

# tmssoftware.com

## Modern web development with TMS WEB Core

### develop • faster



 [facebook.com/tmssoftware](https://facebook.com/tmssoftware)  
 [@tmssoftwarenews](https://twitter.com/tmssoftwarenews)  
 [youtube.com/tmssoftwareTV](https://youtube.com/tmssoftwareTV)

# Overview

## **Part 1: Classification, introduction, setup and the first app with TMS WEB Core**

- Web applications as a modern business application
- Single page application as a modern web application
- Technology and architecture of TMS WEB Core
- Installation of TMS WEB Core
- First application with TMS WEB Core
- Components - An overview

Part 2: Application Development and Features of TMS WEB Core

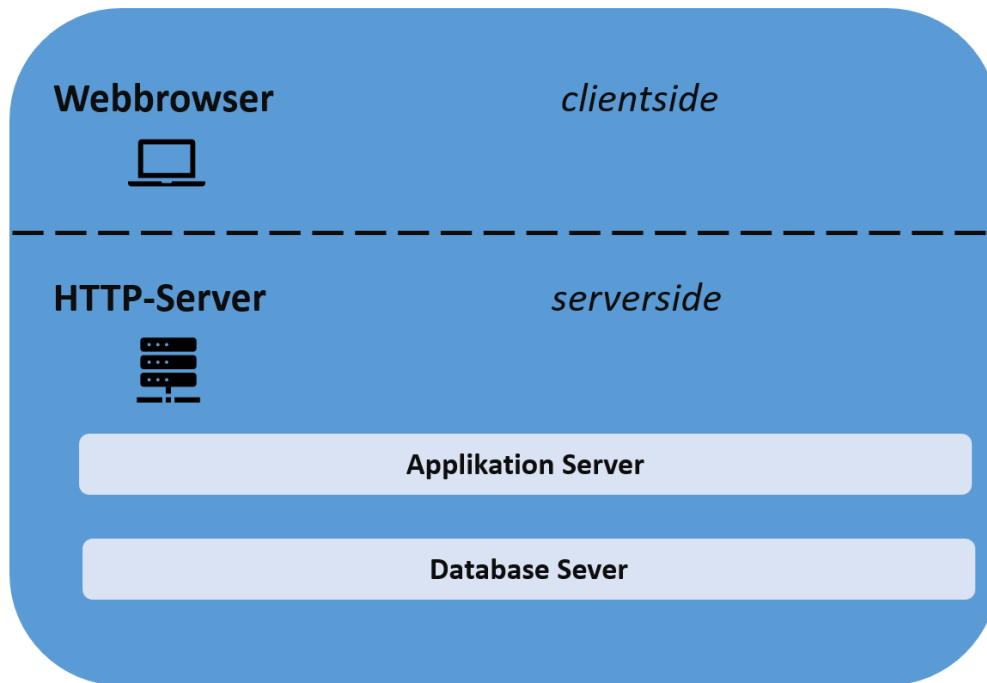
Part 3: Web, Mobile and Desktop Applications with TMS WEB Core

*Note: This set of slides is intended for users of the integrated development environment Delphi. Alternatively, if you would like to use Visual Studio Code, then download the appropriately adapted set of slides.*

# Advantages of Web Applications

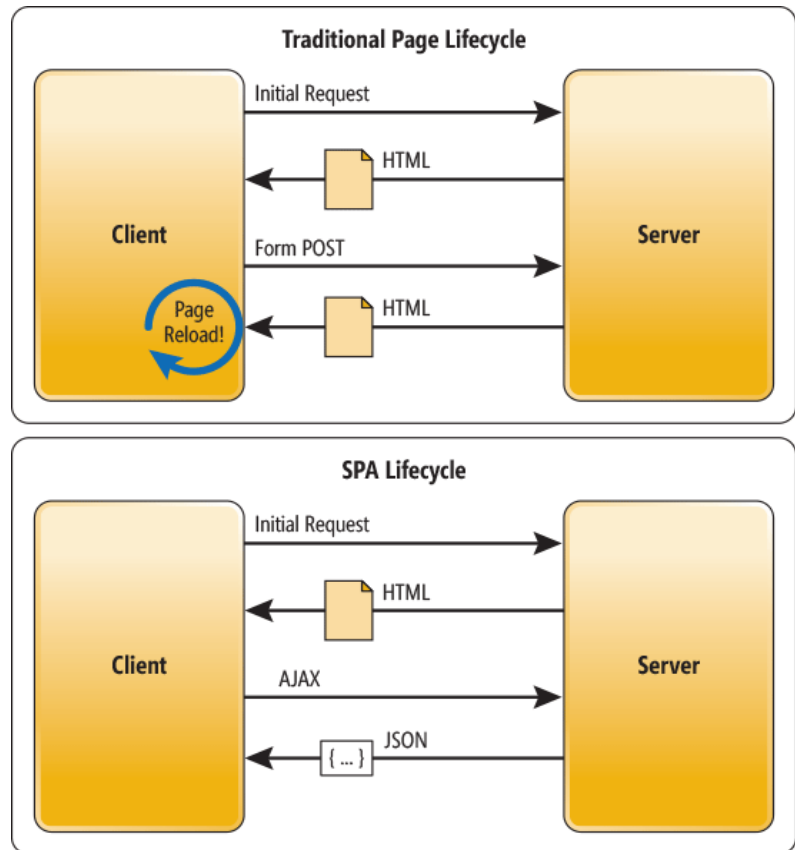
- central installation and execution on the server
- multi-user operation
- lower hardware requirements than when running natively on the client
- can be used across platforms and devices, as it is executed in the browser
- user experience is now largely comparable to desktop applications
- only a few restrictions on system and hardware access
- offline operation is now also possible thanks to PWA technology (Progressive Web App).

# Web Architecture



- the web architecture is a special form of the client-server architecture
- differentiation between client-side and server-side web applications
- server-side web applications: complete program logic is processed on the server and sent to the client (browser) as HTML/CSS
- client-side web applications: some or all of the program logic is executed in the browser using JavaScript

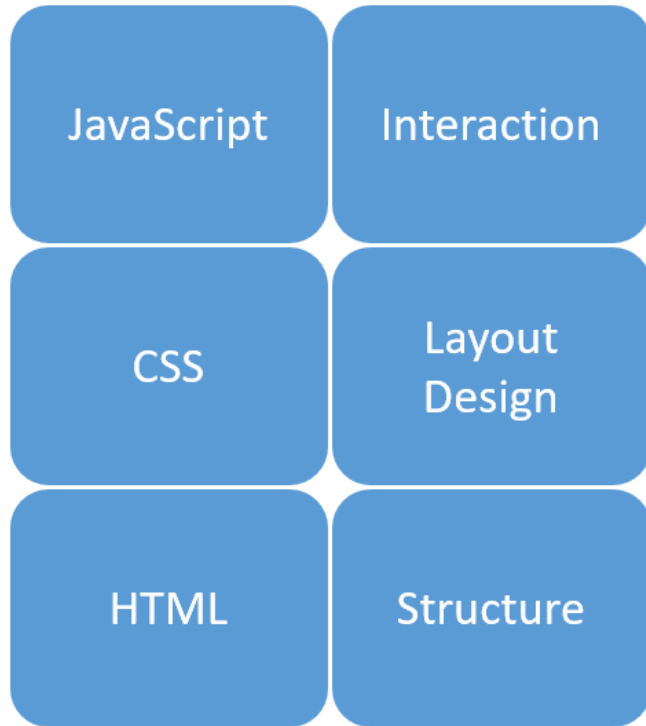
# SPA – Modern Web App



- SPA stands for Single Page Application
- client-side web application
- logic in the browser instead of on the server
- changes in the view are made by manipulating the DOM
- complete reload of the page is not necessary
- responsive
- good user experience

Source: <https://docs.microsoft.com/en-us/archive/msdn-magazine/2013/november/asp-net-single-page-applications-build-modern-responsive-web-apps-with-asp-net>

# Technology of the Web



```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" href="/css/style.css">
  </head>
  <body>
    ...
    ...
    <script type="text/javascript" src="scripts.js"></script>
  </body>
</html>
```

# Web Application: Developer Perspective

- technologies: HTML, CSS and JavaScript
- integration of a large number of libraries
- diverse combination of libraries and frameworks
- free choice of tooling, i.e. development environment, editor, etc.
- difficult to learn, because there are a lot of different technologies
- design of the user interface usually code-based
- development efficiency is therefore often not as high as when using highly integrated development environments for desktop applications

