



# Front-end workshop



# Goals of this course

- ▶ Getting to know practical skills of using node.js environment and programming applications in Javascript
- ▶ Understanding the foundations of Javascript language necessary to continue exploring advanced technologies
- ▶ Introduction to npm environment
- ▶ Introduction to NoSQL databases on the example of MongoDB
- ▶ Learning about React.js library



# Introduction / Presentation of participants

- ▶ Name and Surname, Company
- ▶ Type of current work
- ▶ Experience in programming
- ▶ Reasons for attending the course (expectations)



# Introduction / Preparation of participants

- ▶ Programming skills Javascript
  - ▶ HTML
  - ▶ DOM
  - ▶ AJAX
- ▶ Ability to create and edit text files
- ▶ Usage of tool / IDE (Integrated Development Environment)



# Introduction

1. Introduction to npm
  - ▶ JavaScript package manager
  - ▶ The package.json file and its meaning
  - ▶ npm commands and the ability to use npm for Angular, node.js and similar technologies
2. Remind of JavaScript
3. Basis of ES6



# Introduction / Course contents

- 4 MongoDB database basics
5. Efficient NoSQL schema with data nesting and search
6. Node.js modules
7. Request parsing and sending response
8. Dynamic HTML rendering (Server)
9. Express.js
10. Working with Node.js and SQL databases (MySQL)
11. Working with Node.js and NoSQL databases (MongoDB) and Mongoose
12. Sessions and cookies
13. User authentication and authorization
14. Payment processing using Stripe.js
15. REST API along with authentication process
16. Developing a real-time Node.js application using Websockets
17. Automated testing (Unit tests)



# Module PLAN

- ▶ Web servers (www)
- ▶ HTTP protocol basics
- ▶ Web applications
  - ▶ Web application development
  - ▶ Web application components:
    - ▶ HTML files
    - ▶ resources
    - ▶ cascading stylesheet files(CSS)
    - ▶ JavaScript files



# Web servers

- ▶ Web applications are typically stored on web servers.
- ▶ The server's task is to:
  - ▶ Handle Internet requests
  - ▶ Store an application files
  - ▶ Provide access to resources.
- ▶ A web server is a software that listens for requests, provides access to resources, and generates a response.
- ▶ The web server can receive requests from multiple clients and handles them separately for each client.
- ▶ After receiving a request, the server locates the resource(files) or generates them and sends the contents back to the client. In some cases, requests are forwarded to other services that construct the content of the response.
- ▶ Dynamic content can be generated on the server or on the client's computer.
- ▶ Web applications use dynamic content in the client software using JavaScript and on the server using the server application.



# Web architecture (www)

SERVER



WEB SERVER

WWW NETWORK

CLIENT





# HOW A WEB SERVER WORKS?

CLIENT



REQUEST

RESPONSE

WEB SERVER



- ▶ Web servers work in three stages:
  1. The request is always initiated by the client.
  2. The server locates the resource and constructs a response.
  3. The response with the resource contents is sent back to the client.



# HOW A WEB SERVER WORKS?

CLIENT



REQUEST

WEB SERVER



RESPONSE

SERVER



DB SERVER



- ▶ When external services are added, the process remains very similar.
  1. The client sends a request.
  2. The server tries to locate the resource and then it finds that it requires an external application.
  3. This application generates communication with other resources, such as a database or another server.
  4. The web server constructs a response and sends it back to the client.
- ▶ Corporate application servers are complete servers that contain an application engine, which can load data from external resources and a Web server to handle Web requests.



# WEB CLIENT

- ▶ What is a client?
- ▶ A client is an application running on a computer that can execute HTTP requests and interprets HTTP responses.
  - ▶ Web browsers
    - ▶ Firefox, Chrome, Internet Explorer, Safari, ...
  - ▶ Other applications
    - ▶ e.g. mobile weather application
    - ▶ news reader



## HTTP PROTOCOL

- ▶ HTTP is a communication protocol that is based on TCP connections over IP client and server.
- ▶ In the HTTP protocol, all requests are independent.
- ▶ The client can send additional requests, but each request is completely new to the server and is handled independently from previous requests.
- ▶ No session management within the HTTP protocol.
- ▶ HTTP is stateless.
- ▶ Session management is done by sending an identifier as part of the request, which can refer to a specific session.



# HTTP REQUEST

- ▶ A request contains some information that helps the server generate a response.
- ▶ Each request contains a Request Line, which consists of
  - ▶ Uniform Resource Identifier (URI), which is the name of the requested resource
  - ▶ The method used
  - ▶ The version
- ▶ The HTTP protocols are defined by the following methods:
  - ▶ OPTIONS
  - ▶ GET
  - ▶ HEAD
  - ▶ POST
  - ▶ PUT
  - ▶ DELETE
  - ▶ TRACE
  - ▶ CONNECT



# HTTP REQUEST

- ▶ Browsers use only GET or POST.
- ▶ HTTP request headers are key-value pairs that are used to provide additional information.
- ▶ accept-encoding: specifies which encoding is expected in the response.
- ▶ user-agent: specifies the client application which is used. When you use a browser, the browser name and version are included in this header.
- ▶ Identifiers and information about session, such as cookies, may be included in request headers.
- ▶ For more information about HTTP headers, check out <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>.
- ▶ The message body may contain additional information provided by the user, such as form details, when using the POST method.



# HTTP REQUEST

Necessary information about the request

## Request Line

- method
- URI
- version

## Headers

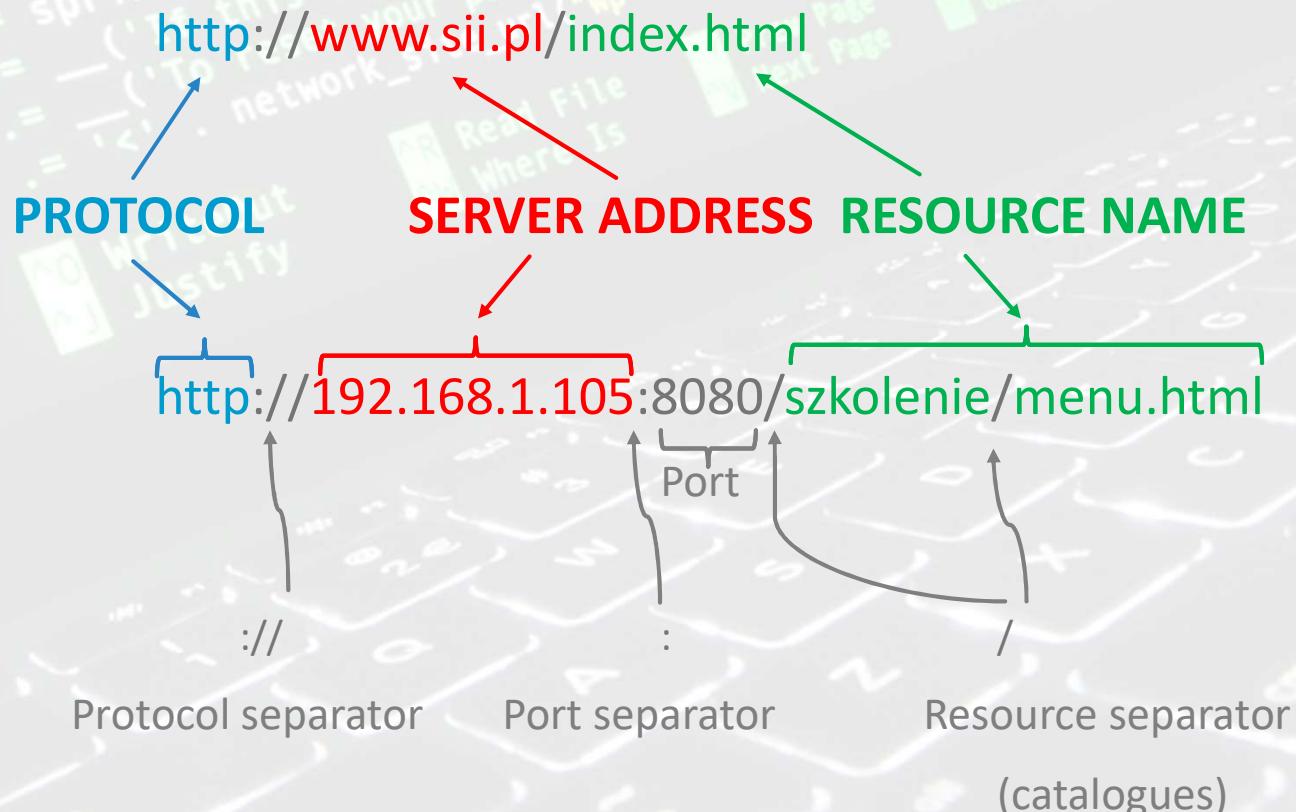
- Name-value pairs

## Message Body

- Additional data



# HTTP REQUEST: URL





# HTTP RESPONSE

Response status

## Status Line

- Version
- Status
- Reason of status

Additional information regarding responses

## Headers

- Name-value pairs

Content of responses

## Message Body

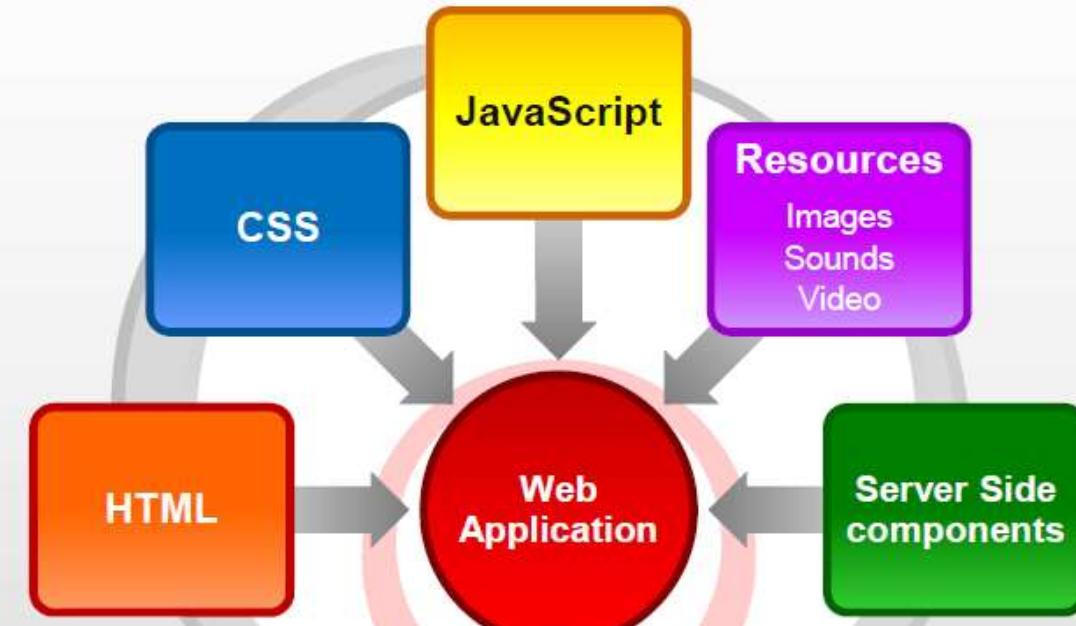
- Response content



# HTTP RESPONSE

- ▶ The HTTP response must contain a status line.
- ▶ Status codes are grouped as follows:
  - ▶ 1xx: Informational, for example **101 - Switching Protocols**
  - ▶ 2xx: Success, for example 200 - OK
  - ▶ 3xx: **Forwarding**. Further action is required to complete the request, for example 301 - Forwarded permanently.
  - ▶ 4xx: Client Error. The request contains a problem and could not be fulfilled, for example 404 - Not Found.
  - ▶ 5xx: Server Error. There was a problem on the server, for example 500 - Server Error.
- ▶ A list of response codes can be found at <http://www.w3.org/Protocols/rfc2616/rfc2616-section10.html>
- ▶ Response headers contain additional information about the response, such as content, type, content size, user cookies, session ID, etc.
- ▶ There must be a blank line between the headers and the body.
- ▶ The body contains the actual content of the requested resource. The body of the response is optional.

# WEB APPLICATIONS





# WEB APPLICATIONS

## ► Technical Definition

- A set of files stored locally or on a web server that can run in a web browser

## ► Conceptual definitione

- An application that runs in a web browser that:
  - Provides the user with a solution to a problem
  - Is self-contained and focused
  - Has a rich user interface
  - Takes advantage of the capabilities of the user's device



# WEB APPLICATIONS

- ▶ A web application is a collection of files stored in a directory that can be aliased by a URL.
- ▶ Together, the files describe the user interface and contain the application logic that runs in a web browser.
- ▶ The user interface is defined using HTML. A consistent look and interface (colors, fonts, and spacing) can be defined using CSS.
- ▶ The application logic and user interactions are defined by JavaScript.
- ▶ It is possible to load additional resources to enhance the application, such as images, videos, sound, ...
- ▶ Applications can include dynamic server components such as servlets, JSP, Facelets, REST, services, etc. Server components provide dynamic data, user interfaces, styles, etc.
- ▶ Server software (e.g., Java EE) provides all the server-side technologies to create these components.

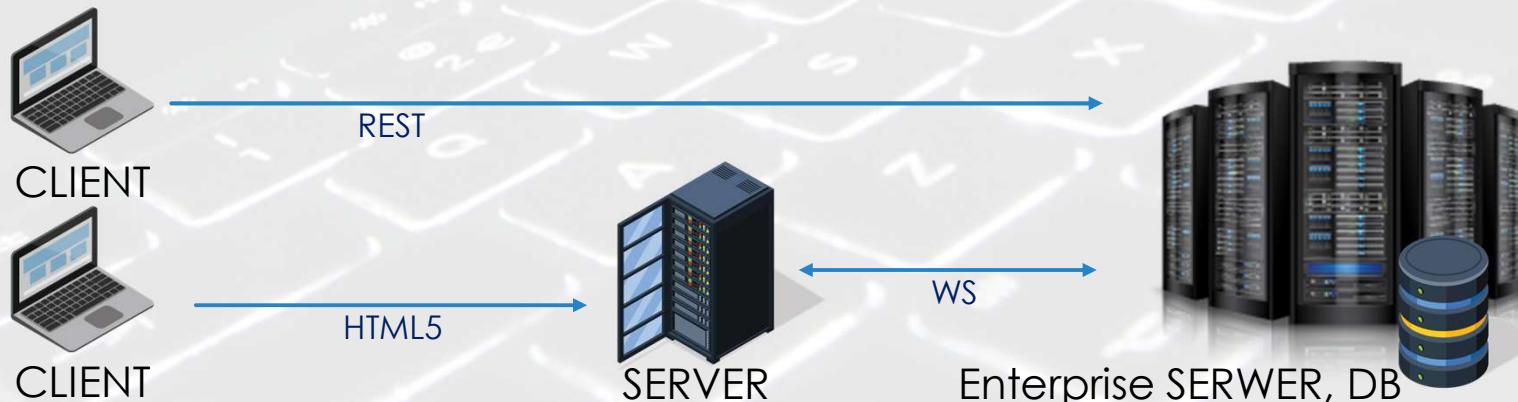


# WEBSITE vs. WEB APPLICATIONS

- ▶ Similarities:
  - ▶ Both could be run in web browsers
  - ▶ Have HTML5, CSS3, JavaScript and additional resources
  - ▶ Can be stored locally or on web servers
- ▶ Website
  - ▶ Is a collection of information organized into pages
  - ▶ Relies on navigation
  - ▶ It isn't very interactive
- ▶ Web Application
  - ▶ Solves one problem at a time
  - ▶ It's interactive
  - ▶ It's self-contained
  - ▶ It's dynamic

# Server Side Web Application

- ▶ Generates dynamic content from data sources
  - ▶ Generates dynamic pages, images, and resources
  - ▶ Provides authentication, session management and security
  - ▶ Stores user and service information
  - ▶ Provides XML and REST web services (e.g., returns JSON)
  - ▶ Provides AJAX and WebSocket server **endpoints**





# Server Side Web Application

- ▶ Dynamic content can be generated by applications, which is running on servers.
- ▶ The web application downloads the dynamic content generated by the servers and presents the content using separate requests that will only receive dynamic information.
- ▶ The layout and displayed content are defined and controlled by the client application.
- ▶ Interactivity and dynamic content is achieved by usage of client-side scripting.
- ▶ Another way to get dynamic content is to generate the content and layout on the server. The server receives the request, then generates the content as a static page, which returne to the client to only display the information.
- ▶ Interactivity and dynamic content is achieved by navigation and hyperlinks.





# HTML

- ▶ HTML is the **standard markup language** for web pages.
- ▶ HTML means **Hyper Text Markup Language**
- ▶ You can use HTML to create your own website.
  
- ▶ This course is consistent with the latest HTML5 standard.
  
