders, spacing, layout and many other typographic characteristics, and can do so independently for on-screen and printed views.

CSS also defines non-visual styles, such as reading speed and emphasis for aural text readers. The W3C has now deprecated the use of all presentational HTML markup.

For example, under pre-CSS HTML, a heading element defined with red text would be written as:

```
<h1><font color="red"> Chapter 1.
</font></h1>
```

Using CSS, the same element can be coded using style properties instead of HTML presentational attributes:

```
<h1 style="color: red;"> Chapter 1.
</h1>
```

The advantages of this may not be immediately clear (*since the second form is actually more verbose*), but the power of CSS becomes more apparent when the style properties are placed in an internal style element or, **even better, an external CSS file**. For example, suppose the document contains the style element:

```
<style>
h1 {color: red;}
</style>
```

All h1 elements in the document will then automatically become red without requiring any explicit code.
If the author later wanted to make h1 elements blue instead, this could be done by changing the style element to:

```
<style>
h1 {color: blue;}
</style>
```

rather than by laboriously going through the document and changing the color for each individual h1 element.

The styles can also be placed in an external CSS file, as described below, and loaded using syntax similar to:

```
<link href="path/to/file.css"
rel="stylesheet" type="text/css">
```

This further decouples the styling from the HTML document, and makes it possible to restyle multiple documents by **simply editing a shared external CSS file.**

SOURCES
CSS information can be provided from various sources. These sources can be the **web browser**, the **user** and the **author.**
The information from the author can be further classified into inline, media type, importance, selector specificity, rule order, inheritance and property definition.
CSS style information can be in a separate document or it can be embedded into an HTML document.
Multiple style sheets can be imported.
Different styles can be applied depending on the output device being used;
for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium.

the style sheet with the highest priority controls the content display.
Declarations not set in the highest priority source are passed on to a source of lower priority, such as the user agent style.
This process is called cascading.
One of the goals of CSS is to allow users greater control over presentation.
Someone who finds red italic headings difficult to read may apply a different style sheet.
Depending on the browser and the web site, a user may choose from various style sheets provided by the designers, or may remove all added styles and view the site using the browser's default styling, or may override just the red italic heading style without altering other attributes.

CSS PRIORITY SCHEME (HIGHEST TO LOWEST)

Priority	CSS source type	Description
1	Importance	The `'!important'` annotation overwrites the previous priority types
2	Inline	A style applied to an HTML element via HTML 'style' attribute
3	Media Type	A property definition applies to all media types, unless a media specific CSS is defined
4	User defined	Most browsers have the accessibility feature: a user defined CSS
5	Selector specificity	A specific contextual selector **`(#heading p)`** overwrites generic definition
6	Rule order	Last rule declaration has a higher priority
7	Parent inheritance	If a property is not specified, it is inherited from a parent element
8	CSS property definition in HTML document	CSS rule or CSS inline style overwrites a default browser value
9	Browser default	The lowest priority: browser default value is determined by W3C initial value specifications

SPECIFICITY
Specificity refers to the relative weights of various rules.
It determines which styles apply to an element when more than one rule could apply.
Based on specification, a simple selector (e.g. **`H1`**) has a specificity of 1,
class selectors have a specificity of **`1,0,`**
and ID selectors a specificity of **`1,0,0.`**

Because the specificity values do not carry over as in the decimal system, commas are used to separate the "digits" *(a CSS rule having 11 elements and 11 classes would have a specificity of* **`11,11`**, *not 121*).
Thus the following rules selectors result in the indicated specificity:

Selectors	Specificity
`H1 {color: white;}`	`0, 0, 0, 1`
`P EM {color: green;}`	`0, 0, 0, 2`
`.grape {color: red;}`	`0, 0, 1, 0`
`P.bright {color: blue;}`	`0, 0, 1, 1`
`P.bright EM.dark {color: yellow;}`	`0, 0, 2, 2`
`#id218 {color: brown;}`	`0, 1, 0, 0`
`style=" "`	`1, 0, 0, 0`

EXAMPLE Consider this HTML fragment:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <style>
    #xyz { color: blue; }
    </style>
  </head>
  <body>
    <p id="xyz" style="color: green;"> To demonstrate specificity </p>
  </body>
</html>
```

In the above example, the declaration in the **style** attribute overrides the one in the **<style>** element because it has a higher specificity, and thus, the paragraph appears green.

Inheritance is a key feature in CSS; it relies on the ancestor-descendant relationship to operate.
Inheritance is the mechanism by which properties are applied not only to a specified element, but also to its descendants.
Inheritance relies on the document tree, which is the hierarchy of XHTML elements in a page based on nesting.
Descendant elements may inherit CSS property values from any ancestor element enclosing them.

In general, descendant elements inherit text-related properties, but they are box-related properties are not inherited.
Properties that can be inherited are
color, font, letter-spacing, line-height, list-style, text-align, text-indent, text-transform, visibility, white-space and word-spacing.

Properties that cannot be inherited are
background, border, display, float and clear, height, and width, margin, min- and max-height and -width, outline, overflow, padding, position, text-decoration, vertical-align and z-index.

Inheritance can be used to avoid declaring certain properties over and over again in a style sheet, allowing for shorter CSS.

Inheritance in CSS is not the same as inheritance in class-based programming languages, where it is possible to define class B as "like class A, but with modifications".
With CSS, it is possible to style an element with "class A, but with modifications".
However, it is not possible to define a CSS class B like that, which could then be used to style multiple elements without having to repeat the modifications.

Example
Given the following style sheet:

```
h1 {
   color: pink;
}
```

Suppose there is an h1 element with an emphasizing element (em) inside:

```
<h1>
   This is to <em>illustrate</em> inheritance
</h1>
```

If no color is assigned to the em element, the emphasized word "illustrate" inherits the color of the parent element, h1. The style sheet h1 has the color pink, hence, the em element is likewise pink.

WHITESPACE
Whitespace between properties and selectors is ignored. This code snippet:

```
body{overflow:hidden;background:#000000;background-image:url(images/bg.gif);
background-repeat:no-repeat;background-position:left top;}
```

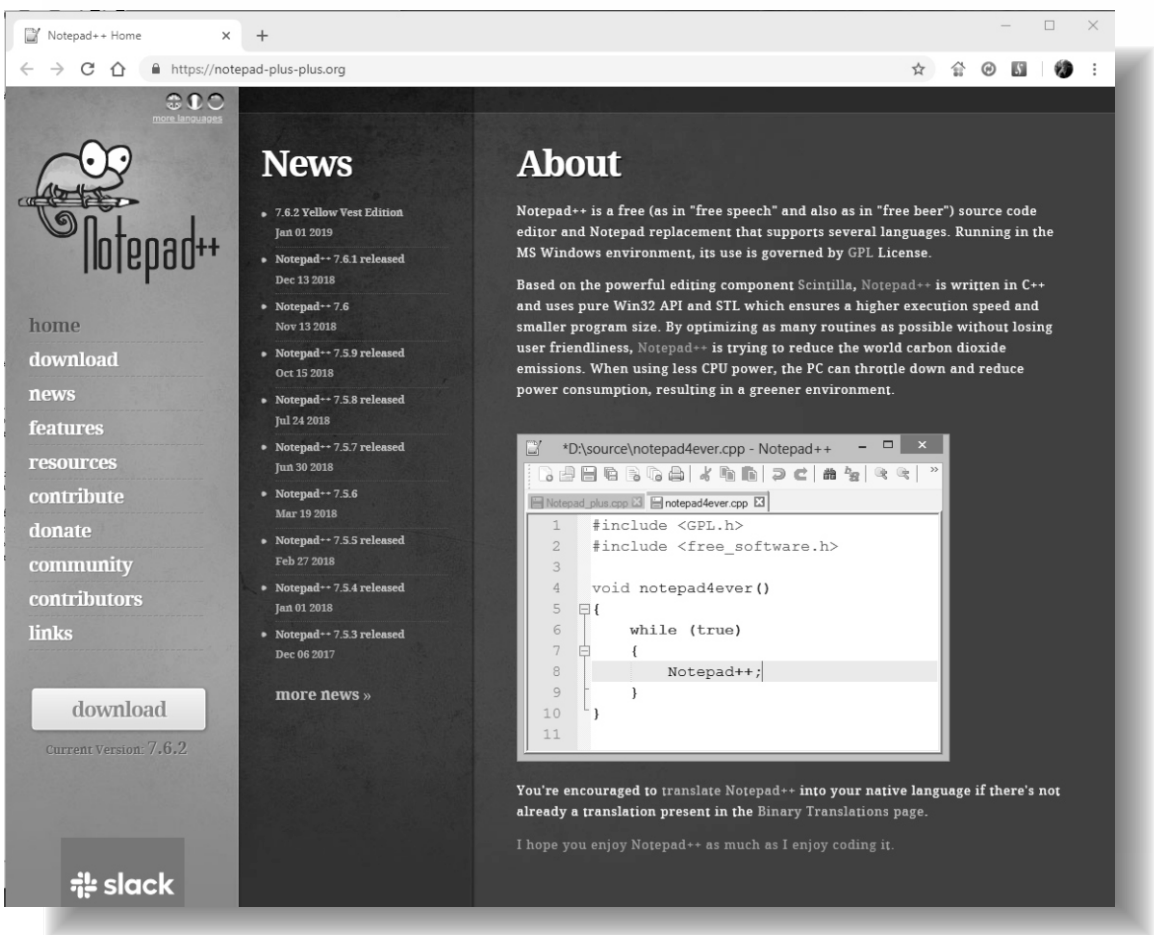
is functionally equivalent to this one:

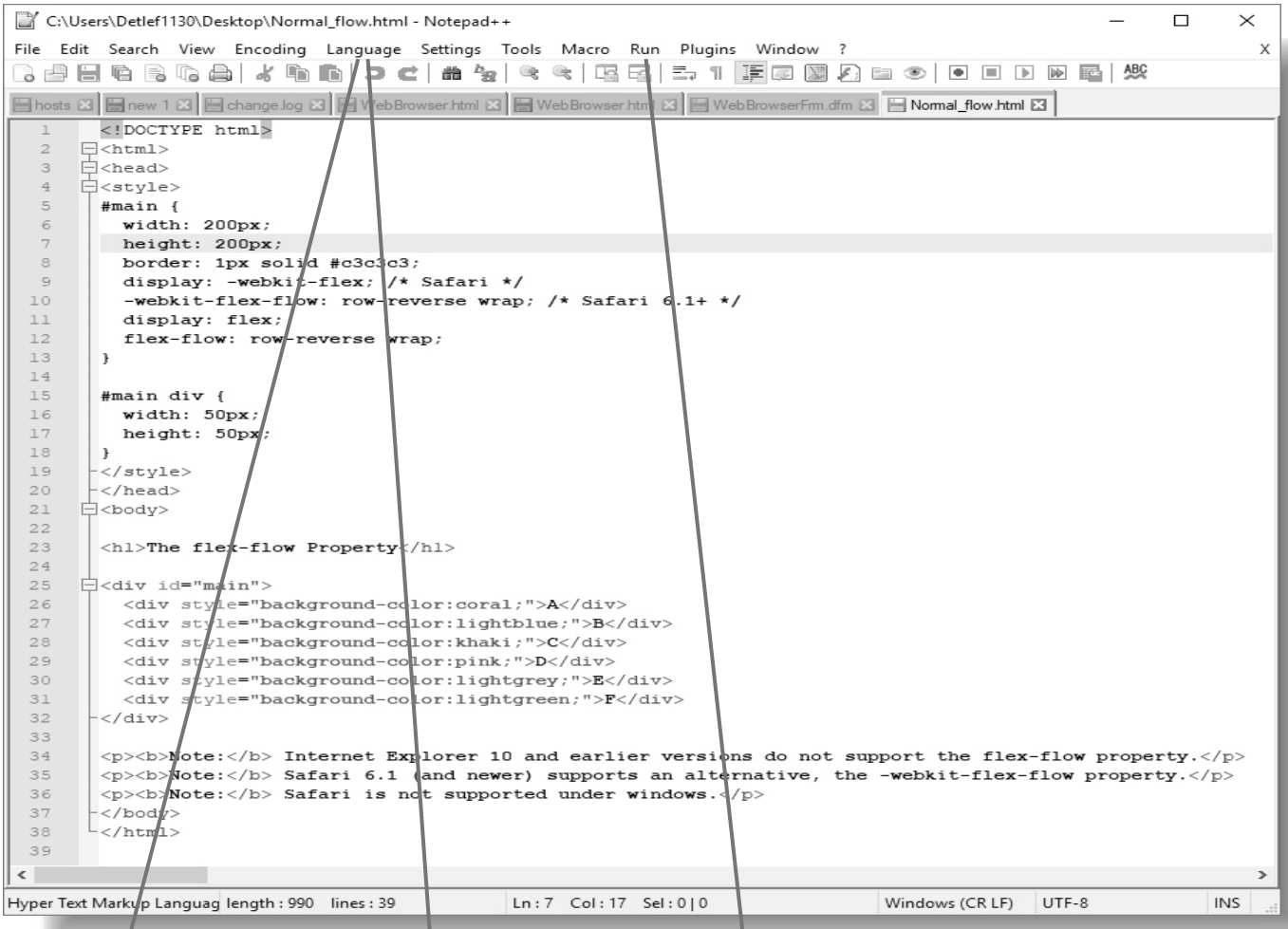
```
body {
    overflow: hidden;
    background: #000000;
    background-image: url(images/bg.gif);
    background-repeat: no-repeat;
    background-position: left top;
}
```

One common way to format CSS for readability is to indent each property and give it its own line. In addition to formatting CSS for readability, you can use shorthand properties to write out the code faster which also gets processed more quickly when being rendered, like so:

```
body {
    overflow: hidden;
    background: #000 url(images/bg.gif) no-repeat left top;
}
```

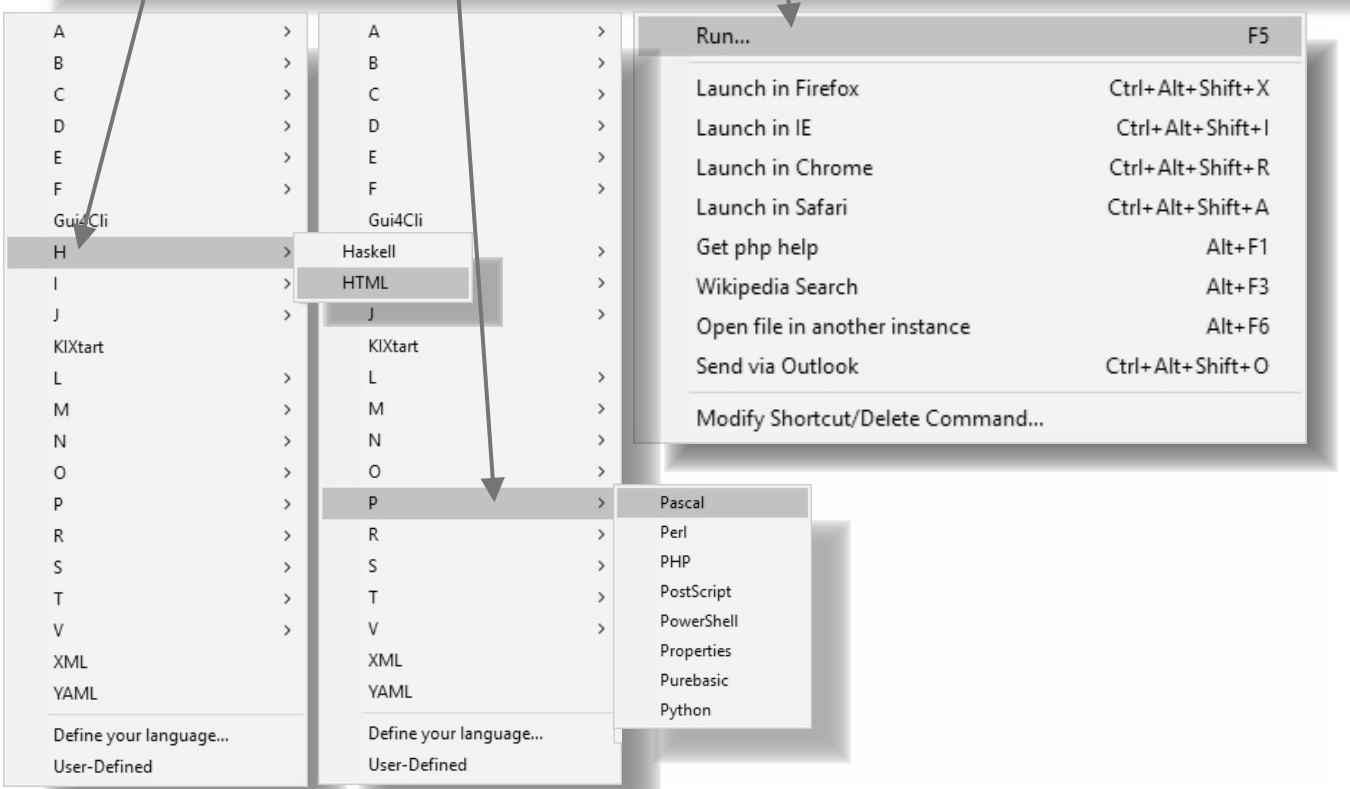
To make things more understandable I have created examples about all the following positioning scheme -desciptions. You can create these very simple your self:
Create an **`.html`** file give it some name - maybe the subject of what it is about.
You need to edit that file, to create your own page:
I would suggest **Notepad ++** you can find it at **`https://notepad-plus-plus.org/.`** It is simply fantastic. It is free, very easy to use and works prompt. In **Notepad++** you edit your file and then save it with your name. So you could take any page from the browser save it and then alter it and save it again. Thats all.




C:\Users\Detlef1130\Desktop\Normal_flow.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
hosts new 1 change.log WebBrowser.html WebBrowser.htm WebBrowserFrm.dfm Normal_flow.html
<!DOCTYPE html>
<html>
<head>
<style>
#main {
width: 200px;
height: 200px;
border: 1px solid #c3c3c3;
display: -webkit-flex; /* Safari */
-webkit-flex-flow: row-reverse wrap; /* Safari 6.1+ */
display: flex;
flex-flow: row-reverse wrap;
}
#main div {
width: 50px;
height: 50px;
}
</style>
</head>
<body>
<h1>The flex-flow Property</h1>

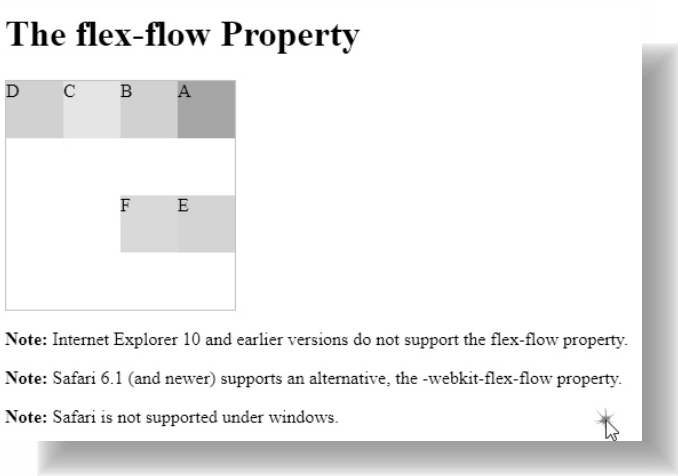
A
B
C
D
E
F

Note: Internet Explorer 10 and earlier versions do not support the flex-flow property.
Note: Safari 6.1 (and newer) supports an alternative, the -webkit-flex-flow property.
Note: Safari is not supported under windows.
</body>
</html>
Hyper Text Markup Languag length : 990 lines : 39
Ln : 7 Col : 17 Sel : 0 | 0
Windows (CR LF) UTF-8 INS


A
B
C
D
E
F
Gui4Cli
H
I
J
KIXtart
L
M
N
O
P
R
S
T
V
XML
YAML
Define your language...
User-Defined
Haskell
HTML
Pascal
Perl
PHP
PostScript
PowerShell
Properties
Purebasic
Python
Run... F5
Launch in Firefox Ctrl+Alt+Shift+X
Launch in IE Ctrl+Alt+Shift+I
Launch in Chrome Ctrl+Alt+Shift+R
Launch in Safari Ctrl+Alt+Shift+A
Get php help Alt+F1
Wikipedia Search Alt+F3
Open file in another instance Alt+F6
Send via Outlook Ctrl+Alt+Shift+O
Modify Shortcut/Delete Command...

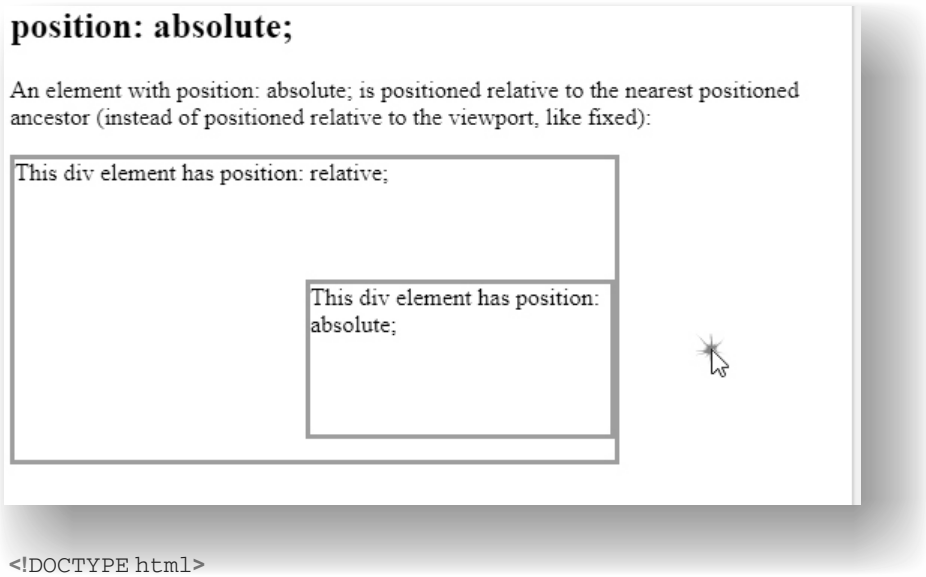
POSITIONING
CSS 2.1 defines three positioning schemes:

Normal flow
Inline items are laid out in the same way as the letters in words in text, one after the other across the available space until there is no more room, then starting a new line below. Block items stack vertically, like paragraphs and like the items in a bulleted list. Normal flow also includes relative positioning of block or inline items, and run-in boxes.




```
<!DOCTYPE html>
<html>
<head>
<style>
#main {
 width: 200px;
  height: 200px;
  border: 1px solid #c3c3c3;
  display: -webkit-flex; /* Safari */
  -webkit-flex-flow: row-reverse wrap; /* Safari 6.1+ */
  display: flex;
  flex-flow: row-reverse wrap;
}

#main div {
  width: 50px;
  height: 50px;
}
</style>
</head>
<body>

<h1>The flex-flow Property</h1>

 <div id="main">
  <div style="background-color:coral;">A</div>
  <div style="background-color:lightblue;">B</div>
  <div style="background-color:khaki;">C</div>
  <div style="background-color:pink;">D</div>
  <div style="background-color:lightgrey;">E</div>
 <div style="background-color:lightgreen;">F</div>
</div>

<p><b>Note:</b> Internet Explorer 10 and earlier versions do not support the flex-flow property.</p>
<p><b>Note:</b> Safari 6.1 (and newer) supports an alternative, the -webkit-flex-flow property.</p>
<p><b>Note:</b> Safari is not supported under windows.</p>
</body>
</html>
```

Absolute
Specifies absolute positioning. The element is positioned in relation to its nearest non-static ancestor.




```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}

div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: absolute;</h2>

<p>An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):</p>

<div class="relative">This div element has position: relative;
 <div class="absolute">This div element has position: absolute;</div>
</div>

</body>
</html>
```

Fixed
The item is absolutely positioned in a fixed position on the screen even as the rest of the document is scrolled

position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: fixed;</h2>

<p>An element with position: fixed; is positioned relative to the viewport, which means
it always stays in the same place even if the page is scrolled:</p>

<div class="fixed">
This div element has position: fixed;
</div>

</body>
</html>
```

FLOAT AND CLEAR
The float property may have one of three values. Absolutely positioned or fixed items cannot be floated!
Other elements normally flow around floated items, unless they are prevented from doing so by their clear property.