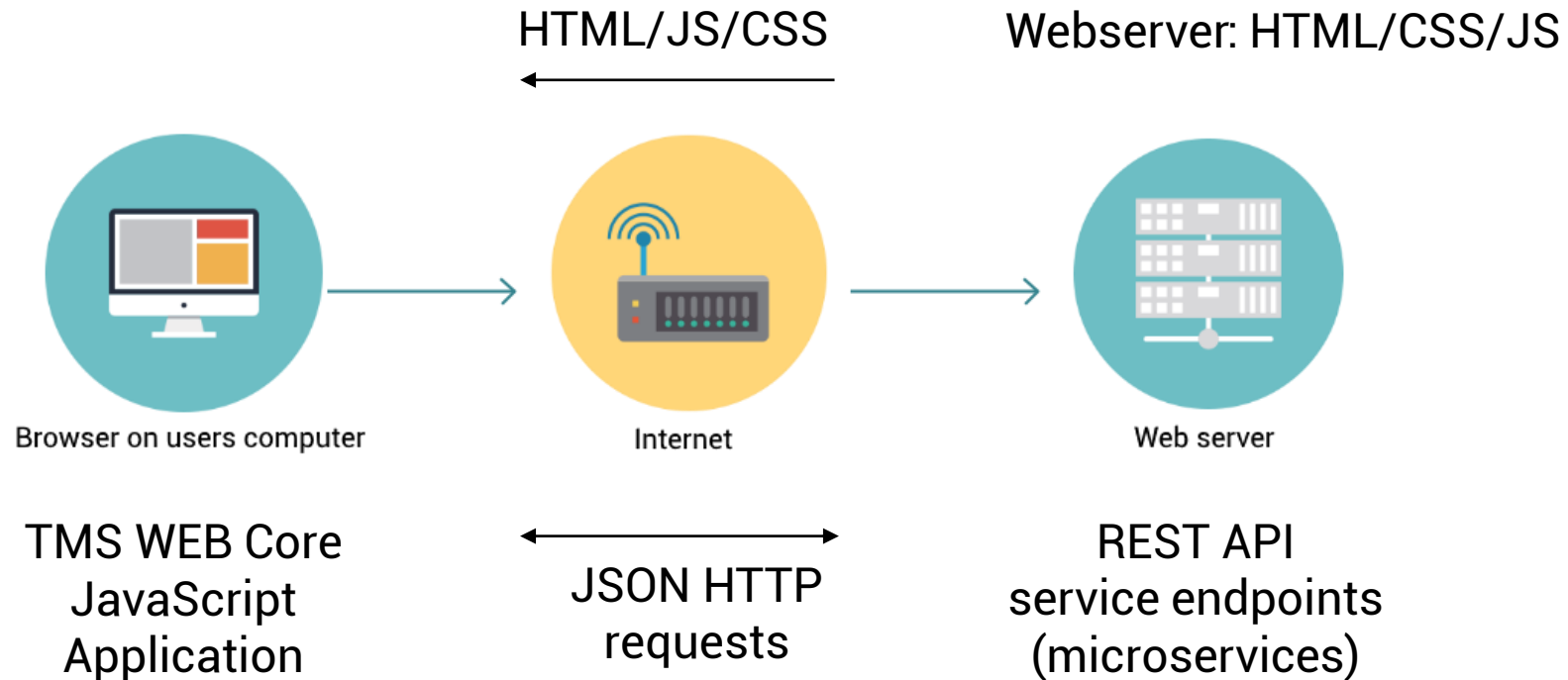
# Technologies for Web Applications

- basic technologies HTML, CSS, JavaScript
- libraries e.g. Bootstrap for layout and design
- libraries e.g. React for component-based development
- frameworks e.g. Angular for complex application structures
- PHP as server-side scripting language
- TypeScript as an alternative programming language for JavaScript (translation via transpiler)
- ...
- TMS WEB Core as a complete development system based on Object Pascal, component based, IDE support, visual designer

# System Requirements

- integrated development environment Delphi, Visual Studio or Lazarus
- operating system according to the development environment
- web server: Internet Information Service, Apache, node.js, Nginx
- web browser: all browsers that process HTML5 are suitable
  - Windows: Google Chrome, Microsoft Edge, Mozilla Firefox, Opera (limitations), Internet Explorer (limitations)
  - macOS: AppleSafari
  - iOS: AppleSafari
  - Android: Chrome

# Dataflow – TMS WEB Core

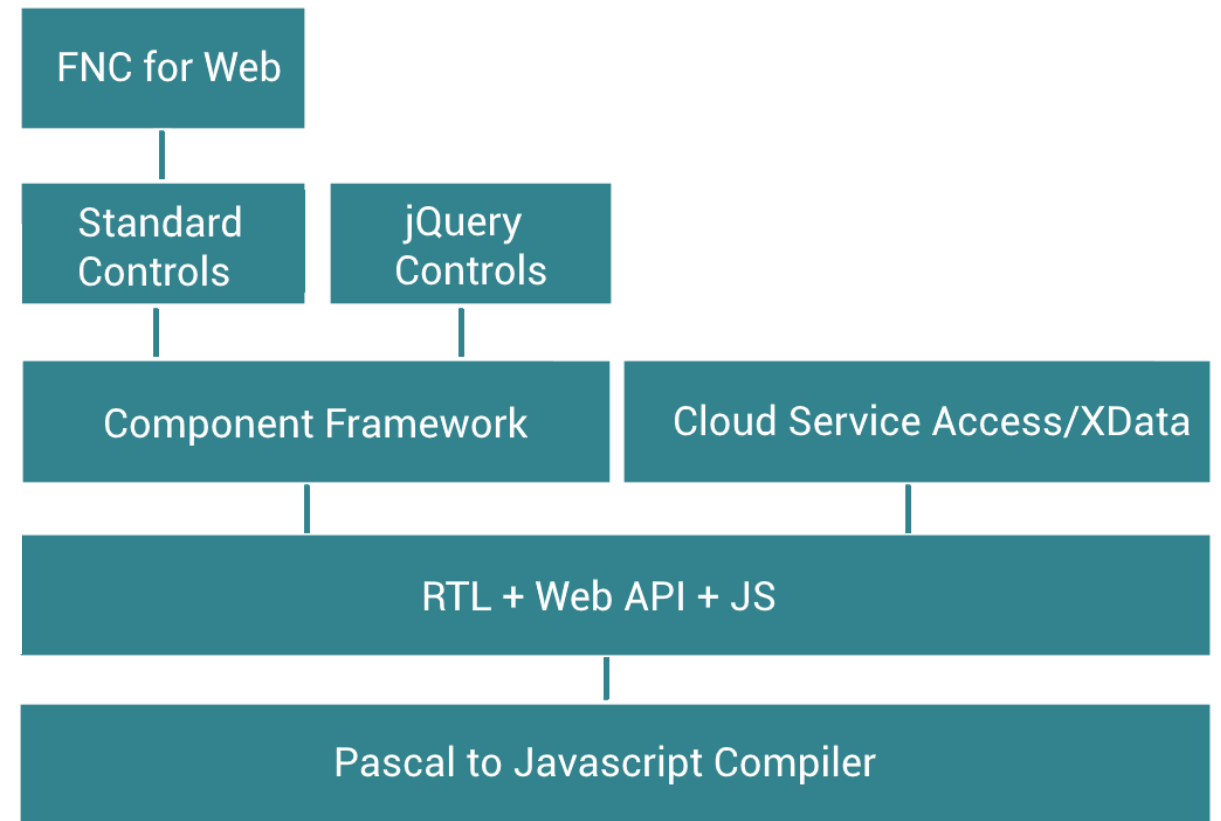


Source: TMS, <https://www.tmssoftware.com/>

# Architecture – TMS WEB Core

Legend:

- FNC: Framework Neutral Components, TMS user interface components, etc. for use in TMS WEB Core
- RTL: Runtime Library
- JS: JavaScript



Source: TMS, <https://www.tmssoftware.com/>

# Core features of TMS WEB Core

- consistent development of a web application using Object Pascal
- no use of HTML, CSS and JavaScript necessary; expansion and use of CSS and JavaScript libraries is possible
- useing of IDEs such as Delphi or Visual Studio Code is possible
- graphic designer, including WYSIWYG
- visual and non-visual components for accelerated development approach
- Object Pascal as a powerful object-oriented programming language
- transpiler to translate Object Pascal to JavaScript

# Object Pascal as a programming language

**Object Pascal** is a term for several (partially) compatible programming languages, all of which extend the Pascal language with object-oriented programming. The best-known variant is Delphi, which is used in the development environment of the same name.