- ▶ **HTML is easy to learn** ☺



# HTML

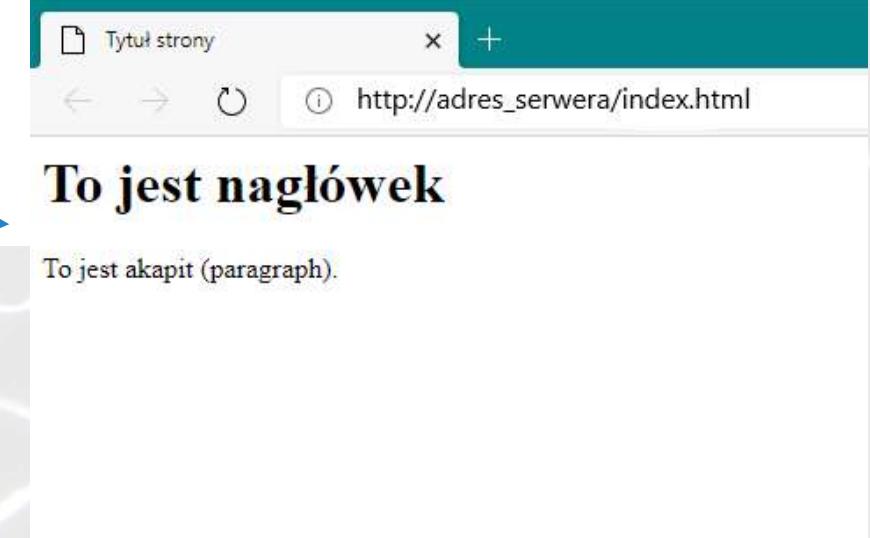
► An example of simple HTML code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page title</title>
  </head>

  <body>

    <h1>To jest nagłówek</h1>
    <p>To jest akapit (paragraph).</p>

  </body>
</html>
```





# HTML

- ▶ What is HTML?
- ▶ HTML means **Hyper Text Markup Language**
- ▶ HTML is the standard **markup language** for creating web pages
- ▶ HTML describes the structure of a web page
- ▶ HTML consists of a number of elements
- ▶ HTML elements inform the browser how to display content
- ▶ HTML elements describe pieces of content, such as "this is a header," "this is a paragraph," "this is a link," etc.



# HTML

- ▶ **<!DOCTYPE html>** declaration specifies that this document is an HTML5 document
- ▶ **<html>** element is the main element of the HTML page
- ▶ **<head>** element contains meta information about the HTML page
- ▶ **<title>** element defines the title of the HTML page (which is displayed in the browser title bar or page tab)
- ▶ **<body>** element defines the body of the document; it is the container for all visible content, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- ▶ **<h1>** element defines a large header
- ▶ **<p>** element defines a paragraph



# HTML

- ▶ What is an HTML element (**TAG**)?
- ▶ HTML element (Tag HTML) is defined by a start tag, some content and **an end tag**:
- ▶ **<tagname> Content ... </tagname>**
- ▶ HTML element **is all the marks from the start tag to the end tag**
- ▶ **<h1> My first headline</h1>**
- ▶ **<p> My first paragraph. </p>**
- ▶ **NOTE:** Some tags, such as **<br>**, have no content and don't have to be finished with an **end tag**



# HTML

- ▶ The structure of an HTML document is similar to XML.
- ▶ HTML elements **can be nested** (it means that elements can contain other elements).
- ▶ All HTML documents consist of **nested** HTML elements.
- ▶ The `<html>` element is the root element and defines the entire HTML document.
- ▶ It has the start tag `<html>` and an end tag `</html>`.
  - ▶ Then inside the `<html>` element there is an `<body>` element
  - ▶ `<body>` element defines the body of the document.
  - ▶ It has a start tag `<body>` and an end tag `</body>`.
  - ▶ Then inside the `<body>` there are two other elements: `<h1>` i `<p>`

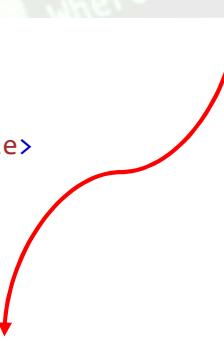


# HTML

- ▶ Always remember about an end tag
- ▶ Some HTML elements will display correctly even if you forget the end tag

```
<!DOCTYPE html>
<html>
  <head>
    <title>Tytuł strony</title>
  </head>

  <body>
    <h1>To jest nagłówek</h1>
    <p>To jest akapit (paragraph).</p>
  </body>
</html>
```



- ▶ However, never trust this!
- ▶ If you forget the end tag, unexpected results and errors may occur!



# HTML

- ▶ Empty HTML elements
- ▶ HTML elements without content are called empty elements.
- ▶ The `<br>` tag defines a line split and is an empty element without a closing tag:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Tytuł strony</title>
  </head>

  <body>
    <h1>To jest nagłówek</h1>
    <p>To jest akapit (paragraph) <br> z dwoma liniami tekstu. </p>
  </body>
</html>
```





# HTML

- ▶ HTML is not case-sensitive
- ▶ HTML tags are not case-sensitive: `<P>` means the same as `<p>`.
  
- ▶ The HTML standard does not require lowercase tags. However, in HTML it's recommended to use lowercase and it requires lowercase for more stringent document types such as XHTML.

# HTML

- ▶ The following pages present a list of HTML tags with their descriptions:

Tag	Opis
<!--...-->	Defines comment
<!DOCTYPE>	Specifies the document type
<a>	Defines a hyperlink
<abbr>	Defines an acronym or abbreviation
<acronym>	Not supported in HTML5. Use <abbr> instead.
	Defines acronym
<address>	Defines contact information for the author/owner of the document
<applet>	Not supported in HTML5. Use <embed> or <object> instead.
	Defines an embedded applet
<area>	Defines the area inside the image map
<article>	Defines the article
	Defines content independently of page content
<aside>	
<audio>	Defines embedded audio content

# HTML

- ▶ The following pages present a list of HTML tags with their descriptions:

Tag	Opis
<code>&lt;b&gt;</code>	Defines bold text
<code>&lt;base&gt;</code>	Specifies the base URL/target for all relative URLs in the document  <b>Not supported in HTML5. Use CSS instead.</b>
<code>&lt;basefont&gt;</code>	Sets the default color, size and font for all text in the document
<code>&lt;bdi&gt;</code>	Isolates part of the text that can be formatted in a different direction than other text outside of it
<code>&lt;bdo&gt;</code>	Sends the current text direction
<code>&lt;big&gt;</code>	<b>Not supported in HTML5. Use CSS instead.</b>  Defines large text
<code>&lt;blockquote&gt;</code>	Defines a section quoted from another source
<code>&lt;body&gt;</code>	Defines the content of the document
<code>&lt;br&gt;</code>	Defines the split of a single line
<code>&lt;button&gt;</code>	Defines a clickable button

# HTML

- ▶ The following pages present a list of HTML tags with their descriptions:

Tag	Opis
<code>&lt;canvas&gt;</code>	Used to draw on-the-fly graphics using scripts (usually JavaScript)
<code>&lt;caption&gt;</code>	Define table signature
<code>&lt;center&gt;</code>	<b>Not supported in HTML5. Use CSS instead.</b>
<code>&lt;cite&gt;</code>	Defines centered text
<code>&lt;code&gt;</code>	Defines the title of the work
<code>&lt;col&gt;</code>	Defines a piece of computer code
<code>&lt;colgroup&gt;</code>	Defines column properties for each column in a <code>&lt;colgroup&gt;</code> element
<code>&lt;data&gt;</code>	Specifies a group of one or more columns in a table for formatting
<code>&lt;datalist&gt;</code>	Adds a machine-readable translation of the given content
<code>&lt;dd&gt;</code>	Specifies a list of predefined input control options
<code>&lt;del&gt;</code>	Defines the description / term value in the description list
<code>&lt;details&gt;</code>	Defines text that has been removed from the document
<code>&lt;dfn&gt;</code>	Defines additional details that the user can show or hide
<code>&lt;dialog&gt;</code>	Specifies the term that will be defined in the content
	Defines a dialog box or window

# HTML

- ▶ The following pages present a list of HTML tags with their descriptions:

Tag	Opis
<code>&lt;dir&gt;</code>	Not supported in HTML5. Use <code>&lt;ul&gt;</code> instead.
	Defines a list of directories
<code>&lt;div&gt;</code>	Defines a section in the document
<code>&lt;dl&gt;</code>	Defines a list of descriptions
<code>&lt;dt&gt;</code>	Defines a term / name in the description list
<code>&lt;em&gt;</code>	Defines the highlighted text
<code>&lt;embed&gt;</code>	Defines a container for an external application
<code>&lt;fieldset&gt;</code>	Groups related elements in a form
<code>&lt;figcaption&gt;</code>	Defines a signature for the <code>&lt;figure&gt;</code> element
<code>&lt;figure&gt;</code>	Defines independent content
	Not supported in HTML5. Use CSS instead.
<code>&lt;font&gt;</code>	Defines the font, color and size of text
<code>&lt;footer&gt;</code>	Defines a footer for a document or section
<code>&lt;form&gt;</code>	Defines an HTML form for user input

# HTML

- ▶ The following pages present a list of HTML tags with their descriptions:

Tag	Opis
<code>&lt;frame&gt;</code>	Not supported in HTML5. Defines a window (frame) in a set of frames
<code>&lt;frameset&gt;</code>	Not supported in HTML5. Defines a set of frames
<code>&lt;h1&gt; to &lt;h6&gt;</code>	Defines HTML headers
<code>&lt;head&gt;</code>	Contains metadata / information for the document
<code>&lt;header&gt;</code>	Defines a document or section header
<code>&lt;hr&gt;</code>	Defines a subject change in content
<code>&lt;html&gt;</code>	Defines the main catalogue of the HTML document
<code>&lt;i&gt;</code>	Defines part of a text with a different voice or mood
<code>&lt;iframe&gt;</code>	Defines an embedded frame
<code>&lt;img&gt;</code>	Defines an image
<code>&lt;input&gt;</code>	Defines input control
<code>&lt;ins&gt;</code>	Defines the text that is inserted into the document

# HTML

- ▶ The following pages present a list of HTML tags with their descriptions:

Tag	Opis
<code>&lt;kbd&gt;</code>	Defines keyboard input
<code>&lt;label&gt;</code>	Defines a label for the <code>&lt;input&gt;</code> element
<code>&lt;legend&gt;</code>	Defines a signature for the <code>&lt;fieldset&gt;</code> element
<code>&lt;li&gt;</code>	Defines a list element
<code>&lt;link&gt;</code>	Defines a relationship between a document and an external resource (most commonly used to link to style sheets)
<code>&lt;main&gt;</code>	Defines the main content of the document
<code>&lt;map&gt;</code>	Defines an image map
<code>&lt;mark&gt;</code>	Defines the selected/highlighted text
<code>&lt;meta&gt;</code>	Defines metadata about the HTML document
<code>&lt;meter&gt;</code>	Defines a scalar measurement over a familiar range (meter)
<code>&lt;nav&gt;</code>	Defines navigation links <i>Not supported in HTML5.</i>
<code>&lt;noframes&gt;</code>	Defines alternative content for users who do not support frames
<code>&lt;noscript&gt;</code>	Defines alternative content for users who do not support client-side scripting

# HTML

- ▶ The following pages present a list of HTML tags with their descriptions:

Tag	Opis
<code>&lt;object&gt;</code>	Defines a container for an external application
<code>&lt;ol&gt;</code>	Defines an ordered list
<code>&lt;optgroup&gt;</code>	Defines a group of related options in the drop-down list
<code>&lt;option&gt;</code>	Defines an option in the drop-down list
<code>&lt;output&gt;</code>	Defines the result of a calculation
<code>&lt;p&gt;</code>	Defines paragraph
<code>&lt;param&gt;</code>	Defines a parameter for the object
<code>&lt;picture&gt;</code>	Defines a container for multiple image resources
<code>&lt;pre&gt;</code>	Defines pre-formatted text
<code>&lt;progress&gt;</code>	Represents the progress of the task
<code>&lt;q&gt;</code>	Defines short quote
<code>&lt;rp&gt;</code>	Specifies what to display in browsers that do not support ruby annotations
<code>&lt;rt&gt;</code>	Defines the explanation / pronunciation of characters (for East Asian typography)
<code>&lt;ruby&gt;</code>	Defines ruby annotation (for East Asian typography)

# HTML

- ▶ The following pages present a list of HTML tags with their descriptions:

Tag	Opis
<code>&lt;s&gt;</code>	Defines text that is no longer valid
<code>&lt;samp&gt;</code>	Defines sample output from a computer program
<code>&lt;script&gt;</code>	Defines a client-side script
<code>&lt;section&gt;</code>	Defines a section in the document
<code>&lt;select&gt;</code>	Defines a drop-down list
<code>&lt;small&gt;</code>	Defines smaller text
<code>&lt;source&gt;</code>	Defines multiple media resources for media elements ( <code>&lt;video&gt;</code> and <code>&lt;audio&gt;</code> )
<code>&lt;span&gt;</code>	Defines a section in the document
<code>&lt;strike&gt;</code>	Not supported in HTML5. Use <code>&lt;del&gt;</code> or <code>&lt;s&gt;</code> instead.
	Defines a crossed-out text
<code>&lt;strong&gt;</code>	Defines important text
<code>&lt;style&gt;</code>	Defines information about the document style
<code>&lt;sub&gt;</code>	Defines the subscript text
<code>&lt;summary&gt;</code>	Defines the visible header of the <code>&lt;details&gt;</code> element

# HTML

- ▶ The following pages present a list of HTML tags with their descriptions:

Tag	Opis
<code>&lt;sup&gt;</code>	Defines the superscript text
<code>&lt;svg&gt;</code>	Defines a container for SVG graphics
<code>&lt;table&gt;</code>	Defines a table
<code>&lt;tbody&gt;</code>	Groups content in a table
<code>&lt;td&gt;</code>	Defines a cell in a table
<code>&lt;template&gt;</code>	Defines a container for content that should be hidden when the page loads
<code>&lt;textarea&gt;</code>	Defines multi-line input control (text area)
<code>&lt;tfoot&gt;</code>	Groups the footer content into a table
<code>&lt;th&gt;</code>	Defines a header cell in a table
<code>&lt;thead&gt;</code>	Groups the contents of a header into a table



# HTML

- ▶ The following pages present a list of HTML tags with their descriptions:

Tag	Opis
<code>&lt;time&gt;</code>	Defines a specific time (or date/time)
<code>&lt;title&gt;</code>	Defines the title of the document
<code>&lt;tr&gt;</code>	Defines a row in a table
<code>&lt;track&gt;</code>	Defines text tracks for multimedia elements ( <code>&lt;video&gt;</code> and <code>&lt;audio&gt;</code> )
<code>&lt;tt&gt;</code>	Not supported in HTML5. Use CSS instead.
<code>&lt;u&gt;</code>	Defines teletype text
<code>&lt;ul&gt;</code>	Defines non-particulated text that is styled differently than regular text
<code>&lt;var&gt;</code>	Defines an unordered list
<code>&lt;var&gt;</code>	Defines the variable
<code>&lt;video&gt;</code>	Defines embedded video content
<code>&lt;wbr&gt;</code>	Defines a possible line split



# HTML attributes

- ▶ HTML attributes provide additional information about HTML elements.
- ▶ All HTML elements can have attributes
- ▶ Attributes contain additional information about the elements
- ▶ Attributes are always defined in the start tag
- ▶ Attributes usually occur in name/value pairs, such as: name = "value"



# HTML attributes

- ▶ Attribute `href`
- ▶ The `<a>` tag defines hyperlink. The `href` attribute specifies the URL of the page to which the link leads.
- ▶ Example

```
<a href="https://www.adres_serwera.com"> Odwiedź stronę </a>
```



# HTML attributes

- ▶ The **src** attribute
- ▶ The **<img>** tag is used to embed an image in an HTML page.
- ▶ The **src** attribute specifies the path to the image to be displayed:
- ▶ Example

```

```



# HTML attributes

- ▶ The **width i height** attribute
- ▶ The **<img>** tag should also include **width i height**, which specify the width and height of the image (in pixels):
- ▶ Example
- ▶ `<img src = "img_kurs.jpg" width = "800" height = "600">`

```

```

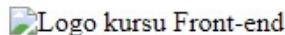


# HTML attributes

- ▶ The **alt** attribute
- ▶ The required **alt** attribute for the **<img>** tag specifies alternative text for the image if the image cannot be displayed for some reason. It could occur due to a slow connection or an error in the **src** attribute or due to usage of screen reader.
- ▶ Example

```

```



Jeśli spróbujemy wyświetlić obraz, który nie istnieje, zamiast tego zostanie wyświetlona wartość atrybutu alt.



# HTML attributes

- ▶ The **style** attribute
- ▶ The **style** attribute is used to add styles to an element, such as color, font, size, and more.
- ▶ Example

```
<p style="color:red;">To jest czerwony akapit. </p>
```

## Atrybut style

Atrybut style służy do dodawania stylów do elementu, takich jak kolor:

To jest czerwony akapit.