In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
 float: right;
}
</style>
</head>
<body>

<p>In this example, the image will float to the right in the paragraph, and the text in the
paragraph will wrap around the image.</p>

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;margin-left:15px;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum
interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl
est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet.
Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus
interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim
ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus
gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor
porta. Cras ac leo purus. Mauris quis diam velit.</p>

</body>
</html>
```

left
The item floats to the left of the line that it would have appeared in; other items may flow around its right side.

right
The item floats to the right of the line that it would have appeared in; other items may flow around its left side.

clear
Forces the element to appear underneath ('clear') floated elements to the left (clear:left), ight (clear:right) or both sides (clear:both).

In this example, the image is taller than the element containing it, and it is floated, so it overflows outside of its container:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

Add a clearfix class with overflow: auto; to the containing element, to fix this problem:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
border: 3px solid #4CAF50;
  padding: 5px;
}

.img1 {
  float: right;
}

.clearfix {
  overflow: auto;
}

.img2 {
  float: right;
}
</style>
</head>
<body>

<p>In this example, the image is taller than the element containing it, and it is floated, so it overflows outside of its container:</p>

<div>
 <img class="img1" src="pineapple.jpg" alt="Pineapple" width="170" height="170">
 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...
</div>

<p style="clear:right">Add a clearfix class with overflow: auto; to the containing element, to fix this problem:</p>

<div class="clearfix">
 <img class="img2" src="pineapple.jpg" alt="Pineapple" width="170" height="170">
 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...
</div>

</body>
</html>
```

WA

INTRODUCTION

WebAssembly (often shortened to Wasm) is a standard that defines a binary format and a corresponding assembly-like text format for executables used by web pages.

The purpose of Wasm is to enable the JavaScript engine of a web browser to execute page scripts nearly as fast as native machine code. But this is not a full replacement for JavaScript; rather, Wasm is only intended for performance-critical portions of page scripts. Wasm code runs in the same sandbox as regular script code.

The World Wide Web Consortium (W3C) maintains the standard with contributions from Mozilla, Microsoft, Google, and Apple. `https://webassembly.org/`

WEBASSEMBLY code is intended to be run on a portable abstract structured stack machine, which is designed to be faster to parse than JavaScript, as well as faster to execute, and to enable very compact code representation.

HISTORY

Vendor-specific precursor technologies are Google **Native Client (NaCl)** and asm.js. The initial implementation of **WebAssembly** support in browsers was based on the feature set of **asm.js.**

It was first announced on 17 June **2015** and on 15 March 2016 was demonstrated executing **Unity's Angry Bots in Firefox**, **Chromium, Google Chrome, and Microsoft Edge**.

In March 2017, the design of the minimum viable product was declared to be finished and the preview phase ended. In late September 2017, Safari 11 was released with support.

As of December 2018, the website Can I use `https://caniuse.com/`. 54% global web browser support for WebAssembly (*and indirectly, through polyfill, others are supported*). In February 2018 the **WebAssembly Working Group** published three public working drafts for the **Core Specification, JavaScript Interface, and Web API.**

SUPPORT

In November 2017, Mozilla declared support "in all major browsers", after WebAssembly was enabled by default in Edge 16. For backward compatibility, **WASM** can be compiled into `asm.js` by a JavaScript polyfill and **executed on incompatible browsers this way.** **Emscripten** can compile to **WASM**using **LLVM** in the backend.

Its initial aim is to support compilation from **C** and **C++,** though support for other source languages such as **Rust** and **.NET** languages is also emerging.

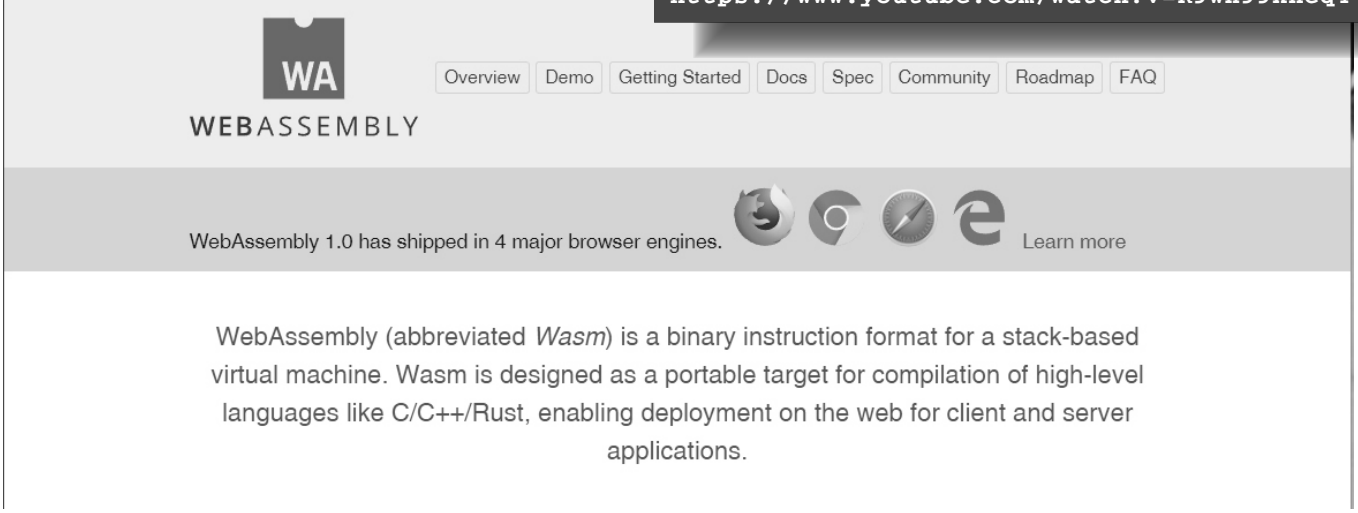
PAS2JS will in future also work on it.

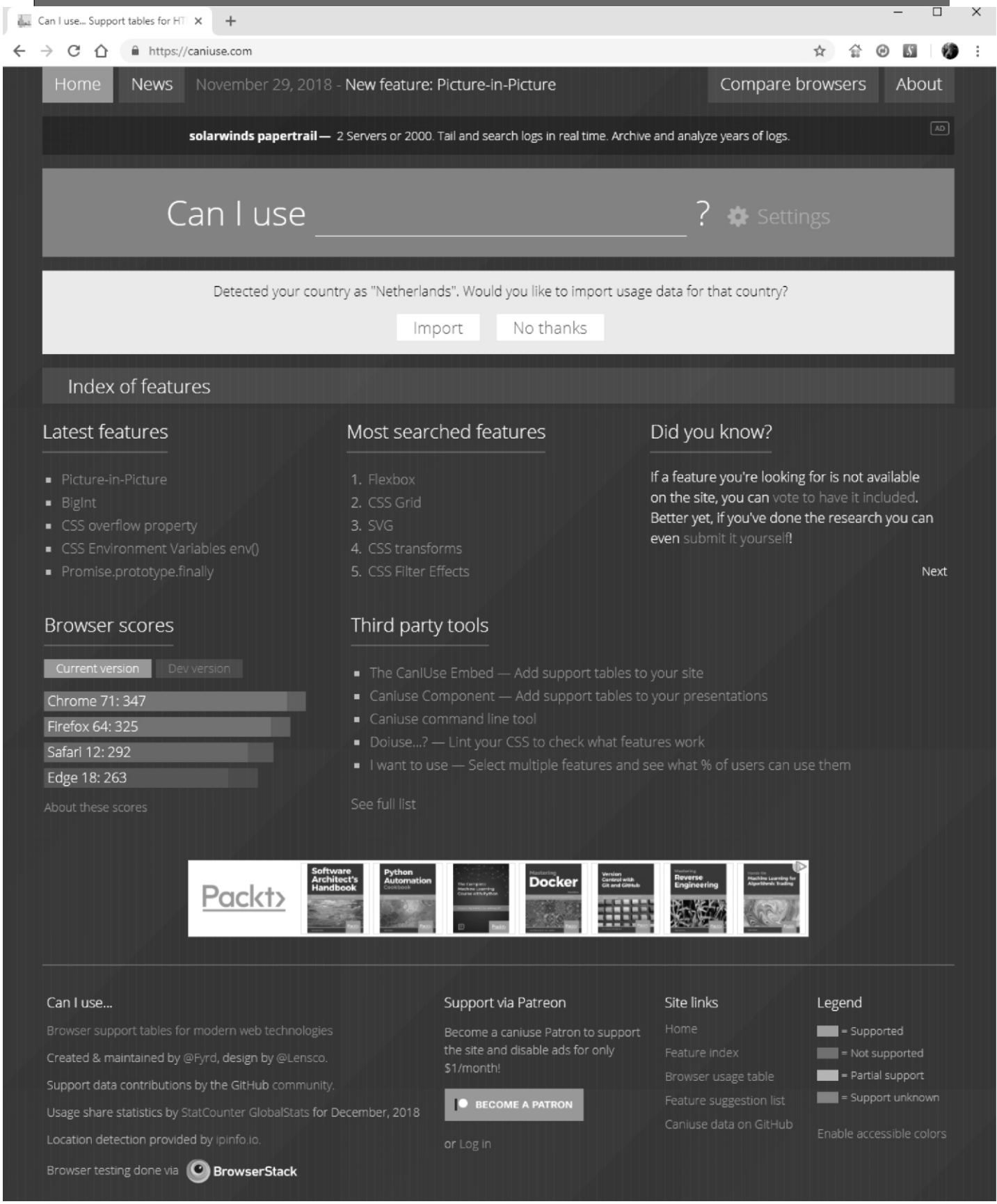
After the **minimum viable product (MVP)** release, there are plans to support garbage collection which would make **WebAssembly** a compilation target for garbage-collected programming languages like **Java, C#** (supported via **Blazor**) and **Go.**

```
(module
  (import "math" "exp" (func $exp (param f64) (result f64)))
  (func (export "doubleExp") (param $0 f64) (result f64)
    (f64.mul
      (call $exp
        (get_local $0)
      )
      (f64.const 2)
    )
  )
)
```

EXTERNAL LINKS

W3C Community Group:
`https://www.w3.org/community/webassembly/`
WebAssembly Design:
`https://github.com/WebAssembly/design`
"WebAssembly". MDN Web Docs. – with info on browser compatibility and specifications
(WebAssembly JavaScript API):
`https://developer.mozilla.org/en-US/docs/WebAssembly`
WebAssembly: What and What Next? (youtube):
`https://www.youtube.com/watch?v=R9wn99Xheq4`




Can I use... Support tables for HT
https://caniuse.com
Home
News
November 29, 2018 - New feature: Picture-in-Picture
Compare browsers
About
solarwinds papertrail — 2 Servers or 2000. Tail and search logs in real time. Archive and analyze years of logs.
AD
Can I use ? Settings
Detected your country as "Netherlands". Would you like to import usage data for that country?
Import
No thanks
Index of features
Latest features
Picture-in-Picture
BigInt
CSS overflow property
CSS Environment Variables env()
Promise.prototype.finally
Most searched features
1. Flexbox
2. CSS Grid
3. SVG
4. CSS transforms
5. CSS Filter Effects
Did you know?
If a feature you're looking for is not available on the site, you can vote to have it included. Better yet, if you've done the research you can even submit it yourself!
Next
Browser scores
Current version
Dev version
Chrome 71: 347
Firefox 64: 325
Safari 12: 292
Edge 18: 263
About these scores
Third party tools
The CanIUse Embed — Add support tables to your site
Caniuse Component — Add support tables to your presentations
Caniuse command line tool
Doiuse...? — Lint your CSS to check what features work
I want to use — Select multiple features and see what % of users can use them
See full list
Packt
Software Architect's Handbook
Python Automation Cookbook
Mastering Docker
Version Control with Git and GitHub
Reverse Engineering
Machine Learning for Algorithmic Trading
Can I use...
Browser support tables for modern web technologies
Created & maintained by @Fyrd, design by @Lensco.
Support data contributions by the GitHub community.
Usage share statistics by StatCounter GlobalStats for December, 2018
Location detection provided by ipinfo.io.
Browser testing done via BrowserStack
Support via Patreon
Become a caniuse Patron to support the site and disable ads for only $1/month!
BECOME A PATRON
or Log in
Site links
Home
Feature index
Browser usage table
Feature suggestion list
Caniuse data on GitHub
Legend
= Supported
= Not supported
= Partial support
= Support unknown
Enable accessible colors

JS

JavaScript, often abbreviated as JS, is a high-level, interpreted programming language that conforms to the ECMAScript specification. It is a language that is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language*1, **JavaScript** supports event-driven, functional, and imperative (*including object-oriented and prototype-based*) programming styles.
It has APIs for working with **text, arrays, dates, regular expressions, and the DOM**, but the language itself does not include any **I/O** (*input output*), such as **networking, storage, or graphics facilities,** relying for these upon the host environment in which it is embedded.

*(*1: Programming paradigms are a way to classify programming languages based on their features. Languages can be classified into multiple paradigms.)*

INITIALLY only implemented client-side in web browsers, **JavaScript engines** are now embedded in many other types of host software, including server-side in **web servers and databases,** and in **non-web programs such as word processors** and **PDF** software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

The terms **Vanilla JavaScript** and **Vanilla JS** refer to **JavaScript** not extended by any frameworks or additional libraries.
Scripts written in Vanilla JS are plain JavaScript code.

Although there are similarities between **JavaScript** and **Java,** including language name, syntax, and respective standard libraries, **the two languages are distinct and differ greatly in design;**
JavaScript was influenced by programming languages such as **Self** and **Scheme.**

HISTORY

BEGINNINGS AT NETSCAPE

In **1993,** the **National Center for Supercomputing Applications (NCSA)**, a unit of the University of Illinois at Urbana-Champaign, released NCSA Mosaic, the first popular graphical Web browser, which played an important part in expanding the growth of the nascent **World Wide Web** beyond the **NeXTSTEP** niche where the WorldWideWeb had formed three years earlier.

In 1994, a company called **Mosaic Communications** was founded in Mountain View, California and employed many of the original NCSA Mosaic authors to create **Mosaic Netscape.**

However, it intentionally shared no code with NCSA Mosaic. The internal codename for the company's browser was Mozilla, which stood for "Mosaic killer", as the company's goal was to displace NCSA Mosaic as the world's number one web browser.
The first version of the Web browser, Mosaic Netscape 0.9, was released in late 1994.
Within four months it had already taken three-quarters of the browser market and became the main web browser for the 1990s.
To avoid trademark ownership problems with the **NCSA,** the browser was subsequently renamed **Netscape Navigator** in the same year, and the company took the name Netscape Communications.

Netscape

Netscape Communications realized that the Web needed to become more dynamic. Marc Andreessen, the founder of the company believed that HTML needed a "glue language" that was easy to use by Web designers and part-time programmers to assemble components such as images and plugins, where the code could be written directly in the Web page markup.

In 1995, **Netscape Communications** recruited **Brendan Eich** with the goal of embedding the **Scheme** programming language into its **Netscape Navigator.**

Before he could get started, **Netscape Communications** collaborated with **Sun Microsystems** to include in **Netscape Navigator Sun's** more static programming language **Java,** in order to compete with **Microsoft** for user adoption of Web technologies and platforms.

Netscape Communications then decided that the scripting language they wanted to create would complement **Java** and should have a similar syntax, which excluded adopting other languages such as **Perl, Python, TCL, or Scheme**.

To defend the idea of **JavaScript** against competing proposals, the company needed a prototype.
Eich wrote one in 10 days, in May 1995.

Although it was developed under the name **Mocha,** the language was officially called **LiveScript** when it first shipped in beta releases of **Netscape Navigator 2.0** in September 1995, but it was renamed **JavaScript** when it was deployed in the **Netscape Navigator 2.0** beta 3 in December.

To defend the idea of JavaScript against competing proposals, the company needed a prototype. Eich wrote one in 10 days, in May 1995.

The final choice of name caused confusion, giving the impression that the language was a spin-off of the **Java** programming language, and the choice has been characterized as a marketing ploy by **Netscape** to give **JavaScript** the cachet of what was then the hot new Web programming language.

There is a common **misconception** that **JavaScript** was influenced by an earlier Web page **scripting language developed by Nombas named Cmm** (*not to be confused with the later C-- created in 1997*).

Brendan Eich, however, had never heard of **Cmm** before he created **LiveScript**.
Nombas did pitch their **embedded Web page scripting** to Netscape, though Web page scripting was not a new concept, as shown by the **ViolaWWW Web browser. Nombas** later switched to offering **JavaScript** instead of **Cmm** in their **ScriptEase** product and was part of the **TC39** group that standardized **ECMAScript.**

SERVER-SIDE JAVASCRIPT
In December 1995, soon after releasing **JavaScript** for browsers, **Netscape** introduced an implementation of the language for server-side scripting with **Netscape Enterprise Server.**

Since 1996, the **IIS web-server** has supported Microsoft's implementation of server-side **Javascript** -- **JScript—in ASP and .NET pages.**
Since the mid-2000s, additional server-side JavaScript implementations have been introduced, such as Node.js in 2009.

ADOPTION BY MICROSOFT
Microsoft script technologies including **VBScript** and **JScript** were released in 1996.
JScript, a reverse-engineered implementation of Netscape's JavaScript, was part of Internet Explorer 3.

JScript was also available for server-side scripting in Internet Information Server. **Internet Explorer 3 also included Microsoft's first support for CSS and various extensions to HTML,** but in each case the implementation was **noticeably different** from that found in **Netscape Navigator** at the time.
These differences made it difficult for designers and programmers to make a single website work well in both browsers, leading to the use of **"best viewed in Netscape"** and **"best viewed in Internet Explorer"** logos that characterized these early years of the browser wars. **JavaScript** began to acquire a reputation for being one of the roadblocks to a **cross-platform and standards-driven Web.**
Some developers took on the difficult task of trying to make their sites work in both major browsers, but many could not afford the time.
With the release of Internet **Explorer 4, Microsoft** introduced the concept of **Dynamic HTML**, but the differences in language implementations and the different and proprietary **Document Object Models** remained and were obstacles to widespread take-up of **JavaScript** on the Web.

STANDARDIZATION
In November 1996, **Netscape** submitted **JavaScript** to **ECMA** International to carve out a standard specification, which other browser vendors could then implement based on the work done at **Netscape.**
This led to the official release of the language specification **ECMAScript** published in the first edition of the **ECMA-262** standard in June 1997,
with **JavaScript** being the most well known of the implementations. **ActionScript** and **JScript** were other well-known implementations of **ECMAScript.**

The release of **ECMAScript** 2 in June 1998 continued the standards process cycle, conforming some modifications to the **ISO/IEC 16262** international standard. **ECMAScript 3** was released in December 1999 and is the modern-day baseline for JavaScript.
The original **ECMAScript 4** work led by **Waldemar Horwat** (then at **Netscape,** now at **Google**) started in 2000.
Microsoft initially participated and implemented some proposals in their **JScript .NET** language.

Over time it was clear that **Microsoft** had no intention of cooperating or implementing proper **JavaScript** in **Internet Explorer**, even though they had no competing proposal and they had a partial (and diverged at this point) implementation on the .**NET** server side.
So by 2003, the original **ECMAScript 4** work was mothballed.

The next major event was in 2005, with two major happenings in JavaScript's history. First, **Brendan Eich** and **Mozilla** rejoined **Ecma International** as a not-for-profit member and work started on **ECMAScript** for **XML** (E4X), the ECMA-357 standard, which came from ex-Microsoft employees at **BEA Systems** (originally acquired as Crossgain). This led to working jointly with **Macromedia** (later acquired by **Adobe Systems**), who were implementing E4X in ActionScript 3 (ActionScript 3 was a fork of original ECMAScript 4).
So, along with **Macromedia,** work restarted on **ECMAScript 4** with the goal of standardizing what was in **ActionScript 3**. To this end, **Adobe Systems** released the **ActionScript Virtual Machine 2**, code named **Tamarin,** as an open source project.
But Tamarin and **ActionScript 3** were too different from web **JavaScript** to converge, as was realized by the parties in 2007 and 2008.

Alas, there was still turmoil between the various players; **Douglas Crockford** - then at **Yahoo!** - joined forces with **Microsoft in 2007 to oppose ECMAScript 4**, which led to the **ECMAScript 3.1 effort.** The development of ECMAScript 4 was never completed, but that work influenced subsequent versions.

While all of this was happening, the open source and developer communities set to work to revolutionize what could be done with **JavaScript.**

This community effort was sparked in 2005 when Jesse James Garrett released a white paper in which he coined the term Ajax, and described a set of technologies, of which JavaScript was the backbone, used to create web applications where data can be loaded in the background, avoiding the need for full page reloads and leading to more dynamic applications.

This resulted in a renaissance period of **JavaScript** usage spearheaded by open source libraries and the communities that formed around them, with libraries such as **Prototype, jQuery, Dojo Toolkit, MooTools**, and others being released.

In July 2008, the disparate parties on either side came together in Oslo. This led to the eventual agreement in early 2009 to rename ECMAScript 3.1 to ECMAScript 5 and drive the language forward using an agenda that is known as **Harmony.**
ECMAScript 5 was finally released in December 2009.

In June 2011, ECMAScript 5.1 was released to fully align with the third edition of the ISO/IEC 16262 international standard. ECMAScript 2015 was released in June 2015.
ECMAScript 2016 was released in June 2016.
The current version is ECMASCRIPT 2017, released in June 2017.

LATER DEVELOPMENTS
JavaScript has become one of the most popular programming languages on the Web.
However, many professional programmers initially denigrated the language due to the perceived target audience of Web authors and other such "amateurs".

The advent of Ajax returned JavaScript to the spotlight and brought more professional programming attention.

The advent of **Ajax** (See the separate explanation at page 48) returned **JavaScript** to the spotlight and brought more professional programming attention. The result was a proliferation of comprehensive frameworks and libraries, improved **JavaScript** programming practices, and increased usage of **JavaScript** outside Web browsers, as seen by the proliferation of **Server-side JavaScript** platforms.

In January 2009, the **CommonJS** project was founded with the goal of specifying a common standard library mainly for JavaScript development outside the browser.

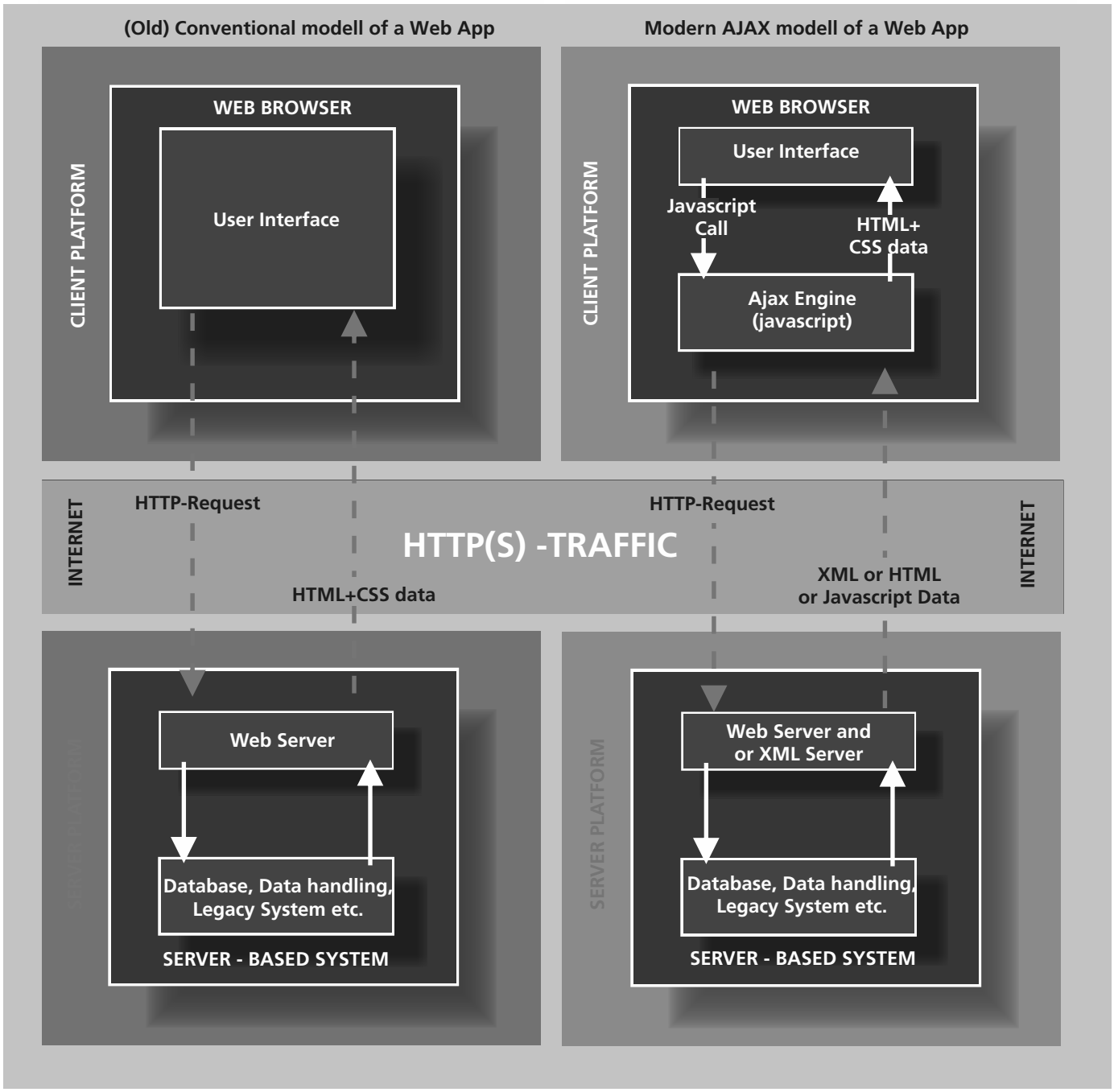
With the rise of **single-page applications** and **JavaScript-heavy** sites, it is increasingly being used as a compile target for **source-to-source compilers** from both **dynamic languages** and **static languages.**

TRADEMARK
"JavaScript" is a trademark of **Oracle** Corporation in the United States. It is used under license for technology invented and implemented by **Netscape Communications** and current entities such as the **Mozilla Foundation.**

AJAX short for "**A**synchronous **J**avaScript **A**nd **X**ML" is a set of Web development techniques using many web technologies on the client side to create asynchronous Web applications.
With **AJAX,** web applications can send and retrieve data from a server asynchronously (*in the background*) without interfering with the display and behavior of the existing page.