Introduction to Object Pascal:

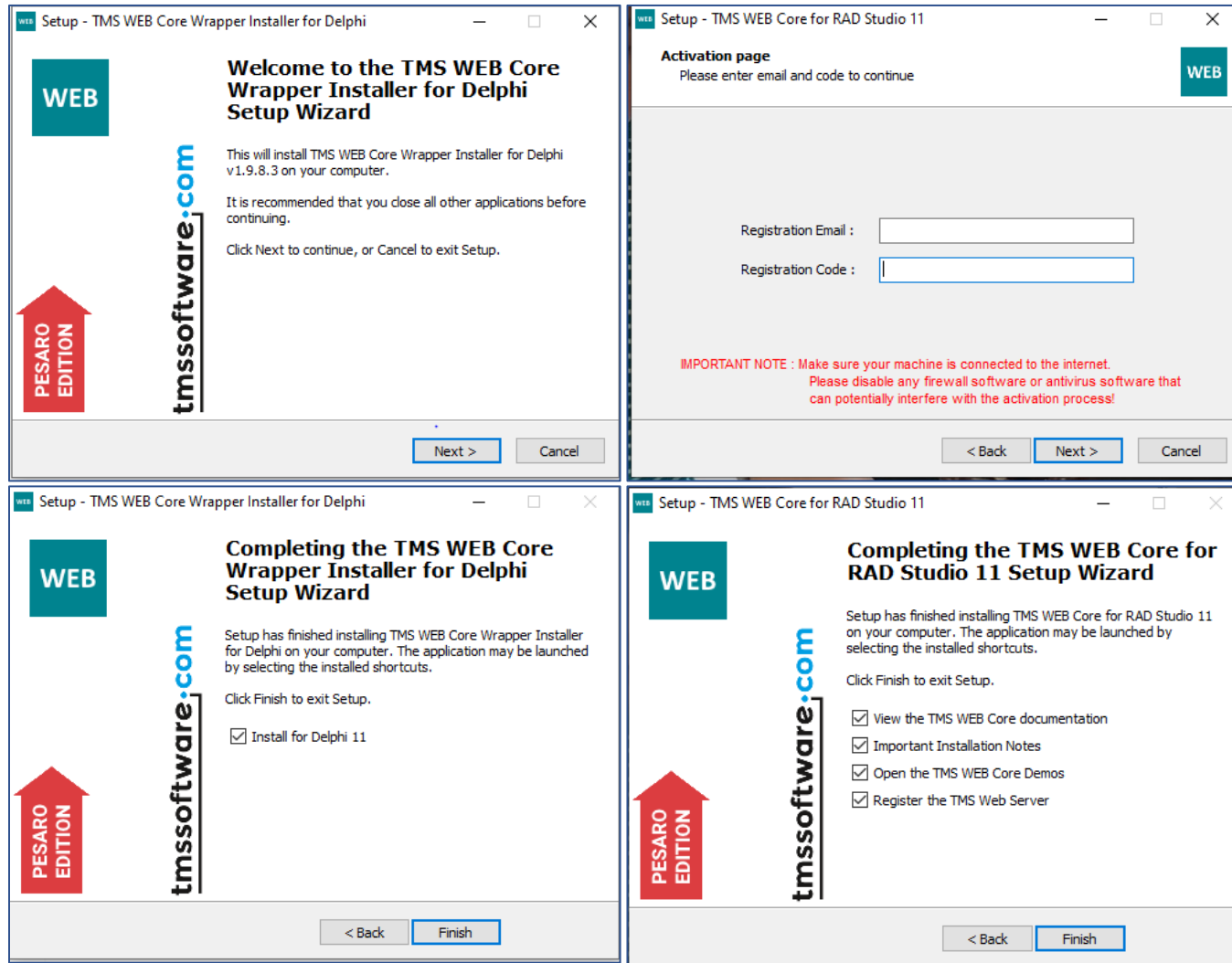
<https://www.delphi-treff.de/object-pascal/program-structure/>

# Development in an IDE

TMS WEB Core supports multiple integrated development environments

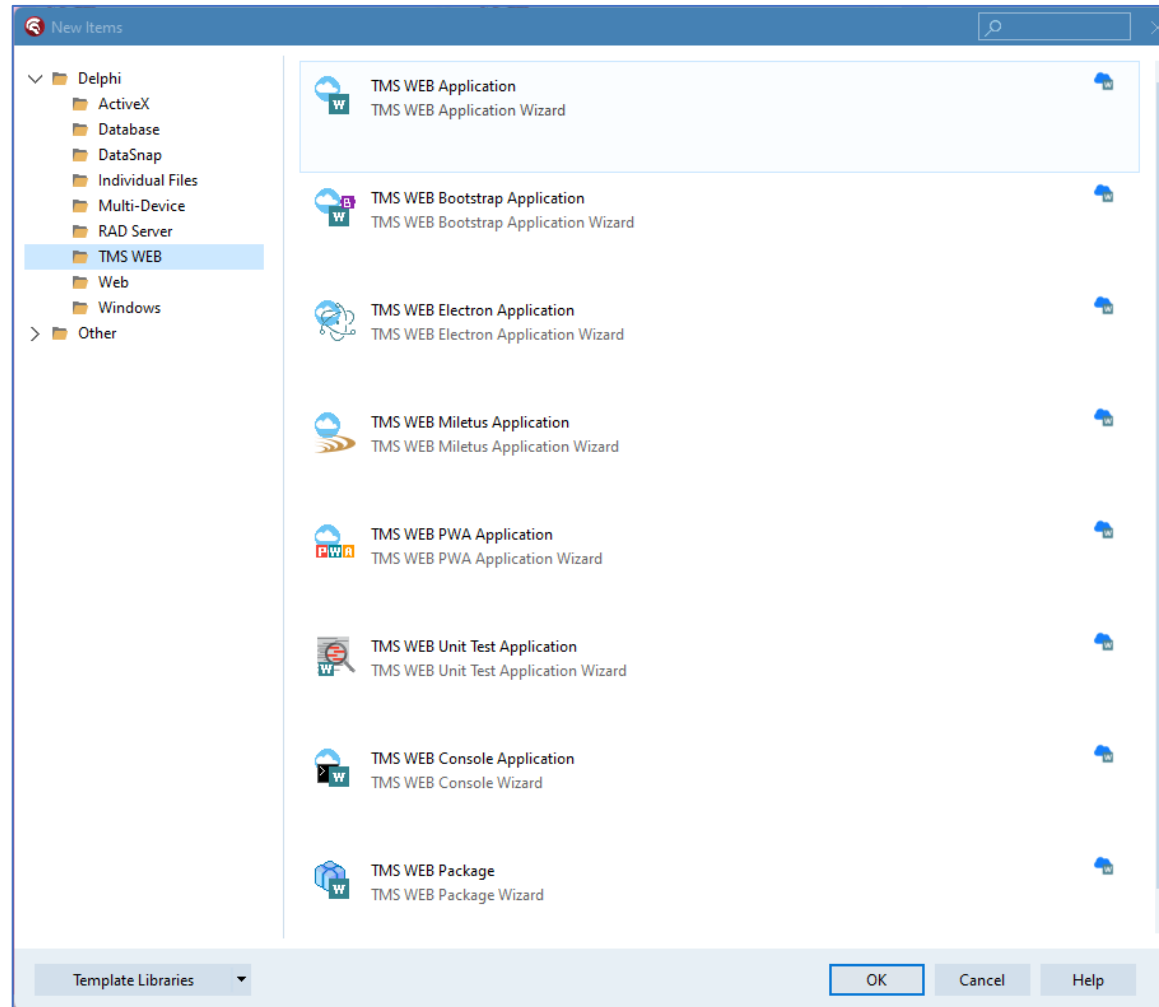
- Delphi: can be used from version Delphi XE7, compatible with all editions, i.e. Community, Professional, Architect and Enterprise
- Microsoft Visual Studio Code: available as a plugin for Windows, macOS or Linux, including WYSIWYG designer
- Lazarus: IDE support from version 2, available for different platforms (Windows, macOS, Linux)