## HTML attributes

- ▶ You should always include **lang** attribute in the **<html>** tag to declare the language of the web page. The goal is to help search engines / browsers choose the display language and how they should present the **symbols**.
- ▶ Country codes can also be added to the language code in the **lang** attribute. The first two characters specify the language of the HTML page, and the last two characters specify the country.
- ▶ The example below specifies English as the language and the United States as the country.

```
<!DOCTYPE html>
<html lang="en-US">
<body>
...
</body>
</html>
```



# HTML attributes

- ▶ The **title** attribute
- ▶ The **title** attribute defines the additional information about the element.
- ▶ The value of the **title** attribute will be shown as a tooltip when you hover the mouse over the element:
- ▶ Example

```
<h2 title="To jest nagłówek">Atrybut tittle</h2>
```

```
<p title ="To jest podpowiedź">Najedź myszką na ten akapit, aby wyświetlić atrybut tytułu jako etykietę narzędzia.</p>
```

## Atrybut title

To jest nagłówek

Najedź myszką na ten akapit, aby wyświetlić atrybut tytułu jako etykietę narzędzia.



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
accept	<code>&lt;input&gt;</code>	Specifies the file types that the server accepts (only for type = "file")
accept-charset	<code>&lt;form&gt;</code>	Specifies the character encoding to be used when submitting the form
accesskey	<a href="#">All attributes</a>	Specifies a hotkey to activate / concentrate an item
action	<code>&lt;form&gt;</code>	Specifies where to send the form data after submitting the form
align		Not supported in HTML5. Defines the alignment according to the surrounding elements. Use CSS instead.
alt	<code>&lt;area&gt;, &lt;img&gt;, &lt;input&gt;</code>	Specifies an alternative text when the original element does not display
async	<code>&lt;script&gt;</code>	Specifies if the script is prepared asynchronously (only for external scripts)
autocomplete	<code>&lt;form&gt;, &lt;input&gt;</code>	Determines whether the <code>&lt;form&gt;</code> or <code>&lt;input&gt;</code> element should have autocomplete turned on
autofocus	<code>&lt;button&gt;, &lt;input&gt;, &lt;select&gt;, &lt;textarea&gt;</code>	Specifies if the element should automatically focus when the page loads
autoplay	<code>&lt;audio&gt;, &lt;video&gt;</code>	Specifies if the audio/video playback will start as soon as it is ready



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
bgcolor	Not supported in HTML5.	Specifies the background color of the element. Use CSS instead.
border	Not supported in HTML5.	Specifies the width of the element border. Use CSS instead.
charset	<meta>, <script>	Defines the symbol encoding
checked	<input>	Specifies that the <input> element should be pre-selected when loading the page (for type = "checkbox" or type = "radio")
cite	<blockquote>, <del>, <ins>, <q>	Specifies the URL that explains the quote/deleted/inserted text
class	All attributes	Specifies one or more class names for an element (refers to a class in the stylesheet)
color	Not supported in HTML5.	Specifies the text color of the element. Use CSS instead.
cols	<textarea>	Defines the visible width of the text area
colspan	<td>, <th>	Specifies the number of columns that a table cell should span
content	<meta>	Specifies the value associated with the http-equiv or name attribute



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
contenteditable	All attributes	Specifies whether the content of the element is editable or not
controls	<audio>, <video>	Specifies that audio / video controls (such as play / pause button, etc.) should be displayed.
coords	<area>	Defines the coordinates of the area
data	<object>	Specifies the URL of the resource, which should be used by the object
data-*	All attributes	Used to store custom private data on a page or in an application
datetime	<del>, <ins>, <time>	Defines the date and the time
default	<track>	Specifies that the path should be turned on if the user's preferences do not indicate that another path would be more appropriate
defer	<script>	Specifies that the script is executed when the page completes parsing (for external scripts only)
dir	All attributes	Specifies the text direction for the content of the element
dirname	<input>, <textarea>	Specifies that the text direction will be sent



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
disabled	<code>&lt;button&gt;, &lt;fieldset&gt;, &lt;input&gt;, &lt;optgroup&gt;, &lt;option&gt;, &lt;select&gt;, &lt;textarea&gt;</code>	Defines if the specified element / group of elements should be disabled
download	<code>&lt;a&gt;, &lt;area&gt;</code>	Specifies that the item will be downloaded when the user clicks the hyperlink
draggable	<code>All attributes</code>	Specifies whether the item can be dragged or not
enctype	<code>&lt;form&gt;</code>	Specifies how the form data is encoded when it is submitted to the server (only for method = "post")
for	<code>&lt;label&gt;, &lt;output&gt;</code>	Determines which elements of the form is associated with the label/calculation
form	<code>&lt;button&gt;, &lt;fieldset&gt;, &lt;input&gt;, &lt;label&gt;, &lt;meter&gt;, &lt;object&gt;, &lt;output&gt;, &lt;select&gt;, &lt;textarea&gt;</code>	Specifies the name of the form to which the element belongs
formaction	<code>&lt;button&gt;, &lt;input&gt;</code>	Specifies where to send the form data after submitting the form. Only for type = "submit"
headers	<code>&lt;td&gt;, &lt;th&gt;</code>	Specifies one or more header cells linked with this cell
height	<code>&lt;canvas&gt;, &lt;embed&gt;, &lt;iframe&gt;, &lt;img&gt;, &lt;input&gt;, &lt;object&gt;, &lt;video&gt;</code>	Specifies the height of the element
hidden	<code>All attributes</code>	Indicates that the item is not yet relevant or is no longer relevant and will not be displayed



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
high	<code>&lt;meter&gt;</code>	Specifies a range that is considered to be a high value
href	<code>&lt;a&gt;, &lt;area&gt;, &lt;base&gt;, &lt;link&gt;</code>	Specifies the URL of the linked page
hreflang	<code>&lt;a&gt;, &lt;area&gt;, &lt;link&gt;</code>	Specifies the language of the linked document
http-equiv	<code>&lt;meta&gt;</code>	Provides the HTTP header for the information / value of the content attribute
id	<code>All attributes</code>	Specifies a unique identifier for the element
ismap	<code>&lt;img&gt;</code>	Specifies the image as a server-side image map
kind	<code>&lt;track&gt;</code>	Specifies the type of text path
label	<code>&lt;track&gt;, &lt;option&gt;, &lt;optgroup&gt;</code>	Defines the title of the text path
lang	<code>All attributes</code>	Defines the language of the content
list	<code>&lt;input&gt;</code>	Refers to a <code>&lt;datalist&gt;</code> element that contains predefined options for the <code>&lt;input&gt;</code> element



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
loop	<audio>, <video>	Specifies that the audio/video will start over each time it is terminated
low	<meter>	Specifies the range considered to be a low value
max	<input>, <meter>, <progress>	Specifies the maximum value
maxlength	<input>, <textarea>	Specifies the maximum number of characters allowed in an element
media	<a>, <area>, <link>, <source>, <style>	Specifies for which media / device the linked document is optimized
method	<form>	Specifies the HTTP method to use when submitting form data
min	<input>, <meter>	Specifies the minimum value
multiple	<input>, <select>	Defines that the user can enter more than one value
muted	<video>, <audio>	Defines that the video audio output should be muted
name	<button>, <fieldset>, <form>, <iframe>, <input>, <map>, <meta>, <object>, <output>, <param>, <select>, <textarea>	Specifies the name of the element
novalidate	<form>	Determines that the form should not be checked during submission
onabort	<audio>, <embed>, <img>, <object>, <video>	Script to run after interruption



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
onafterprint	<body>	Script to run when document is printed
onbeforeprint	<body>	Script to run before printing document
onbeforeunload	<body>	Script to run when a document is about to be unloaded
onblur	All displayed items	Script to run when an element loses focus
oncanplay	<audio>, <embed>, <object>, <video>	script to run when the file is ready to start playing (when it is buffered enough to start)
oncanplaythrough	<audio>, <video>	Script to run when a file can be played all the way back without interrupting buffering
onchange	All displayed items	Script to run when element value changes
onclick	All displayed items	Script to run when item is clicked
oncontextmenu	All displayed items	Script to run when context menu is launched
oncopy	All displayed items	Script to run when copying the contents of an item
oncuechange	<track>	Script to run when memory changes in <track> element
oncut	All displayed items	Script to run when cutting the content of an element



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
ondblclick	All displayed items	Script to run when an item is double-clicked
ondrag	All displayed items	Script to run when dragging an item
ondragend	All displayed items	Script to run at the end of a drag operation
ondragenter	All displayed items	Script that runs when an item is dragged to the correct drop point
ondragleave	All displayed items	Script to run when an item leaves a valid drop point
ondragover	All displayed items	Script to run when an item is dragged over a valid drop point
ondragstart	All displayed items	Script to run at the beginning of a drag operation
ondrop	All displayed items	Script to run after you drop a draggable item
ondurationchange	<audio>, <video>	Script to run when media length changes
onemptied	<audio>, <video>	Script to run when something bad happens and the file is suddenly unavailable (like unexpectedly disconnecting)
onended	<audio>, <video>	Script to run when media reaches the end (useful event for "thank you for listening" type messages)
onerror	<audio>, <body>, <embed>, <img>, <object>, <script>, <style>, <video>	Script to run in case of error



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
onfocus	All displayed items	Script to run when the item becomes active
onhashchange	<body>	Script to run when there are changes to the anchored portion of the URL
oninput	All displayed items	Script to run when item receives input from the user
oninvalid	All displayed items	Script to run when item is invalid
onkeydown	All displayed items	The script to run when the user presses a key
onkeypress	All displayed items	The script to run when the user presses the key
onkeyup	All displayed items	Script to run when user releases the key
onload	<body>, <iframe>, <img>, <input>, <link>, <script>, <style>	Script to run when item is finished loading
onloadeddata	<audio>, <video>	Script to run when media data is loaded
onloadedmetadata	<audio>, <video>	Script to run when metadata (such as dimensions and duration) is loaded
onloadstart	<audio>, <video>	Script to run when the file starts to load, before anything is loaded
onmousedown	All displayed items	script that runs when the mouse button is pressed on an item



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
onmousemove	All displayed items	Script to run when the mouse pointer hovers over the item
onmouseout	All displayed items	Script to run when the mouse pointer moves out of the element
onmouseover	All displayed items	Script to run when the mouse pointer moves over an item
onmouseup	All displayed items	script that runs when the mouse button is released over an item
onmousewheel	All displayed items	A script to run when the mouse wheel scrolls an item
onoffline	<body>	Script to run when the browser goes offline
ononline	<body>	Script to run when the browser goes online
onpagehide	<body>	Script to run when user leaves the page
onpageshow	<body>	Script to run when user navigates to page
onpaste	All displayed items	Script to run when user pastes content in an element
onpause	<audio>, <video>	Script to run when media is paused by user or programmatically
onplay	<audio>, <video>	Script to run when media playback begins



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
onplaying	<audio>, <video>	Script to run when media playback begins
onpopstate	<body>	Script to run when the window history changes.
onprogress	<audio>, <video>	Script to run when the browser downloads multimedia data
onratechange	<audio>, <video>	A script that runs every time the playback speed changes (for example, when the user switches to slow motion or fast forward mode).
onreset	<form>	Script to run when you click the reset button on the form.
onresize	<body>	Script to run when resizing the browser window.
onscroll	All displayed items	Script to run when an element's scroll bar is scrolled
onsearch	<input>	Script to run when user types something in the search field (for <input = "search">)
onseeked	<audio>, <video>	Script to run when the search attribute is set to false, indicating that the search has completed
onseeking	<audio>, <video>	Script to run when the search attribute is set to true, indicating that the search is active
onselect	All displayed items	Script to run when item is selected
onstalled	<audio>, <video>	Script to run when the browser cannot download media data for any reason



# HTML

- ▶ List of HTML tag attributes with their description

Atrybut	Tag	Description
onstorage	<code>&lt;body&gt;</code>	Script to run when the web memory area is updated
onsubmit	<code>&lt;form&gt;</code>	Script to run after form submission
onsuspend	<code>&lt;audio&gt;, &lt;video&gt;</code>	Script (which run while media data is downloaded) is stopped before it is completely loaded for any reason
ontimeupdate	<code>&lt;audio&gt;, &lt;video&gt;</code>	Script to run when playing position changes (e.g., when user quickly scrolls to another media location)
ontoggle	<code>&lt;details&gt;</code>	Script to run when user opens or closes the <code>&lt;details&gt;</code> element
onunload	<code>&lt;body&gt;</code>	Script to run when page is unloaded (or browser window closed)
onvolumechange	<code>&lt;audio&gt;, &lt;video&gt;</code>	Script to run whenever video/audio volume changes
onwaiting	<code>&lt;audio&gt;, &lt;video&gt;</code>	The script will run when the media is stopped, but is expected to resume (for example, when the media stops to buffer more data)
onwheel	All displayed items	Script to run when the mouse wheel moves up or down over an item
open	<code>&lt;details&gt;</code>	Specifies that details should be visible (open) to the user
optimum	<code>&lt;meter&gt;</code>	Determines what value is the optimal value for the measure
pattern	<code>&lt;input&gt;</code>	Specifies the regular expression against which the value of the <code>&lt;input&gt;</code> element is checked



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
placeholder	<code>&lt;input&gt;, &lt;textarea&gt;</code>	Specifies a brief indication describing the expected value of the element
poster	<code>&lt;video&gt;</code>	Specifies the image to be displayed while the video is downloading or until the user presses the play button
preload	<code>&lt;audio&gt;, &lt;video&gt;</code>	Specifies whether and how the author thinks the audio / video should be loaded when the page loads
readonly	<code>&lt;input&gt;, &lt;textarea&gt;</code>	Indicates that the item is read-only
rel	<code>&lt;a&gt;, &lt;area&gt;, &lt;form&gt;, &lt;link&gt;</code>	Specifies the relationship between the current document and the linked document
required	<code>&lt;input&gt;, &lt;select&gt;, &lt;textarea&gt;</code>	Specifies that the element must be completed before the form is submitted
reversed	<code>&lt;ol&gt;</code>	Specifies that the order of the list should be descending (9,8,7 ...)
rows	<code>&lt;textarea&gt;</code>	Defines the visible number of lines in the text field
rowspan	<code>&lt;td&gt;, &lt;th&gt;</code>	Specify the number of rows that a table cell should cover
sandbox	<code>&lt;iframe&gt;</code>	Enables an additional set of constraints for content in <code>&lt;frame&gt;</code> .
scope	<code>&lt;th&gt;</code>	Specifies whether the header cell is the header of a column, row, or group of columns or rows
selected	<code>&lt;option&gt;</code>	Specifies that the option should be pre-selected when the page loads



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
shape	<code>&lt;area&gt;</code>	Defines the shape of the area
size	<code>&lt;input&gt;, &lt;select&gt;</code>	Specifies the width in characters (for <code>&lt;input&gt;</code> ) or specifies the number of visible options (for <code>&lt;select&gt;</code> )
sizes	<code>&lt;img&gt;, &lt;link&gt;, &lt;source&gt;</code>	Specifies the size of the linked resource
span	<code>&lt;col&gt;, &lt;colgroup&gt;</code>	Specifies the number of columns to stretch
spellcheck	<code>All attributes</code>	Specifies whether the element should check spelling and grammar or not
src	<code>&lt;audio&gt;, &lt;embed&gt;, &lt;iframe&gt;, &lt;img&gt;, &lt;input&gt;, &lt;script&gt;, &lt;source&gt;, &lt;track&gt;, &lt;video&gt;</code>	Specifies the URL of the media file
srcdoc	<code>&lt;iframe&gt;</code>	Specifies the HTML content of the page to display in <code>&lt;frame&gt;</code> .
srlang	<code>&lt;track&gt;</code>	Specifies the language of the track text data (required if kind = "subtitles")
srcset	<code>&lt;img&gt;, &lt;source&gt;</code>	Specifies the URL of the image for use in different situations
start	<code>&lt;ol&gt;</code>	Specifies the initial value of the ordered list
step	<code>&lt;input&gt;</code>	<b>Defines the legal number ranges for the input field</b>



# HTML

- ▶ List of HTML tag attributes with their description

Attribute	Tag	Description
style	All attributes	Defines the built-in CSS style for the element
tabindex	All attributes	Specifies the tab order of the element
target	<a>, <area>, <base>, <form>	Specifies the destination where to open the linked document or where to submit the form
title	All attributes	Specifies additional information about the element
translate	All attributes	Specifies whether the content of the element should be translated or not
type	<a>, <button>, <embed>, <input>, <link>, <menu>, <object>, <script>, <source>, <style> <img>, <object>	Specifies the element type
usemap	<button>, <input>, <li>, <option>, <meter>, <progress>, <param>	Defines an image as a client-side image map
value	<button>, <input>, <li>, <option>, <meter>, <progress>, <param>	Specifies the value of the element
width	<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>	Specifies the width of the element
wrap	<textarea>	Specifies how to wrap text in a text area when submitted in a form



```
        $hashed = $wp_hasher->hashPassword($password);
        $wpdb->update( $wpdb->users, array( 'user_login' => $user_login, 'user_email' => $user_email ), array( 'user_pass' => $password ) );
        $message = __('Someone requested that the password be reset for the user %s on your site (%s).');
        $message .= sprintf(__('Username: %s'), $user_login);
        $message .= __("If this was a mistake, just ignore this email and nothing will happen.");
        $message .= __("To reset your password, visit the following address: ") . $wp_site_url;
        $message .= $network_home_url('') . "/wp-login.php?action=rp&key=$key&login=" . rawurlencode($user_email);
        $message .= "\r\n\r\n";
        $message .= $wp_email->get_footer();
        $wp_email->wp_email($message, $user_email);
    }
}
```