By decoupling the data interchange layer from the **presentation layer**, **AJAX** allows web pages, and by extension web applications, to **change content dynamically** without the need to reload the entire page.

In practice, modern implementations commonly utilize **JSON instead of XML** due to the advantages of JSON being native to **JavaScript.**
Ajax is not a single technology, but rather a group of technologies. **HTML** and **CSS** can be used in combination to mark up and style information.
The webpage can then be modified by **JavaScript** to **dynamically display - and allow the user to interact with - the new information**. The built-in `XMLHttpRequest` object within JavaScript is commonly used to execute **Ajax** on webpages allowing websites to **load content onto the screen without refreshing the page.** AJAX is not a new technology, or different language, just existing technologies used in new ways.



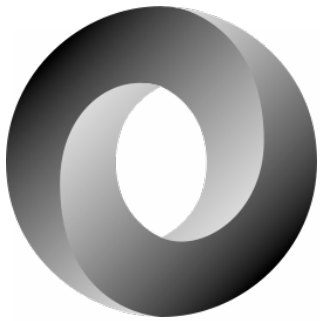
TECHNOLOGIES
The term AJAX has come to represent a broad group of Web technologies that can be used to implement a Web application that communicates with a server in the background, **without interfering with the current state of the page.**
In the article that contains the term Ajax, Jesse James Garrett explained these technologies are incorporated:

Technologies incorporated:
1. HTML (or XHTML) and CSS for presentation.
2. The Document Object Model (DOM) for dynamic display of and interaction with data.
3. JSON or XML for the interchange of data, and XSLT for its manipulation.
4. The **`XMLHttpRequest`** object for *asynchronous communication.*
5. JavaScript to bring these technologies together.

Since then, however, there have been a number of developments in the technologies used in an AJAX application, and in the definition of the term AJAX itself.
XML is no longer required for data interchange and, therefore, XSLT is no longer required for the manipulation of data.
JavaScript Object Notation (JSON) is often used as an alternative format for data interchange, although other formats such as preformatted HTML or plain text can also be used.
A variety of popular JavaScript libraries, including JQuery, include abstractions to assist in executing Ajax requests.

Asynchronous HTML and **HTTP (AHAH)** involves using **`XMLHTTPRequest`** to retrieve (X)HTML fragments, which are then inserted directly into the Web page.

For **Delphi** I can mention that **TMS** has created its software on this great new Technique. It is availabale for **Delphi** as well as **Lazarus.**
Delphi has also an other component group: **Intraweb,** that Suite is not available for the modern Webpages, but uses the Conventional approach - which is as a model demonstrated on Page 48 (last page see the diagram).
TMS uses the **Single Page Approach** which is explained on page 51.



JavaScript Object Notation (JSON) is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication, including as a replacement for XML in some AJAX-style systems.

JSON is a language-independent data format. It was derived from JavaScript, but as of 2017 many programming languages include code to generate and parse JSON-format data. The official Internet media type for JSON is application/json. JSON filenames use the extension .json.

Douglas Crockford originally specified the **JSON** format in the early 2000s; two competing standards, RFC 8259 and ECMA-404, defined it in 2017. The **ECMA** standard describes only the allowed syntax, whereas the RFC covers some security and interoperability considerations.

A restricted profile of JSON, known as I-JSON (*short for* "Internet JSON"), seeks to overcome some of the interoperability problems with JSON. It is defined in RFC 7493.

The following example shows a possible JSON representation describing a person.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    „postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      „number": "212 555-1234"
    },
    {
      "type": "office",
      „number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

HISTORY

Douglas Crockford at the Yahoo Building. (2007)

JSON grew out of a need for stateless, real-time server-to-browser communication protocol without using browser plugins such as Flash or Java applets, the dominant methods used in the early 2000s.

Douglas Crockford first specified and popularized the JSON format.
The acronym originated at **State Software**, a company co-founded by **Crockford** and others in March 2001.
The co-founders agreed to build a system that used **standard browser capabilities** and **provided an abstraction layer for Web developers to create stateful Web applications** that had a persistent duplex connection to a Web server by holding two HTTP connections open and recycling them before standard browser time-outs if no further data were exchanged.

The co-founders had a round-table discussion and voted whether to call the data format **JSML** or **JSON,** as well as under what license type to make it available.
Crockford, being inspired by the words of then President Bush, should also be credited with coming up with the "evil-doers" JSON license ("The Software shall be used for Good, not Evil.") in order to open-source the JSON libraries, but force (troll) corporate lawyers, or those who are overly pedantic, to seek to pay for a license from State.
Chip Morningstar developed the idea for the **State Application Framework at State Software**. On the other hand, this clause led to license compatibility problems of the **JSON** license with other open-source licenses.

A precursor to the **JSON** libraries was used in a children's digital asset trading game project named Cartoon Orbit at **Communities.com** (*the State co-founders had all worked at this company previously*) for **Cartoon Network**, which used a browser side plug-in with a proprietary messaging format to manipulate **DHTML** elements (*this system is also owned by 3DO*).

Upon discovery of early **AJAX** capabilities, **digiGroups, Noosh,** and others used frames to pass information into the user browsers' visual field without refreshing a Web application's visual context, realizing real-time rich Web applications using only the standard HTTP, HTML and JavaScript capabilities of Netscape 4.0.5+ and IE 5+.

Crockford then found that **JavaScript** could be used as an **object-based messaging forma**t for such a system.

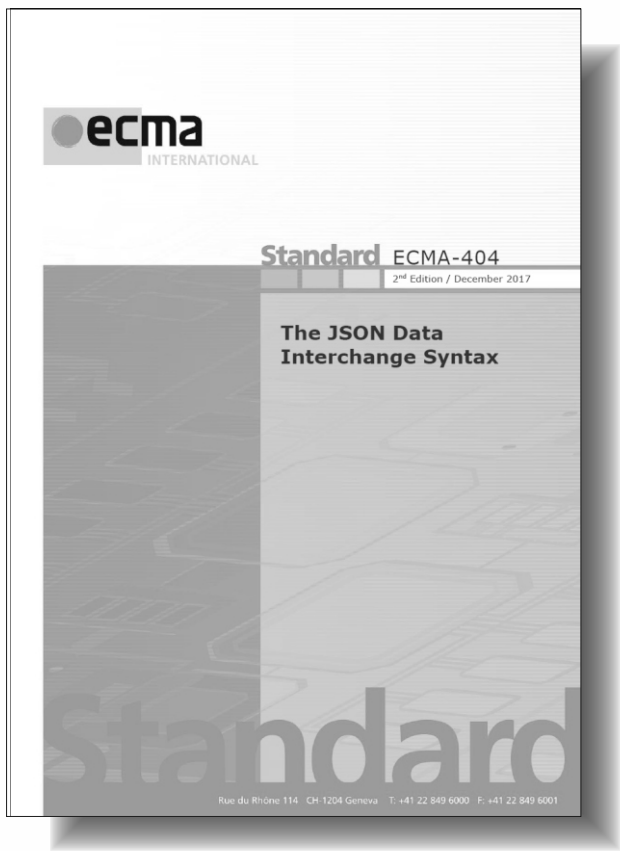
The system was sold to **Sun Microsystems, Amazon.com** and **EDS.** The **JSON.org** website was launched in 2002. In December 2005, **Yahoo!** began offering some of its Web services in **JSON.**

JSON was originally intended to be a subset of the **JavaScript** scripting language and is commonly used with **Javascript,** but it is a language-independent data format.
Code for parsing and generating **JSON data** is readily available in many programming languages. **JSON's** website lists **JSON** libraries by language.

Though **JSON** was originally advertised and believed to be a strict subset of JavaScript and ECMAScript, **it inadvertently allows some unescaped characters in strings that are illegal in JavaScript and ECMAScript string literals**.

The official Website: **`http://www.json.org/`**

You can download at your downloadpage a little book : **The JSON Data Interchange SynTax**



SINGLE-PAGE APPLICATION
A single-page application **(SPA)** is a web application or web site that interacts with the user by **dynamically rewriting the current page rather than loading entire new pages from a server.**
This approach avoids interruption of the user experience between successive pages, **making the application behave more like a desktop application**.

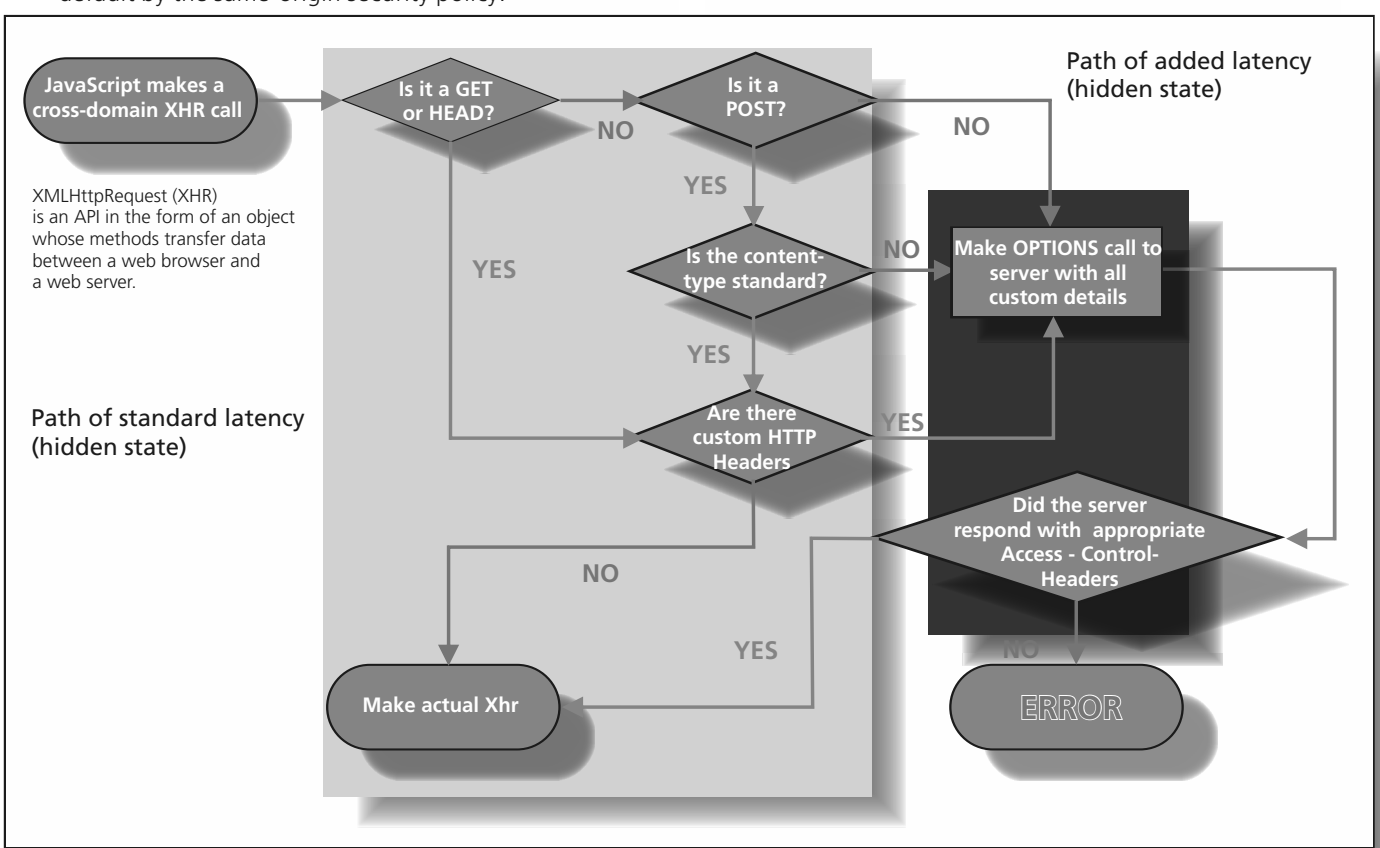
In an **SPA,** either all necessary code – **HTML, JavaScript, and CSS** – is retrieved with a single page load, or the appropriate resources are dynamically loaded and added to the page as necessary, usually in response to user actions.
The page does **not reload** at any point in the process, **nor does control transfer to another page**, although the location hash or the **HTML5 History API** can be used to provide the **perception and navigability** of separate logical pages in the application.
Interaction with the single page application often involves dynamic communication with the web server behind the scenes.

CROSS-ORIGIN RESOURCE SHARING (CORS)
Cross-origin resource sharing **(CORS)** is a mechanism that allows **restricted resources** on a web page to be **requested from another domain** outside the domain from which the first resource was served. A web page may **freely embed cross-origin images, stylesheets, scripts, iframes, and videos.** Certain "cross-domain" requests, **notably AJAX requests**, are forbidden by default by the same-origin security policy.

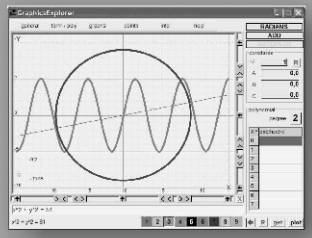
CORS defines a way in which a browser and server can interact to determine whether or not it is safe to allow the **cross-origin request.** It allows for more freedom and functionality than purely same-origin requests, but is **more secure than simply allowing all cross-origin requests**. The specification for CORS was originally published as a W3C Recommendation but that document is obsolete. The current actively-maintained specification that defines CORS is WHATWG's Fetch Living Standard.

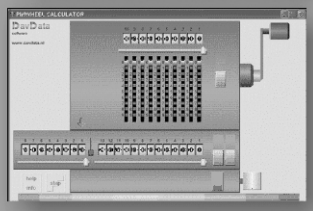
The working of CORS
The **CORS standard** describes new HTTP headers which provide browsers a way to request remote URLs only when they have permission.
Although some validation and authorization can be performed by the server, it is generally the browser's responsibility to support these headers and honor the restrictions they impose.

For **AJAX** and **HTTP** request methods that can modify data (*usually HTTP methods other than GET, or for POST usage with certain MIME types*),
the specification mandates that browsers "preflight" the request, soliciting supported methods from the server with an HTTP OPTIONS request method, and then, upon "approval" from the server, sending the actual request with the actual HTTP request method.
Servers can also notify clients whether "credentials" (*including Cookies and HTTP Authentication data*) should be sent with requests.



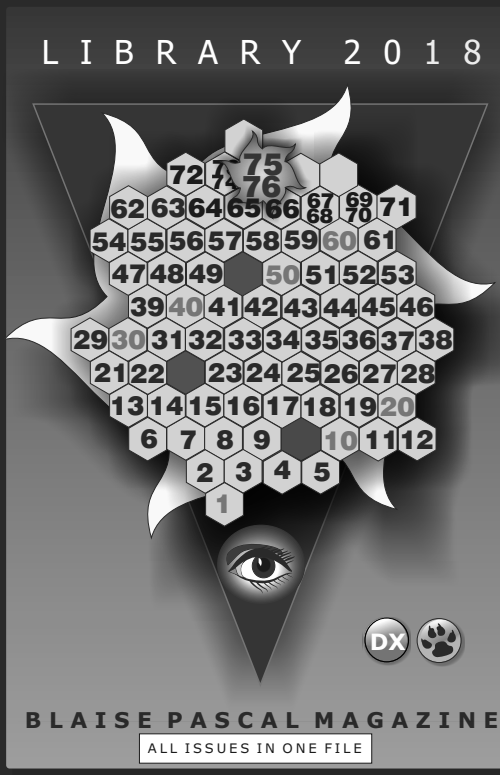
DAVID DIRKSE





```
procedure ;
var
begin
 for i := 1 to 9
do
  begin
```

GRAPHICS COMPUTER MATH & GAMES IN PASCAL



POCKET EDITION

Printed in full color.
A fully indexed PDF book is included + 52 projects

CREDITCARD LIBRARY STICK 16 GB

All issues 1-76 on the USB stick complete searchable 4300 pages -fully indexed including all code

TOTAL PACK: 4 ITEMS

1 BOOK INCLUDING 50 PROJECTS EXCL. SHIPPING	**€ 50**	**total ~~€ 210~~**
2 INCLUDING 1 YEAR DOWNLOAD SUBSRIPTION BLAISE PASCAL MAGAZINE	**€ 60**	
3 LEARN TO PROGRAM USING LAZRUS PDF	**€ 25**	**only € 100**
4 THE NEWEST LIBRARY STICK PDF ISSUES 1-76 INC CODE	**€ 75**	

https://www.blaisepascalmagazine.eu/product-category/bundles/

Official logo of HTML5

Photo of Tim Berners-Lee in April 2009

HYPERTEXT MARKUP LANGUAGE (HTML)

is the standard markup language for creating web pages and web applications. With **Cascading Style Sheets (CSS)** and **JavaScript,** it forms the most important accompanying of cornerstone technologies for the **World Wide Web.**

Web browsers receive **HTML** documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page.
HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets.
Tags such as **`<img />`** and **`<input />`** directly introduce content into the page.
Other tags such as **<p>** surround and provide information about document text and may include other tags as sub-elements.
Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages.
Inclusion of **CSS** defines the look and layout of content.
The **World Wide Web Consortium (W3C)**, maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

HISTORY
Development
In 1980, physicist Tim Berners-Lee, a contractor at **CERN,** proposed and prototyped **ENQUIRE,** a system for **CERN** researchers to use and share documents.
In 1989, Berners-Lee wrote a memo proposing an Internet-based hypertext system.

Berners-Lee specified HTML and wrote the browser and server software in late 1990. That year, **Berners-Lee** and CERN data systems engineer **Robert Cailliau** collaborated on a joint request for funding, but the project **was not formally adopted by CERN.** In his personal notes from 1990 he listed "some of the many areas in which hypertext is used" and put an encyclopedia first.

The first publicly available description of HTML was a document called **"HTML Tags"**, first mentioned on the Internet by Tim Berners-Lee in late 1991. It describes 18 elements comprising the initial, relatively simple design of HTML.
Except for the hyperlink tag, these were strongly influenced by **SGMLguid**, an in-house Standard Generalized Markup Language **(SGML)-based** documentation format at CERN. Eleven of these elements still exist in HTML 4.

HTML is a markup language that web browsers use to interpret and compose text, images, and other material into visual or audible web pages. Default characteristics for every item of HTML markup are defined in the browser, and these characteristics can be altered or enhanced by the web page designer's additional use of CSS.

Many of the text elements are found in the 1988 ISO technical report TR 9537 Techniques for using **SGML,** which in turn covers the features of **early text formatting languages such as that used by the RUNOFF command developed in the early 1960s for the CTSS** (**Compatible Time-Sharing System**) **operating system:** these formatting commands were derived from the commands used by typesetters to manually format documents.
However, the **SGML** concept of generalized markup is based on elements (***nested annotated ranges with attributes***) rather than merely print effects, with also the separation of structure and markup; **HTML has been progressively moved in this direction with CSS.**

Berners-Lee considered **HTML to be an application of SGML**. It was formally defined as such by the Internet Engineering Task Force (IETF) with the mid-1993 publication of the first proposal for an HTML specification, the "**Hypertext Markup Language (HTML)**" Internet Draft by **Berners-Lee** and **Dan Connolly**, which included an SGML Document type definition to define the grammar.

The draft expired after six months, but was notable for its acknowledgment of the NCSA Mosaic browser's custom tag for embedding in-line images, reflecting the IETF's philosophy of basing standards on successful prototypes.

Similarly, Dave Raggett's competing Internet-Draft, "HTML+ (Hypertext Markup Format)", from late 1993, suggested standardizing already-implemented features like tables and fill-out forms.
After the HTML and HTML+ drafts expired in early 1994, the IETF created an HTML Working Group, which in 1995 completed "HTML 2.0", the first HTML specification intended to be treated as a standard against which future implementations should be based.

Further development under the supervision of the IETF was stalled by competing interests.
Since 1996, the HTML specifications have been maintained, with input from commercial software vendors, by the World Wide Web Consortium (W3C).

However, in 2000, HTML also became an international standard. HTML 4.01 was published in late 1999, with further errata published through 2001.
In 2004, development began on HTML5 in the Web Hypertext Application Technology Working Group (WHATWG), which became a joint deliverable with the W3C in 2008, and completed and standardized on 28 October 2014.

MARKUP
HTML markup consists of several key components, including those called tags (and their attributes), character-based data types, character references and entity references.
HTML tags most commonly come in pairs like **<h1>** and **</h1>**, although some represent empty elements and so are unpaired, for example **<img>**.
The first tag in such a pair is the start tag, and the second is the end tag (*they are also called opening tags and closing tags*).

Another important component is the HTML document type declaration, which triggers standards mode rendering.

The following is an example of the classic "Hello, World!" program:

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

The text between **<html>** and **</html>** describes the web page, and the text between **<body>** and **</body>** is the visible page content. The markup text **<title>** This is a title **</title>** defines the browser page title.

The Document Type Declaration **<!DOCTYPE html>** is for HTML5. If a declaration is not included, various browsers will revert to "quirks mode" for rendering.

ELEMENTS
HTML documents imply a structure of nested HTML elements. These are indicated in the document by HTML tags, enclosed in angle brackets thus: ** .**
In the simple, general case, the extent of an element is indicated by a pair of tags: a "start tag" ** ** and "end tag" ** .** The text content of the element, if any, is placed between these tags.

Tags may also enclose further tag markup between the start and end, including a mixture of tags and text.
This indicates further *(nested)* elements, as children of the parent element.

The start tag may also include attributes within the tag. These indicate other information, such as identifiers for sections within the document, identifiers used to bind style information to the presentation of the document, and for some tags such as the **<img>** used to embed images, the reference to the image resource.

Some elements, such as the line break **
,** do not permit any embedded content, either text or further tags. These require only a single empty tag (*akin to a start tag*) and do not use an end tag.

Many tags, particularly the closing end tag for the very commonly used paragraph element ** ,** are optional.
An HTML browser or other agent can infer the closure for the end of an element from the context and the structural rules defined by the HTML standard.
These rules are complex and not widely understood by most HTML coders.

The general form of an HTML element is therefore:
<tag attribute1="value1" attribute2="value2">''content''</tag>.
Some HTML elements are defined as empty elements and take the form
<tag attribute1="value1" attribute2="value2">.
Empty elements may enclose no content, for instance, the **
** tag or the inline **<img>** tag.
The name of an HTML element is the name used in the tags.
Note that the end tag's name is preceded by a slash character, /, and that in empty elements the end tag is neither required nor allowed.
If attributes are not mentioned, default values are used in each case.

FEATURES

Markup

HTML 5 introduces elements and attributes that reflect typical usage on modern websites. Some of them are semantic replacements for common uses of generic block **( )** and inline **()** elements, for example **<nav>** (*website navigation block*), **<footer>** (*usually referring to bottom of web page or to last lines of HTML code*), or **<audio>** and **<video>** instead of **<object>.**

Some deprecated elements from HTML 4.01 have been dropped, including purely presentational elements such as **<font>** and ** ,** whose effects have long been superseded by the more capable Cascading Style Sheets. There is also a renewed emphasis on the importance of DOM scripting in Web behavior.

The HTML 5 syntax is no longer based on SGML despite the similarity of its markup. It has, however, been designed to be backward-compatible with common parsing of older versions of HTML.
It comes with a new introductory line that looks like an SGML document type declaration, **<!DOCTYPE html>**, which triggers the standards-compliant rendering mode. Since 5 January 2009, HTML 5 also includes Web Forms 2.0, a previously separate WHATWG specification.

New APIs

HTML5 related APIs
In addition to specifying markup, HTML 5 specifies scripting application programming interfaces (APIs) that can be used with JavaScript. Existing Document Object Model (DOM) interfaces are extended and de facto features documented. There are also new APIs, such as:

- **Canvas;**
- **Timed Media Playback;**
- **Offline;**
- **Editable content;**
- **Drag and drop;**
- **History;**
- **MIME type and protocol handler registration;**
- **Microdata;**
- **Web Messaging;**
- **Web Storage**
  a key-value pair storage framework that provides behaviour similar to cookies but with larger storage capacity and improved API.