# TMS WEB Core – Installation



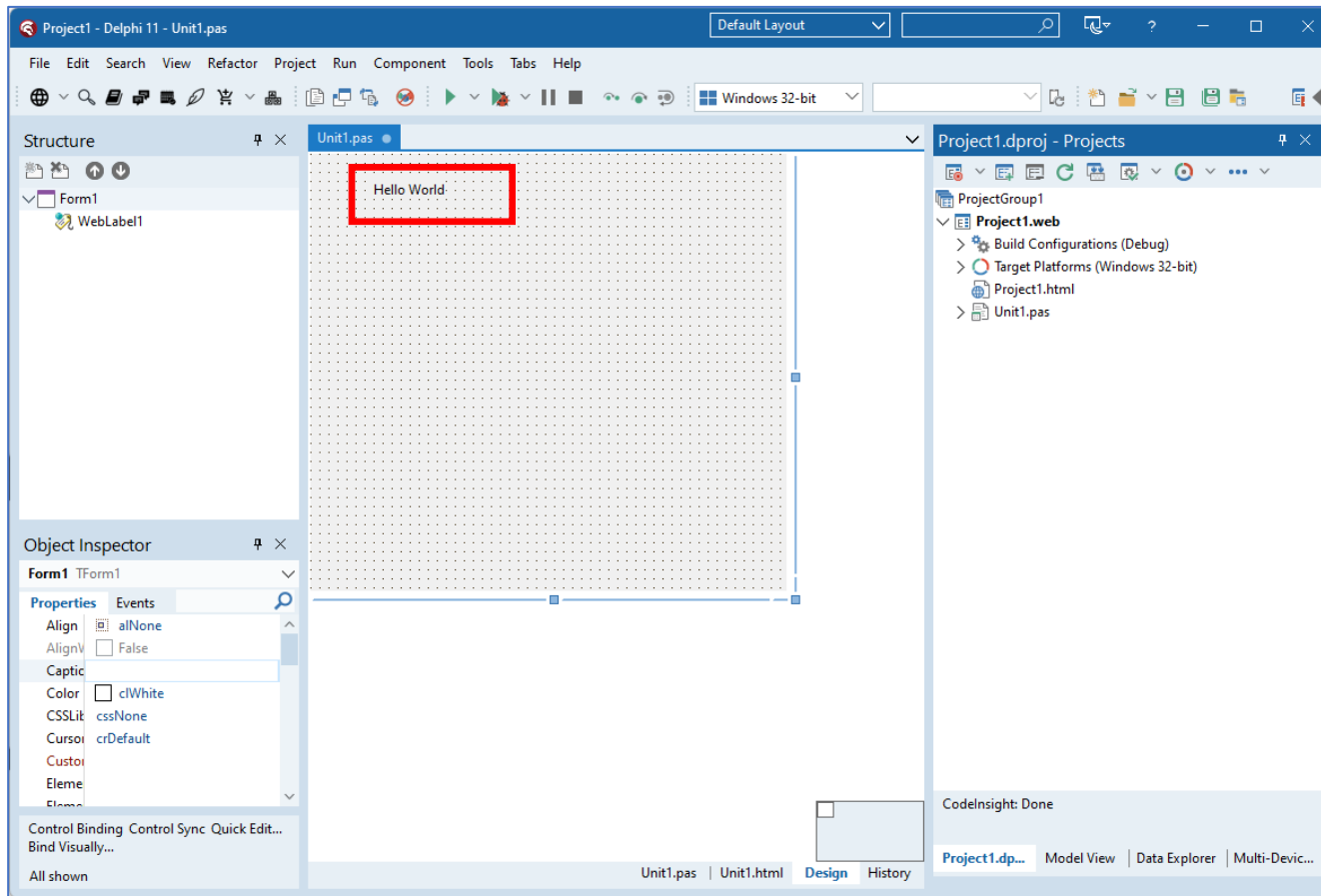
- Installation of TMS WEB Core on Windows
- Integration into the Delphi development environment
- Installation includes:
  - components
  - local development server
  - demos
  - documentation

# TMS WEB Core Project Types



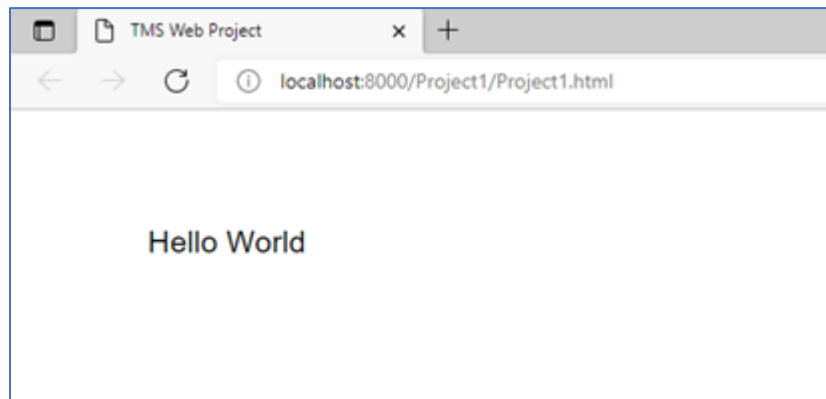
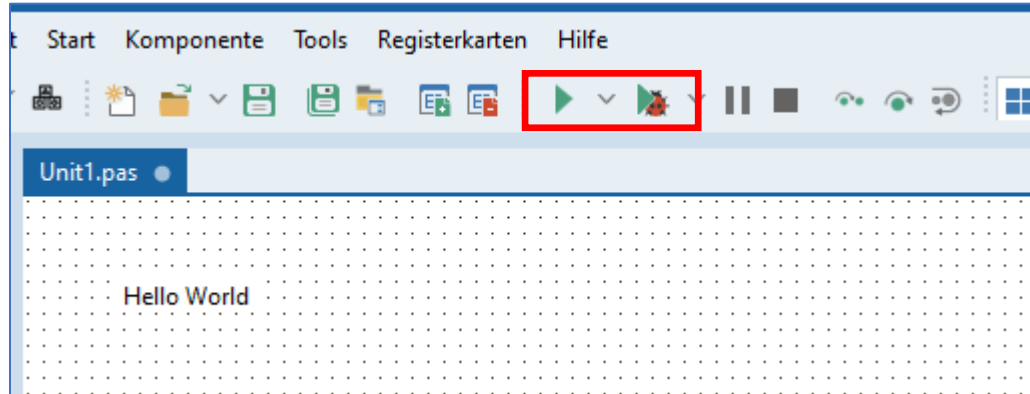
- *TMS WEB Core application: A classic web application.*
- *TMS WEB Bootstrap Application: Web application that uses Bootstrap*
- *TMS WEB Electron application: A web application based on the Electron framework to create cross-platform desktop applications*
- *TMS WEB Miletus Application: Create a desktop application for Windows, macOS and Linux from a web application. Electron is not used*
- *TMS WEB PWA Application: Creates a web application with the option of offline use*

# TMS WEB Application "Hello World" in Delphi



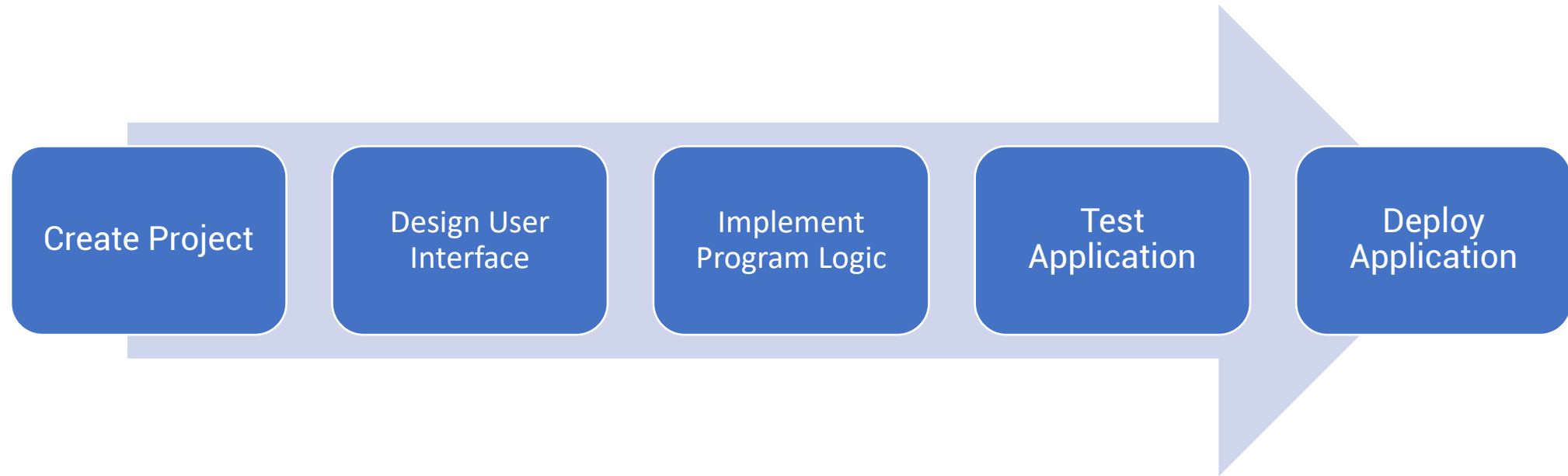
- immediately after executing the project assistant, you come to the graphical designer of Delphi
- the user interface of the web application can be designed here
- in the example a label (TWebLabel) is dragged onto the form (page)
- the design can be edited by changing of the properties

# TMS WEB Application "Hello World" Started in the Browser



- the web application can be started directly from Delphi
- a start is possible with and without execution of the debugger
- the locally installed server is started to test and debug the application
- the server listens on port 8000 (localhost)
- the application runs directly in the local browser