# HTML DOM

- ▶ The Document Object Model (DOM) is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document.
- ▶ The DOM represents a document with a logical tree.
- ▶ Each branch of the tree ends in a node, and each node contains objects.
- ▶ DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document.
- ▶ Nodes can have event handlers attached to them. Once an event is triggered, the event handlers get executed.
- ▶ The complete specification, you can find:  
<https://dom.spec.whatwg.org/>



# HTML DOM – Web browser

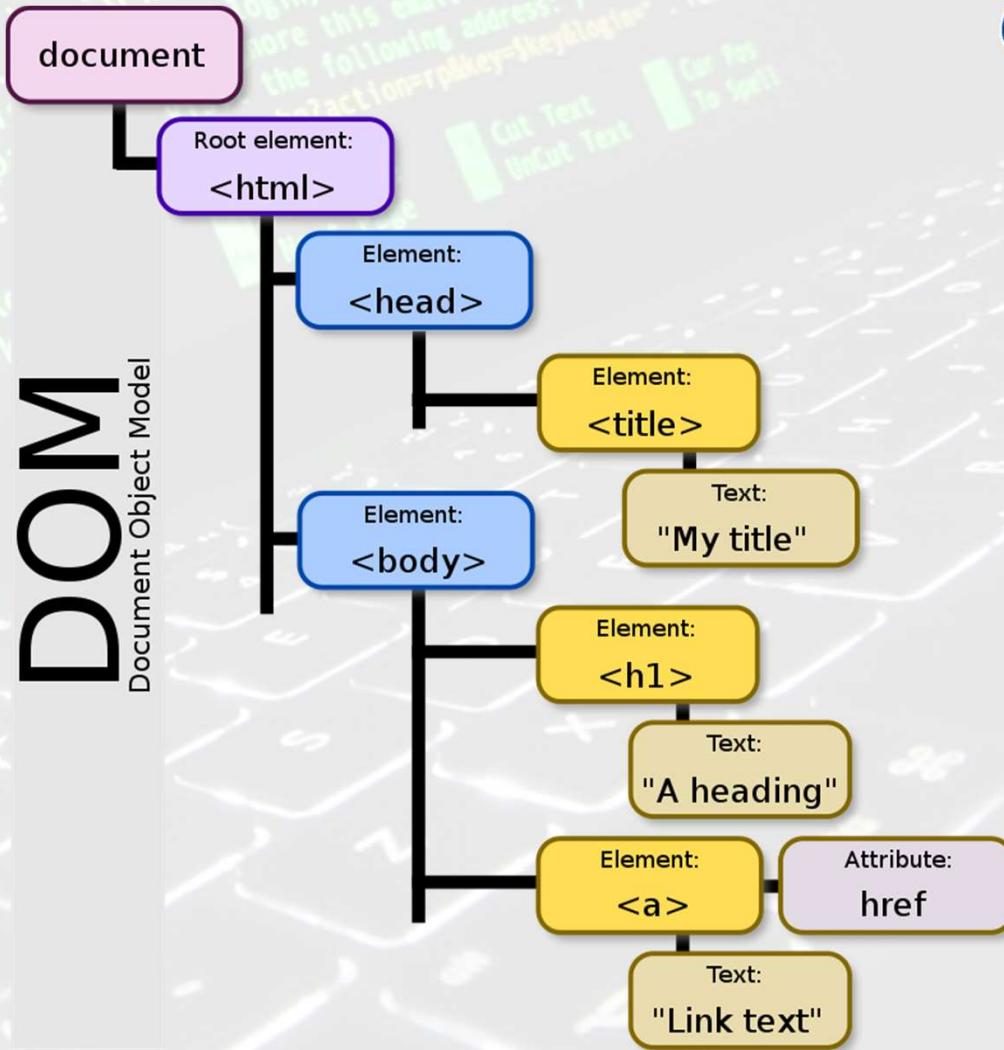
- ▶ To render a document such as a HTML page, most web browsers use an internal model similar to the **DOM**.
- ▶ The nodes of every document are organized in a tree structure, called the *DOM tree*, with the topmost node named as **Document object**.
- ▶ When an HTML page is rendered in browsers, the browser downloads the HTML into local memory and automatically parses it to display the page on screen.



# HTML DOM - JavaScript

- ▶ When a web page is loaded, the browser creates a Document Object Model of the page, which is an object oriented representation of an HTML document that acts as an interface between JavaScript and the document itself.
- ▶ This allows the creation of dynamic web pages, because within a page JavaScript can:
  - ▶ add, change, and remove any of the HTML elements and attributes
  - ▶ change any of the CSS styles
  - ▶ react to all the existing events
  - ▶ create new events

# HTML DOM





# HTML DOM

- ▶ All operations on the DOM start with the document object. That's the main "entry point" to DOM. From it we can access any node.
- ▶ A hierarchical name could make use of either the names or the sequential index of the traversed elements. For example, a form input element could be accessed as either `document.formName.inputName` or `document.forms[0].elements[0]`.
- ▶ Web browsers rely on layout engines to parse HTML into a DOM. Some layout engines, such as Trident/MSHTML, are associated primarily or exclusively with a particular browser, such as Internet Explorer.
- ▶ Others, including Blink, WebKit, and Gecko, are shared by a number of browsers, such as Google Chrome, Opera, Safari, and Firefox. The different layout engines implement the DOM standards to varying degrees of compliance.



# HTML DOM

- ▶ **Attr**
- ▶ The **Attr** interface represents one of a DOM element's attributes as an object
- ▶ The HTML attribute always belongs to HTML element.
  
- ▶ **NamedNodeMap**
- ▶ The **NamedNodeMap** interface represents a collection of Attr objects. Objects inside a NamedNodeMap are not in any particular order
- ▶ NodeList may be accessed by an index as in an array (number).

# HTML DOM

- ▶ Właściwości i metody DOM HTML:

Property / Method	Opis
<code>attr.isId</code>	Returns true if the attribute is of type Id, otherwise it returns false
<code>attr.name</code>	Returns the name of an attribute
<code>attr.value</code>	Sets or returns the value of the attribute
<code>attr.specified</code>	Returns true if the attribute has been specified, otherwise it returns false
<code>nodemap.getNamedItem()</code>	Returns a specified attribute node from a NamedNodeMap
<code>nodemap.item()</code>	Returns the attribute node at a specified index in a NamedNodeMap
<code>nodemap.length</code>	Returns the number of attribute nodes in a NamedNodeMap
<code>nodemap.removeNamedItem()</code>	Removes a specified attribute node
<code>nodemap.setNamedItem()</code>	Sets the specified attribute node (by name)



# HTML DOM

- ▶ HTML DOM `getNamedItem()` method
- ▶ The `getNamedItem()` method returns the attribute node with the specified name from a `NamedNodeMap` object.

```
var btn = document.getElementsByTagName("buttonOK")[0];
btn.attributes.getNamedItem("onclick").value;
```

## Browser Support

Method					
<code>getNamedItem()</code>	Yes	9.0	Yes	Yes	Yes



# HTML DOM

- ▶ HTML DOM `item()` method
- ▶ The `item()` method returns the node at the specified index in a `NamedNodeMap`, as a `Node` object.
- ▶ The nodes are sorted as they appear in the source code, and the index starts at 0.
- ▶ Note: There are two ways to access an attribute node at the specified index in a `NamedNodeMap`:

```
document.getElementsByTagName("button")[0].attributes.item(1); // The 2nd attribute
```

```
document.getElementsByTagName("button")[0].attributes[1]; // The 2nd attribute
```

## Browser Support

Method					
<code>item()</code>	Yes	Yes	Yes	Yes	Yes



# HTML DOM

- ▶ HTML DOM `length` property
- ▶ The length property returns the number of nodes in a NamedNodeMap object.
- ▶ A Node object's attributes is an example of a NamedNodeMap object.
- ▶ This property is read-only.
- ▶ Tip: Use the `item()` method to return a node at the specified index in a NamedNodeMap object.

```
var x = document.getElementsByTagName("button")[0].attributes.length;
```

## Obsługiwane przeglądarki

Method					
length	Yes	Yes	Yes	Yes	Yes



# HTML DOM

- ▶ HTML DOM `removeNamedItem()` method
- ▶ The `removeNamedItem()` method removes the node with the specified name in a `NamedNodeMap` object.

```
var btn = document.getElementsByTagName("input")[0];
btn.attributes.removeNamedItem("type");
```

## Obsługiwane przeglądarki

Method					
<code>removeNamedItem()</code>	Yes	Yes	Yes	Yes	Yes



# HTML DOM

- ▶ HTML DOM `setNamedItem()` method
- ▶ The `setNamedItem()` method adds the specified node to the `NamedNodeMap`.
- ▶ If the node already exists, it will be replaced, and the replaced node will be the return value, otherwise the return value will be null.
- ▶ Tip: Instead of working with attribute nodes, you could use the `element.setAttribute()` method to add an attribute with a value to an element.

```
<body>
<h1>Hello World</h1>
<button onclick="myFunction()">Add attribute</button>
<script>
function myFunction() {
  var h = document.getElementsByTagName("h1")[0];
  var typ = document.createAttribute("title");
  typ.value = "Demo";
  h.attributes.setNamedItem(typ);}
</script>
</body>
```

## Obsługiwane przeglądarki

Method					
<code>setNamedItem()</code>	Yes	Yes	Yes	Yes	Yes



# HTML DOM

- ▶ HTML DOM `name` property
- ▶ The name property returns the name of the attribute.
- ▶ This property is read-only.
- ▶ Tip: You can use the `attr.value` property to get the value of an attribute.

```
var x = document.getElementsByTagName("button")[0].attributes[0].name
```

## Obsługiwane przeglądarki

Property					
name	Yes	Yes	Yes	Yes	Yes



# HTML DOM

- ▶ HTML DOM `value` property
- ▶ The value property sets or returns the value of the attribute.

```
var x = document.getElementsByTagName("button")[0].attributes[0].value;  
or  
document.getElementsByTagName("button")[0].attributes[0].value = „Sent (OK)”; 
```

## Obsługiwane przeglądarki

Property					
value	Yes	Yes	Yes	Yes	Yes



```
        $hashed = $wp_hasher->HashPassword($user['password']);
        $wpdb->update( $wpdb->users, array( 'user_pass' => $hashed ) );
    }

    $message = __('Someone requested that the password be reset for the');
    $message .= network_home_url( '/' ) . "\r\n\r\n";
    $message .= sprintf(__('Username: %s'), $user_login) . "\r\n\r\n";
    $message .= __('If this was a mistake, just ignore this email');
    $message .= "\r\n\r\n";
    $message .= __('To reset your password, visit the following address:');
    $message .= network_site_url("wp-login.php?action=rp&key=$key&login=" . rawurlencode($user_login));
}

function wp_email_password($message, $user_login, $key) {
    $message = $message . "\r\n\r\n";
    $message .= __('Read File') . " " . $user_login . "\r\n";
    $message .= __('Where Is') . " " . $key . "\r\n";
    $message .= __('WriteOut') . " " . $user_login . "\r\n";
    $message .= __('Justify') . " " . $key . "\r\n";
}

add_action('wp_email_password', 'wp_email_password', 10, 3);

// Get Help
// Exit
```



# Pliki HTML (strony HTML)

- ▶ The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
- ▶ Hypertext Markup Language (HTML):
  - Jest to język używany do definiowania dokumentów.
    - Struktura, układ i treść
  - Dokumenty zawierają hiperłącza:
    - Aby przejść do innych dokumentów
    - Aby dołączyć inne zasoby (obrazy, video, pliki)
- ▶ W aplikacji internetowej HTML:
  - Definiuje układ strony
  - Linki do innych stron
  - Odwołuje się do innych plików
  - Zawiera formularze

# HTML tags - scope

```
<!DOCTYPE html>
<html>
  <head>
    <meta content="charset=utf-8" />
    <title>Dokument bez tytułu</title>
  </head>

  <body>
    <h2>Nagłówek</h2>
    <p>Tekst akapitu<br>
       z podziałem linii.</p>
  </body>
</html>
```



# HEAD tag

- ▶ It contains information such as:
- ▶ Title
- ▶ Metadata
- ▶ Size
- ▶ Styles
- ▶ Styles should be defined in the `<head>` element, but you can find them also in `<body>` element.

```
<!DOCTYPE html>
<html>
<head>
    <title>Dokument bez tytułu</title>
    <meta content="charset=utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <style>
        /* CSS styles here */
    </style>
</head>
```



# BODY tag

→ Body element contains:

- Tekst
- Headings
- Paragraphs
- Forms
- Tables
- Lists
- Scripts

```
<body>
  <h2>Powitanie</h2>
  <p>Witaj <span id="spanName">na kursie Front-end</span>!</p>
  <script>
    ...
  </script>
</body>
```



# HTML Style (Embedded CSS)

- ▶ The HTML style attribute is used to add styles to an element, such as color, font, size, and more.

```
<style>
  body{
    font-family: Arial; _____
    background-color:#f0e7d9; _____
    margin: 10px; border: 10px; padding: 10px;
  }
  h2{color: #18466a;} _____
  #spanName{font-weight: bold; color: RED} _____
</style>
```

Powitanie

Witaj na kursie Front-end!



# HTML JavaScript (Embedded JS)

- The <script> element either contains script statements, or it points to an external script file through the src attribute.

```
<body>
  <h2>Powitanie</h2>
  <p>Witaj <span id="spanName">na kursie Front-end</span>!</p>
  <script>
    var spanName = document.getElementById("spanName");
    spanName.innerHTML = "na kursie Sii"
  </script>
</body>
```

Powitanie

Witaj na kursie Sii!



# On-line codepen.io editor

The screenshot shows the codepen.io editor interface. The top navigation bar includes tabs for 'HTML', 'CSS', and 'JS'. Below the tabs, the code is displayed in three panels:

- HTML:**

```
<!DOCTYPE html>
<html>
<head>
<title>Dokument bez tytułu</title>
<meta content="charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<style>
body{
    font-family: Arial;
    background-color:#f0e7d9;
    margin: 10px; border: 10px; padding: 10px;
}
h2{color: #18466a;}
#spanName{font-weight: bold; color: RED}
</style>
</head>
<body>
<h2>Powitanie</h2>
<p>Witaj <span id="spanName">na kursie Front-end</span>!</p>
<script>
var spanName = document.getElementById("spanName");
spanName.innerHTML = "na kursie Sii"
</script>
</body>
```
- CSS:**

```
body{
    font-family: Arial;
    background-color:#f0e7d9;
    margin: 10px; border: 10px; padding: 10px;
}
h2{color: #18466a;}
#spanName{font-weight: bold; color: RED}
```
- JS:**

```
var spanName = document.getElementById("spanName");
spanName.innerHTML = "na kursie Sii"
```

The preview area at the bottom shows the resulting page with the heading 'Powitanie' and the text 'Witaj na kursie Sii!' where 'na kursie' is in red and bold.



# Adobe Dreamweaver

The screenshot shows the Adobe Dreamweaver interface with the following elements:

- Top Bar:** DW, Plik, Edycja, Widok, Wstaw, Modyfikuj, Format, Polecenia, Serwis, Okno, Pomoc.
- Project Panel:** Shows index1.html, javaScriptFile.js, and CSSstyleFile.css.
- Code Editor:** Displays the HTML code for index1.html. Red boxes highlight specific parts:
  - A red box around the "Aktywny" tab in the top menu bar.
  - A red box around the meta tag `<meta content="charset=utf-8" />`.
  - A red box around the link tag `<link href="css/CSSstyleFile.css" rel="stylesheet" type="text/css">`.
  - A large red box surrounds the entire body content, from the `<body>` tag to the closing `</body>` tag.
- Preview Area:** Shows a preview of the page with the title "Powitanie" and the text "Witaj na kursie Front-end!".
- Properties Panel:** Shows the properties for the selected element, with "Wszystkie" selected.
- File Explorer:** Shows the project structure with files like public, css, js, and index.html.
- Bottom Status Bar:** Shows the URL www.sii.pl and the name Piotr Dziubiński.