Not all of the above technologies are included in the W3C HTML 5 specification, though they are in the WHATWG HTML specification.
Some related technologies, which are not part of either the W3C HTML 5 or the WHATWG HTML specification, are as follows.

The W3C publishes specifications for these separately:

- Geolocation;
- IndexedDB
  an indexed hierarchical key-value store (formerly WebSimpleDB);
- File
  an API intended to handle file uploads and file manipulation;
- Directories and System
  an API intended to satisfy client-side-storage use cases not well served by databases;
- File Writer – an API for writing to files from web applications;[
- Web Audio
  a high-level JavaScript API for processing and synthesizing audio in web applications;
- ClassList.
- Web cryptography API
- WebRTC
- Web SQL Database
  a local SQL Database (*no longer maintained*);

HTML 5 cannot provide animation within web pages. Additional JavaScript or CSS3 is necessary for animating HTML elements. Animation is also possible using JavaScript and HTML 4, and within SVG elements through SMIL, although browser support of the latter remains uneven as of 2011.

Error handling

Text document with **red question mark.svg**
This article possibly contains inappropriate or misinterpreted citations that do not verify the text.
HTML 5 is designed so that old browsers can safely ignore new HTML 5 constructs.

Popularity

According to a report released on 30 September 2011, 34 of the world's top 100 Web sites were using HTML 5 – the adoption led by search engines and social networks.
Another report released in August 2013 has shown that 153 of the Fortune 500 U.S. companies implemented HTML5 on their corporate websites.

Since 2014, HTML 5 is at least partially supported by most popular layout engines.

barnsten
• development tools • training
• consultancy • hands-on workshops
• components • support

DELPHI MEETUP Amsterdam : donderdag 7 maart 2019

Event details

Delphi Software ontwikkeling bij albelli

Delphi Meetup Amsterdam : donderdag 7 maart 2019

Er staat weer een nieuwe Meetup op het programma.
We zijn uitgenodigd bij **albelli,** gevestigd in de **Silver Tower** met onderstaand **waanzinnig uitzicht over Amsterdam**.

Afbeeldingsresultaat voor albelli uitzicht

Op 7 maart a.s. laten zij zien hoe bij albelli gebruik wordt gemaakt van Delphi voor ontwikkeling van hun toonaangevende software voor het maken van fotobeken voor meer dan vier miljoen klanten binnen diverse landen in Europa.

SALE

RAD Studio™ is the fastest way to design and develop modern, cross-platform native apps and services. It's easier than ever to create stunning, high performing apps for Windows, macOS, iOS, Android and Linux Server (Linux Server is supported in Delphi Enterprise or higher), using the same native code base. Share visually designed UIs across multiple platforms that make use of native controls and platform behaviors, and leverage powerful and modern languages with enhancements that help you code faster. Developers pick RAD Studio™ because it delivers 5x the speed for development and deployment across multiple desktop and mobile platforms.

GET 10% OFF

We celebrate the 10.3 Rio version with 10% discount on all versions.
Promotion ends February 13, 2019.

Order now also the CData Enterprise Connectors with 10% discount.
https://www.barnsten.com/default/promotions/cdata-enterprise-plus-68
Order your InterBase licenses with 10% discount.
https://www.barnsten.com/default/promotions/interbase-2017-server-additional-users

barnsten

- development tools
- consultancy
- components
- training
- hands-on workshops
- support

Delphi 10.3 Rio Professional - Named License (Full-Version)

~~€1,719.00~~ €1,547.10

Add to Cart

Delphi 10.3 Rio Enterprise - Named License (Full-Version)

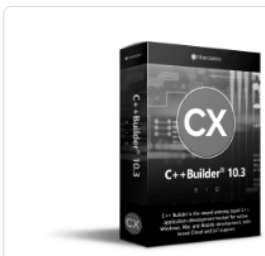
~~€4,041.00~~ €3,636.90

Add to Cart

Delphi 10.3 Rio Architect - Named License (Full-Version)

~~€6,466.00~~ €5,819.40

Add to Cart



C++Builder 10.3 Rio Professional - Named License (Full-Version)

~~€1,719.00~~ €1,547.10

Add to Cart

C++Builder 10.3 Rio Enterprise - Named License (Full-Version)

~~€4,042.00~~ €3,637.80

Add to Cart

C++Builder 10.3 Rio Architect - Named License (Full-Version)

~~€6,467.00~~ €5,820.30

Add to Cart

RAD Studio 10.3 Rio Professional - Named License (Full-Version)

~~€2,922.00~~ €2,629.80

Add to Cart

RAD Studio 10.3 Rio Enterprise - Named License (Full-Version)

~~€4,850.00~~ €4,365.00

Add to Cart

RAD Studio 10.3 Rio Architect - Named License (Full-Version)

~~€7,275.00~~ €6,547.50

Add to Cart

ABOUT PAS2JS

PAS2JS is a compiler/transpiler to translate programs written in Pascal (subset of Delphi/ObjFPC syntax) to JavaScript.
The goal is to use strong typing, while still be able to use low level whenever you choose.
The compiled Pascal functions can be used in DOM events or called by JavaScript.
pas2js is written completely in FPC, runs on many platforms like Windows, Mac and Linux and more. It is built modular consisting of the following parts:

- file cache - loading, caching files, converting to **`UTF-8`**
- file resolver - handling search paths, finding used units and include files
- scanner - reading tokens, handling compiler directives like **`$IfDef`** and **`$Include`**
- parser - reading the tokens, checking syntax, creating Pascal nodes
- resolver - resolving references, type checking and checking duplicate identifiers
- use analyzer - finding unused identifiers, emit hints and warning
- converter - translating Pascal nodes into JavaScript nodes
- compiler - handling config files, parameters, compiling recursively all used units, writes js
- command line interface - a small wrapper to embed the compiler into a console program
- library and interface - a small wrapper to embed the compiler into a library

Each part is tested separately and is used by other FPC tools as well.
For example the scanner and parser are used by fpdoc too. Thus they are tested and extended by other programmers, reducing greatly the work for developing **PAS2JS**.
Consistency is kept by several test suites, containing thousands of tests.

Note: The modular structure allows to compile any parts or the whole compiler into an IDE addon (not yet started).

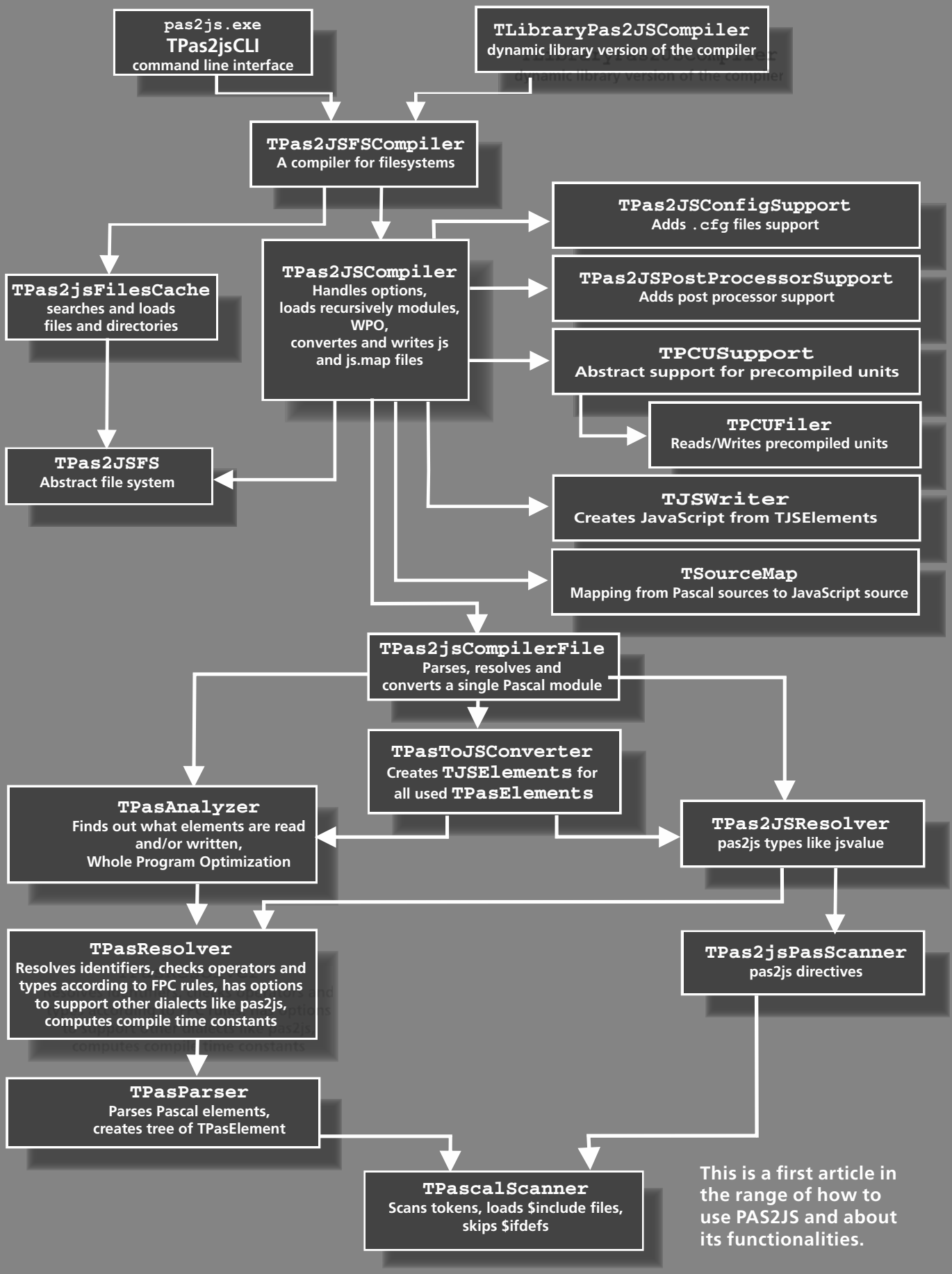
Delphi and ObjFPC mode

Delphi mode
Defines macro **DELPHI**
Assigning a function to a function type variable does not require the @ operator. For example, you can write either **`OnGetThing:=GetValue; or OnGetThing:=@GetValue;.`**
A function type variable reference without brackets is treated as a call. For example: If **`OnGetThing`** is a variable of type function: integer you can write: If **`OnGetThing=3`** then ;.
You must use the @@ operator to get the procedure **`address`** (i.e. JS reference) of a procedure type variable.
For example instead of If **`OnClick=nil`** then ; you must use **`if @@OnClick=nil then ;.`**
Every procedure/method overload needs the **`'overload'`** modifier.

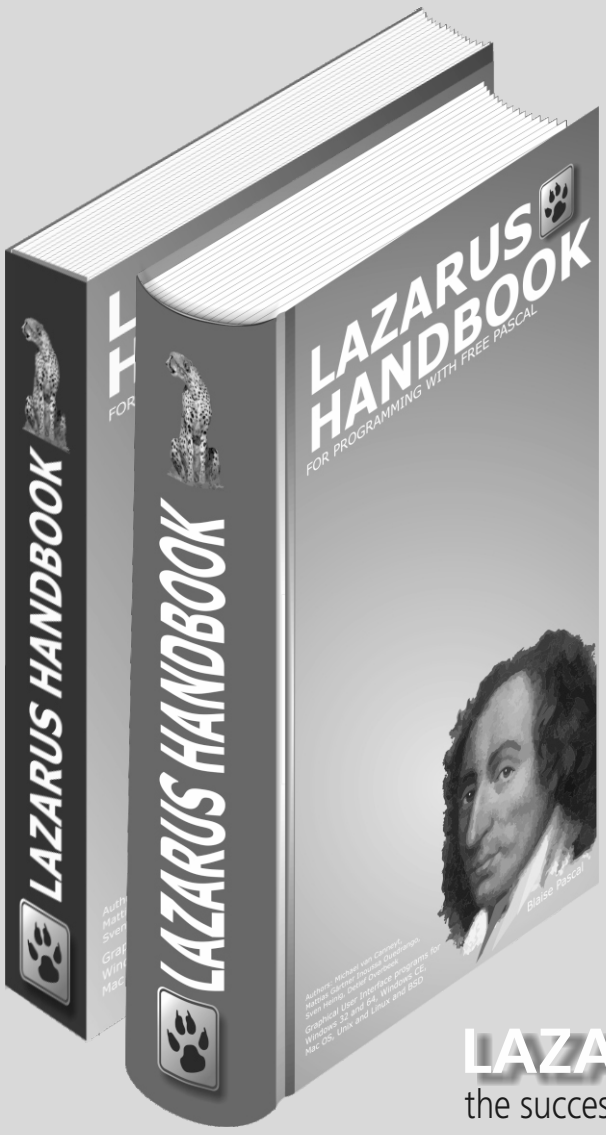
ObjFPC mode

This the default mode of **PAS2JS** and is generally more strict than the Delphi mode, and allows some more operations.
Defines macro OBJFPC
Assigning a function to a function type variable requires the @ operator. For example:
`OnGetThing:=@GetValue;.`
A function type variable always needs brackets to be called. For example: If **`OnGetThing`** is a variable of type function: integer then this is allowed: **`If OnGetThing()=3`** then ;.
While this gives an error: If **`OnGetThing=3`** then ;.
You can compare a procedure type with nil. For example If **`OnClick=nil`** then ;.
You can compare a procedure type with a procedure address (i.e. JS reference).
For example **`If OnClick=@OnFormClick then ;.`**
The procedure modifier **`'overload'`** can be omitted when all overloads are in one scope, e.g. a unit or a class.
And if one procedure has such modifier all procedures with same name and in same scope are overloads as well.

This is part of the information you can find at:
`https://www.blaisepascalmagazine.eu/about-pas2js/`



This is a first article in the range of how to use PAS2JS and about its functionalities.



Pocket version
and the hardcover version.
Details available on the website;

LAZARUS HANDBOOK

the successor for the "Lazarus complete guide"

As promised we have a preview file available for download for subscribers who have ordered a copy of the Lazarus Handbook in advance of its publication. This pdf is for pre-order customers only.
If you have pre-ordered the book, please re-send your email address to office@blaisepascal.eu
(so we can verify the request) and we will send you by return an address from which you can download the preview file.

The authors and editors are still working on the book, adding new material and updating the content for Lazarus version 2.0.
Since there is a continued delay in publication we have produced an interim pdf of chapters that are already complete.
The final book's pdf will be considerably larger than this preview. We apologise for the delay.
This is partly owing to the need to edit and produce this magazine,
and partly because some authors are also working on other projects for us, such as Pas2js.

The definitive version of the new book will be published as soon as possible,
and an announcement of the publication date will be made in due course.

If you want updates about this book please go to:
https://www.blaisepascalmagazine.eu/blog/update-about-the-lazarus-handbook/

BY DETLEF OVERBEEK

INTRODUCTION

kbmMW has a special way to install and I try here to explain that all to you.

By now I have some experience in the way of downloading and installing various items.
The organisation that Kim Madsen uses for his component suite has a different approach: first of all - as always you better organize the way you want to handle things. It will help you not to make all the wrong steps many people do when they choose to install his free Memtable, or eventually the professional or the enterprise version of his suite.

The installation of the kbmMW Suite has some special items, which might be new for you: first of all - to download your code you need to go to the website:
`https://portal.components4developers.com/`
You need to be careful with your passwords etc. because after years, if you update the code regularly, itwill not change so you will get no new one with each new version. You can of course ask for it again but it' s better to keep it at hand. Its a logical way to do this, but quite opposite to what Embarcadero does. So your code is used for all future updates new versions etc.

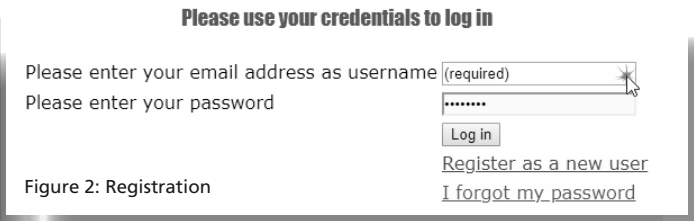


Figure 2: Registration

Once your used to it it's very simple. The length of the activation code is good to fill a complete book. Your credentials will be asked and then the page appears where you can download what you need or have bought. So simply click on the download mark and you can start to install. One thing is important:
Before you install the **Pro-version** or the **Enterprise-Mversion** you **need to install the memtable first.** The **Standard-Version** and the **Free-Version (CodeGear)** can be downloaded at the same page. (*The procedure is equal. If you do not have a contact or subscription simply ask for one. The answer is prompt*).
If you don't it will not work correctly - it's mentioned in the **installation files,** but in your enthusiasm you might forget that. Now the first things are done and you can start to install. I suggest you to create an Installer - Directory like **`C:/KBMMemtable_INSTALL_7810.`** (*or give your own name*).

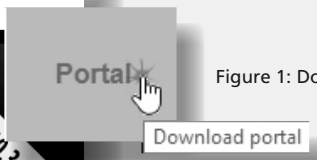


Figure 1: Download Portal

From here the install takes place: You can create your own Directory to make the install towards. You need to know that installing this program is not the installing into your Delphi version. First the components etc will be put in to an installer Wizard which will guide you through all possible settings and that is quite a lot.
I first will show the short method for the MemTable. Starting at figure15 on page 66
I will also show the options you will have by the wizzard that is used, for the Pro- or Enterprise Version.
NOTE: once the installment is done you need to show Delphi the right path and you also will need to give that path in the Project Details. I will explain how to.
The following Figure shows what will be installed and where.

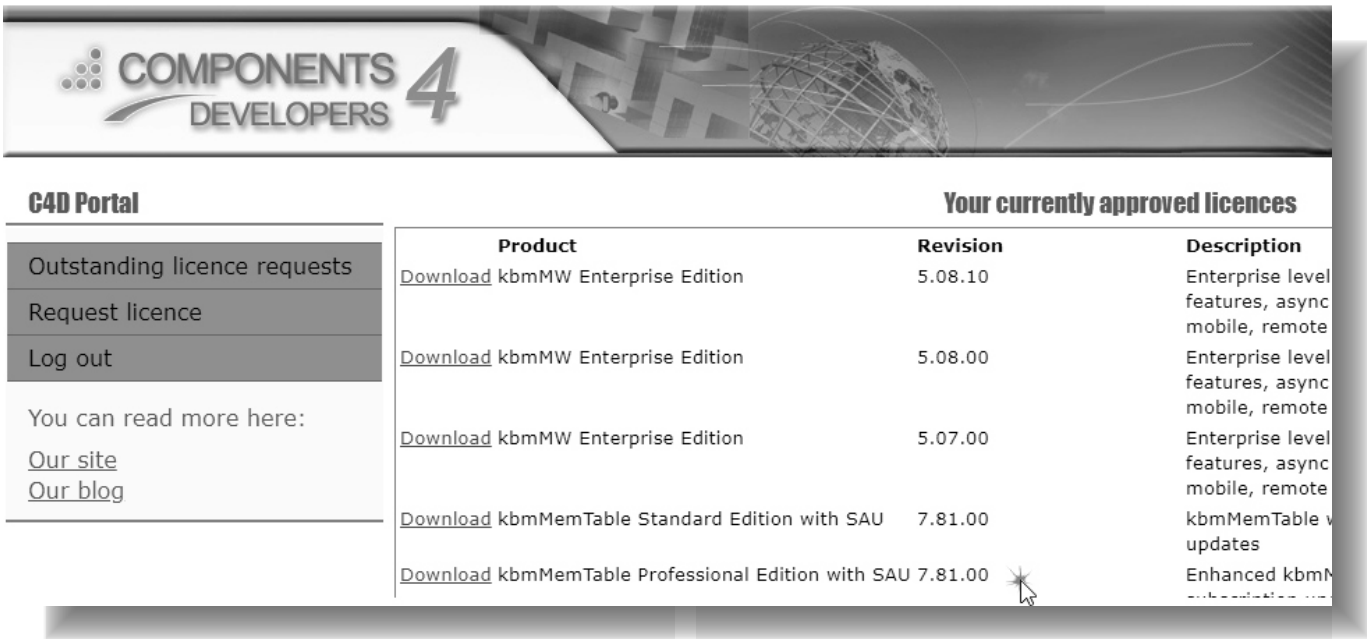


Figure 3: approved licences

NOTE: All pre-installations:
- Unless you have specific reasons not to, remove old kbmMemTable installation completely, including all `kbmMem*` and `kbmSQL* dcu's, bpl's, dcp's, obj's, dll's, dcuil's, bpi's, lib's, ppu's, o's etc.`

kbmMemTable exists in 3 versions:
CodeGear Edition (CG)
Standard Edition (Std)
Professional Edition (Pro)

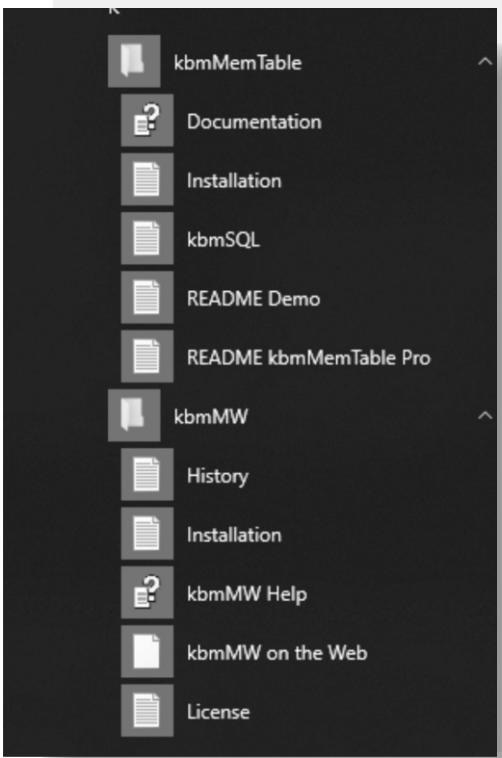


Figure 4: after insatllation in win 10

Lazarus 1.2.4 with FreePascal 2.6.4 or newer

- Open Package: kbmMemRunLazXXX.lpk
- Recompile Clean
(*in the More.. button on the package toolbar*)
- Open Package: **`kbmMemDesLazXXX.lpk`**
- Recompile Clean (*in the More.. button on the package toolbar*)
- Install (*Recompile IDE*)

If the installation fails
(*linker says it can't find kbmMemRunLazXXX*),
please click on Tools/Build Lazarus with profile:
Build All
Then the package should be installed.

You will find 4 new components in the 'kbmMemTable' tab after a successfull installation.
I will describe that in the next issue 78 exactly, and show in detail how to.

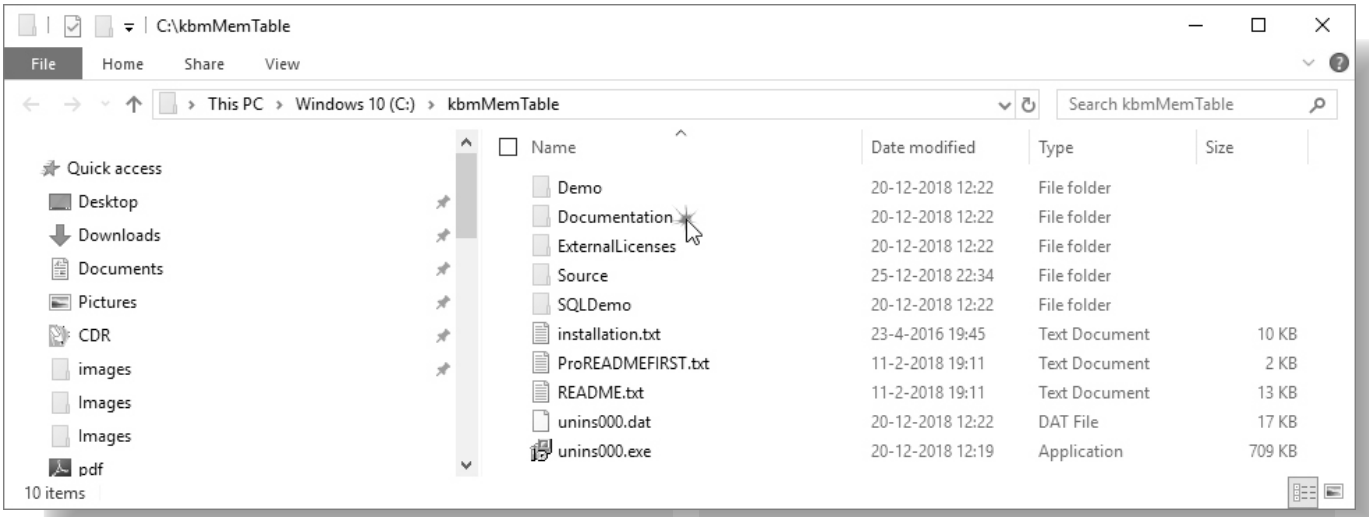


Figure 5: Overview of all the directories after insatllation in win 10

From here the install takes place by the wizzard:
You can create your own Directory to make the install to. You need to know that installing this program is not the installing into your Delphi version.
The following Figure shows what will be installed and where.