# Example: Web Application with TMS WEB Core



# Example: Credit Calculator Requirements

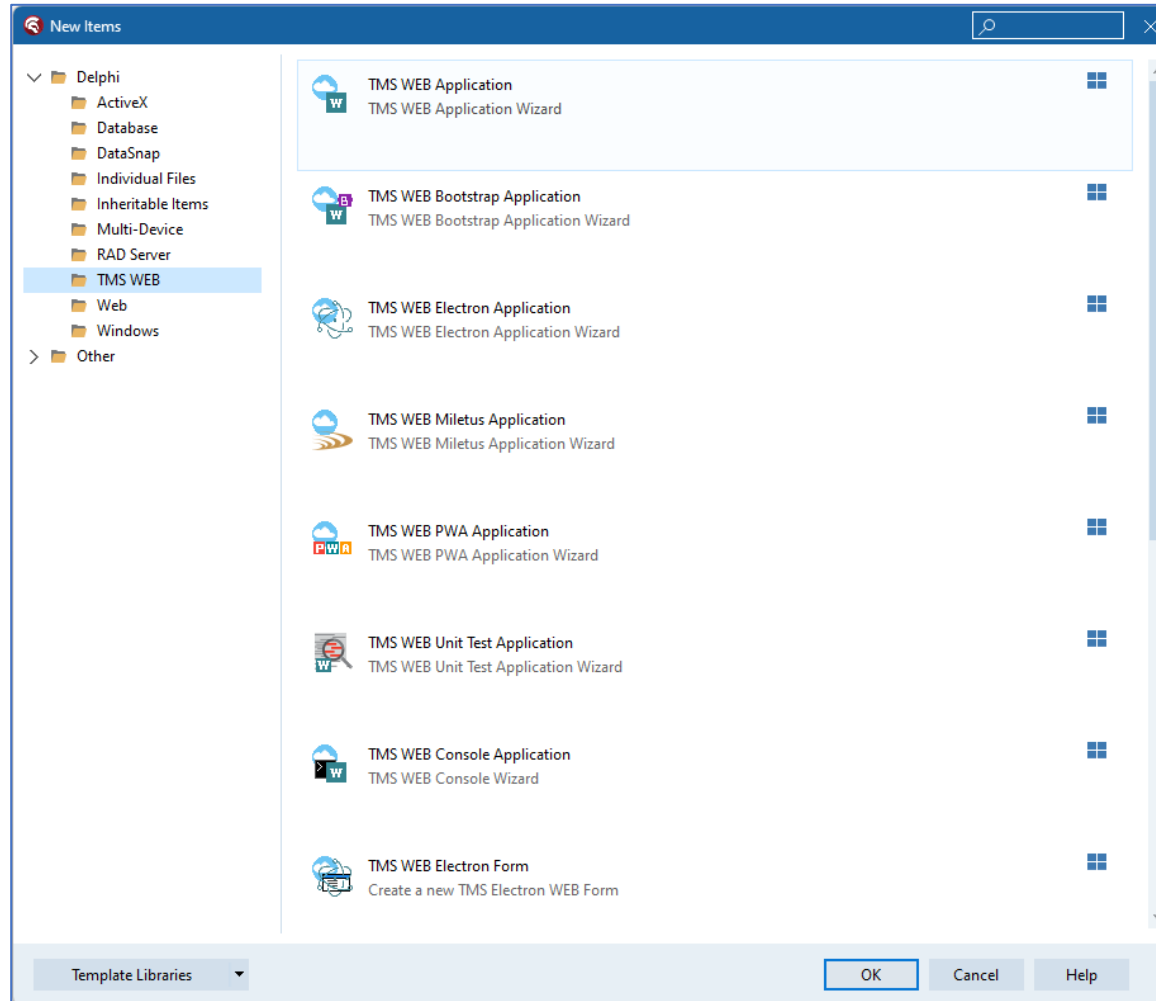
- Implementation of a simple credit calculator
- Input data:
  - credit value
  - interest rate
  - repayment rate
  - fixed interest period (term)
- Output: Calculation of the repayment plan

Download the  
Quellcode:  
<https://www.tmssoftware.com/site/academic.asp>



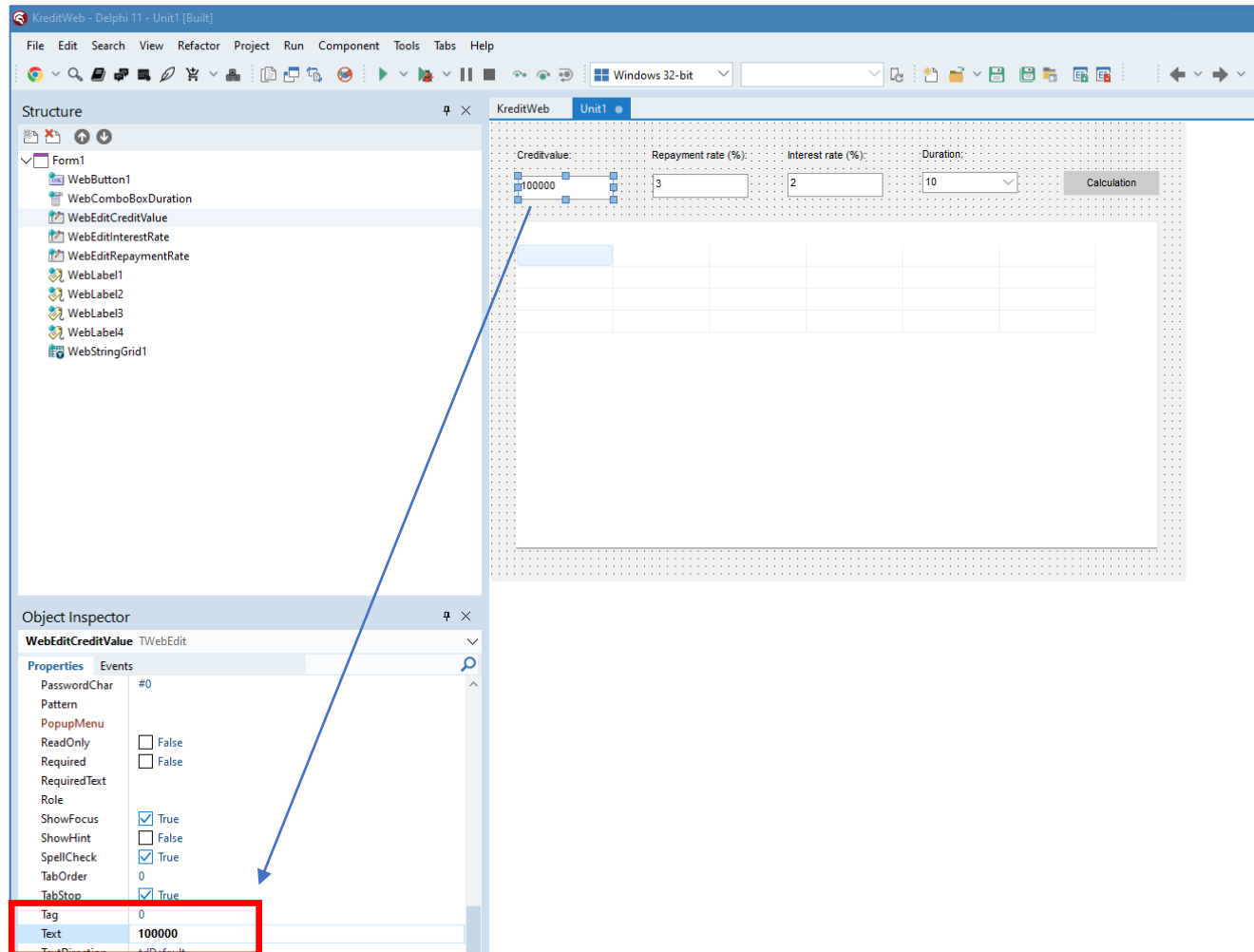
Source: <https://pixabay.com/de/photos/anlage-finanzen-zeit-kapitalrendite-3247252/>