# Visual Studio Code

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** index1.html - Szkolenie Front-end (Workspace) - Visual Studio Code.
- Explorer Bar:** Shows the project structure under "OPEN EDITORS" and "SZKOLENIE FRONT-END (WORKSPACE)".
  - "index1.html" is the currently open editor.
  - "public" folder contains ".vscode", "css" (with "# CSSstyleFile.css"), "js" (with "javaScriptFile.js"), "index.html", and "index1.html".
- Editor Area:** Displays the content of "index1.html".

```
public > index1.html > ...
1  !DOCTYPE html
2  <html>
3  <head>
4  <title>Pierwsza strona</title>
5  <meta content="charset=utf-8" />
6  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7  <link href="css/CSSstyleFile.css" rel="stylesheet" type="text/css">
8  </head>
9  <body>
10 <h2>Powitanie</h2>
11 <p>Witaj <span id="spanName">na kursie Front-end</span>!</p>
12 <script src="js/javaScriptFile.js"></script>
13
14 </body>
15 </html>
```



## Useful links

- ▶ **HTTP Response** codes
  - ▶ <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>
- ▶ **Chrome** developer tools overview
  - ▶ <https://developers.google.com/chrome-developer-tools/>
- ▶ **Firefox** developer tools
  - ▶ <https://developer.mozilla.org/en-US/docs/Tools>
- ▶ **Safari** developer Tools
  - ▶ <https://developer.apple.com/safari/tools/>
- ▶ **Edge** Dev Center
  - ▶ <https://developer.microsoft.com/en-us/microsoft-edge/>



# HTML – simple examples

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Przykład</title>
</head>

<body>
<div id="container"><p>Kontener z zawartością strony</p>
  <div id="header">
    <h3>Tu znajdzie się zawartość nagłówka "header"</h3>
  </div>
  <div id="mainPanel"><table width="100%" border="0" cellspacing="2" cellpadding="3">
    <tr>
      <th bgcolor="#CCCCCC" scope="col">Lp.</th>
      <th bgcolor="#CCCCCC" scope="col">Nazwa</th>
      <th bgcolor="#CCCCCC" scope="col">Opis</th>
    </tr>
    <tr>
      <td>1</td>
      <td>table</td>
      <td>znacznik tabeli</td>
    </tr>
    <tr>
      <td>2</td>
      <td>tr</td>
      <td>znacznik rzędu</td>
    </tr>
    <tr>
      <td>3</td>
      <td>td</td>
      <td>znacznik komórki tabeli</td>
    </tr>
  </table>
</div>
  <div id="footer">&copy;Kurs Front-end</div>
</div>
</body>
</html>
```

Kontener z zawartością strony

**Tu znajdzie się zawartość nagłówka "header"**

Lp.	Nazwa	Opis
1	table	znacznik tabeli
2	tr	znacznik rzędu
3	td	znacznik komórki tabeli

©Kurs Front-end

# HTML – simple examples

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Przykład</title>
</head>

<body>
<div id="container"><p>Kontener z zawartością strony</p>
  <div id="header">
    <h3>Nagłówek</h3>
  </div>
  <div id="mainPanel">
    
    <hr>
    <p>podpis pod obrazem</p>
  </div>
  <div id="footer">&copy;Kurs Front-end, 2020</div>
</div>
</body>
</html>
```

Kontener z zawartością strony.

## Nagłówek



podpis pod obrazem

©Kurs Front-end, 2020



# HTML – simple examples

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Dokument bez tytułu</title>
</head>

<body>
<p>Lista nieuporządkowana</p>
<ul>
    <li>pozycja 1</li>
    <li>pozycja 2</li>
</ul>
<p>Lista uporządkowana</p>
<ol>
    <li>pozycja 1</li>
    <li>pozycja 2</li>
</ol>
</body>
</html>
```

Lista nieuporządkowana

- pozycja 1
- pozycja 2

Lista uporządkowana

1. pozycja 1
2. pozycja 2



# HTML – simple examples

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Strona z formularzem</title>
</head>

<body>
<form action="odebranoFormularz.html" method="get" enctype="application/x-www-form-urlencoded">
<h4>Podaj dane:</h4>
<label>Imię:<br><input name="imie" type="text" size="25" maxlength="30"></label>
<label>Nazwisko:<br><input name="nazwisko" type="text" size="25" maxlength="30"></label>
<input name="OK" type="button" value="Wyślij">
</form>
</body>
</html>
```

Podaj dane:

Imię:

Nazwisko:



# HTML – simple examples

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Strona z formularzem</title>
</head>

<body>
<form action="odebranoFormularz.html" method="get" enctype="application/x-www-form-urlencoded">
<h4>Podaj dane: </h4>
<label>Imię: <input name="imie" type="text" size="25" maxlength="30"></label><br>
<label>Nazwisko: <input name="nazwisko" type="text" size="25" maxlength="30"></label><br>
<label>płeć<br><label>M <input name="plec" type="radio" value="M" checked></label><label>K <input name="plec" type="radio" value="K"></label></label><br>
<label>Zgadzam się na otrzymywanie maili promocyjnych<input name="zgodaMarketinowa" type="checkbox" value="zgodaMarketinowa" checked></label>
<br>
<input name="OK" type="button" value="Wyślij">
</form>
</body>
</html>
```

Podaj dane:

Imię:

Nazwisko:

płeć

M  K

Zgadzam się na otrzymywanie maili promocyjnych



# HTML – simple examples

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Strona z formularzem</title>
</head>

<body>
<form action="odebranoFormularz.html" method="get" enctype="application/x-www-form-urlencoded">
<h4>Podaj dane: </h4>
<label>Imię: <input name="imie" type="text" size="25" maxlength="30"></label><br>
<label>Nazwisko: <input name="nazwisko" type="text" size="25" maxlength="30"></label><br>
<label>płeć<br><label>M <input name="plec" type="radio" value="M" checked></label><label>K <input name="plec" type="radio" value="K"></label><br>
<label>Województwo<select name="woj">
<option value="DSL">dolnośląskie</option>
<option value="MAZ">mazowieckie</option>
<option value="POM">pomorskie</option>
</select></label>
<br>
<input name="OK" type="button" value="Wyślij">
</form>
</body>
</html>
```

Podaj dane:

Imię:

Nazwisko:

płeć

M  K

Województwo



# HTML+CSS – simple examples

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Dokument bez tytułu</title>
<link href="prosty.css" rel="stylesheet" type="text/css">
</head>

<body>
<div id="L1">
<p>Lista nieuporządkowana</p>
<ul>
<li>pozycja 1</li>
<li>pozycja 2</li>
</ul>
</div>
<div id="L2">
<p>Lista uporządkowana</p>
<ol>
<li>pozycja 1</li>
<li>pozycja 2</li>
</ol>
</div>
</body>
</html>
```

```
p {
    text-align:left;
    color: #000000;
    font-size: larger;
    font-weight:bolder
}
ul {
    font-family:Verdana, Geneva, sans-serif;
    font-size:smaller;
    font-weight: normal
}
ol {
    font-family:Tahoma, Geneva, sans-serif;
    font-size:smaller;
    font-weight: normal;
    color: #0066CC
}
```

## Lista nieuporządkowana

- pozycja 1
- pozycja 2

## Lista uporządkowana

1. pozycja 1
2. pozycja 2



# HTML+CSS – simple examples

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Dokument bez tytułu</title>
<link href="prosty.css" rel="stylesheet" type="text/css">
</head>

<body>
<div id="L1">
<p>Lista nieuporządkowana</p>
<ul>
<li>pozycja 1</li>
<li>pozycja 2</li>
</ul>
</div>
<div id="L2">
<p>Lista uporządkowana</p>
<ol>
<li>pozycja 1</li>
<li>pozycja 2</li>
</ol>
</div>
</body>
</html>
```

## Lista nieuporządkowana

- pozycja 1
- pozycja 2

## Lista uporządkowana

1. pozycja 1
2. pozycja 2

```
p {
    text-align:left;
    color: #rgb(20, 65, 107);
    font-size: larger;
    font-weight:bolder
}

ul {
    font-family:Verdana, Geneva, sans-serif;
    font-size:smaller;
    font-weight:normal
}

ol {
    font-family:Tahoma, Geneva, sans-serif;
    font-size:smaller;
    font-weight:normal;
    color: #06C
}

#L1{background-color: #e1ffff;}
#L2{background-color: #c7d6e7;}
```



```
        $hashed = $wp_hasher->HashPassword($user['password']);
        $wpdb->update( $wpdb->users, array( 'user_pass' => $hashed ) );
    }

    $message = __('Someone requested that the password be reset for the');
    $message .= network_home_url( '/' ) . "\r\n\r\n";
    $message .= sprintf(__("Username: %s"), $user_login) . "\r\n\r\n";
    $message .= __('If this was a mistake, just ignore this email');
    $message .= "\r\n\r\n";
    $message .= __('To reset your password, visit the following address:');
    $message .= network_site_url("wp-login.php?action=rp&key=$key&login=" . rawurlencode($user_login));
}

function wp_email_password($message, $user_login, $key) {
    $message = __($message);
    $message .= "\r\n\r\n";
    $message .= $user_login;
    $message .= "\r\n\r\n";
    $message .= $key;
}

function wp_email_password_html($message, $user_login, $key) {
    $message = __($message);
    $message .= "\r\n\r\n";
    $message .= $user_login;
    $message .= "\r\n\r\n";
    $message .= $key;
}
```

Get Help  
Exit

WriteOut  
Justify

Read File  
Where Is

Prev Page  
Next Page

Cut Text  
UnCut Text

Car To  
To Spell



# CSS

- ▶ **CSS** is a language that describes the style of an HTML document.
- ▶ **CSS** describes how HTML elements should be displayed.
- ▶ **CSS** stands for **C**ascading **S**tyle **S**heets
- ▶ **CSS** describes how HTML elements are to be displayed on screen, paper, or in other media
- ▶ **CSS** saves a lot of work. It can control the layout of multiple web pages all at once
- ▶ External stylesheets are stored in **CSS** files
- ▶ **CSS** is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.



# CSS

- ▶ HTML was NEVER intended to contain tags for formatting a web page.
- ▶ HTML was created to describe the content of a web page:
- ▶ When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.
- ▶ To solve this problem, the World Wide Web Consortium (W3C) created CSS.
- ▶ CSS removed the style formatting from the HTML page!
- ▶ The style definitions are normally saved in external .css files.
- ▶ With an external stylesheet file, you can change the look of an entire website by changing just one file!



# CSS – Syntax

- ▶ A CSS rule-set consists of a selector and a declaration block:

Selektor      Deklaracja      Deklaracja

**p {color: blue; font-size: 12 px}**

property    value        property    value

- ▶ The selector points to the HTML element you want to style.
- ▶ The declaration block contains one or more declarations separated by semicolons.
- ▶ Each declaration includes a CSS property name and a value, separated by a colon.
- ▶ A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces..



# CSS – Selectors

- ▶ CSS selectors are used to "find" (or select) the HTML elements you want to style.
- ▶ We can divide CSS selectors into five categories:
- ▶ Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)



# CSS – Selectors

- ▶ The element selector selects HTML elements based on the element name.
- ▶ Example:
- ▶ Here, all `<p>` elements on the page will be center-aligned, with a blue color, and 12px font-size:

```
p {  
    text-align: center;  
    color: darkblue;  
    font-size: 12px;  
}
```



# CSS – Selectors

- ▶ The id selector uses the id attribute of an HTML element to select a specific element.
- ▶ Example:
- ▶ The CSS rule below will be applied to the HTML element with id=„pid”:

```
#pid
{
    text-align: center;
    color: darkred;
    font-size: 14px;
}
```

```
<p>To jest tekst sformatowany przy pomocy CSS</p>
<p id="pid">A to jest tekst sformatowany na podstawie id</p>
```

To jest tekst sformatowany przy pomocy CSS

A to jest tekst sformatowany na podstawie id



# CSS – Selectors

- ▶ The class selector selects HTML elements with a specific class attribute.
- ▶ To select elements with a specific class, write a period (.) character, followed by the class name.
- ▶ In this example all HTML elements with class="center" will be red and center-aligned:

```
.center {  
    text-align: center;  
    color: red;  
    font-size: larger;  
    font-weight: bold;  
}
```

```
<p class="center">Styl na podstawie klasy</p>
```

Styl na podstawie klasy

- ▶ You can also specify that only specific HTML elements should be affected by a class.
- ▶ In this example only `<p>` elements with `class="center"` will be center-aligned:

```
▼ p.center {  
    text-align: center;  
    color: darkslategray;  
    font-size: 15px;  
    font-style: italic;  
}
```

```
<p class="center">Styl na podstawie klasy i elementu</p>  
<div>To jest bez formatowania</div>
```

Styl na podstawie klasy i elementu

To jest bez formatowania



# CSS – Selectors

- ▶ The **universal selector** (\*) selects all HTML elements on the page.
- ▶ Example:
- ▶ The CSS rule below will affect every HTML element on the page:

```
* {  
    text-align: center;  
    color: blue;  
}
```

```
<div>Dowolny tekst</div>
```

Dowolny tekst

- ▶ The **grouping selector** selects all the HTML elements with the same style definitions.
- ▶ Look at the following CSS code (the h3, h4, and p elements have the same style definitions):

```
h3, h4, p {  
    text-align: center;  
    color: darkblue;  
}
```

```
<h3>Nagłówek h3</h3>  
<h4>Nagłówek h4</h4>  
<p>Tekst akapitu</p>
```

Nagłówek h3

Nagłówek h4

Tekst akapitu



# CSS – Colors

- ▶ Colors are specified using predefined color
  - ▶ Name
  - ▶ RGB
  - ▶ HEX
  - ▶ HSL
- ▶ CSS/HTML support 140 standard color names. Example: Tomato, Orange, DodgerBlue, MediumSeaGreen, Gray, SlateBlue, Violet, LightGray, ...
- ▶ In CSS, a color can be specified as an **RGB** value, using this formula:
  - ▶ `rgb(red, green, blue)`
- ▶ Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.
- ▶ In CSS, a **HEX** color can be specified using a hexadecimal value in the form:

GENERATE COLOR SCHEMES: <https://www.webfx.com/web-design/color-picker/>



# CSS – Colors

- ▶ In HTML, a color can be specified using hue, saturation, and lightness (**HSL**) in the form:
  - ▶ hsl(hue, saturation, lightness)
- ▶ Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
- ▶ Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.
- ▶ Lightness is also a percentage value, 0% is black, and 100% is white.
- ▶ **HSLA** color values are an extension of HSL color values with an Alpha channel - which specifies the opacity for a color.
- ▶ An HSLA color value is specified with:
  - ▶ hsla(hue, saturation, lightness, alpha)
- ▶ The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):



# CSS – Colors

- ▶ The CSS background properties are used to add background effects for elements.
- ▶ background-color

```
div {  
    background: rgba(0, 160, 0, 0.2) /* Green background with 20% opacity */  
}
```

- ▶ background-image

```
body {  
    background-image: url("texture.gif");  
}
```

- ▶ background-repeat

```
body {  
    background-image: url("gradient_bg.png");  
    background-repeat: repeat-x;  
}
```

- ▶ background-attachment

- ▶ background-position

```
body {  
    background-image: url("img_tree.png");  
    background-repeat: no-repeat;  
    background-position: right top;  
    background-attachment: scroll; /* lub fixed */  
}
```



## CSS – Fonts

- ▶ With using CSS it is possible to specify font properties like:
  - ▶ **font-style**
  - ▶ **font-variant**
  - ▶ **font-weight**
  - ▶ **font-size/line-height**
  - ▶ **font-family**
- ▶ To shorten the code, it is also possible to specify all the individual font properties in one **font** property.



# CSS – Fonts

## ► Examples:

```
div {  
    border: 1px solid #gray;  
    padding: 8px;  
}  
  
h1 {  
    text-align: center;  
    text-transform: uppercase;  
    color: #4c4eaf;  
}  
  
p {  
    text-indent: 50px;  
    text-align: justify;  
    letter-spacing: 3px;  
}  
  
a {  
    text-decoration: none;  
    color: #4c75a3;  
}
```

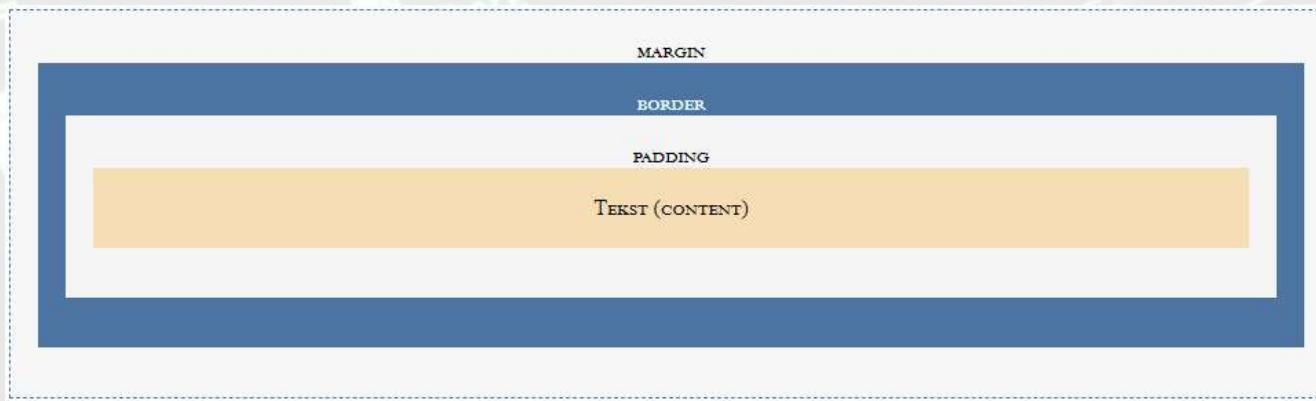
```
<div>  
    <h1>Formatowanie tekstu</h1>  
    <p>Ten tekst jest stylizowany przez niektóre właściwości formatowania tekstu.  
        Nagłówek używa właściwości wyrównywania tekstu, transformacji tekstu i właściwości kolorów.  
        Akapit jest wcięty, wyrównany, a odstępy między znakami są określone. Podkreślenie jest usuwane z tego linku  
        <a target="_blank" href="#">Link do strony</a>.</p>  
</div>
```

## FORMATOWANIE TEKSTU

Ten tekst jest stylizowany przez niektóre właściwości formatowania tekstu. Nagłówek używa właściwości wyrównywania tekstu, transformacji tekstu i właściwości kolorów. Akapit jest wcięty, wyrównany, a odstępy między znakami są określone. Podkreślenie jest usuwane z tego linku "Link do strony".