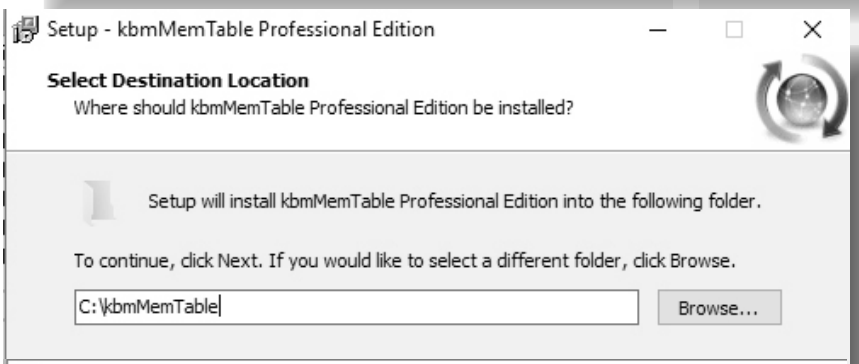


Figure 6: Destination location in the Wizard

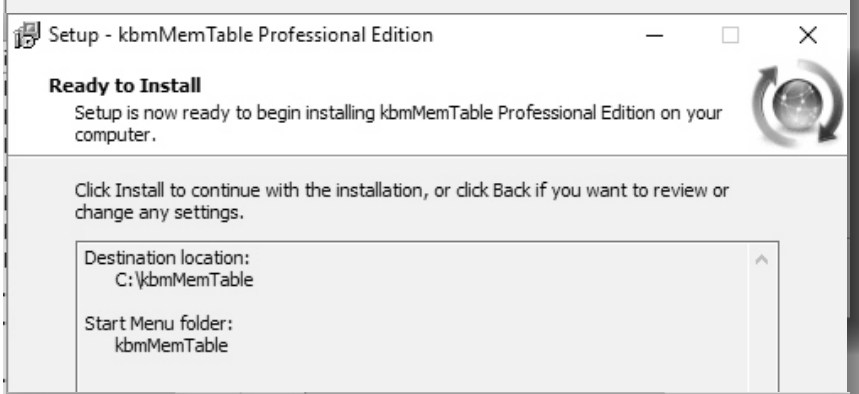


Figure 7: Shows your Installation Dir

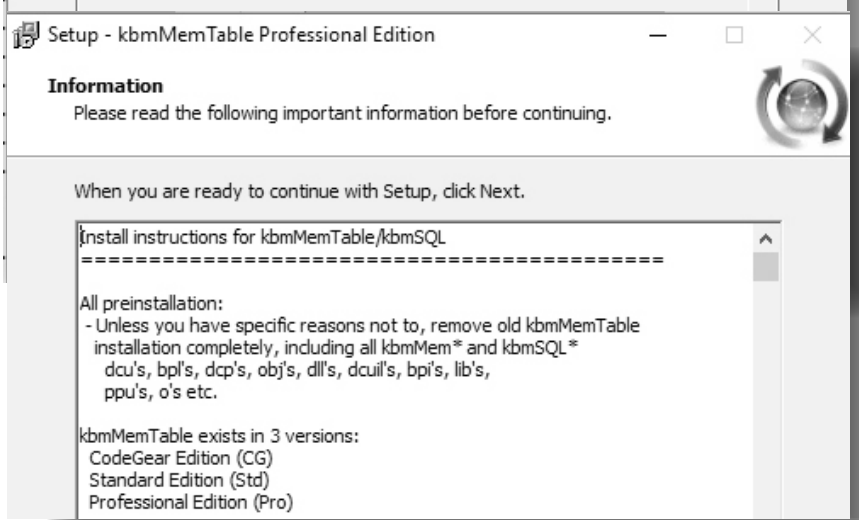


Figure 8: Install instructions

Figure 9: Completing

To install in Delphi:

- Open **kbmMemRunD101XXX.dproj** or your version number, you can find that in the list
- **Compile** <--

- Add
  <the kbmMemTable source dir>
  \D101\Win32 or
  ...\...Win64 or
  ...\...OSX32 or
  ...\...IOS32 or
  ...\...IOS64 or
  ...\...Android directory
  to the search path depending on what environment you want to compile against.
  See the next page for installation figures (page 65)

- Open **kbmMemDesD101XXX.dproj**
- **Compile**
- **Install.**

- **To compile mobile applications, please add the kbmMemTable source directory to Tools/Options/Library/Library Path for the relevant target (Android, IOS Device, IOS Simulator etc)**

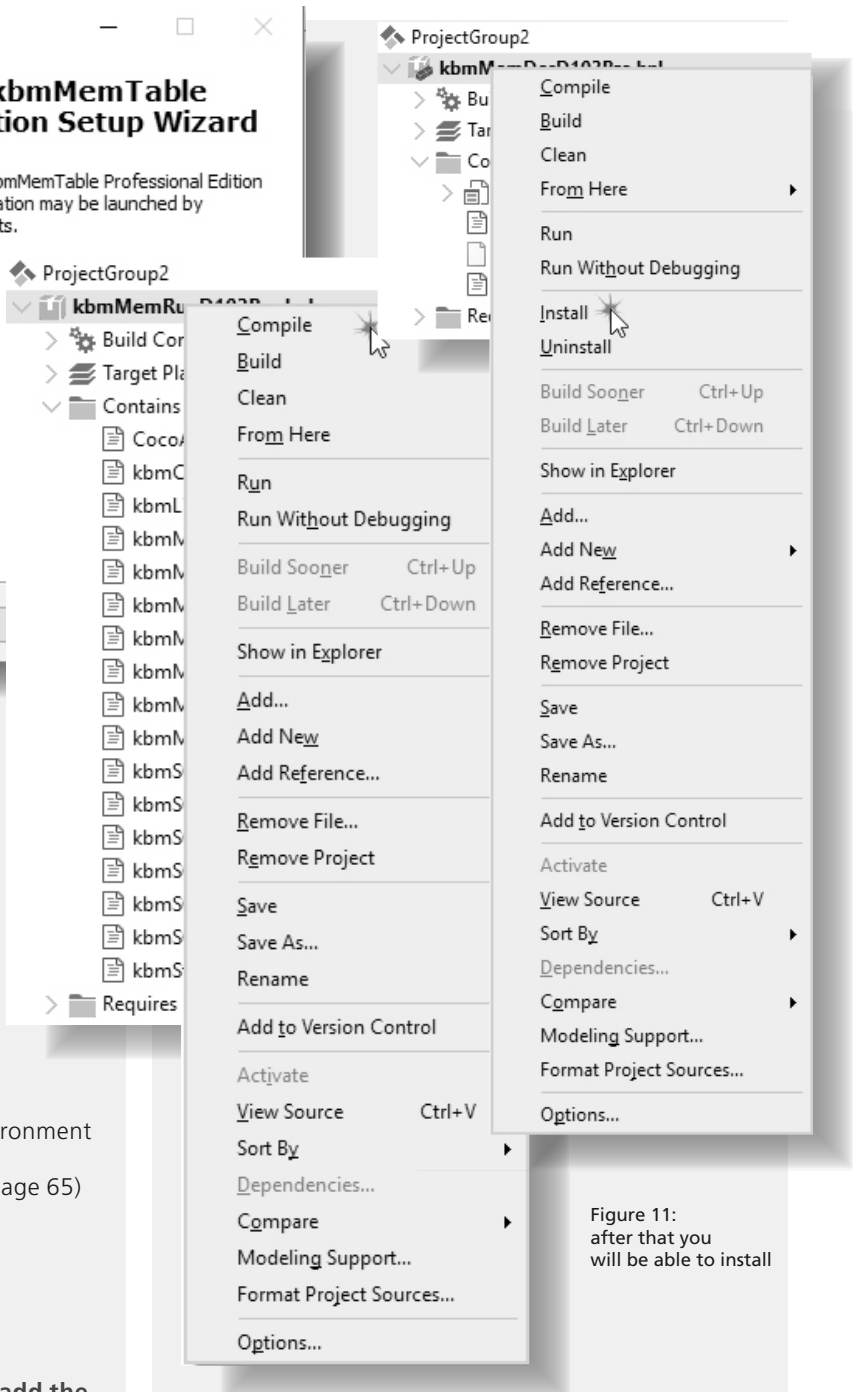


Figure 10: Compilation

Figure 11:
after that you
will be able to install

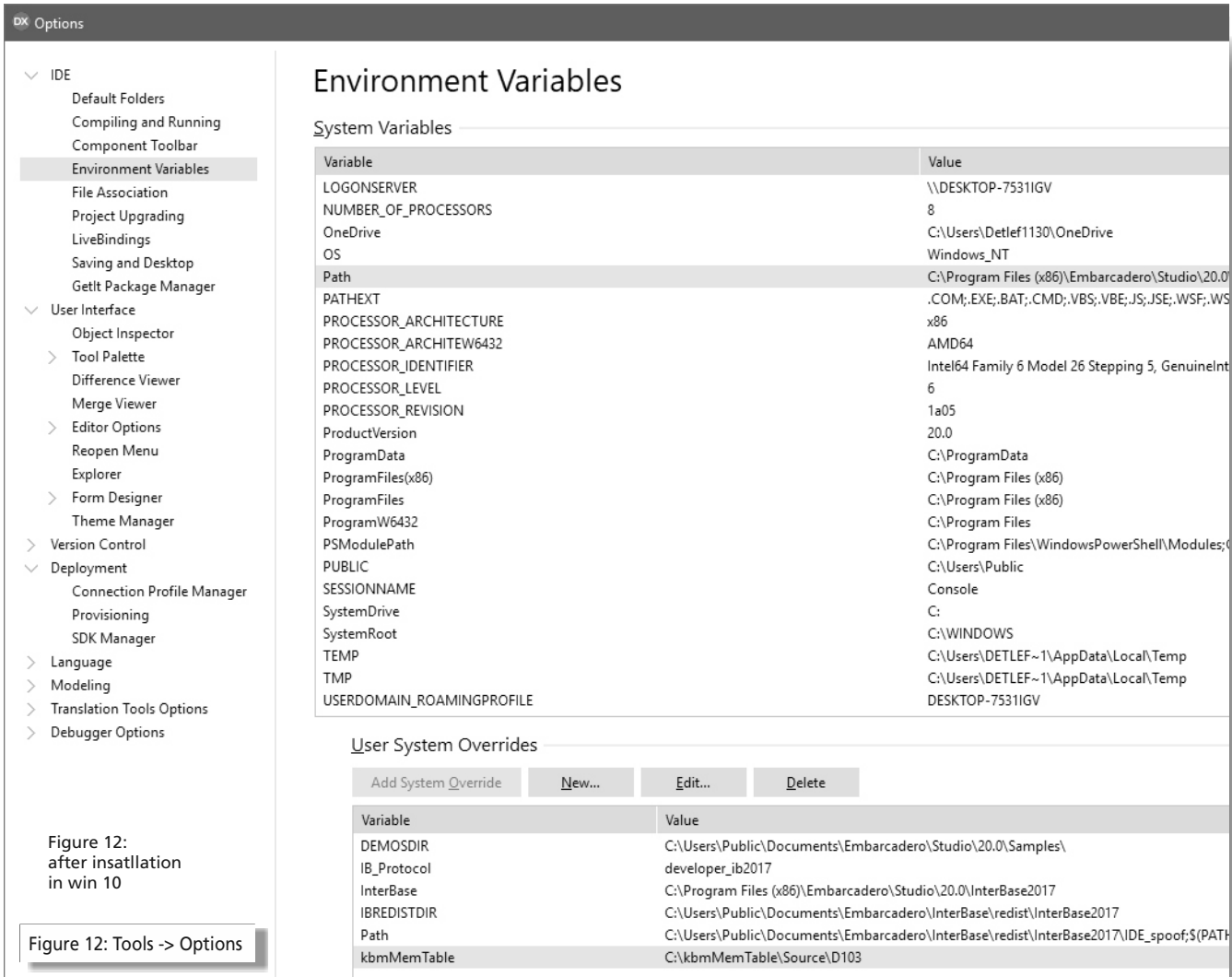


Figure 12:
after insatllation
in win 10

Figure 12: Tools -> Options

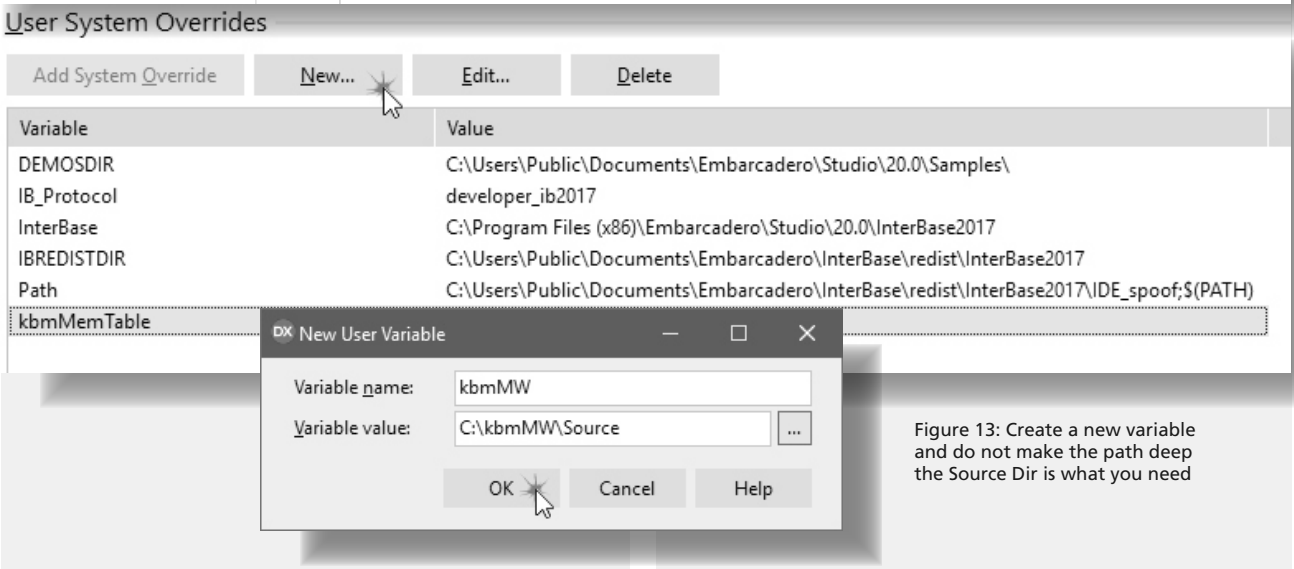
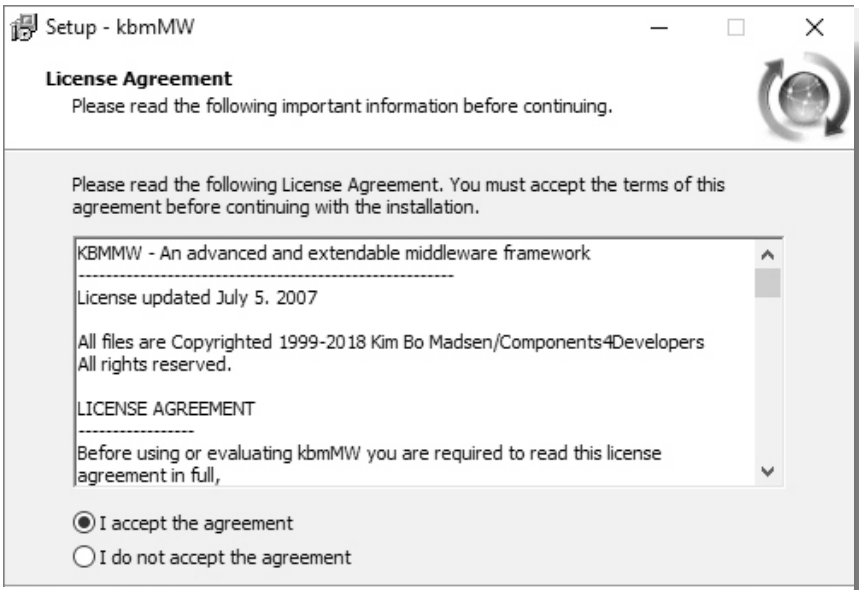


Figure 13: Create a new variable and do not make the path deep the Source Dir is what you need