# Example: Credit Calculator – Create Project



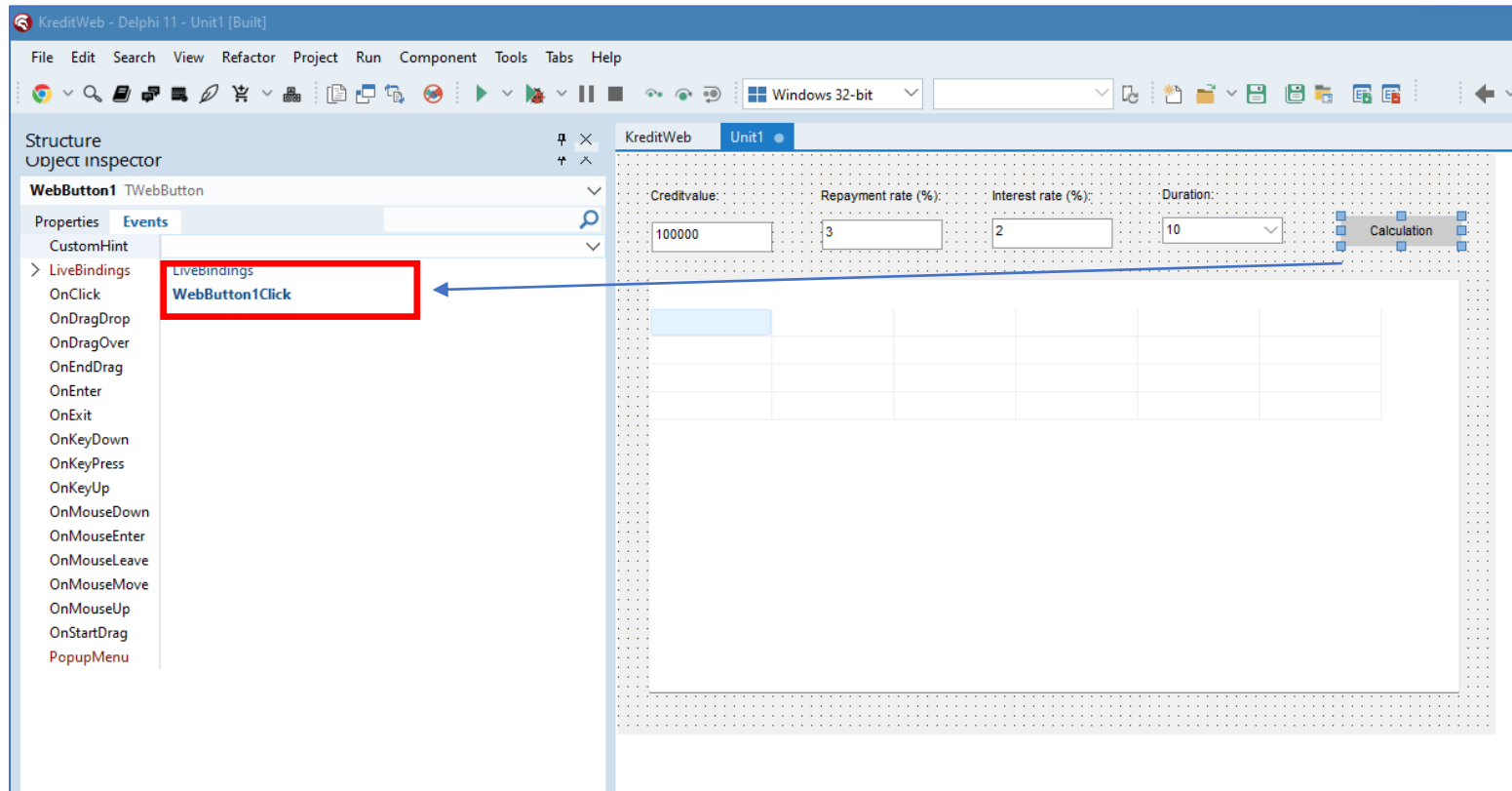
- start a new project with the project template
- create the project folder and subfolders
- TMS WEB application template is selected for a web application

# Example: Credit Calculator - User Interface - Set Properties



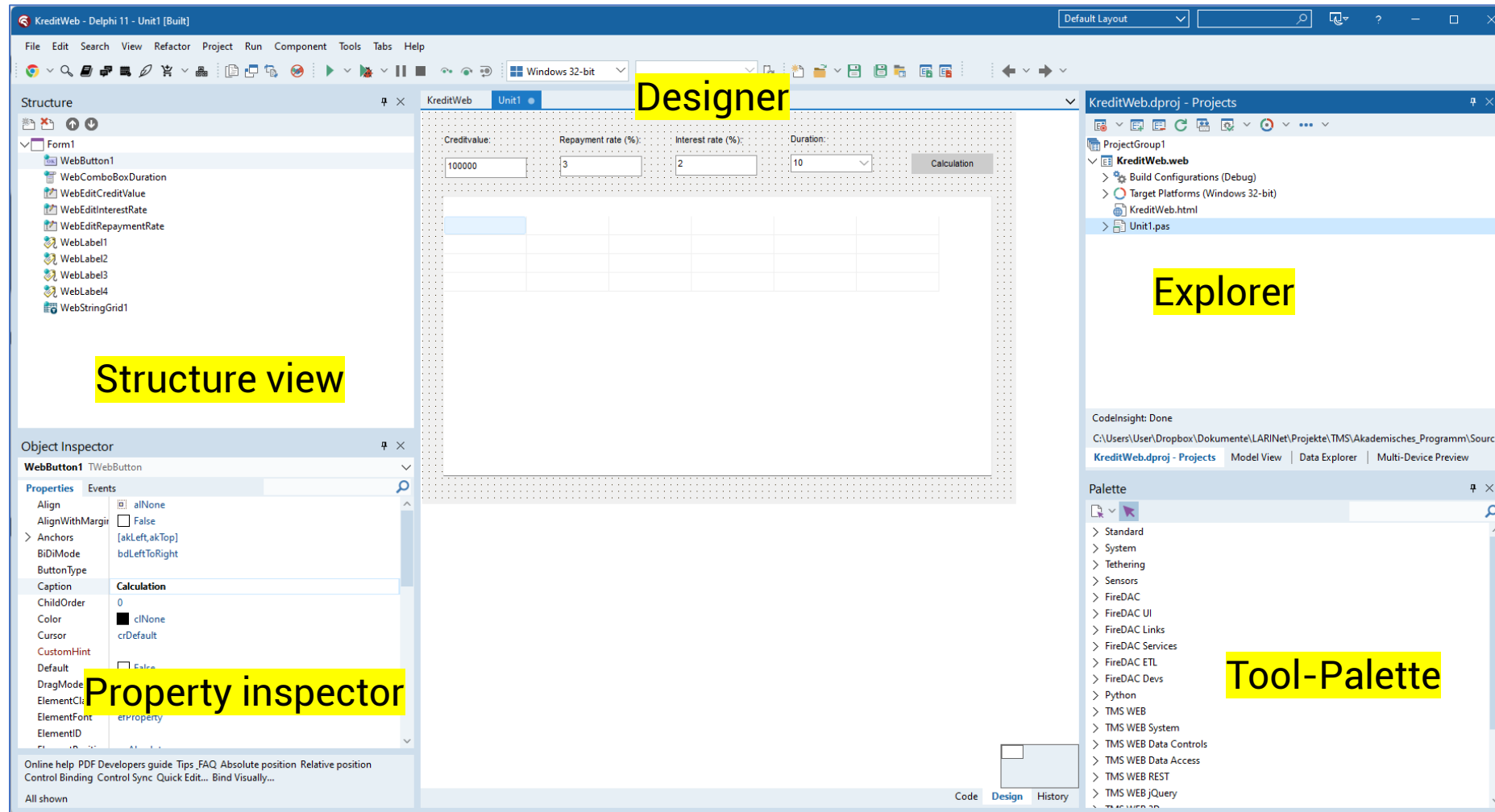
- the values of the properties are defined for each visual component
- select the component and set the properties in the object inspector

# Example: Credit Calculator - User Interface - Bind Events

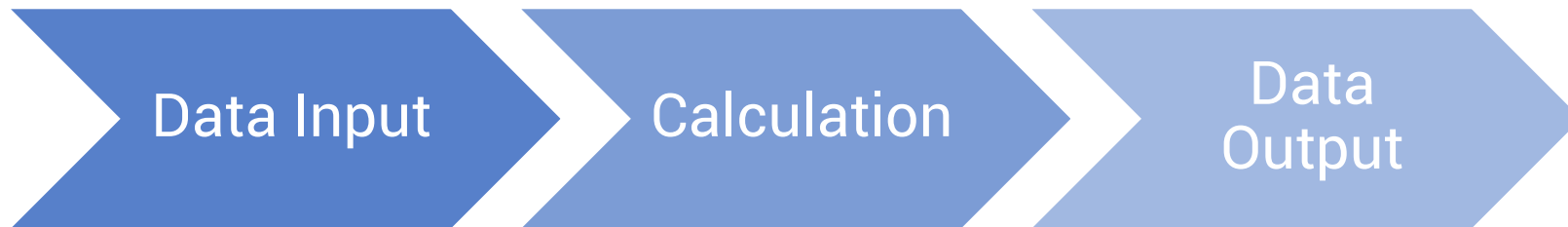


- bind the events for the components
- for example, the *OnClick* event for a button

# Example: Credit Calculator – View in Delphi



# Example: Credit calculator – Program Logic



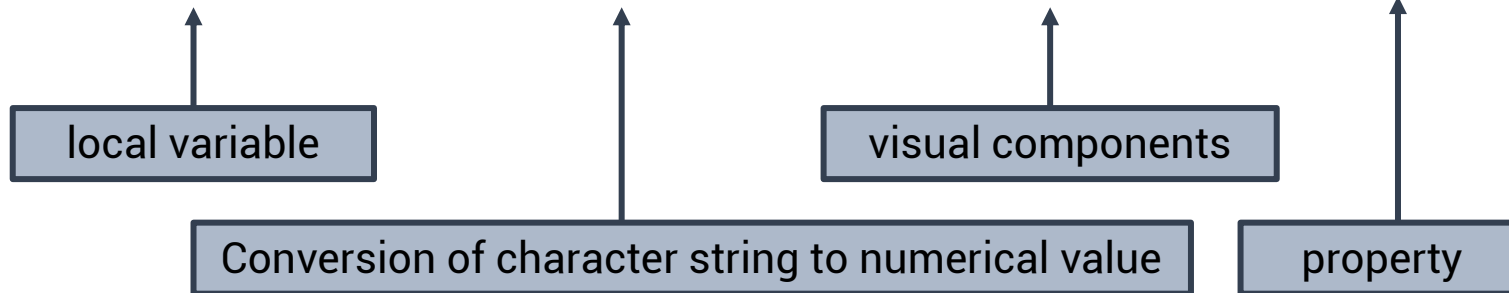
- read values from the text fields (TWebEdit).
- check values if necessary
- assign values to variables