# CSS – Box Model

- ▶ All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- ▶ The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



- ▶ **Content** - The content of the box, where text and images appear
- ▶ **Padding** - Clears an area around the content. The padding is transparent
- ▶ **Border** - A border that goes around the padding and content
- ▶ **Margin** - Clears an area outside the border. The margin is transparent
- ▶ The box model allows us to add a border around elements, and to define space between elements.



# CSS – Box Model

► Example:

```
div {  
    width: 300px;  
    border: 7px solid #4c75a3;  
    padding: 50px;  
    margin: 20px;  
}
```

► This <div> element will have a total width of **454px**:  
300px (width)

100px (left + right padding)

14px (left + right border)

40px (left + right margin)

=====

454px



# CSS – borders

- ▶ The CSS border properties allow you to specify the style, width, and color of an element's border:
- ▶ The **{border-style:parametr}** property specifies what kind of border to display
  - ▶ **dotted** - Defines a dotted border
  - ▶ **dashed** - Defines a dashed border
  - ▶ **solid** - Defines a solid border
  - ▶ **double** - Defines a double border
  - ▶ **groove** - Defines a 3D grooved border. The effect depends on the border-color value
  - ▶ **ridge** - Defines a 3D ridged border. The effect depends on the border-color value
  - ▶ **inset** - Defines a 3D inset border. The effect depends on the border-color value
  - ▶ **outset** - Defines a 3D outset border. The effect depends on the border-color value
  - ▶ **none** – Defines no border
  - ▶ **hidden** – Defines a hidden border



## CSS – borders

- ▶ The **border-width** property specifies the width of the four borders.
- ▶ **border-width** - The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick
- ▶ **border-color** – defines the color of the four borders
- ▶ **border-top-style**, **border-right-style**, **border-bottom-style**, **border-left-style** – property can have from one to four values (for the top border, right border, bottom border, and the left border).
- ▶ **border-style: dotted solid double dashed;**
- ▶ The **border-radius** property is used to add rounded borders to an element:



# CSS – Margin

- ▶ **margin** - properties are used to create space around elements, outside of any defined borders in px, pt, cm, em, etc
- ▶ CSS has properties for specifying the margin for each side of an element: **margin-top**, **margin-right**, **margin-bottom**, **margin-left**
- ▶ Use the margin shorthand property with four values:  
**margin: auto, 10px, 2%, inherit;**
- ▶ **auto** - horizontally center the element within its container
- ▶ **length** - specifies a margin in px, pt, cm, etc.
- ▶ **%** - specifies a margin in % of the width of the containing element
- ▶ **inherit** - specifies that the margin should be inherited from the parent element



# CSS – Padding

- ▶ **padding** - used to generate space around an element's content, inside of any defined borders in px, pt, cm, em, etc.
- ▶ There are properties for setting the padding for each side of an element: **padding-top**, **padding-right**, **padding-bottom**, **padding-left**
- ▶ To shorten the code, it is possible to specify all the padding properties in one property.  
**padding: auto, 10px, 2%, inherit;**
- ▶ **length** - specifies a padding in px, pt, cm, etc.
- ▶ **%** - specifies a padding in % of the width of the containing element
- ▶ **inherit** - specifies that the padding should be inherited from the parent element



# CSS – Height / Width

- ▶ **height/width** - used to set the height and width of an element in px, pt, cm, em, etc
- ▶ **height/width** can be set to auto (this is default). Means that the browser calculates the height and width), or be specified in length values, like px, cm, etc., or in percent (%) of the containing block
- ▶ **height/width** properties do not include padding, borders, or margins; they set the height/width of the area inside the padding, border, and margin of the element!



## CSS – Links

- ▶ Links can be styled with any CSS property (e.g. color, font-family, background, etc.).
- ▶ In addition, links can be styled differently depending on what state they are in.
- ▶ The four links states are:
  - ▶ `a:link` - a normal, unvisited link
  - `a:visited` - a link the user has visited
  - `a:hover` - a link when the user mouses over it
  - `a:active` - a link the moment it is clicked
- ▶ When setting the style for several link states, there are some order rules:
  - ▶ `a:hover` **MUST** come after `a:link` and `a:visited`
  - ▶ `a:active` **MUST** come after `a:hover`
  - ▶ The `text-decoration` property is mostly used to remove underlines from links:



# CSS – Display

- ▶ The **display** property specifies if/how an element is displayed.
- ▶ Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.



# CSS – Display

- ▶ A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).
- ▶ Examples of block-level elements:
  - <div>
  - <h1> - <h6>
  - <p>
  - <form>
  - <header>
  - <footer>
  - <section>



# CSS – Display

- ▶ An **inline** element does not start on a new line and only takes up as much width as necessary.
- ▶ Examples of inline elements:
  - <span>
  - <a>
  - <img>



# CSS – Display

- ▶ **display:none;** is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.
- ▶ The **<script>** element uses **display:none;** as default.
- ▶ As mentioned, every element has a default display value. However, you can override this.
- ▶ Setting the display property of an element only changes how the element is displayed, **NOT what kind of element it is.**
- ▶ So, an **inline element** with **display: block;** is **not allowed** to have other block elements inside it.



## CSS – Display – Hide an Element

- ▶ Hiding an element can be done by setting the display property to none (`display:none`).
- ▶ The element will be hidden, and the page will be displayed as if the element **is not there**:
  
- ▶ `visibility:hidden;` also hides an element.
- ▶ However, the element will still take up the same space as before. The element will be hidden, but **still affect the layout**:



# CSS – Display

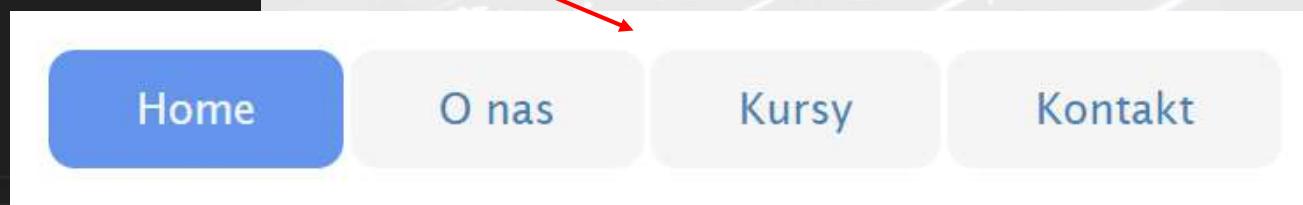
- ▶ Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.
- ▶ A common example is making inline <li> elements for horizontal menus.
- ▶ Example:



```
<!DOCTYPE html>
<html>
<head>
<style>
li {
    display: inline;
}
a {
    border: thin steelblue 1px;
    border-radius: 15px;
    padding: 20px 50px 20px 50px;
    color: steelblue;
    background-color: whitesmoke;
    text-decoration:none;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans';
    font-size:24px
}
#menu{
    margin: 130px 20px 20px 20px;
    border: thin;
}
a:hover, a:active {
    background-color: cornflowerblue;
    color: whitesmoke;
}
</style>
</head>
<body>

<div id="menu">
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">O nas</a></li>
        <li><a href="#">Kursy</a></li>
        <li><a href="#">Kontakt</a></li>
    </ul>
</div>

</body>
</html>
```





## CSS – Tables

- ▶ To specify table borders in CSS, use the **border** property.
- ▶ The example below specifies a black border for **<table>**, **<th>**, and **<td>** elements:

```
table, th, td {  
    border: 1px solid black;  
}
```

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol	Zieliński	67/80



## CSS – Tabeles

- ▶ The **border-collapse** property sets whether the table borders should be collapsed into a single border:

```
table {  
    border-collapse: collapse;  
}  
  
table, th, td {  
    border: 1px solid black;  
}
```

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol	Zielinski	67/80



## CSS – Tabeles

- ▶ Właściwość **border-collapse** określa, czy ramki tabeli powinny zostać zwinięte w jedną ramkę. Taki wygląd jest zazwyczaj zdecydowanie lepszy.

```
table {  
    border-collapse: collapse;  
}  
  
table, th, td {  
    border: 1px solid black;  
}
```

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol	Zielinski	67/80

- ▶ If you only want a border around the table, only specify the **border** property for **<table>**



## CSS – Tabeles

- ▶ **width** and **height** of a table are defined by the width and height properties.
- ▶ The example below sets the width of the table to 80%, and the height of the **<th>** elements to 40px

```
table {  
    width: 80%;  
    border-collapse: collapse;  
}  
table, th, td {  
    border: 1px solid black;  
}  
th {  
    height: 40px;  
}
```

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol	Zieliński	67/80



# CSS – Tabeles

- ▶ The **text-align** property sets the horizontal alignment (like left, right, or center) of the content in **<th>** or **<td>**.
- ▶ By default, the content of **<th>** elements are center-aligned and the content of **<td>** elements are left-aligned.
- ▶ The following example left-aligns the text in **<th>** elements:

```
table {  
    width: 80%;  
    border-collapse: collapse;  
}  
table, th, td {  
    border: 1px solid black;  
}  
th {  
    height: 40px;  
    text-align:left;  
}
```

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol	Zieliński	67/80



## CSS – Tables

- To control the space between the border and the content in a table, use the **padding** property on **<td>** and **<th>** elements:

```
table {  
    width: 80%;  
    border-collapse: collapse;  
}  
table, th, td {  
    border: 1px solid black;  
}  
th, td {  
    padding: 15px;  
    text-align: left;  
}
```

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol	Zieliński	67/80



## CSS – Tables

- ▶ Add the **border-bottom** property to <th> and <td> for horizontal dividers:

```
table {  
    width: 80%;  
    border-collapse: collapse;  
}  
table, th, td {  
    border-bottom: 1px solid #DDDDDD;  
}  
th, td {  
    padding: 15px;  
    text-align: left;  
}
```

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol	Zieliński	67/80



## CSS – Tabeles

- ▶ Use the **:hover** selector on <tr> to highlight table rows on mouse over:

```
table {  
    width: 80%;  
    border-collapse: collapse;  
}  
  
table, th, td {  
    border-bottom: 1px solid #DDDDDD;  
}  
  
th, td {  
    padding: 15px;  
    text-align: left;  
}  
  
tr:hover {  
    background-color: #f5f5f5;  
}
```

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol	Zieliński	67/80



## CSS – Tabeles

- For zebra-striped tables, use the **nth-child()** selector and add a background-color to all even (or odd) table rows:

```
table {  
    width: 80%;  
    border-collapse: collapse;  
}  
table, th, td {  
    border-bottom: 1px solid #DDDDDD;  
}  
th, td {  
    padding: 15px;  
    text-align: left;  
}  
tr:nth-child(even) {  
    background-color: #f3f3f3;  
}  
tr:hover {  
    background-color: #f5f5f5;  
}
```

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol	Zieliński	67/80



# CSS – Tabele

- ▶ The example below specifies the **background color** and text color of <th> elements:

```
table {  
    width: 80%;  
    border-collapse: collapse;  
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',  
    'Lucida Sans Unicode', Geneva, Verdana, sans-serif;  
}  
table, th, td {  
    border-bottom: 1px solid ■#DDDDDD;  
}  
th {  
    background-color: ■#5a6f9b;  
    color: ■white;  
}  
th, td {  
    padding: 15px;  
    text-align: left;  
}  
tr:nth-child(even) {  
    background-color: ■#f3f3f3;  
}  
tr:hover {  
    background-color: ■#f5f5f5;  
}
```



# CSS – Tabele

► Excercise:

► Add header color and font style.

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol	Zieliński	67/80



## CSS – Position

- ▶ The position **position** specifies the type of positioning method used for an element (**static**, **relative**, **fixed**, **absolute**, **sticky**)
  
- ▶ **position:static;**
- ▶ HTML elements are positioned static by default.
- ▶ Static positioned elements are not affected by the top, bottom, left, and right properties.
- ▶ An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page.



## CSS – Position

- ▶ The position **position** specifies the type of positioning method used for an element (**static**, **relative**, **fixed**, **absolute**, **sticky**)
- ▶ **position: relative;**
- ▶ An element with **position: relative;** is positioned relative to its normal position.
- ▶ Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.
- ▶ Other content will not be adjusted to fit into any gap left by the element.



## CSS – Position

- ▶ The position **position** specifies the type of positioning method used for an element (**static**, **relative**, **fixed**, **absolute**, **sticky**)
- ▶ **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. The **top**, **right**, **bottom**, **left** properties are used to position the element.
- ▶ A fixed element does not leave a gap in the page where it would normally have been located.



# CSS – Position

► **position:fixed;**

**position:fixed;**

Element z position:fixed; jest umieszczony względem obszaru wyświetlania, co oznacza, że zawsze pozostaje w tym samym miejscu, nawet jeśli strona jest przewijana.

Ten kontener ma ustawioną właściwość  
position:fixed;



# CSS – Position

- ▶ The position **position** specifies the type of positioning method used for an element (**static**, **relative**, **fixed**, **absolute**, **sticky**)
- ▶ **position: absolute;**
- ▶ An element with **position: absolute;** is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- ▶ However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
- ▶ Note: A "positioned" element is one whose position is anything except static.



## CSS – Position

- ▶ The position **position** specifies the type of positioning method used for an element (**static**, **relative**, **fixed**, **absolute**, **sticky**)
- ▶ **position:sticky;**
- ▶ An element with **position:sticky;** is positioned based on the user's scroll position.
- ▶ A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).



# CSS – Position

► **position:sticky;**

► Example:

Obszar jest zdefiniowany z padding:bottom:2000px. Przewiń stronę by sprawdzić zachowanie elementu z właściwością position:sticky.

IE/Edge 15 i wersje wcześniejsze nie obsługują position:sticky.

Jestem elementem "lepkim"!

W przykładzie, element div jest umieszczany na górze strony (top: 0), w momencie scrollowania.

Aby usunąć przyleganie (stickiness) przewiń stronę do góry.

Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudianda nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudianda nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

```
div.sticky {  
    position: -webkit-sticky;  
    position: sticky;  
    top: 0;  
    padding: 5px;  
    background-color: #cae8ca;  
    border: 2px solid cornflowerblue;
```

Jestem elementem "lepkim"!

w przykładzie, element div jest umieszczany na górze strony (top: 0), w momencie scrollowania.

Aby usunąć przyleganie (stickiness) przewiń stronę do góry.

Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudianda nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudianda nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.



## CSS – z-index

- ▶ The **z-index** property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
  
- ▶ **z-index:value;**
- ▶ When elements are positioned, they can overlap other elements.
- ▶ The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
- ▶ An element can have a positive or negative stack order.