Figure 14: The window to create the new variable



This is the part where the Enterprise version is shown how to be installed

Figure 15: Install Dir

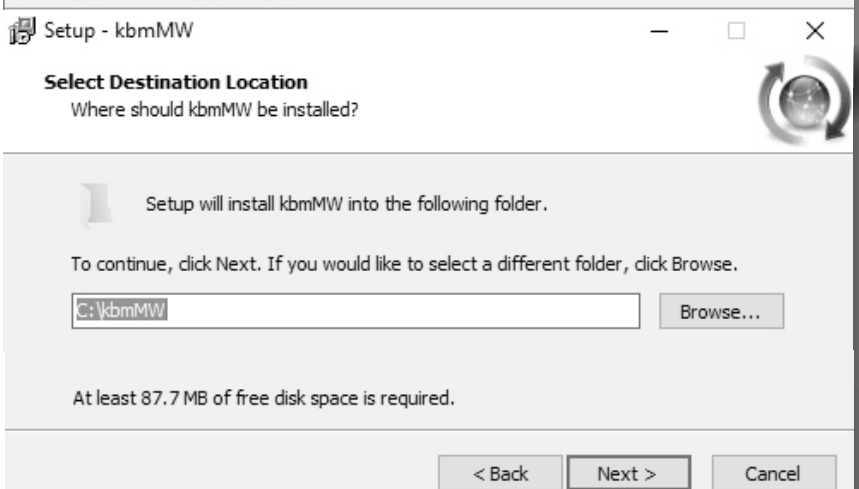


Figure 16: The window to create the new variable

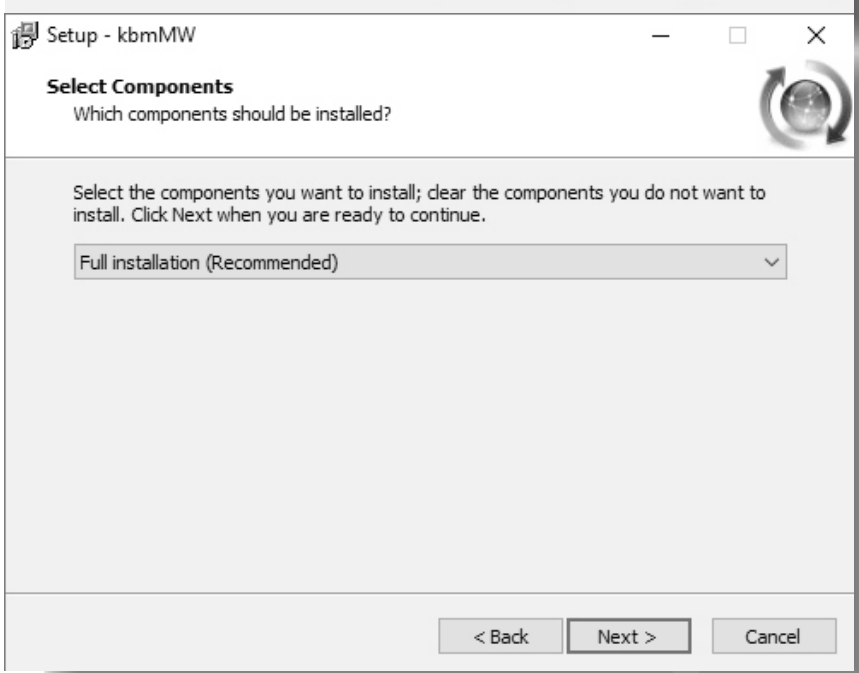


Figure 17: Install selection

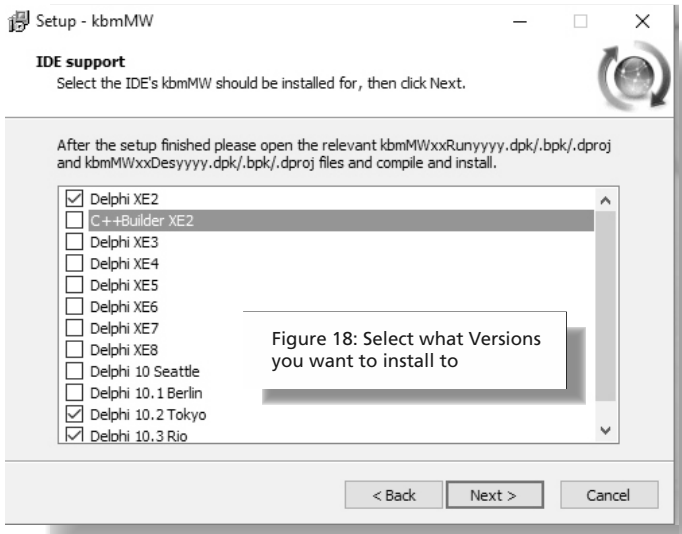


Figure 18: Select what Versions you want to install to

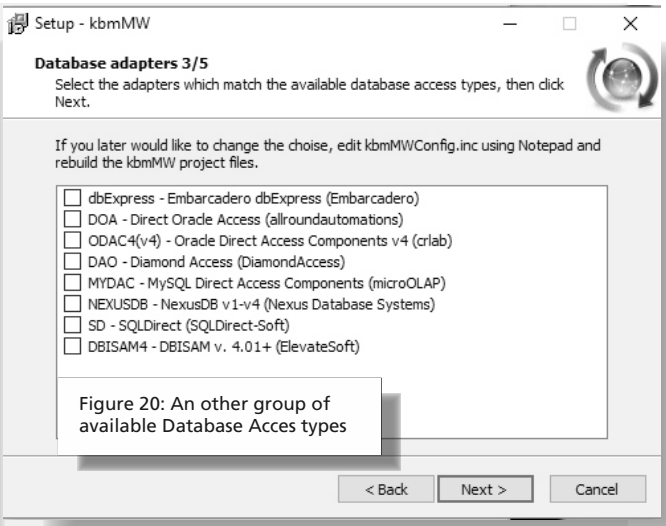


Figure 20: An other group of available Database Acces types

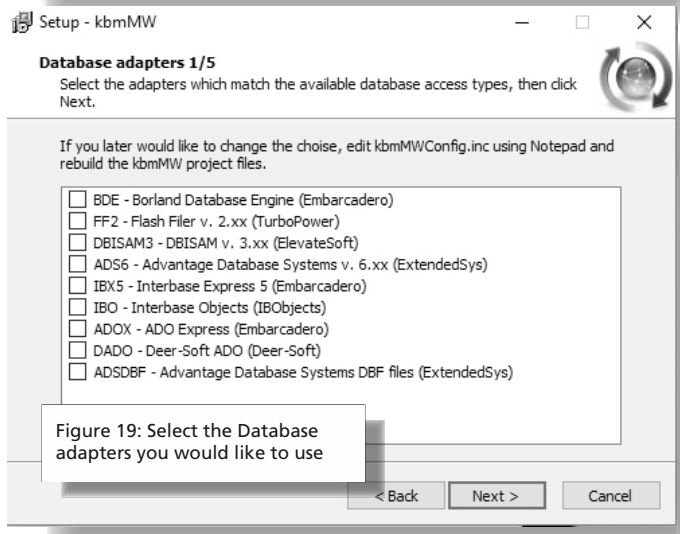


Figure 19: Select the Database adapters you would like to use

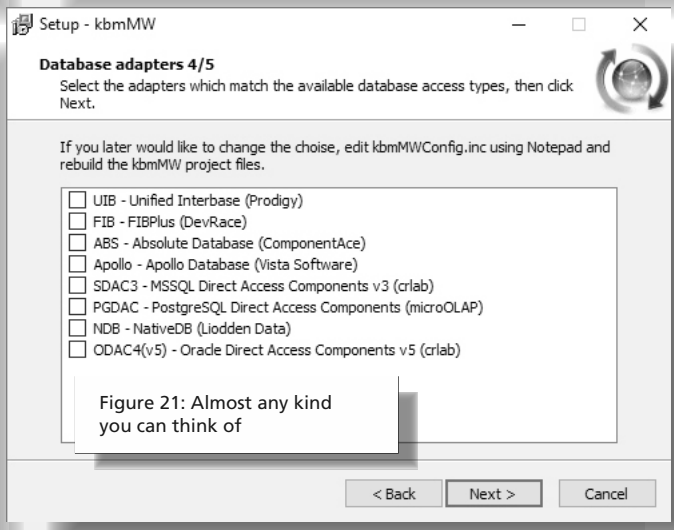


Figure 21: Almost any kind you can think of

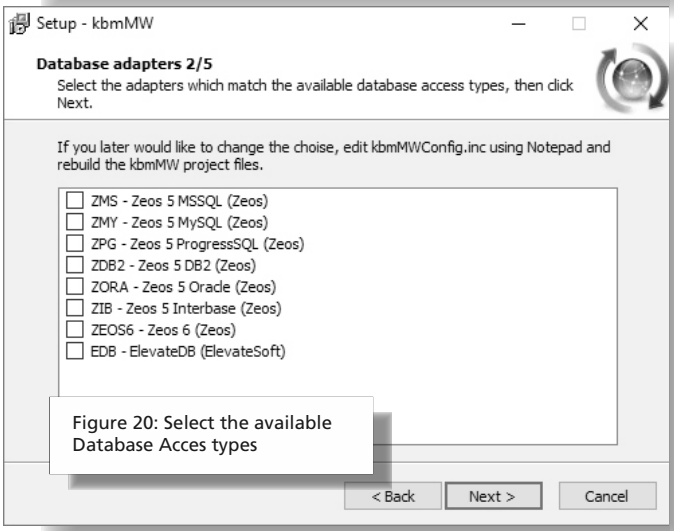


Figure 20: Select the available Database Acces types

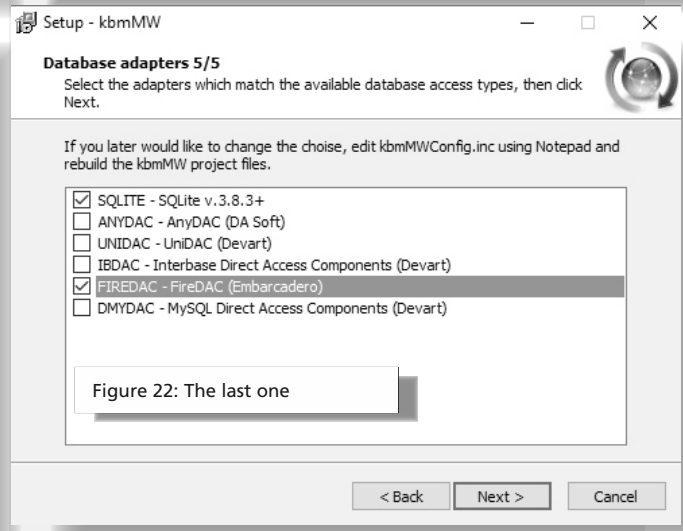


Figure 22: The last one

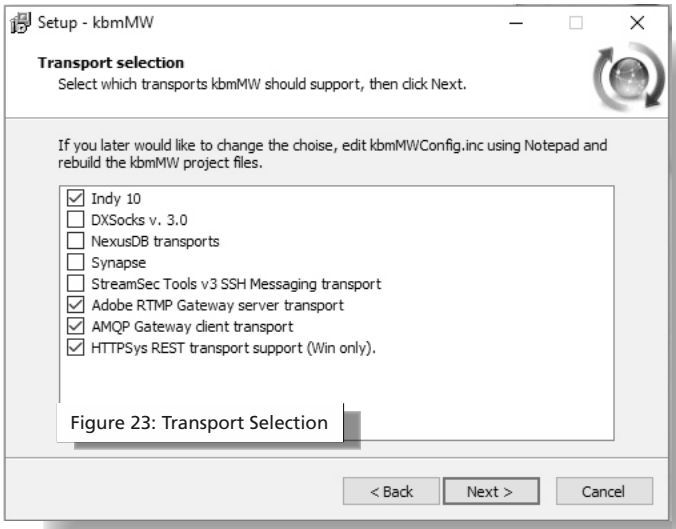


Figure 23: Transport Selection

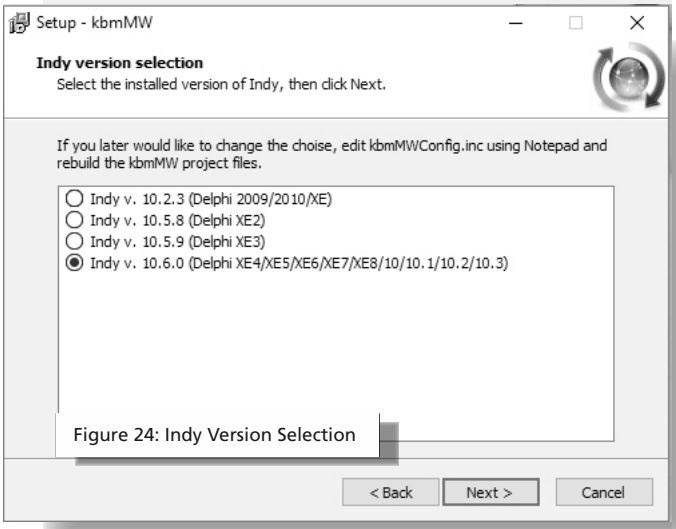


Figure 24: Indy Version Selection

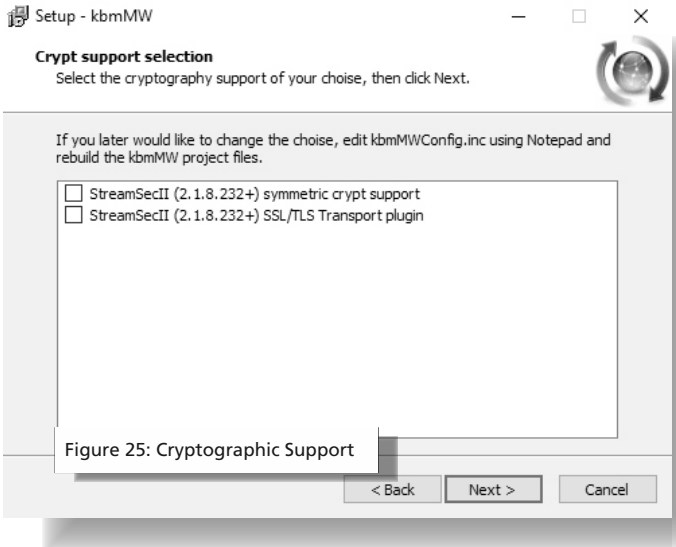


Figure 25: Cryptographic Support

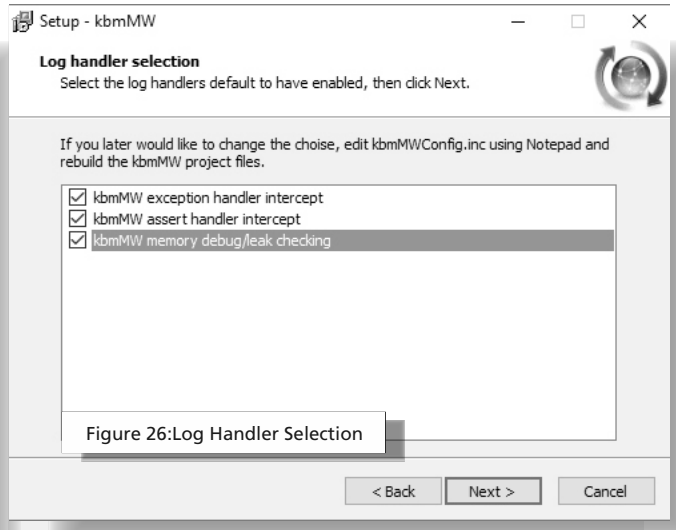


Figure 26:Log Handler Selection

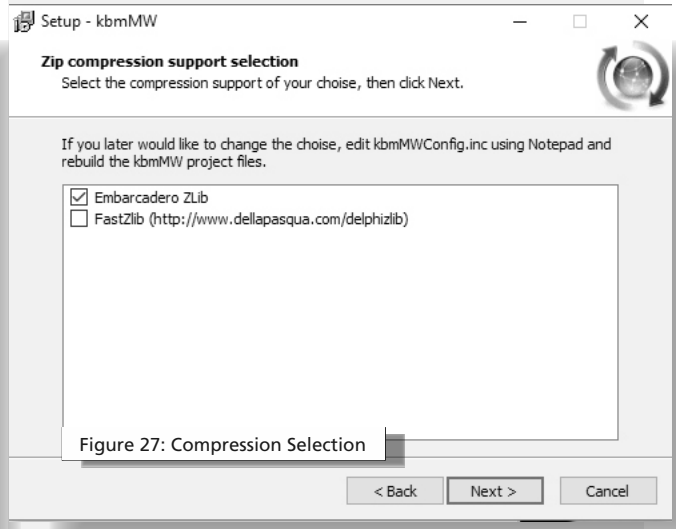


Figure 27: Compression Selection

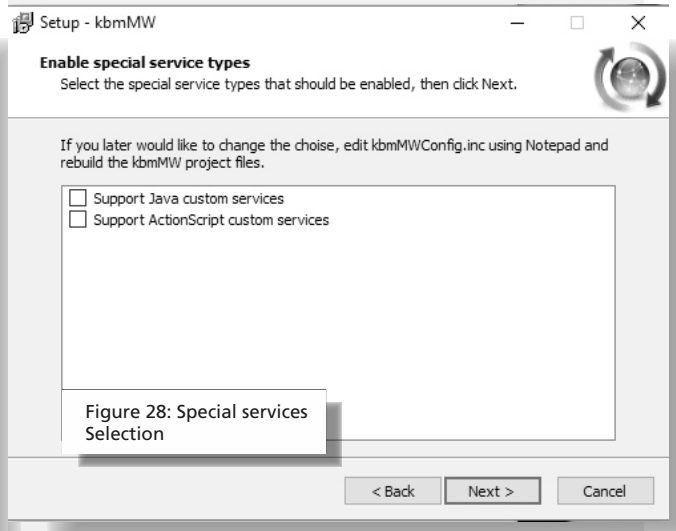


Figure 28: Special services Selection

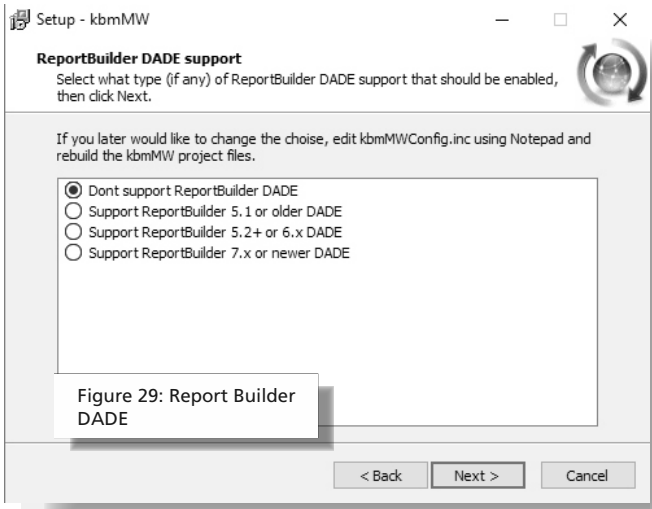


Figure 29: Report Builder DADE

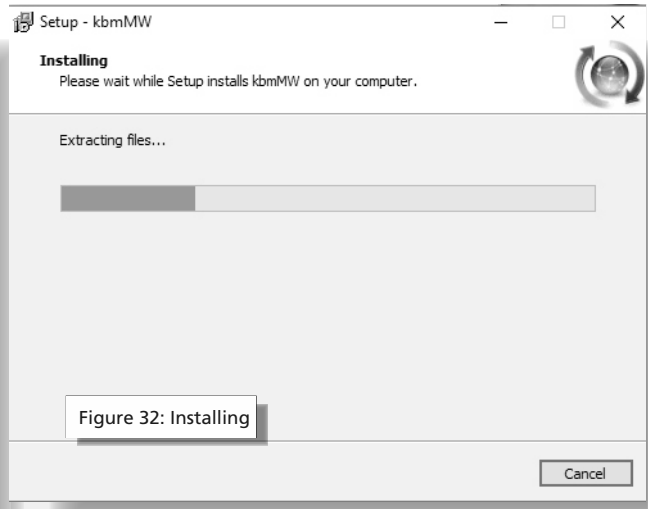


Figure 32: Installing

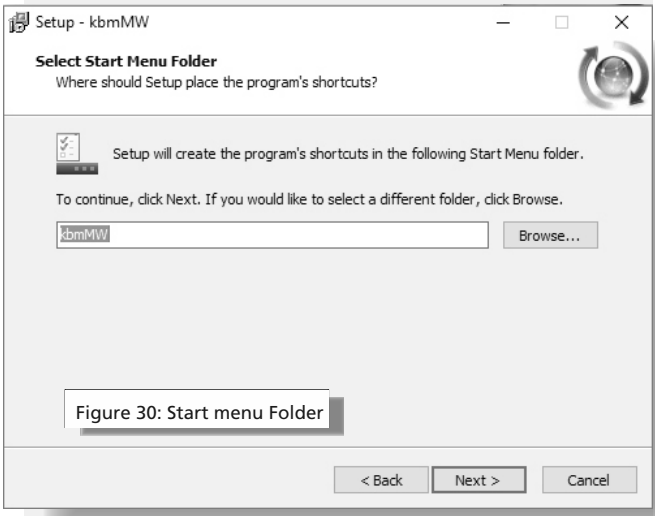


Figure 30: Start menu Folder

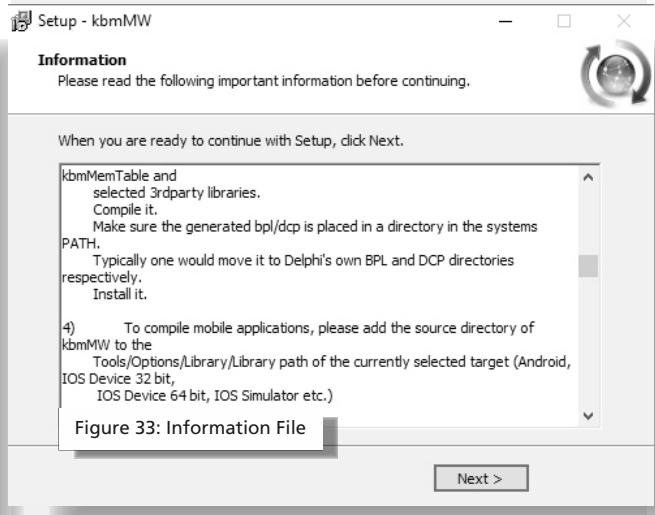


Figure 33: Information File

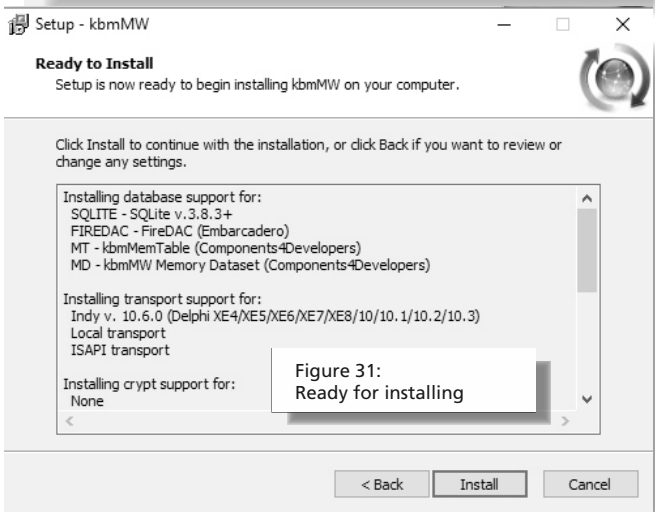


Figure 31:
Ready for installing

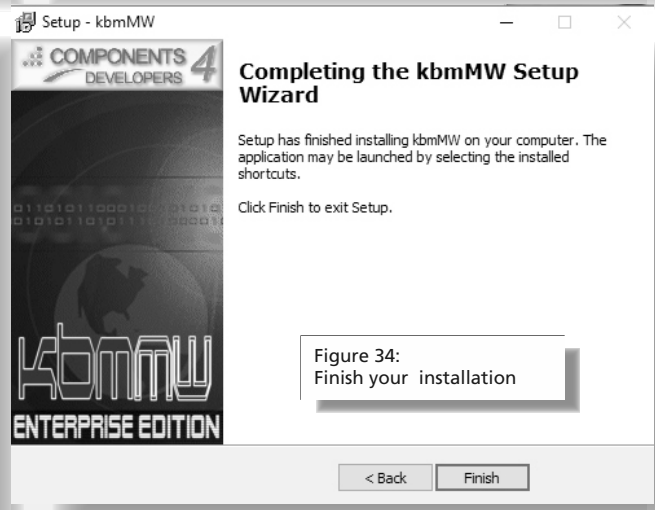


Figure 34:
Finish your installation

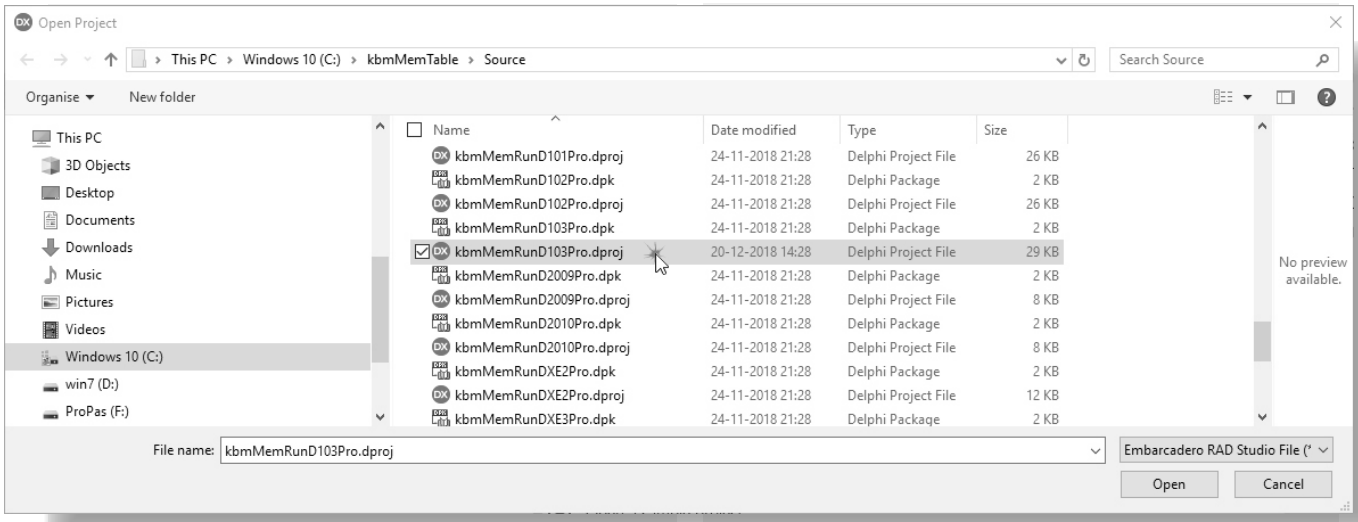


Figure 35: Select your project that you need to to compile the first time

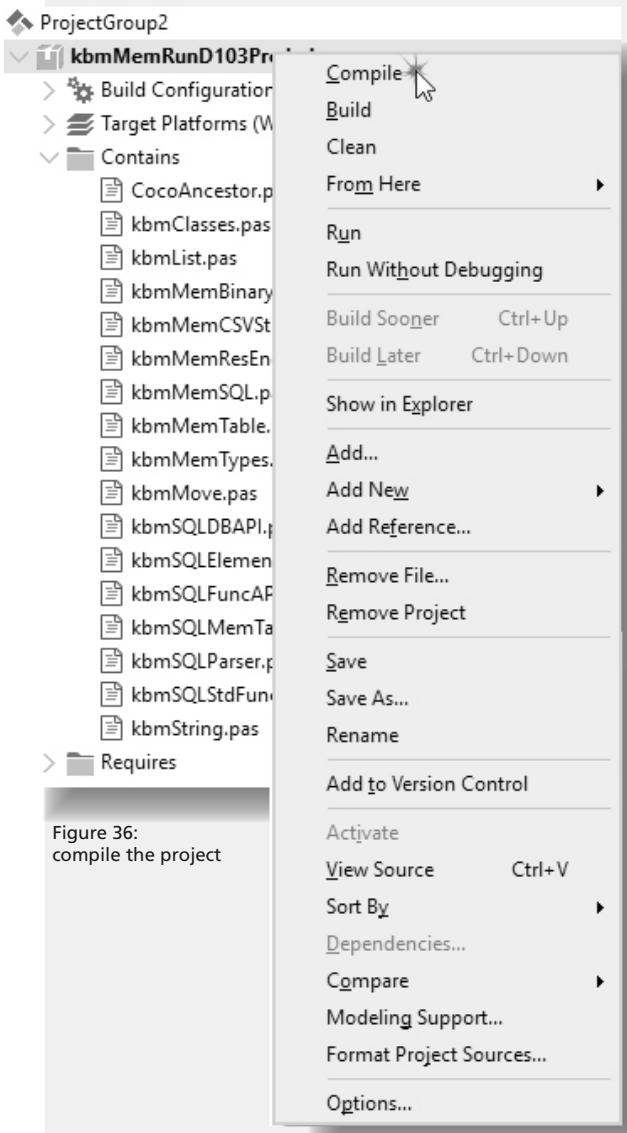


Figure 36:
compile the project

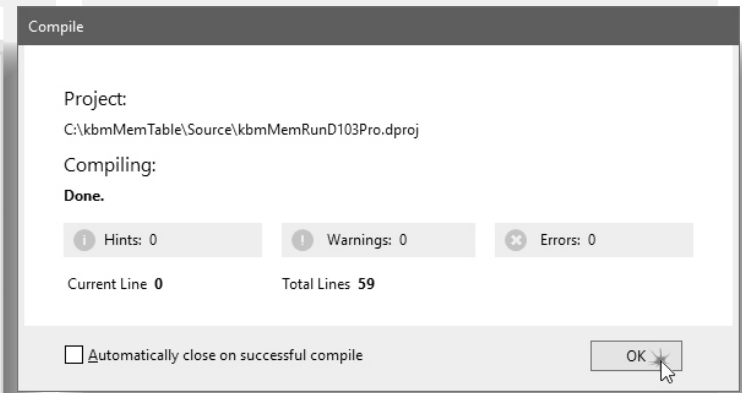


Figure 37: Result of the first compilation

And again the installation figures: It is of course the same way as the Memtable installation.
The installment comes without a ready compiled project because of the large number of installing variations.

kbmMW General
TkbmMWEventCrypt
TkbmMWAuthorizationManager
TkbmMWLZHCompression
TkbmMWLZ4Compression
TkbmMWCrypt
TkbmMWPooledSession
TkbmMWTransactionResolver
TkbmMWEventResolver
TkbmMWMemSQL
TkbmMWTemporarySequencer
TkbmMWDatastoreSequencer
TkbmMWRoundRobinLoadBalancer
TkbmMWRandomLoadBalancer
TkbmMWBestFitLoadBalancer
TkbmMWLoadBalancedServerAnnouncement
TkbmMWMemoryMessageQueue
TkbmMWMessageAction
TkbmMWMessageActionDispatcher
TkbmMWMultithreadMessageActionDispatch
TkbmMWDispatchedMessageAction
TkbmMWProgressManager
TkbmMWSyncMessageProcessor
TkbmMWAsyncMessageProcessor
TkbmMWSyncMessageQueueProcessor
TkbmMWAsyncMessageQueueProcessor
TkbmMWMultithreadMessageQueueProcessc
TkbmMWGroupedMultithreadMessageQueue
TkbmMWFileStoreMessageQueue
TkbmMWFilePool
TkbmMWBPVisualizer
kbmMW Hashes
TkbmMWHashHaval
TkbmMWHashMD4
TkbmMWHashMD5
TkbmMWHashRipeMD128
TkbmMWHashRipeMD160
TkbmMWHashSHA1
TkbmMWHashSHA256
TkbmMWHashSHA384
TkbmMWHashSHA512
TkbmMWHashTiger
kbmMW MetaData
TkbmMWNullMetaData
TkbmMWGenericSQLMetaData
TkbmMWOracleMetaData
TkbmMWInterbaseMetaData
TkbmMWMSSQLMetaData
TkbmMWSQLiteMetaData
kbmMW Clients
TkbmMWSimpleClient
TkbmMWInventoryClient
TkbmMWPooledInventoryClient
TkbmMWPooledSimpleClient
TkbmMWClientConnectionPool
TkbmMWClientQuery
TkbmMWClientStoredProc
TkbmMWClientBriefCaseBinaryStreamFormat
TkbmMWClientTransactionResolver
TkbmMWFileClient
TkbmMWPooledFileClient
kbmMW MT
TkbmMWMTConnectionPool
TkbmMWMTQuery
kbmMW Server
TkbmMWLocalStat
TkbmMWServer
TkbmMWONStat
TkbmMWWinPerfMonStats
kbmMW SQLite
TkbmMWSQLiteConnectionPool
TkbmMWSQLiteQuery
TkbmMWSQLiteResolver
kbmMemTable
TkbmMemTable
TkbmBinaryStreamFormat
TkbmCSVStreamFormat
TkbmMemSQL
kbmMW Client Transports
TkbmMWLocalClientTransport
TkbmMWTCPIPIndyClientTransport
TkbmMWISAPIClientTransport
TkbmMWTCPIPIndyMessagingClientTransport
TkbmMWUDPIndyMessagingClientTransport
kbmMW FireDAC
TkbmMWFireDACConnectionPool
TkbmMWFireDACQuery
TkbmMWFireDACStoredProc
TkbmMWFireDACResolver
kbmMW MD
TkbmMWMDConnectionPool
TkbmMWMDQuery
TkbmMWMDResolver
This is what you get for your money
kbmMW Server Transports
TkbmMWLocalServerTransport
TkbmMWTCPIPIndyServerTransport
TkbmMWHTTPSysServerTransport
TkbmMWISAPIServerTransport
TkbmMWISAPIRESTServerTransport
TkbmMWRTMPGateway
TkbmMWVirtualMessagingServerTransport
TkbmMWTCPIPIndyMessagingServerTransport
TkbmMWUDPIndyMessagingServerTransport
kbmMW Dataset Streamformats
TkbmMWBinaryStreamFormat
TkbmMWXMLStreamFormat
TkbmMWJSONStreamFormat
kbmMW Ciphers
TkbmMWCipherAES
TkbmMWCipherRijndael
TkbmMWCipherBlowfish
TkbmMWCipherCast128
TkbmMWCipherCast256
TkbmMWCipherDES
TkbmMWCipher3DES
TkbmMWCipherICE
TkbmMWCipherThinICE
TkbmMWCipherIDEA
TkbmMWCipherMars
TkbmMWCipherMisty1
TkbmMWCipherRC2
TkbmMWCipherRC4
TkbmMWCipherRC5
TkbmMWCipherRC6
TkbmMWCipherSerpent
TkbmMWCipherTea
TkbmMWCipherTwofish
kbmMW SQL Rewriter
TkbmMWGenericANSI92SQLRewriter
TkbmMWGenericANSI2003SQLRewriter
TkbmMWOracleSQLRewriter
TkbmMWMySQLRewriter
TkbmMWMSSQLRewriter
TkbmMWInterbaseSQLRewriter
TkbmMWPostgreSQLRewriter
TkbmMWSQLiteSQLRewriter
kbmMW Cross Adapter
TkbmMWXConnectionPool
TkbmMWXQuery
TkbmMWXStoredProc
TkbmMWXResolver
TkbmMWXMultiDBConnectionPool

The new release of kbmMW includes additional features in the configuration framework.