- calculate values for year 1 (annuity)
- calculate values for years 2 to n in a loop

- define number of rows for the DataGrid
- output values in the DataGrid

# Example: Credit calculator - Source Code - Read Data from Form

```
creditValue := StrToFloat(WebEditCreditValue.Text);  
duration    := StrToInt(WebComboDuration.Text);  
interestRate := StrToFloat(WebEditInterestRate.Text);  
repaymentRate := StrToFloat(WebEditRepaymentRate.text);
```



# Credit Calculator – Source Code - Initialize DataGrid

```
procedure TForm1.Form1Create(Sender: TObject);
begin
    // Header
    WebStringGrid1.Cells[0,0]:='Year';
    WebStringGrid1.Cells[1,0]:='Value';
    WebStringGrid1.Cells[2,0]:='Interest charges';
    WebStringGrid1.Cells[3,0]:='Repayment';
    WebStringGrid1.Cells[4,0]:='Annuity';
    WebStringGrid1.Cells[5,0]:='Rest Value';
end;
```

- the properties of the visual components can be accessed from the source code
- these can be read and written
- the example shows the WebStringGrid component
- the cell is addressed using the Cells[column, row] syntax and can be read or written
- values for column headings are defined here
- the calculated values are written into the cells in the same way

# Example: Credit Calculator – Source Code - Perform Calculation/Output Data

```
// Year 2 - n
for i := 2 to duration do
begin
  // Calculation
  value := restValue;
  interestCharges := value * interestRate / 100;
  repayment := annuity - interestCharges;
  restValue := value - repayment;
  // Output Data
  WebStringGrid1.Cells[0, i] := IntToStr(i);
  WebStringGrid1.Cells[1, i] := FloatToStrF(value, ffNumber, 8, 2);
  WebStringGrid1.Cells[2, i] := FloatToStrF(interestCharges, ffNumber, 8, 2);
  WebStringGrid1.Cells[3, i] := FloatToStrF(repayment, ffNumber, 8, 2);
  WebStringGrid1.Cells[4, i] := FloatToStrF(annuity, ffNumber, 8, 2);
  WebStringGrid1.Cells[5, i] := FloatToStrF(restValue, ffNumber, 8, 2);
end;
end;
```

- calculation: year 2 to n
- use for...to...do-loop
- output: corresponding cells in the DataGrid.

# Example: Credit Calculator – Run

The screenshot shows a web browser window with the following content:

Browser tabs: TMS Web Project

Address bar: localhost:8000/KreditWeb/KreditWeb.html

Form fields:

- Creditvalue:
- Repayment rate (%):
- Interest rate (%):
- Duration:
- Calculation button

Year	Value	Interest charges	Repayment	Annuity	Rest Value

- the application can be started directly in the RAD Studio
- starts the local web server and shows the app in the browser
- the functionality can be checked directly without deployment to an external web server

# Example: Credit calculator – Debugging - Setting the Breakpoint in Delphi

```
WebStringGrid1.Cells[0, 1] := '1';
WebStringGrid1.Cells[1, 1] := FloatToStrF(value, ffNumber, 8, 2);
WebStringGrid1.Cells[2, 1] := FloatToStrF(interestCharges, ffNumber, 8, 2);
WebStringGrid1.Cells[3, 1] := FloatToStrF(repayment, ffNumber, 8, 2);
WebStringGrid1.Cells[4, 1] := FloatToStrF(annuity, ffNumber, 8, 2);
70 WebStringGrid1.Cells[5, 1] := FloatToStrF(restValue, ffNumber, 8, 2);
. // Year 2 - n
. for i := 2 to duration do
. begin
. // Calculation
75 value := restValue;
. interestCharges := value * interestRate / 100;
. repayment := annuity - interestCharges;
. restValue := value - repayment;
. // Output Data
80 WebStringGrid1.Cells[0, i] := IntToStr(i);
. WebStringGrid1.Cells[1, i] := FloatToStrF(value, ffNumber, 8, 2);
. WebStringGrid1.Cells[2, i] := FloatToStrF(interestCharges, ffNumber, 8, 2);
. WebStringGrid1.Cells[3, i] := FloatToStrF(repayment, ffNumber, 8, 2);
. WebStringGrid1.Cells[4, i] := FloatToStrF(annuity, ffNumber, 8, 2);
. WebStringGrid1.Cells[5, i] := FloatToStrF(restValue, ffNumber, 8, 2);
. end;
end;
```

- direct debugging of the Delphi code using the browsers Google Chrome or Mozilla Firefox
- the application runs in debug mode
- a breakpoint can be set directly in the source code editor
- after starting the application, the compiler stops at this point
- you can check the value of the variable and run the application step by step

# Example: Credit Calculator – Debugging - Active Debugging

The screenshot displays a web browser window with the URL `localhost:8000/KreditWeb/KreditWeb.html`. The browser shows a credit calculator interface with the following input fields:

- Creditvalue: 100000
- Repayment rate (%): 3
- Interest rate (%): 2
- Duration: 10

A "Calculation" button is present. Below the inputs is a table with the following structure:

Year	Value	Interest charges	Repayment	Annuity	Rest Value

The right side of the image shows the Chrome DevTools interface with the "Sources" panel open to the file `Unit1.pas`. The code is as follows:

```
42 repaymentRate: double;  
43  
44 value: Double;  
45 annuity: Double;  
46 interestCharges: Double;  
47 repayment: Double;  
48 restValue: Double;  
49 i: integer;  
50  
51 begin  
52 // Tread Data  
53 creditValue := StrToFloat(WebEditCreditValue.Text);  
54 duration := StrToInt(WebComboBoxDuration.Text);  
55 interestRate := StrToFloat(WebEditInterestRate.Text);  
56 repaymentRate := StrToFloat(WebEditRepaymentRate.Text);  
57 // Set RowCount in DataGrid  
58 WebStringGrid1.RowCount := duration+1;  
59 // Year 1  
60 value := creditValue;  
61 interestCharges := value * interestRate / 100;  
62 repayment := value * repaymentRate / 100;  
63 annuity := interestCharges + repayment;  
64 restValue := value - repayment;  
65 // Output Data  
66 WebStringGrid1.Cells[0, 1] := '1';  
67 WebStringGrid1.Cells[1, 1] := FloatToStrF(value, fNumber, 8, 2);  
68 WebStringGrid1.Cells[2, 1] := FloatToStrF(interestCharges, fNumber, 8, 2);  
69 WebStringGrid1.Cells[3, 1] := FloatToStrF(repayment, fNumber, 8, 2);  
70 WebStringGrid1.Cells[4, 1] := FloatToStrF(annuity, fNumber, 8, 2);  
71 WebStringGrid1.Cells[5, 1] := FloatToStrF(restValue, fNumber, 8, 2);  
72 // Year 2 - n  
73 for i := 2 to duration do  
74 begin  
75 // Calculation  
76 value := restValue;  
77 interestCharges := value * interestRate / 100;  
78 repayment := annuity - interestCharges;  
79 restValue := value - repayment;  
80 // Output Data  
81 WebStringGrid1.Cells[0, i] := IntToStr(i);
```

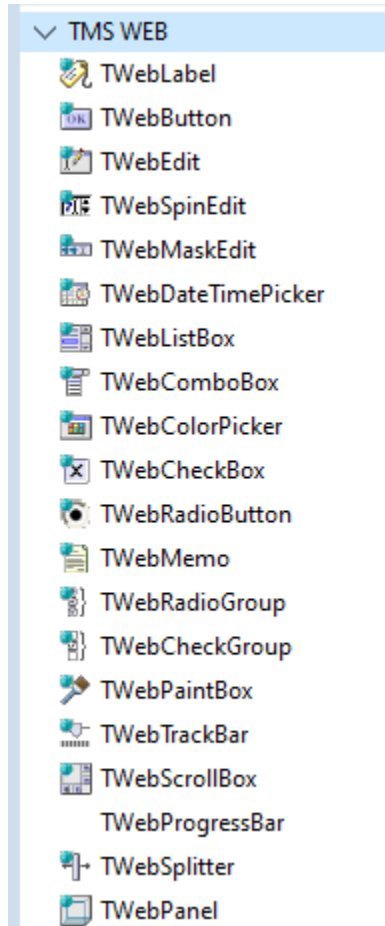
# Example: Credit Calculator – The Result