# CSS – z-index

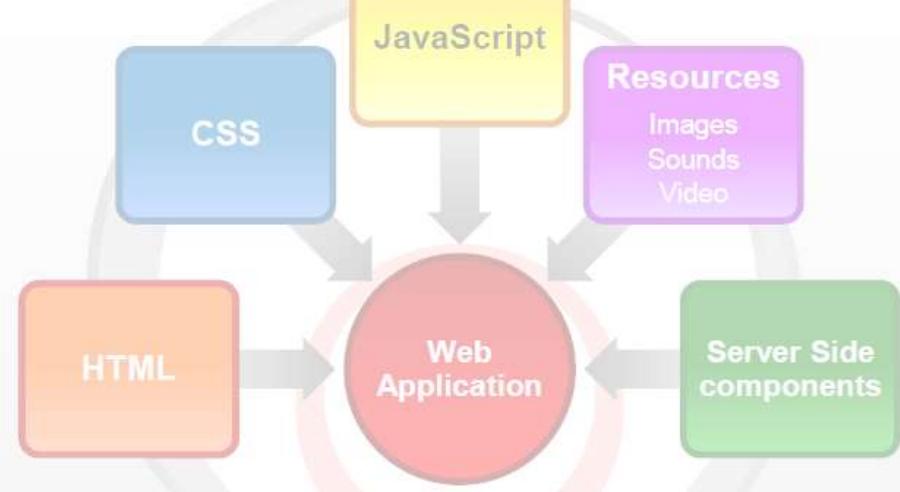
► **z-index:-1;**

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
    position:absolute;
    left:2px;
    top:2px;
    z-index: -1;
    opacity: 30%;
}
</style>
</head>
<body>

<div>
    <h1 style="position: absolute; left:275px;">Tytuł obrazka</h1>
    
    <p style="position: absolute; top:460px; left:200px;">
        oraz tekst wyświetlany nad warstwą zawierającą obraz</p>
</div>

</body>
</html>
```

## Tytuł obrazka



oraz tekst wyświetlany nad warstwą zawierającą obraz



# CSS – Overflow

- ▶ The CSS **overflow** property controls what happens to content that is too big to fit into an area.
- ▶ The **overflow** property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.
- ▶ The **overflow** property has the following values:
  - ▶ **visible** - Default. The overflow is not clipped. It renders outside the element's box
  - ▶ **hidden** - The overflow is clipped, and the rest of the content will be invisible
  - ▶ **scroll** - The overflow is clipped, but a scrollbar is added to see the rest of the content
  - ▶ **auto** - If overflow is clipped, a scrollbar should be added to see the rest of the content



# CSS – Float

- ▶ The CSS **float** property specifies how an element should float.
- ▶ The CSS clear property specifies what elements can **float** beside the cleared element and on which side.
- ▶ The **float** property is used for positioning and layout on web pages.
  - **left** - The element floats to the left of its container
  - **right** - The element floats to the right of its container
  - **none** - The element does not float (will be displayed just where it occurs in the text). This is default
  - **inherit** - The element inherits the float value of its parent
- ▶ In its simplest use, the **float** property can be used to wrap text around images.



# CSS – Opacity

- ▶ The **opacity** property specifies the opacity/transparency of an element.
- ▶ Przezroczysty obraz:
- ▶ The **opacity** property can take a value from 0.0 - 1.0. The lower value, the more transparent
- ▶ A lower value makes the element more transparent.
- ▶ The opacity property is often used together with the :hover selector to change the opacity on mouse-over:

```
img {  
    opacity: 0.5;  
}  
  
img:hover {  
    opacity: 1.0;  
}
```



# CSS – Pseudo-classes

- ▶ A pseudo-class is used to define a special state of an element.
- ▶ For example, it can be used to:
  - Style an element when a user mouses over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus



# CSS – Pseudo-classes

- ▶ Links can be displayed in different ways:

```
/* unvisited link */
a:link {
    color: #cornflowerblue;
    text-decoration:none;
}

/* visited link */
a:visited {
    color: #slategray;
    text-decoration:none;
}

/* mouse over link */
a:hover {
    color: #darkblue;
    text-decoration:none;
}

/* selected link */
a:active {
    color: #0000FF;
    text-decoration:none;
}
```



# CSS – Pseudo-classes

- The :first-child pseudo-class matches a specified element that is the first child of another element.

```
<!DOCTYPE html>
<html>
<head>
<style>
p:first-child {
    color: blue;
}
</style>
</head>
<body>

<p>Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum,
    altera fabulas ut quo. Atqui ...</p>
<p>Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum,
    altera fabulas ut quo. Atqui ...</p>

</body>
</html>
```

  Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui ...

  Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui ...



# CSS – Pseudo-classes

Selektor	Przykład	Opis
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus



# CSS – Pseudo-classes

Selektor	Przykład	Opis
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(language)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element



# CSS – Pseudo-classes

Selektor	Przykład	Opis
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range



# CSS – Pseudo-classes

Selektor	Przykład	Opis
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links



# CSS – Pseudo-elements

Selektor	Przykład	Opis
::after	p::after	Insert content after every <p> element
::before	p::before	Insert content before every <p> element
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user



```
        $hashed = $wp_hasher->HashPassword($user['password']);
        $wpdb->update( $wpdb->users, array( 'user_pass' => $hashed ) );
    }

    $message = __('Someone requested that the password be reset for the');
    $message .= network_home_url( '/' ) . "\r\n\r\n";
    $message .= sprintf(__('Username: %s'), $user_login) . "\r\n\r\n";
    $message .= __('If this was a mistake, just ignore this email.');
    $message .= __("To reset your password, visit the following address:") . "\r\n\r\n";
    $message .= network_site_url("wp-login.php?action=rp&key=$key&login=" . rawurlencode($user_login));
}

function wp_email_password($message, $user_login, $key) {
    $message = __("Get Help") . "\r\n\r\n";
    $message .= __("Exit") . "\r\n\r\n";
    $message .= __("WriteOut") . "\r\n\r\n";
    $message .= __("Justify") . "\r\n\r\n";
    $message .= __("Read File") . "\r\n\r\n";
    $message .= __("Where Is") . "\r\n\r\n";
    $message .= __("Prev Page") . "\r\n\r\n";
    $message .= __("Next Page") . "\r\n\r\n";
    $message .= __("Cut Text") . "\r\n\r\n";
    $message .= __("UnCut Text") . "\r\n\r\n";
    $message .= __("Car To") . "\r\n\r\n";
    $message .= __("To Spell") . "\r\n\r\n";
}
```



# JavaScript

- ▶ JavaScript (JS) was initially created to “make web pages alive”.
- ▶ The programs in this language are called scripts. They can be written right in a web page’s HTML and run automatically as the page loads.
- ▶ Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.
- ▶ In this aspect, JavaScript is very different from another language called Java.



# JavaScript

- ▶ JavaScript syntax is the set of rules, how JavaScript programs are constructed.
- ▶ JavaScript scripts are run via a web browser.
- ▶ JavaScript syntaxes are constructed of:
  - variables,
  - operators,
  - expressions,
  - keywords
  - comments.
- ▶ This instruction sets "Welcome to the course" inside an HTML element with id = "last name".

```
document.getElementById("name").innerHTML = "Witaj na kursie";
```



# JavaScript

JavaScript > JS lesson1.js > ...

```
1  var x = 2020;
2  var car = Object.create(Car.prototype);
3  var witaj = function myFunction(){
4    console.log("Witaj na kursie");
5 }
```

- ▶ In the example above, three basic statements are displayed: a variable, an object and a function declarations.
- ▶ JavaScript has dynamic types. This means that the same variable can be used to hold different data types.
- ▶ Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.
- ▶ ECMAScript is the official name of the JavaScript language.
- ▶ The ECMA-262 specification contains the most in-depth, detailed and formalized information about JavaScript. It defines the language.



# JavaScript

```
document.getElementById("nazwa") != document.getElementById("nazwa")
```

- All JavaScript identifiers are **case sensitive**. The variables `myVar` and `myvar`, are two different variables:

```
var myVar = 5;  
var myvar = "text";  
var MyVar = document.getElementById("name");
```

```
var myVar, myvar, MyVar;  
myVar = 5;  
myvar = "text";  
MyVar = document.getElementById("name");
```



# JavaScript

- ▶ ECMAScript 6 - ECMAScript 2015 accepts Unicode wersji 3.0 characters.
- ▶ Whitespace can appear between any two tokens (a token is an atomic lexis) and at the beginning or end. They can also appear in a comment.
- ▶ In ECMAScript, acceptable white space is tab (\u0009), vertical tab (\u000B), form feed (\u000C), space (\u0020), no-break space (\u00A0), or any other Unicode space separator. JavaScript ignores spaces between tokens.
- ▶ Line terminators (newline characters) affect the automatic insertion of semicolons in JavaScript. A line terminator cannot appear in any token except a string literal (for example, "This is a string").
- ▶ A Unicode escape sequence consists of exactly four hexadecimal digits following \u.

```
var x = 10; // To jest komentarz gdzie \u000A jest dozwolone  
  
var s = "\u00A9 Kurs Front-end";  
  
MyVar.innerHTML = s;
```

© Kurs Front-end



# JavaScript

- ▶ An array of commonly used Unicode characters

Unicode value	Name	
\u0009	Tab	<TAB>
\u000B	Vertical tab	<VT>
\u000C	Form Feed	<FF>
\u0020	Space	<SP>
\u000A	Line Feed	<LF>
\u000D	Carriage return	<CR>
\u0008	Backspace	<BS>
\u0009	Horizontal tab	<HT>
\u0022	Quotation mark	"
\u0027	Apostrophe	'
\u005C	Backslash	\



# JavaScript

- There are two types of comments in JavaScript:

- Single-line Comment
  - Multi-line Comment

```
// To jest komentarz jednowierszowy  
  
/* To jest komentarz wielowierszowy  
UWAGA: ten typ nie może być  
zagnieżdżany */
```

- JavaScript Literals are constant values that can be assigned to the variables that are called literals or constants.
- JavaScript Literals are syntactic representations for different types of data like numeric, string, Boolean, array, etc data.
- Literals in JavaScript provide a means of representing particular or some specific values in our program.

```
null  
"True"  
'true'  
true  
123.45
```



# JavaScript

- ▶ Semicolons separate JavaScript statements.
- ▶ On the web, you might see examples without semicolons.
- ▶ Ending statements with semicolon is not required, but highly recommended.
- ▶ In some cases, semicolons can be omitted because they are automatically inserted into the source code, except for **do-while**, **continue**, **break**, **return** oraz **throw**. The following code shows an example of inserting a semicolon in some situations.
- ▶ The newline is important for automatically inserting a semicolon. If there is a new line between two statements and the first statement does not have a semicolon, it is automatically inserted.

```
var x = 3.14  
var pi = "PI";
```



# JavaScript

- ▶ JavaScript does not enforce the use of a semicolon at the end of a statement. Instead, the semicolon is optional, and the JavaScript interpreter will "intelligently" add them when it runs the code.
- ▶ In JavaScript, a semicolon is automatically inserted when:
  - ▶ two statements are separated by a line terminator
  - ▶ two statements are separated by a closing brace ('}')
- ▶ in practice the lack of semicolons makes your program harder to debug. Because of this, it is universally recognized as a best practice to use semicolons at the end of statements, anyway.

A screenshot of a code editor window. The code shown is:

```
a = b + c  
(d + e) = f;  
a = b + c(d + e) = f;
```

A red arrow points from the closing brace ')' in the second line to the opening parenthesis '(' in the third line, highlighting a common mistake where a developer forgets to add a semicolon after the second line's assignment.



# JavaScript - Identifiers

- ▶ Identifiers are names.
- ▶ In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).
- ▶ The rules for legal names are much the same in most programming languages.
- ▶ In JavaScript, the first character must be a letter, or an underscore (`_`), or a dollar sign (`$`).
- ▶ Subsequent characters may be letters, digits, underscores, or dollar signs.

```
toJestZmienna
to_te_z_jest_zmienna
_inna_zmienna
$zmienna
MojaZmienna
```



# JavaScript – reserved words

- ▶ The following are reserved as future keywords by the ECMAScript specification. They have no special functionality at present, but they might at some future time, so they cannot be used as identifiers. These are always reserved: `enum`, `extends`, `super`, `const`, `export` and `import`.
- ▶ The following are only reserved when they are found in strict mode code: `implements`, `let`, `private`, `public`, `yield`, `interface`, `package`, `protected` and `static`.

<code>break</code>	<code>do</code>	<code>instanceof</code>	<code>typeof</code>	<code>case</code>
<code>else</code>	<code>new</code>	<code>var</code>	<code>catch</code>	<code>finally</code>
<code>return</code>	<code>void</code>	<code>continue</code>	<code>for</code>	<code>switch</code>
<code>while</code>	<code>debugger</code>	<code>function</code>	<code>this</code>	<code>with</code>
<code>default</code>	<code>if</code>	<code>throw</code>	<code>delete</code>	<code>in</code>
<code>try</code>				



# JavaScript Arithmetic Operators

- Arithmetic operators perform arithmetic on numbers ( literals or variables).
- A typical arithmetic operation operates on two numbers.
- The numbers (in an arithmetic operation) are called **operands**.
- The operation (to be performed between the two operands) is defined by an **operator**.

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus (Remainder)
++	Increment pre-increment or post-increment
--	Decrement (decrement) pre-decrement lub post-decrement



# JavaScript – operators

- ▶ The + operator can also be used to add (concatenate) strings.
- ▶ When used on strings, the + operator is called the concatenation operator.

```
var s = "To" + " " + "jest napis";
// zmienna s ma wartość To jest napis
```

- ▶ Adding two numbers, will return the sum, but adding a number and a string will return a string:

```
var s = "To" + " " + "jest napis";
// zmienna s ma wartość To jest napis

var newS = 10.59 + " " + s;
var s2 = s + 2**3;
```



# JavaScript - Assignment Operators

- Assignment operators assign values to JavaScript variables.
- Arithmetic operators allow you to perform operations on numbers (or strings).

Operator	Example	Same As
=	<code>x = y</code>	<code>x = y</code>
+=	<code>x += y</code>	<code>x = x + y</code>
-=	<code>x -= y</code>	<code>x = x - y</code>
*=	<code>x *= y</code>	<code>x = x * y</code>
/=	<code>x /= y</code>	<code>x = x / y</code>
%=	<code>x %= y</code>	<code>x = x % y</code>
**=	<code>x **= y</code>	<code>x = x ** y</code>



## JavaScript - Comparison Operators

- Comparison and Logical operators are used to test for true or false.
- Comparison operators can be used in conditional statements to compare values and take action depending on the result
- JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

`(test) ? (val1) : (val2).`

Operator	Description
<code>==</code>	equal to
<code>===</code>	equal value and <b>equal type</b>
<code>!=</code>	not equal
<code>!==</code>	not equal value or not equal type
<code>&gt;</code>	greater than
<code>&lt;</code>	less than
<code>&gt;=</code>	greater than or equal to
<code>&lt;=</code>	mniejsze lub równe
<code>?</code>	less than or equal to



## JavaScript – Logical Operators and Type Operators

- ▶ Logical operators are used to determine the logic between variables or values.
- ▶ You can use the `typeof` operator to find the data type of a JavaScript variable.

Operator	Description
<code>&amp;&amp;</code>	AND
<code>  </code>	OR
!	NOT
Operator	Description
<code>typeof</code>	checks the type of object.
<code>instanceof</code>	checks if the object is an instance of given type



# JavaScript - Operator Precedence Values

- W JavaScript jako pierwsze wykonywane są operacje w nawiasach.

Value	Operator	Description	Example
21	( )	Expression grouping	(3 + 4)
20	.	Member	person.name
20	[ ]	Member	person["name"]
20	()	Function call	myFunction()
20	new	Create	new Date()



# JavaScript - Operator Precedence Values

Value	Operator	Description	Example
18	<code>++</code>	Postfix Increment	<code>i++</code>
18	<code>--</code>	Postfix Decrement	<code>i--</code>
17	<code>++</code>	Prefix Increment	<code>+i</code>
17	<code>--</code>	Prefix Decrement	<code>--i</code>
17	<code>!</code>	Logical not	<code>!(x==y)</code>
17	<code>typeof</code>	Type	<code>typeof x</code>



# JavaScript - Operator Precedence Values

Value	Operator	Description	Example
16	<code>**</code>	Exponentiation (ES2016)	<code>10 ** 2</code>
15	<code>*</code>	Multiplication	<code>10 * 5</code>
15	<code>/</code>	Division	<code>10 / 5</code>
15	<code>%</code>	Division Remainder	<code>10 % 5</code>
14	<code>+</code>	Addition	<code>10 + 5</code>
14	<code>-</code>	Subtraction	<code>10 - 5</code>



# JavaScript - Operator Precedence Values

Value	Operator	Description	Example
13	<code>&lt;&lt;</code>	Shift left	<code>x &lt;&lt; 2</code>
13	<code>&gt;&gt;</code>	Shift right	<code>x &gt;&gt; 2</code>
13	<code>&gt;&gt;&gt;</code>	Shift right (unsigned)	<code>x &gt;&gt;&gt; 2</code>
12	<code>&lt;</code>	Less than	<code>x &lt; y</code>
12	<code>&lt;=</code>	Less than or equal	<code>x &lt;= y</code>
12	<code>&gt;</code>	Greater than	<code>x &gt; y</code>
12	<code>&gt;=</code>	Greater than or equal	<code>x &gt;= y</code>
12	<code>in</code>	Property in Object	"PI" in Math
12	<code>instanceof</code>	Instance of Object	<code>instanceof Array</code>



# JavaScript - Operator Precedence Values

Value	Operator	Description	Example
11	<code>==</code>	Equal	<code>x == y</code>
11	<code>===</code>	Strict equal	<code>x === y</code>
11	<code>!=</code>	Unequal	<code>x != y</code>
11	<code>!==</code>	Strict unequal	<code>x !== y</code>
10	<code>&amp;</code>	Bitwise AND	<code>x &amp; y</code>
9	<code>^</code>	Bitwise XOR	<code>x ^ y</code>
8	<code> </code>	Bitwise OR	<code>x   y</code>
7	<code>&amp;&amp;</code>	Logical AND	<code>x &amp;&amp; y</code>
6	<code>  </code>	Logical OR	<code>x    y</code>
5	<code>??</code>	Nullish Coalescing	<code>x ?? y</code>
4	<code>? :</code>	Condition	<code>? "Yes" : "No"</code>



# JavaScript - Operator Precedence Values

Value	Operator	Description	Example
3	<code>+ =</code>	Assignment	<code>x += y</code>
3	<code>/ =</code>	Assignment	<code>x /= y</code>
3	<code>- =</code>	Assignment	<code>x -= y</code>
3	<code>* =</code>	Assignment	<code>x *= y</code>
3	<code>% =</code>	Assignment	<code>x %= y</code>
3	<code>&lt;&lt; =</code>	Assignment	<code>x &lt;&lt;= y</code>
3	<code>&gt;&gt; =</code>	Assignment	<code>x &gt;&gt;= y</code>
3	<code>&gt;&gt;&gt; =</code>	Assignment	<code>x &gt;&gt;&gt;= y</code>
3	<code>&amp; =</code>	Assignment	<code>x &amp;= y</code>
3	<code>^ =</code>	Assignment	<code>x ^= y</code>
3	<code>  =</code>	Assignment	<code>x  = y</code>



# JavaScript - Operator Precedence Values

Value	Operator	Description	Example
2	<code>yield</code>	Pause Function	<code>yield x</code>
1	,	Comma	<code>5, 6</code>



# JavaScript - Data Types

- ▶ JavaScript is a dynamically typed (loosely typed) language. JavaScript automatically determines the variables' data type for you.
- ▶ It also means that a variable can be of one data type and later it can be changed to another data type.

Typ	Example	Description
string	var txt = "abc";	represents textual data
number	var num = 5.0;	an integer or a floating-point number
boolean	var b = true;	Any of two values: true or false
Array	var arr = ["x", "y", "z"];	tablica obiektów
Object	var person = {name:"Jan", surname:"Kowalski", age:30, sex:"M"};	key-value pairs of collection of data
undefined	var animal;	a data type whose variable is not initialized
null	person = null;	denotes a null value



# JavaScript – Data Types

- ▶ In strings we can use double or single quotes.
- ▶ JavaScript has only one type of number.
- ▶ Numbers can be written with or without decimals.
- ▶ Arrays are a special type of objects.
- ▶ **null** value doesn't change object type.
- ▶ **undefined** and **null** are equal but not of the same types.