If you do not know what the configuration framework is, please go read the blog post: **REST easy with kbmMW #7 – Configuration.**

Lots of the smart features in **kbmMW** utilize attributes for defining various metadata for parts in your smart **kbmMW** based application server.

For example the **kbmMW_Service attribute** that defines various things about the service class.
However the only way to tell **kbmMW** to grab the settings for the service definition from a configuration file, required you to either use the old (*non smart*) way of defining services and registering them, or it required you to lookup the service's service definition (**`TkbmMWCustomServiceDefinition`** *and descendants*) after the **`AutoRegisterServices`** call, and modify its properties.

AND THAT IS NOT SMART.
So of course **kbmMW** needs to be smart.
It should be as easy as possible to define your stuff, so you only have to focus on your business code and not on all the framework related stuff around it.

All settings based attributes will now have access to fetch configuration settings directly. Settings based attributes includes all attributes descending from **`TkbmMWCustomSettingsAttribute`**. **Currently that includes `kbmMW_Auth`, `kbmMW_Service`, `kbmMW_Rest`, `kbmMW_Config`, `kbmMW_HTTP`, `kbmMW_Ignore`, `kbmMW_Table` and `kbmMW_Field` attributes.**

Let us look at an example:

```
[kbmMW_Service('name:SMARTDEMO, version:1.0,
minInstances:$(service.smartDemo.minInstances=32),
maxInstances:$(service.smartDemo.maxInstances=128)')]
```

The shown attribute, placed before a **kbmMW service class definition**, tells **kbmMW** what the suggested name for the service is, which version it has, how many instances will minimum be spawned and how many will maximum be spawned.
As you can see, the **`minInstances`** and **`maxInstances`** settings has a "different" syntax than usual.
Instead of just containing a simple number, it now contains a **`$(configpath[=defaultvalue])`** formatted value.

Basically **`minInstances`** now looks in the default configuration manager *(referred to via the global variable Config*), for a value at the configuration path **`service.smartDemo.minInstances.`**
If one is found, then the read number will be used.
In the above example there is also a **default value (32),** in case there are no values at the configuration path.

Another example:

```
[kbmMW_Auth('role:SomeRole, grant:$(auth.service.grant=true)')]
```

This attribute can be placed multiple places, like **before methods** or as in this case, **before the service class definition**, where it defines whom have access to accessing the service.
Usually **`grant`** would have the value **`true`** or **`false`**, but in this case we detect the value from the configuration at the configuration path **`auth.service.grant`**. Default value is **`true.`**

I recommend using **the configuration manager** including these new features, rather than any homecooked configuration system, simply because it provides an application wide simple way to access and store configuration data in any of the formats supported by **kbmMW (Ini files, Windows registry, XML files, JSON files and YAML files in addition to a virtual configuration manager which for example could be used to store the configuration in a database**).

kbmMW contains some nice features to get easy access to a headers and cookies. For example via attributes:

```
[kbmMW_Service('name:SMARTDEMO')]
[kbmMW_Rest('path:/myserver')]
TMyService = class(TkbmMWCustomHTTPSmartService)
public
  [kbmMW_Rest('method:get, path: "echoheader", responseMimeType:"text/plain"')]
  function EchoHeader([kbmMW_Arg(mwatHeader,'Accept')] const AHeaderValue:string):string;
  [kbmMW_Rest('method:get, path: "echocookie", responseMimeType:"text/plain"')]
  function EchoCookie([kbmMW_Arg(mwatCookie,'MyCookie')] const ACookieValue:string):string;
end;

...

function TkbmMWCustomService2.EchoHeader(const AHeaderValue:string):string;
begin
   Result:=AHeaderValue;
end;

function TkbmMWCustomService2.EchoCookie(const ACookieValue:string):string;
begin
   Result:=ACookieValue;
end;
```

In the above example, accessing the server from a HTTP client like this:

http://localhost:1111/myserver/echoheader

will result in the return of the value of the Accept header, which in my case resulted in:

text/html,application/xhtml+xml,application/xml;q="0.9",image/webp,image/apng,*/*;q="0.8"

Similarly requesting echocookie will return the value of the cookie named MyCookie, that the browser may have sent to the server. In my case, because there is no cookie defined with that name in the browser, the result will be empty.

This way you can easily get access to numerous header and/or cookie values.

But in the cookie scenario, we did not find any cookie because none was defined. So how do we return a cookie to the web client, that it can send to us next time?

Let's modify the EchoCookie method to return the cookie it received from the web client (as before), but then explicitly either create a new cookie in the client or update its existing value.

```
function TMyService.EchoCookie(const ACookieValue:string):string;
var
 hlp:TkbmMWHTTPTransportStreamHelper;
 cookie:TkbmMWHTTPCookie;
begin
 Result:=ACookieValue;

 // Set and return a new cookievalue that basically contains current date time in ISO8601 format.
 hlp:=TkbmMWHTTPTransportStreamHelper(ResponseTransportStream.Helper);
 cookie:=TkbmMWHTTPCookie(hlp.Cookies.ValueByName['MyCookie']);
 if cookie=nil then
  cookie:=TkbmMWHTTPCookie(hlp.Cookies.New('MyCookie'));
 cookie.Value:=TkbmMWDateTime.Now.ISO8601String;
 cookie.Expires:=TkbmMWDateTime.Now.IncHour(1); // Expires one hour from now.
end;
```

This example introduced two classes, **TkbmMWHTTPTransportHelper,** and **TkbmMWHTTPCookie.**

With each call to a method in a service, two transport streams are made available, a request transport stream **(self.RequestTransportStream)** which contains what we received from the caller augmented with various data the application server sees fit to augment with, and a response transport stream (**self.ResponseTransportStream**) which will contain the data we want to return to the caller.

Part of the response transport stream will be the result of this method (a string) along with the response MimeType we defined using the attributes. However there are many other header type values that can be returned, like cookies and such (*when it is a response for a HTTP type client*).

Each transport stream has a helper attached, which makes certain specialized features easier available. When the transport stream format of the transport, has been set to REST (*which it is in our case*), we can rest assured that the helper must be a **TkbmMWHTTPTransportHelper** or a descendant of that. So casting it nonchalantly will work fine in this case, but if the EchoCookie has dual use, meaning it can be called from a non REST type client (which it could if we added the attribute [kbmMW_Method] in the interface section for that particular method, then we should actively test if the helper is indeed a TkbmMWHTTPTransportHelper before using it as such. That's not the case in our sample here, so let's continue.
Now we have access to a helper object (hlp). In that helper object you will find loads of nice things, like access to the Cookies property, where you can get access to all cookies provided by the web client (browser) if you are looking via the RequestTransportStream's helper, or access to all the cookies you want to return to the web client if you are looking via the ResponseTransportStream.
Thus we check if MyCookie exists. If it does not, then we create a new one which is automatically added. Afterwards we update its value and when the cookie expires.
So now when we first time call the echocookie REST call, we will again receive an empty result (still no cookies defined), but 2nd time we make the call, the ISO8601 timestamp of the first call is returned as the cookie value, and the cookie is then updated again and its expiration extended.
As you may have guessed by now, the hlp instance is also the entrance to accessing and manipulating header values.

Let us make a method that returns any header value for any header name we specify in the call:

```
[kbmMW_Rest('method:get, path:"echoanyheader/{AHeaderName}",
responseMimeType:"text/plain"')]
function EchoAnyHeader([kbmMW_Rest('value: "{AHeaderName}"')]
const AHeaderName:string):string;
```

And its implementation:

```
function TMyService.EchoAnyHeader(const AHeaderName:string):string;
var
  hlp:TkbmMWHTTPTransportStreamHelper;
begin
   hlp:=TkbmMWHTTPTransportStreamHelper(RequestTransportStream.Helper);
   Result:=hlp.Header.ValueByName[AHeaderName];
end;
```

It again cast to get a specialized helper, but now on the RequestTransportStream, and then return the value of a specific header item.

http://localhost:1111/myserver/echoanyheader/user-agent

Will in my case return

Mozilla/5.0 (Windows NT 10.0;Win64;x64) AppleWebKit/537.36 (KHTML,like Gecko) Chrome/70.0.3538.110 Safari/537.36

Some HTTP headers may contain qualifiers or options as part of the headers value. For example the Accept-Language header that often is sent with every browser request, and that tells the server which spoken language(s) the client will accept the response in (or more accurately, prefer the response in). Example:

Accept-Language: de

However if this value was provided by the browser, the server should send data localized into German if at all possible, else it should be localized into English.

Hence the Accept-Language is in fact a value that contains underlying fields (de and en) which in turn may have options attached to them (q).

kbmMW provides a way to access these (already parsed) values too:

```
function TMyService.EchoAnyHeader(const AHeaderName:string):string;
var
 i:integer;
 hlp:TkbmMWHTTPTransportStreamHelper;
 headervalue:TkbmMWHTTPMimeHeaderValue;
 field:TkbmMWHTTPMimeHeaderValueField;
 option:TkbmMWHTTPMimeHeaderValueFieldOption;
begin
 hlp:=TkbmMWHTTPTransportStreamHelper(RequestTransportStream.Helper);
 headervalue:=TkbmMWHTTPMimeHeaderValue(hlp.Header.ValueByName['Accept-Language']);
 if headervalue<>nil then
 begin
  // Access the de, en etc. fields of the header value.
  for i:=0 to headervalue.Fields.Count-1 do
  begin
    field:=TkbmMWHTTPMimeHeaderFieldValue(headervalue.Fields.Values[i]);
    // field.Name to access the fields name. Eg. de or en in this case.
    // field.AsString to get or set the textual representation of the field.
    for j:=0 to field.Options.Count-1 do
    begin
      option:=TkbmMWHTTPMimeHeaderValueFieldOption(field.Options.Values[i]);
      // option.Name is in this case 'q'
      // option.AsString to get or set the textual representation of the field.
      // In this case it could be 1.0 or 0.5 etc.
    end;
  end;
 end;
end;
```

As you can see, each level has the AsString and AsEncodedString property accessible, which makes it possible to set or get the textual representation of that particular value in the mime header value hierarchy.

If you like kbmMW, share the word. Reshare the blog posts and let others know about the product!

Essentially help us to help you.

The upcoming kbmMW update which were supposed to be a minor bug-fix update, will however also contain a new major feature. A client stub generator framework.

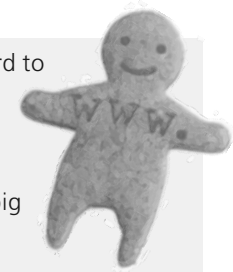
What is a client stub generator framework?

It is a framework which, on the basis of smart services, can generate code that can be used directly by clients of various types, to access the kbmMW based application servers HTTP smart services.

Currently kbmMW already contains a smart client feature, which makes it very easy to write native kbmMW clients that can access smart service based features. Since the smart client relies on late binding, the developer however do not get much IDE and compiler assistance regarding arguments and their types. For that to happen, the compiler will need some generated code that explains the exposed server side functions and methods for the compiler and the IDE.

This is where a stub generator comes into play. This article will however not talk about how to generate Delphi client side stubs for smart services, as that particular feature will probably not make it into the upcoming update, although preparations for it is already in the code.

Instead I have focused on utilizing the stub generator framework to implement another even more complex set of code, that fulfills a typical requirement, found in the REST producing world. I particularly recognized it well over a year ago, when I was consulting large companies doing Java code.



In the Java world, it is a defacto standard to document REST interfaces using something that, at the time, was called Swagger. Later it has been renamed to OpenAPI, which is now recognized and supported by the absolute majority of big players supporting REST.

OpenAPI provides a description of your REST interface, which can be used both for documentation, but also for automatically generating (stub) code for all sorts of development environments, to make it easy for those environments to utilize the features published via the REST interface.

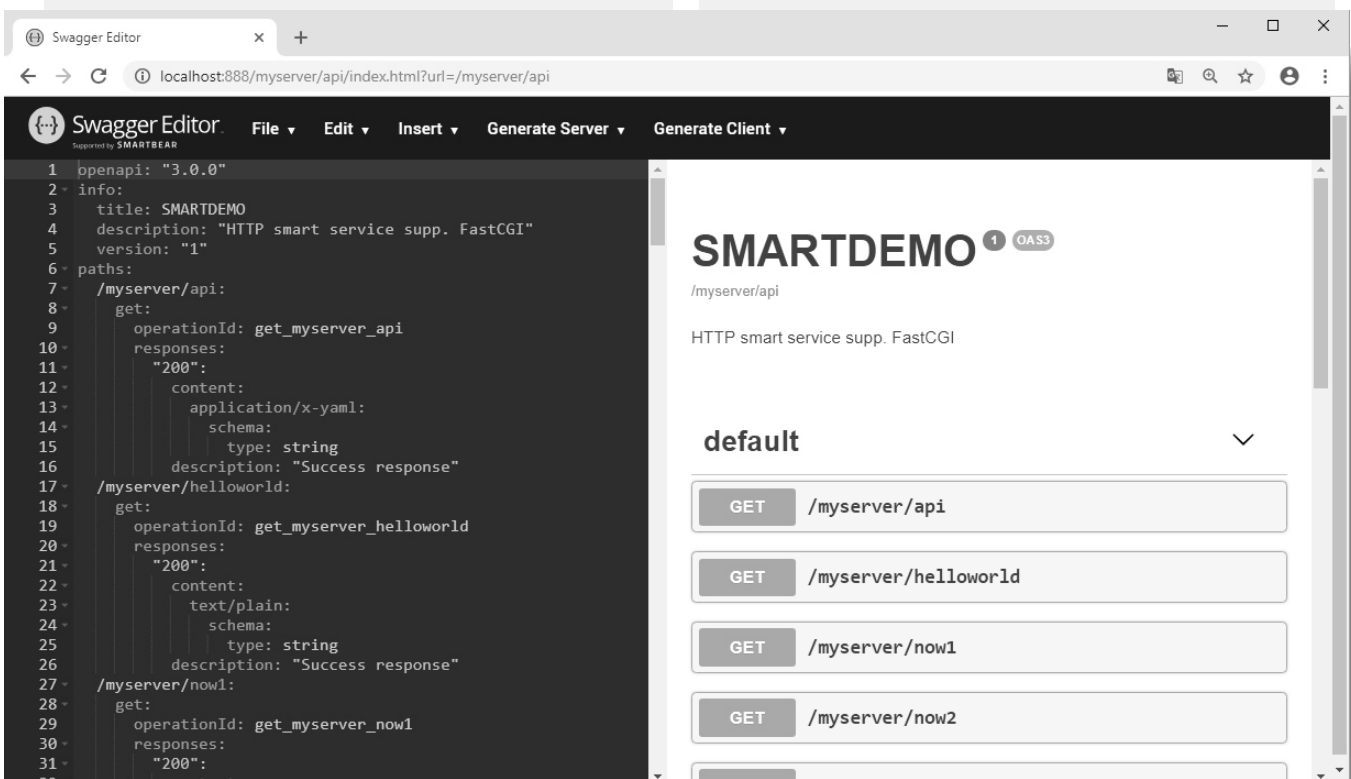
You can read much more about OpenAPI here:
`https://swagger.io/`

The OpenAPI initiative has not only produced a defacto standard but also various tools one can use to generate, edit, view and test REST interface descriptions (in common terms called Swagger files).

One such tool is the Swagger-UI (user interface) which consists of Javascript and HTML which can be served by a web server to provide a simple to use user interface to the REST interfaces that has been exposed in the server.

kbmMW now supports all this fully.

Let's simply start with showing how the REST interface of the kbmMW SimpleInvocation demo server looks like, using Swagger-UI:



POST /myserver/echobytes

GET /myserver/echoreversedconfigstring

GET /someabspath/addnumbers Adds two numbers and returns result

Parameters Try it out

Name	Description
arg1 * required integer *(query)*	First numeric argument
arg2 * required integer *(query)*	Second numeric argument

Responses

Code	Description	Links
200	*The result of the added numbers* text/plain Controls Accept header.	*No links*

POST /myserver/storeperson

On the left, one can see an OpenAPI declaration of the exposed REST interfaces, and on the right one can see a user friendly interface where those interfaces can be called by simple button clicks. It is even easy to fill arguments and such that the REST method requires.

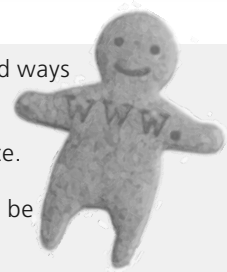
If I scroll the right side down to the **`AddNumbers`** method, and click the bar, it opens up with additional information, and a button that let us try the REST call.

This is imo pretty cool

So how do we OpenAPI enable a REST capable application server?

It is really easy.

To return the OpenAPI specification of the REST services, we simply add another REST exposed method to the service in Unit2.



The reason is that there are two valid ways to generate OpenAPI specifications for REST interfaces that takes or returns objects, inline or by reference. Inline means that each object is explained in detail every place it can be used in all descriptions for the REST calls, while by reference means that OpenAPI style components (objects) are described and referenced where needed.

```
[kbmMW_Rest('method:get, path: "api", responseMimeType:"application/x-yaml"')]
  function OpenAPI:string;
```

We can name the method and its REST path anything you like, but we should provide a correct responseMimeType. The standard for OpenAPI descriptions is to be expressed in YAML, which kbmMW fortunately fully supports. It is also allowed to produce OpenAPI descriptions in JSON, but it is more a way to be compatible with systems that do not support YAML. So in this example, we specify in the mimetype that the response will be YAML.

By reference is the default. However in the sample code, I have chosen to let it be configurable with the help of the kbmMW configuration framework. We could have written: inline:false or inline:true, but the sample instead asks the configuration for the current value of the OpenAPI.inline value that can be either true or false. If no such value is found, kbmMW will use the default value of false (as indicated after the =).

One could also have specified that JSON is preferred. That just requires adding json:true to the settings string like this:

```
// Return OpenAPI specification.
function TkbmMWCustomService2.OpenAPI:string;
begin
  // Return OpenAPI specification for all REST methods in this service
  // as YAML. Add the ASettings value: 'json:true' to return the specification
  // as JSON.
  // Add 'servers: [ "url1", "url2",.. "urln" ]' to ASettings if you want to
  // embed server location information in the specification.
  // Add 'inline:true' to inline object definitions instead of using $ref.
  // The example in the next line utilize the configuration framework to make
  // the setting easily configurable.
  Result:=TkbmMWSmartOpenAPIStubGenerator.GenerateOpenAPI('',self,'inline:$(penAPI.inline=false)');
end;
```

The code in the OpenAPI function is pretty simple. It just calls the GenerateOpenAPI method of the OpenAPI stub generator with the service to be "OpenAPI'ified" and optionally a settings string. The settings string can be empty

In this sample, the settings string contains the value

inline:$(OpenAPI.inline=false)

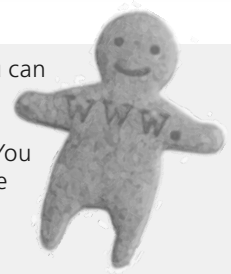
That could obviously also have been made configurable the same way as inline.

However we leave it as is, so the output of the OpenAPI function will be a YAML formatted OpenAPI description of our REST methods in the service.

Hence if we use a browser to open the URL: //localhost:888/myserver/api we will get the complete OpenAPI description:

inline:$(OpenAPI.inline=false), json:true

```
openapi: "3.0.0"
info:
 title: SMARTDEMO
 description: "HTTP smart service supp.
FastCGI"
 version: "1"
paths:
 /myserver/api:
  get:
   operationId: get_myserver_api
   responses:
    "200":
     content:
      application/x-yaml:
       schema:
        type: string
     description: "Success response"
 /myserver/helloworld:
  get:
   operationId: get_myserver_helloworld
   responses:
    "200":
     content:
      text/plain:
       schema:
        type: string
     description: "Success response"
 /myserver/now1:
  get:
   operationId: get_myserver_now1
   responses:
    "200":
     content:
      text/plain:
       schema:
        type: string
        format: date-time
     description: "Success response"
 /myserver/now2:
  get:
   operationId: get_myserver_now2
   responses:
    "200":
     content:
      text/plain:
       schema:
        type: number
        format: double
     description: "Success response"
 /myserver/echostring/{AString}:
  get:
   operationId:
get_myserver_echostring__AString_
   parameters:
    -
     in: path
     name: AString
     required: true
     schema:
      type: string
   responses:
    "200":
     content:
      text/plain:
       schema:
        type: string
     description: "Success response"
```



This is only a piece of the code , you can download the code to study it from your downloadpage.
Feel free to study it if you want to. You may notice that several places, there are facilities to add summaries and descriptions. For example check the addnumbers call:

```
/someabspath/addnumbers:
 get:
  summary: "Adds two numbers and returns result"
  operationId: add_numbers
  parameters:
   -
    in: query
    name: arg1
    required: true
    description: "First numeric argument"
    schema:
     type: integer
     format: int32
   -
    in: query
    name: arg2
    required: true
    description: "Second numeric argument"
    schema:
     type: integer
     format: int32
  responses:
   "200":
    content:
     text/plain:
      schema:
       type: integer
       format: int32
    description: "The result of the added numbers"
```

Where does the summary and descriptions come from?