Name	Änderungsdatum	Typ	Größe
TMSWeb	15.04.2022 08:42	Dateiordner	
Win32	15.04.2022 10:41	Dateiordner	
KreditWeb.dpr	20.08.2021 10:04	DPR-Datei	1 KB
KreditWeb.dproj	15.04.2022 22:05	DPROJ-Datei	61 KB
KreditWeb.dproj.local	15.04.2022 22:05	LOCAL-Datei	1 KB
KreditWeb.html	20.08.2021 10:04	Microsoft Edge H...	1 KB
KreditWeb.identcache	15.04.2022 22:05	IDENTCACHE-Datei	1 KB
Unit1.dfm	15.04.2022 20:11	DFM-Datei	5 KB
Unit1.html	19.08.2021 14:12	Microsoft Edge H...	1 KB
Unit1.pas	15.04.2022 21:37	PAS-Datei	3 KB

- the HTML, CSS and JavaScript files are generated in the output folder for deployment on the server
- then the web application can be accessed via network

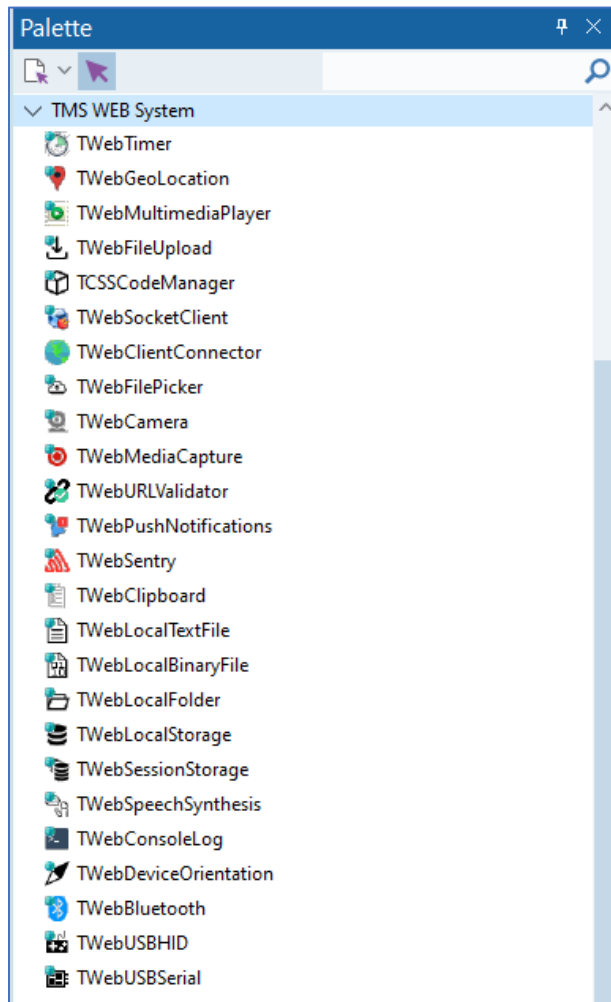
Name	Änderungsdatum	Typ	Größe
KreditWeb.html	15.04.2022 10:41	Microsoft Edge H...	1 KB
KreditWeb.js	15.04.2022 10:41	JavaSkriptdatei	2.418 KB
KreditWeb.js.map	15.04.2022 10:41	Linker Address Map	3.575 KB
Unit1.html	19.08.2021 14:12	Microsoft Edge H...	1 KB

# Components of TMS WEB Core - Visual Components



- the user interface is built of visual controls
- examples:
  - Button: TWebButton
  - Label: TWebLabel
  - Textbox (input): TWebEdit
- TMS WEB Core comes with a large number of basic controls
- these can be expanded:
  - through the FNC controls (TMS)
  - through own development

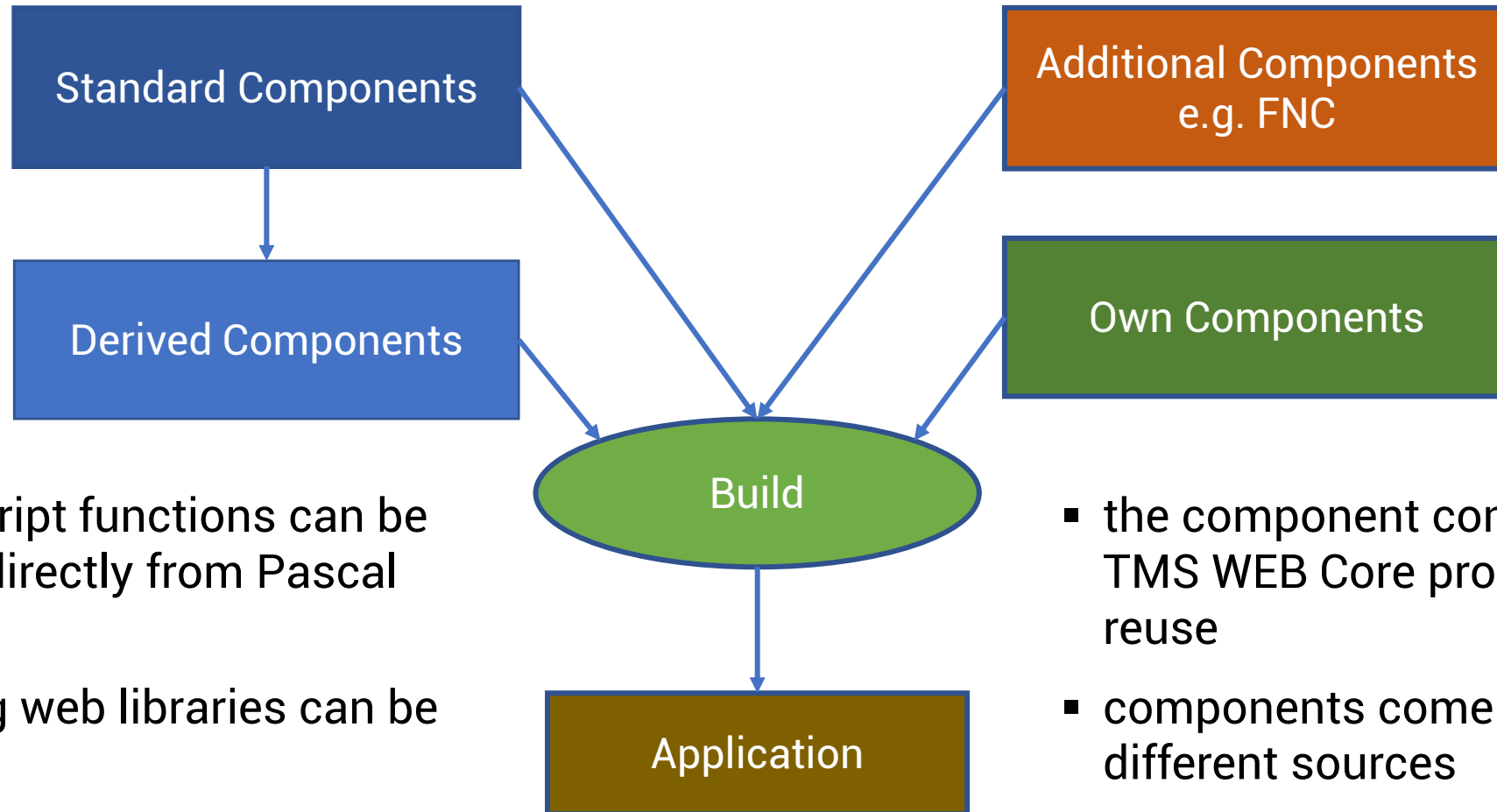
# Components of TMS WEB Core - Non-Visual Components



- TMS WEB Core includes also a lot of non-visual components
- these contain complex functions
- for example, components for camera access (*TWebCamera*), file upload (*TWebFileUpload*) or clipboard access (*TWebClipboard*)
- by dragging a component to the form an object of the class is generated
- the non-visual components are not visible at runtime

# Components of TMS WEB Core

## - Other components



- JavaScript functions can be called directly from Pascal code
- existing web libraries can be used

- the component concept of TMS WEB Core promotes the reuse
- components come from different sources

# Summary

- modern business applications are now often designed as web applications
- advantages: no installation, platform-neutral, cross-device, multi-user capable
- basic technologies: HTML, CSS and JavaScript
- various libraries and frameworks often make things complex
- TMS WEB Core allows programming with Object Pascal, offers a graphic designer and a lot of visual and non-visual components
- application development is accelerated and concentration on business logic is possible

# Outlook

Part 1: Classification, introduction, setup and the first app with TMS WEB Core

**Part 2: Application development and features of TMS WEB Core**

Part 3: Web, Mobile and Desktop Applications with TMS WEB Core

# Ressources

- TMS WEB Core:  
<https://www.tmssoftware.com>
- Delphi Community Edition:  
<https://www.embarcadero.com/products/delphi/starter>
- TMS Academic License:  
<https://www.tmssoftware.com/site/academic.asp>
- Flick, Holger: TMS WEB Core: Developing Web Applications with Delphi,  
Independently published, 2020
- Source code for example Credit Calculator Calculator:  
<https://www.tmssoftware.com/site/academic.asp>

**tmssoftware.com**

develop • web

Framework for creating modern web applications



More info:

<http://web.tmssoftware.com>