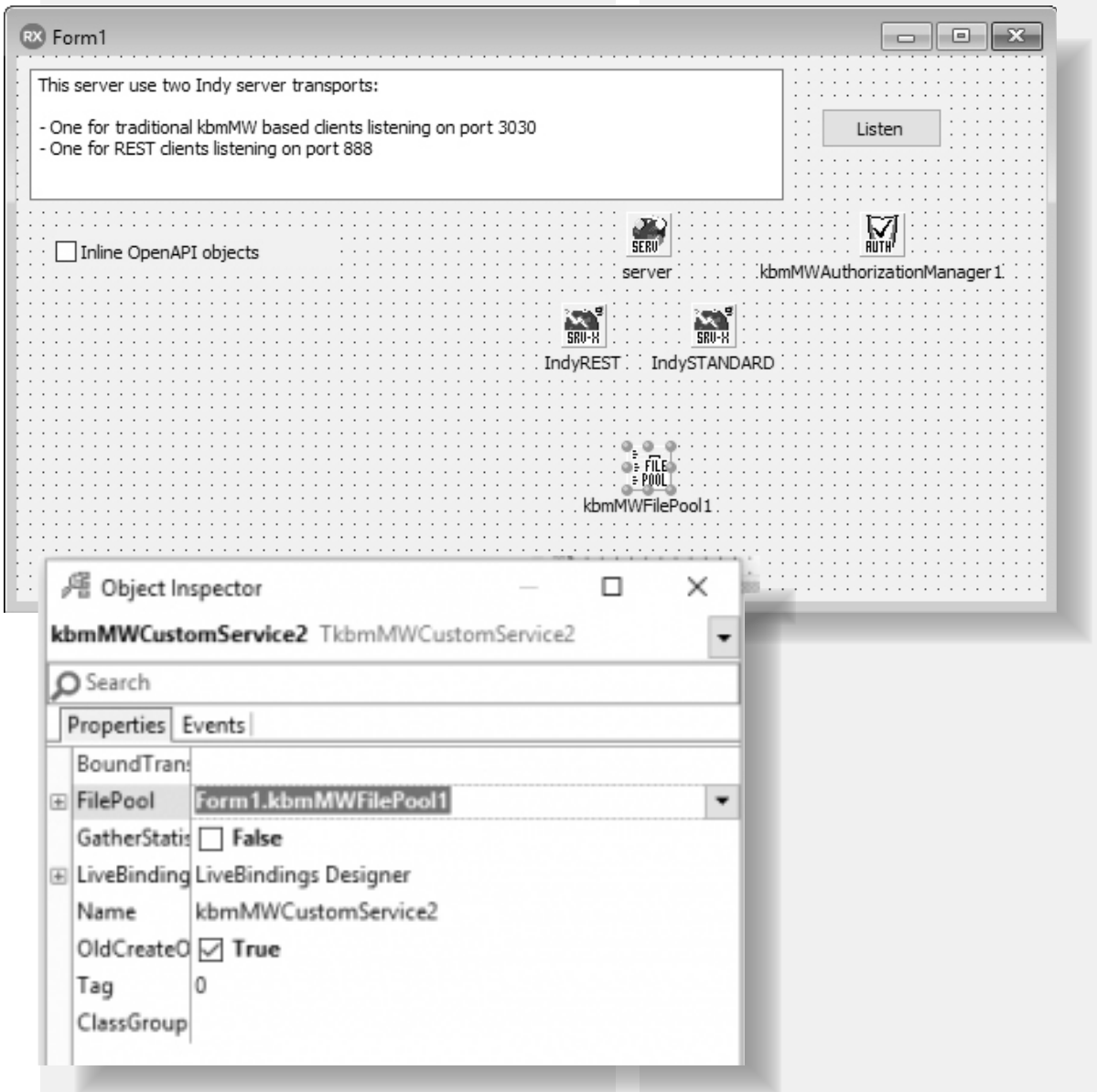
Looking at the definition of the AddNumbers function in Unit2.pas, it becomes obvious:

```
 // Add two numbers.
// It can be called from regular clients, smart clients
// and REST clients.
// It can be called from a browser like this:
// http://.../someabspath/addnumbers?arg1=10&arg2=20
[kbmMW_Method]
[kbmMW_Rest('method:get, path: "/someabspath/addnumbers", '+
      'id:"add_numbers", '+
      'summary:"Adds two numbers and returns result", '+
      'resultDescription:"The result of the added numbers"')]
function AddNumbers([kbmMW_Rest('value: "$arg1",
required: true, description:"First numeric argument"')]
      const AValue1:integer;
           [kbmMW_Rest('value: "$arg2", required: true,
      description:"Second numeric argument"')]
            const AValue2:integer;
           [kbmMW_Arg(mwatRemoteLocation)]
            const ARemoteLocation:string):integer;
```

There is even an id value in the kbmMW_Rest attribute. OpenAPI requires that each REST path must have a unique ID. kbmMW will automatically attempt to generate one, but you can choose your own ID names via the id syntax as seen above. id, summary, description and resultDescription are (in kbmMW) all optional. If OpenAPI require a descriptive value and none has been given, kbmMW provides a default.

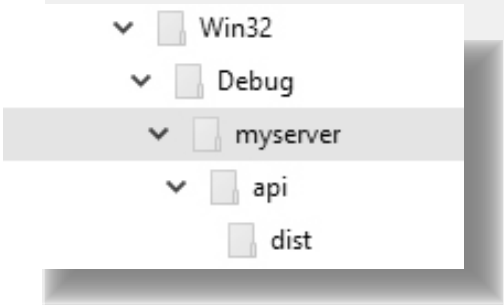
So now we can generate valid OpenAPI descriptions. How do we get around to letting our kbmMW based application server present them using Swagger-UI

Because kbmMW can act as a web server, this is actually also pretty easy. We simply add a TkbmMWFilePool instance to the main form (Unit1) and set the FilePool property of the service datamodule (Unit2) to point to it.

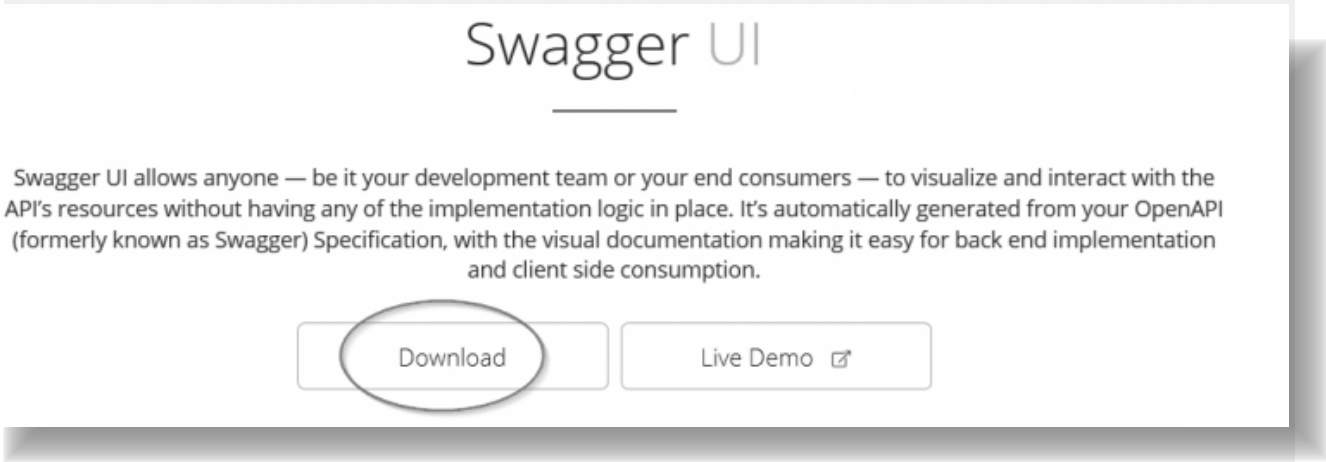


Now kbmMW will work as a regular web server, and attempt to serve a file when it does not find a REST function to be called.

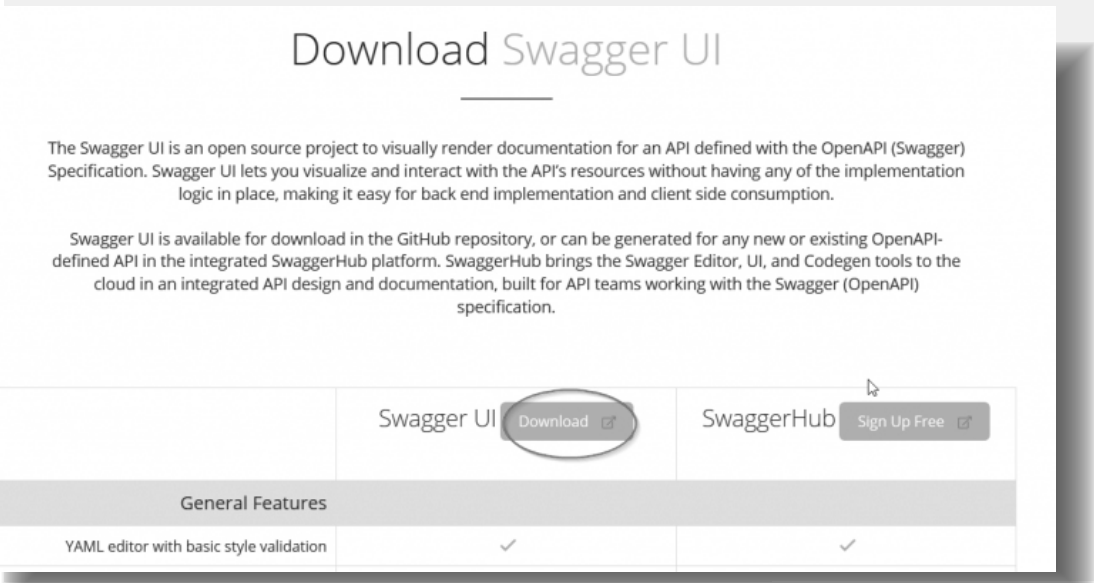
In the same directory as the SimpleInvocation server executable, you should create a directory called MyServer (to match the service) and under it, we add an api directory, both just to match the logical path of the function OpenAPI in Unit2. In fact you do not have to use this specific path hierarchy but I have chosen so for the demo.



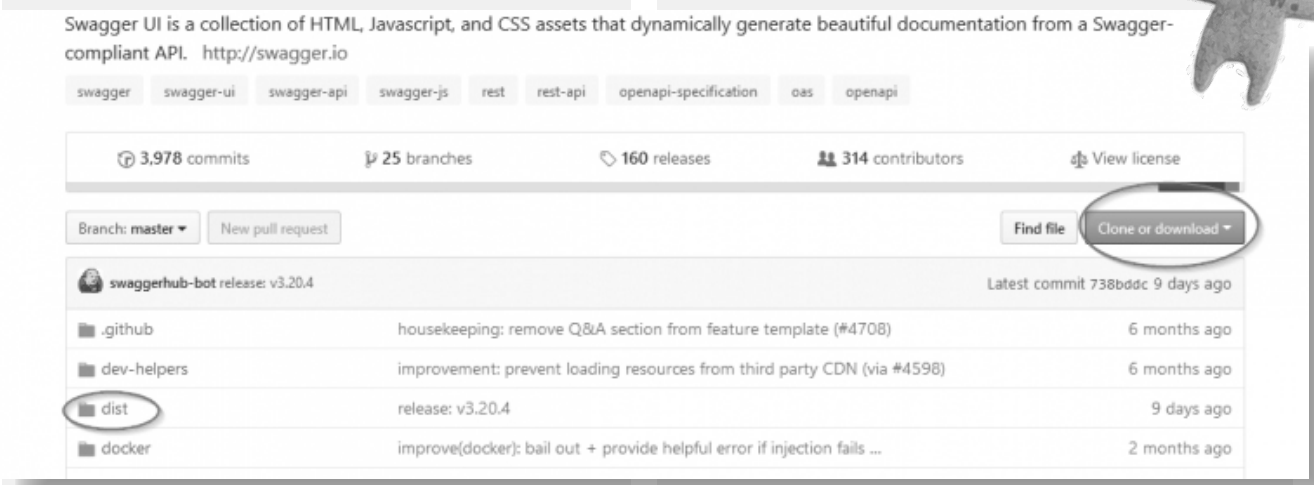
In the api directory we will put files downloaded from https://swagger.io/tools/swagger-ui/



Then



And



You can either download all the files from the dist folder, one by one to the api\dist directory, or you can download everything via the Clone or download button. If you do the later, then open the downloaded zip file and extract the dist folder with contents to the api directory.

Finally add a file called index.html to the api directory. You can copy/paste its contents from this:

```
<!DOCTYPE html>
<!-- HTML for static distribution bundle build -->
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Swagger Editor</title>
  <style>
  * {
    box-sizing: border-box;
  }
  body {
    font-family: Roboto,sans-serif;
    font-size: 9px;
    line-height: 1.42857143;
    color: #444;
    margin: 0px;
  }

  #swagger-editor {
    font-size: 1.3em;
  }

  .container {
    height: 100%;
    max-width: 880px;
    margin-left: auto;
    margin-right: auto;
  }

  #editor-wrapper {
    height: 100%;
    border:1em solid #000;
    border:none;
  }

  .Pane2 {
    overflow-y: scroll;
  }
```

```
  </style>
  <link href="./dist/swagger-editor.css"
rel="stylesheet">
  <link rel="icon" type="image/png"
href="./dist/favicon-32x32.png" sizes="32x32"
/>
  <link rel="icon" type="image/png"
href="./dist/favicon-16x16.png" sizes="16x16"
/>
</head>

<body>
  <div id="swagger-editor"></div>
  <script src="./dist/swagger-editor-
bundle.js"> </script>
  <script src="./dist/swagger-editor-
standalone-preset.js"> </script>
  <script>
  window.onload = function() {
    // Build a system
    const editor = SwaggerEditorBundle({
      dom_id: '#swagger-editor',
      layout: 'StandaloneLayout',
      presets: [
        SwaggerEditorStandalonePreset
      ]
    })

    window.editor = editor
  }
  </script>
```

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
style="position:absolute;width:0;height:0">
   <defs>
     <symbol viewBox="0 0 20 20" id="unlocked">
          <path d="M15.8 8H14V5.6C14 2.703 12.665 1 10 1 7.334 1 6 2.703 6 5.6V6h2v-.801C8 3.754
     8.797 3 10 3c1.203 0 2 .754 2 2.199V8H4c-.553 0-1 .646-1 1.199V17c0 .549.428 1.139.951
     1.30711.197.387C5.672 18.861 6.55 19 7.1 19h5.8c.549 0 1.428-.139 1.951-.30711.196-
    .387c.524-.167.953-.757.953-1.306V9.199C17 8.646 16.352 8 15.8 8z"></path>
     </symbol>

     <symbol viewBox="0 0 20 20" id="locked">
       <path d="M15.8 8H14V5.6C14 2.703 12.665 1 10 1 7.334 1 6 2.703 6 5.6V8H4c-.553 0-1 .646-1
     1.199V17c0 .549.428 1.139.951 1.30711.197.387C5.672 18.861 6.55 19 7.1 19h5.8c.549 0
     1.428-.139 1.951-.30711.196-.387c.524-.167.953-.757.953-1.306V9.199C17 8.646 16.352
     8 15.8 8zM12 8H8V5.199C8 3.754 8.797 3 10 3c1.203 0 2 .754 2 2.199V8z"/>
     </symbol>

     <symbol viewBox="0 0 20 20" id="close">
       <path d="M14.348 14.849c-.469.469-1.229.469-1.697 0L10 11.8191-2.651 3.029c-.469.469-
      1.229.469-1.697 0-.469-.469-.469-1.229 0-1.69712.758-3.15-2.759-3.152c-.469-.469-
      .469-1.228 0-1.697.469-.469 1.228-.469 1.697 0L10 8.18312.651-3.031c.469-.469 1.228-.469
       1.697 0 .469.469.469 1.229 0 1.6971-2.758 3.152 2.758 3.15c.469.469.469 1.229 0 1.698z"/>
     </symbol>

     <symbol viewBox="0 0 20 20" id="large-arrow">
       <path d="M13.25 10L6.109 2.58c-.268-.27-.268-.707 0-.979.268-.27.701-.27.969 017.83
     7.908c.268.271.268.709 0 .9791-7.83 7.908c-.268.271-.701.27-.969 0-.268-.269-.268-.707
     0-.979L13.25 10z"/>
     </symbol>

     <symbol viewBox="0 0 20 20" id="large-arrow-down">
       <path d="M17.418 6.109c.272-.268.709-.268.979 0s.271.701 0 .9691-7.908 7.83c-.27.268-
       .707.268-.979 01-7.908-7.83c-.27-.268-.27-.701 0-.969.271-.268.709-.268.979 0L10
       13.2517.418-7.141z"/>
     </symbol>


     <symbol viewBox="0 0 24 24" id="jump-to">
       <path d="M19 7v4H5.83l3.58-3.59L8 6l-6 6 6 6 1.41-1.41L5.83 13H21V7z"/>
     </symbol>

     <symbol viewBox="0 0 24 24" id="expand">
       <path d="M10 18h4v-2h-4v2zM3 6v2h18V6H3zm3 7h12v-2H6v2z"/>
     </symbol>

   </defs>
 </svg>

</body>

</html>
```

Now you are ready to rock and roll. Start the server. Then start a browser and type:
`http://localhost:888/myserver/api/index.html?url=/myserver/api`

This instructs kbmMW to serve the index.html file, which in turn requests remaining files needed for producing the Swagger-UI interface. Finally we tell Swagger-UI to load the OpenAPI description from the /myserver/api URL (*which, if you remember, will call the function OpenAPI in Unit2*).

You should now end up with a view similar to what was shown in start of this blog post.
From within the Swagger-UI interface, you can generate both server skeletons and client stubs for all the REST features exposed by kbmMW.

Happy Swagging

If you like kbmMW, share the word. Reshare the blog posts and let others know about the product!

Essentially help us to help you

- ADVERTISEMENT -

COMPONENTS 4 DEVELOPERS

We are happy to announce v5.08.10 of our popular middleware for Delphi and C++Builder.

We strive hard to ensure **kbmMW** continues to set the bar for what an n-tier product must be capable of in the real world!

Notice that kbmMemTable v. 7.81.00 or newer is a prerequisite to this update.
This release is primarily a bugfix release.

Professional and Enterprise Edition is available for all with a current active SAU. If your SAU has run out, please visit our shop to extend it with another 12 months.

CodeGear Edition may be available for free,
but only supports a specific Delphi/Win32 SKU,
contains a limited feature set and does not include source.

Please visit `https://portal.components4developers.com` to download.

—-

kbmMW is the premiere n-tier product for Delphi, C++Builder and FPC on .Net, Win32, Win64, Linux, Java, PHP, Android, IOS, embedded devices, websites, mainframes and more.

Please visit `www.components4developers.com` for more information about **kbmMW**.

—-

Components4Developers is a company established in 1999 with the purpose of providing high quality development tools for developers and enterprises. The primary focus is on SOA, EAI and systems integration via our flagship product kbmMW.

kbmMW is a portable, highly scalable, high end application server and enterprise architecture integration (EAI) development framework for Win32, ..Net and Linux with clients residing on Win32, .Net, Linux, Unix, Mainframes, Minis, Embedded and many other places. It is currently used as the backbone in hundreds of central systems, in hospitals, courts, private, industries, offshore industry, finance, telecom, governments, schools, laboratories, rentals, culture institutions, FDA approved medical devices, military and more.

NEW IN THIS VERSION

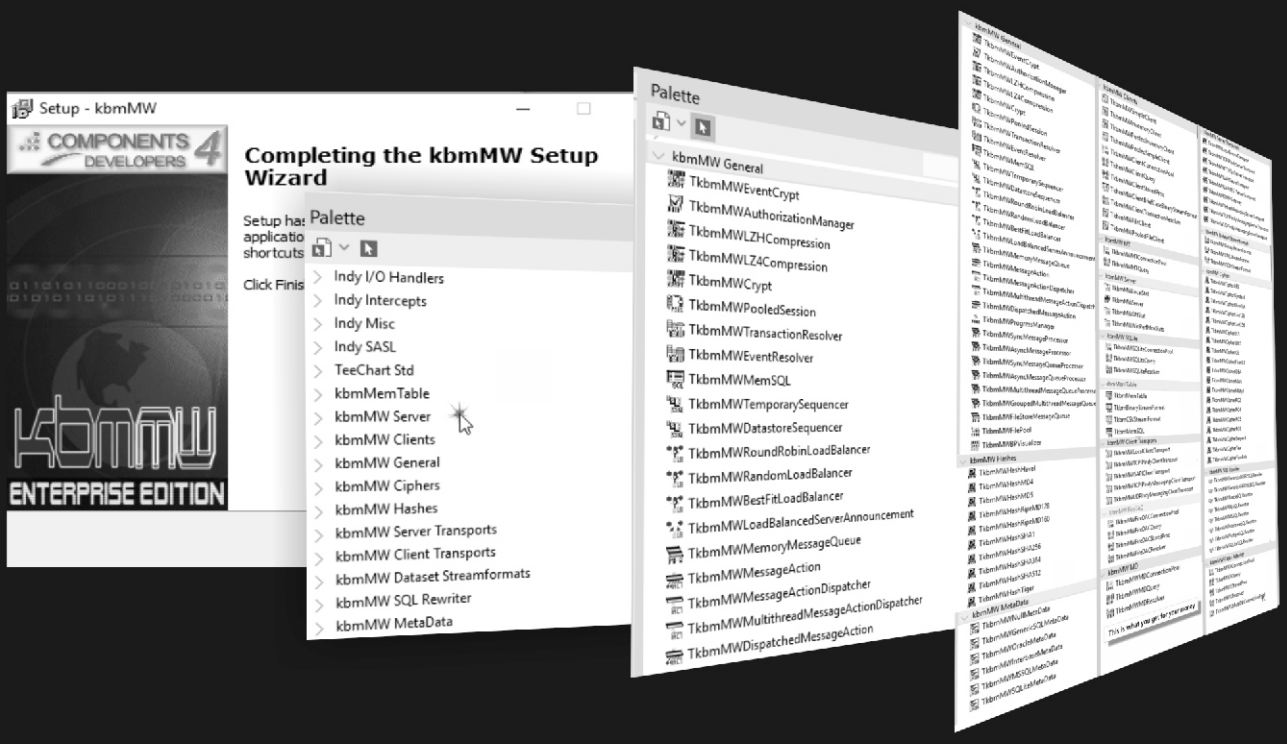
- Added multiple overloaded variants of kbmMWBackupFile method in kbmMWGlobal.pas to allow for more features.
- Added boolean properties Modified and Normalized to TkbmMWDuration to make it easy to figure out if a duration normalization needs to be done.
- Added support for MaxBackupCount in TkbmMWLocalFileLogManager.

Changes/minor additions

- Changed TkbmMWScheduledEvent.GetIntervalMSecs's algorithm in calculating minimum required probe (sampling) interval.
- Added class function GetIsManaged(const AValue:TValue):boolean;
  class function GetIsInterface(const AValue:TValue):boolean;
  class function GetIsProbablyRefCountedObject(const AObject:TObject):boolean;
  to TkbmMWRTTI (kbmMWRTTI.pas).
- Changed TkbmMWLocalFileLogManager backup handling to use kbmMWBackupFile instead of its own scheme.
  That includes adding features to specify max number of log backups via the property MaxBackupCount. Default there is no limit.
- Added NormalizedInterval to TkbmMWScheduledEvent which returns a normalized interval (TkbmMWDuration).
- Added ProbeIntervalMSecs to TkbmMWScheduledEvent which returns the minimum needed probe (sampling) interval (in msecs) to live up to the scheduled events interval settings.

Fixes

- Fixed TkbmMWScheduler.WaitRuns not waiting at least one run.
  Amongst others, affected SameFile in TkbmMWFileClient.
- Fixed compilation in older SKU's. Tested with XE5.
- Fixed compilation for Android.
- Fixed some SmartService leak scenarios.
- Fixed rounding problem in TkbmMWDuration.Normalize method resulting in slightly off normalized msecs value.
- Fixed TkbmMWScheduledEvent.IsTimeToExecute issue resulting in sometimes wandering outside scheduled second intervals due to interval creep.
- Fixed some demos.



KBMMW PROFESSIONAL AND ENTERPRISE EDITION

V. 5.08.10 BETA RELEASED!

NEW! OPENAPI, SWAGGERUI AND DELPHI CLIENT STUB SUPPORT!

- RAD Studio XE2 to 10.3 Rio supported
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 and OSX client and server support
- Native high performance 100% developer defined application server
- Full support for centralized and distributed load balancing and failover
- Advanced ORM/OPF support including support of existing databases
- Advanced logging support
- Advanced configuration framework
- Advanced scheduling support for easy access to multithread programming
- Advanced smart service and clients for very easy publication of functionality
- High quality random functions.
- High quality pronouncable password generators.
- High performance LZ4 and Jpeg compression
- Complete object notation framework including full support for YAML, BSON, Messagepack, JSON and XML
- Advanced object and value marshalling to and from YAML, BSON, Messagepack, JSON and XML
- High performance native TCP transport support
- High performance HTTPSys transport for Windows.
- CORS support in REST/HTML services.
- Native PHP, Java, OCX, ANSI C, C#, Apache Flex client support!
- High speed, unified database access (35+ supported database APIs) with connection pooling, metadata and data caching on all tiers
- Multi head access to the application server, via REST/AJAX, native binary, Publish/Subscribe, SOAP, XML, RTMP from web browsers, embedded devices, linked application servers, PCs, mobile devices, Java systems and many more clients
- Complete support for hosting FastCGI based applications (PHP/Ruby/Perl/Python typically)
- Native complete AMQP 0.91 support (Advanced Message Queuing Protocol)
- Complete end 2 end secure brandable Remote Desktop with near realtime HD video, 8 monitor support, texture detection, compression and clipboard sharing.
- Bundling kbmMemTable Professional which is the fastest and most feature rich in memory table for Embarcadero products.

kbmMemTable is the fastest and most feature rich in memory table for Embarcadero products.

- **Easily supports large datasets with millions of records**
- **Easy data streaming support**
- **Optional to use native SQL engine**
- **Supports nested transactions and undo**
- **Native and fast build in M/D, aggregation/grouping, range selection features**
- **Advanced indexing features for extreme performance**

COMPONENTS 4 DEVELOPERS

DX

EESB, SOA,MoM, EAI TOOLS FOR INTELLIGENT SOLUTIONS. kbmMW IS THE PREMIERE N-TIER PRODUCT FOR DELPHI / C++BUILDER