```
typeof undefined          // undefined
typeof null              // object

null === undefined       // false
null == undefined        // true
```

- ▶ In JS you can compare **null** i **undefined**.



# JavaScript – Data Types

```
var str1 = "To jest tekst";
var str2 = 'To jest tekst';
var str2 = "Apostrof ' " + ' cudzysłów " można stosować zamiennie';

var x = "Tekst";
x = 5;
x = 5.0;

var x = 5;
var y = 5;
var z = 6;
var o = "5";
(x == y)          // true
(x == z)          // false
(x == o)          // true
(x === o)         //false

var arr = ["x", "y", "z"]; // typ array(tablica)
var osoba = {imie:"Jan", nazwisko:"Kowalski", wiek:30, plec:"M"}; // typ object
osoba = null; // wartość null, typ object
osoba = undefined; // wartość undefined, typ undefined
var animal; // wartość undefined, typ undefined
var person = null; // wartość null, typ undefined
```



# JavaScript – `typeof`

- ▶ You can use the `typeof` operator to find the data type of a JavaScript variable.
- ▶ `typeof` can be used to for primitive and complex types

```
typeof "abc"                  // typ "string"
typeof 1.234                  // typ "number"
typeof true                   // typ "boolean"
typeof false                  // typ "boolean"
typeof x                      // typ "undefined" (jeśli x nie zostało zdefiniowane)
typeof {imie:"Jan", wiek:30} // typ "object"
typeof [1,2,3,4]              // typ "object" (nie "array" !!!)
typeof null                   // typ "object"
typeof function myFunc(){}
// typ "function"
```



# JavaScript – statements

- ▶ The statements are executed, one by one, in the same order as they are written.
- ▶ Semicolons separate JavaScript statements.
- ▶ JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.
- ▶ The purpose of code blocks is to define statements to be executed together.
- ▶ One place you will find statements grouped together in blocks, is in JavaScript functions:
- ▶ Empty instruction is deined as single semicolon (;). It does nothing.

```
{  
    instrukcja_1;  
    instrukcja_2;  
  
    instrukcja_n;  
}  
  
;  
  
var x = 999;
```



# JavaScript – The *for* Loop

- ▶ **Statement 1** is executed (one time) before the execution of the code block.
  - ▶ **Statement 2** defines the condition for executing the code block.
  - ▶ **Statement 3** is executed (every time) after the code block has been executed.
- 
- ▶ From the example below , you can read:
  - ▶ Statement 1 sets a variable before the loop starts (var i = 1).
  - ▶ Statement 2 defines the condition for the loop to run (i must be less or equal to 10).
  - ▶ Statement 3 increases a value (i++) each time the code block in the loop has been executed.

```
for(var i=1; i<=10; i++) {           //pętla z iteracją po zmiennej i
    console.log(i);
}
```



# JavaScript – The `for/in` Loop

- ▶ The JavaScript `for in` statement loops through the properties of an Object:

```
var myObject = {  
    one: "value of property ONE",  
    two: "value of property TWO"  
};  
  
for(var mObj in myObject) {          //pętla z iteracją po właściwościach obiektu myObject  
    if(myObject.hasOwnProperty(mObj)){  
        console.log(mObj);  
    }  
}
```

## Example Explained

- ▶ The for in loop iterates over a `myObject` object
- ▶ Each iteration returns a key (`mObj`)
- ▶ The key is used to display the key
- ▶ The value of the key is `myObject[x]`
- ▶ The `hasOwnProperty()` method returns a boolean indicating whether the object has the specified property as its own property (as opposed to inheriting it).



# JavaScript –`while` and `do/while` loops

- ▶ The `while` loop loops through a block of code as long as a specified condition is `true`. In the following example, the code in the loop will run, over and over again, as long as a variable (`i`) is less than 10:

```
var i = 0;                                //pętla while z iteracją po zmiennej i
while(i<10) {
    console.log(++i);
}
```

- ▶ The `do/while` loop is a variant of the `while` loop. This loop will execute the code block once, before checking if the condition is `true`, then it will repeat the loop as long as the condition is `true`.
- ▶ The example below uses a do while loop. The loop will always be executed at least once, even if the condition is `false`, because the code block is executed before the condition is tested:

```
var i = 0;                                //pętla do/while z iteracją po zmiennej i
do{
    console.log(++i);
} while(i<10);
```



# JavaScript – if, switch

- ▶ Use **if** to specify a block of code to be executed, if a specified condition is **true**
- ▶ Use **else** to specify a block of code to be executed, if the same condition is false
- ▶ Use **else if** to specify a new condition to test, if the first condition is false

```
if(b==true){  
    instrukcje  
}
```

- ▶ The **switch** expression is evaluated once. The value of the expression is compared with the values of each case. If there is a match, the associated block of code is executed. If there is no match, the **default** code block is executed.

```
switch(expression) {  
    case label_1:  
        <statements1>  
        [break;]  
    case label_2:  
        <statements2>  
        [break;]  
    ...  
    default:  
        <statementsn>  
        [break;]  
}
```



# JavaScript – return, with

- When JavaScript reaches a `return` statement, the function will stop executing.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- Functions often compute a `return value`. The return value is "returned" back to the "caller":

```
function multiply(x,y) {  
    var result = x * y;  
    return result;  
}
```

- The `with` statement extends the scope chain for a statement.
- Using `with` is not recommended, and is forbidden in ECMAScript 5 strict mode

```
with (object) {  
    <instrukcje wykorzystujące właściwości i metody zadeklarowanego obiektu>  
}
```



# JavaScript – try...catch...finally

- ▶ The **try** statement marks a block of statements to try and specifies a response should an exception be thrown..
- ▶ **Try** statements: The statements to be executed.
- ▶ **Catch** statements: Statement that is executed if an exception is thrown in the try-block.
- ▶ **Exception** var: An optional identifier to hold an exception object for the associated catch-block.
- ▶ **Finally** statements: Statements that are executed after the try statement completes. These statements execute regardless of whether an exception was thrown or caught.

```
function myFunction() {  
    try {  
        functionCanThrowAnException();  
        if(condition) {  
            throw "MyException";  
        }  
    }  
    catch (e) {  
        logError(e);  
    }  
    finally {  
        closeConnection();  
    }  
}
```



# JavaScript – break and continue

- ▶ The **labeled** statement can be used with **break** or **continue** statements. It is prefixing a statement with an identifier which you can refer to. Etykieta może być dowolnym identyfikatorem, który nie jest słowem zastrzeżonym.
- ▶ The **break** statement terminates the current loop, switch, or label statement and transfers program control to the statement following the terminated statement.
- ▶ The **continue** statement terminates execution of the statements in the current iteration of the current or labeled loop, and continues execution of the loop with the next iteration.
- ▶ The **debugger** statement invokes any available debugging functionality, such as setting a breakpoint. If no debugging functionality is available, this statement has no effect.



# JavaScript – *labels* and debugger

```
myLabel:  
    var x =5;  
  
    function IneedToDebug(){  
        debugger;  
        // linie wykonujące debuggowanie kodu  
    }  
  
    for (var i=0; i<10; i++){  
        if(i==5) continue;  
        console.log('Iteracja w pętli:' + i)  
    }
```



# JavaScript – *break* i *continue*

- ▶ The **break** statement terminates the **while** loop
- ▶ In contrast to the break statement, **continue** does not terminate the execution of the loop entirely: instead,
  - ▶ In a **while** loop, it jumps back to the condition.
  - ▶ In a **for** loop, it jumps to the update expression.
- ▶ The **continue** statement can include an optional label that allows the program to jump to the next iteration of a labeled loop statement instead of the current loop. In this case, the continue statement needs to be nested within this labeled statement.



# JavaScript – break i continue

```
myLabel:  
for (var k=0; k<2;k++){  
  
    for (var i=0; i<10; i++){  
        if(i==5) continue myLabel;  
  
        console.log('Iteracja w petli:' + k*10+i);  
    }  
}
```

```
Iteracja w petli:00  
Iteracja w petli:01  
Iteracja w petli:02  
Iteracja w petli:03  
Iteracja w petli:04  
Iteracja w petli:100  
Iteracja w petli:101  
Iteracja w petli:102  
Iteracja w petli:103  
Iteracja w petli:104
```

```
for (var k=0; k<2;k++){  
    myLabel:  
    for (var i=0; i<10; i++){  
        if(i==5) continue myLabel;  
  
        console.log('Iteracja w petli:' + k*10+i);  
    }  
}
```

```
Iteracja w petli:00  
Iteracja w petli:01  
Iteracja w petli:02  
Iteracja w petli:03  
Iteracja w petli:04  
Iteracja w petli:06  
Iteracja w petli:07  
Iteracja w petli:08  
Iteracja w petli:09  
Iteracja w petli:100  
Iteracja w petli:101  
Iteracja w petli:102  
Iteracja w petli:103  
Iteracja w petli:104  
Iteracja w petli:106  
Iteracja w petli:107  
Iteracja w petli:108  
Iteracja w petli:109
```



# JavaScript – break i continue

```
for (var k=0; k<2;k++){  
    myLabel:  
        for (var i=0; i<10; i++){  
            if(i==5) break myLabel;  
  
            console.log('Iteracja w petli:' + k*10+i);  
        }  
}
```

```
Iteracja w petli:00  
Iteracja w petli:01  
Iteracja w petli:02  
Iteracja w petli:03  
Iteracja w petli:04  
Iteracja w petli:100  
Iteracja w petli:101  
Iteracja w petli:102  
Iteracja w petli:103  
Iteracja w petli:104
```

```
myLabel:  
for (var k=0; k<2;k++){  
  
    for (var i=0; i<10; i++){  
        if(i==5) break myLabel;  
  
        console.log('Iteracja w petli:' + k*10+i);  
    }  
}
```

```
Iteracja w petli:00  
Iteracja w petli:01  
Iteracja w petli:02  
Iteracja w petli:03  
Iteracja w petli:04
```



# JavaScript – break i continue

```
foo:  
  for (var i = 1; i <= 5; i++){  
    for (var j = 1; j <= 3; j++) {  
      console.log("i = " + i + ", j = " + j);  
      if ((i + j) > 4) break foo;  
    } //koniec petli  
  } //koniec petli
```

i = 1, j = 1
i = 1, j = 2
i = 1, j = 3
koniec petli
i = 2, j = 1
i = 2, j = 2
i = 2, j = 3

- ▶ Why doesn't "koniec petli" display twice?



# JavaScript – Exercises

- ▶ Excercise 1:

Using a `for` loop, print on the console the factorial value of  $1!$  up to  $10!$

- ▶ Excercise 1a:

Using a `while` or `do / while` loop, print on the console the factorial value of  $1!$  up to  $10!$

- ▶ Excercise 2:

Using `for` loop, print on console values  $1,2,3,5,8,13,21,34,55,89$  (first ten numbers of Fibonacci sequence).

- ▶ Excercise 3:

Using `for` and `if` statement, print on console first ten even values of the Fibonacci sequence.

- ▶ Excercise 4:

Monday is the first day of the week, use a `while` loop and a `switch` statement to print to the console:

1. Monday
2. Tuesday
- ...



# JavaScript – Exercises

- ▶ Excercise 5:

Print the values on the console: 1-5, 2-4, 3-3, 3-2, 3-1

- ▶ Excercise 6:

We throw 2 dice, print on the console all combinations of throws where the sum of the dice is even

- ▶ Excercise 7:

We have 20 white balls and 20 red balls. We draw one ball and if it is white, we randomly choose from the others until we draw white again.

Print on the console list possible results for 10 draws.

For example, after drawing the red one, we print:

red

and after drawing the white one, three red ones and white one, print :

white-red-red-red-white

**Tip: Use Math.random()**



# JavaScript – *string* (*properties*)

Method	Description
<code>constructor</code>	Creates a new String object.
<code>length</code>	The length property of a String object contains the length of the string
<code>prototype</code>	It allows you to add properties to a String object.



# JavaScript – string (methods)

Metoda	Opis
<code>charAt()</code>	Returns the character (exactly one UTF-16 code unit) at the specified index
<code>charCodeAt()</code>	Returns a number that is the UTF-16 code unit value at the given index.
<code>concat()</code>	Combines the text of two (or more) strings and returns a new string.
<code>endsWith()</code>	Determines whether a string ends with the characters of the string searchString.
<code>fromCharCode()</code>	Returns a string created by using the specified sequence of Unicode values.
<code>includes()</code>	Determines whether the calling string contains searchString.
<code>indexOf()</code>	Returns the index within the calling String object of the first occurrence of searchValue, or -1 if not found.
<code>lastIndexOf()</code>	Returns the index within the calling String object of the last occurrence of searchValue, or -1 if not found.



# JavaScript – string (methods)

Method	Description
<code>localeCompare()</code>	Returns a number indicating whether the reference string compareString comes before, after, or is equivalent to the given string in sort order.
<code>match()</code>	Used to match regular expression regexp against a string.
<code>repeat()</code>	Returns a string consisting of the elements of the object repeated count times.
<code>replace()</code>	Used to replace occurrences of searchFor using replaceWith. searchFor may be a string or Regular Expression, and replaceWith may be a string or function.
<code>search()</code>	Search for a match between a regular expression regexp and the calling string.
<code>slice()</code>	Extracts a section of a string and returns a new string.
<code>split()</code>	Returns an array of strings populated by splitting the calling string at occurrences of the substring sep.
<code>startsWith()</code>	Determines whether the calling string begins with the characters of string searchString.



# JavaScript – string (methods)

Method	Description
<code>substr()</code>	Returns a new string containing characters of the calling string from (or between) the specified index (or indeces).
<code>substring()</code>	Returns a new string containing characters of the calling string from (or between) the specified index (or indeces).
<code>toLocaleLowerCase()</code>	The characters within a string are converted to lowercase while respecting the current locale.
<code>toLocaleUpperCase()</code>	The characters within a string are converted to uppercase while respecting the current locale.
<code>toLowerCase()</code>	Returns the calling string value converted to lowercase.
<code>toString()</code>	Returns a string representing the specified object.
<code>toUpperCase()</code>	Returns the calling string value converted to uppercase.
<code>trim()</code>	Trims whitespace from the beginning and end of the string. Part of the ECMAScript 5 standard.
<code>valueOf()</code>	Returns the primitive value of the specified object.



# JavaScript – number (*properties*)

Method	Description
	Creates a new Number value
<code>Number()</code>	The largest positive representable number.
<code>MAX_VALUE</code>	The smallest positive representable number—that is, the positive number closest to zero (without actually being zero).
<code>MIN_VALUE</code>	Special value representing negative infinity. Returned on overflow.
<code>NEGATIVE_INFINITY</code>	Special "Not a Number" value.
<code>NaN</code>	Special value representing infinity. Returned on overflow.
<code>POSITIVE_INFINITY</code>	Allows the addition of properties to the Number object.
<code>prototype</code>	



# JavaScript – number (methods)

Method	Description
<code>isFinite()</code>	Determine whether the passed value is a finite number.
<code>isInteger()</code>	Determine whether the passed value is an integer.
<code>isNaN()</code>	Determine whether the passed value is NaN.
<code>isSafeInteger()</code>	Determine whether the passed value is a safe integer (number between -(2 <sup>53</sup> - 1) and 2 <sup>53</sup> - 1).
<code>toExponential(x)</code>	Returns a string representing the number in exponential notation.
<code>toFixed(x)</code>	Returns a string representing the number in fixed-point notation.
<code>toLocaleString()</code>	Returns a string with a language sensitive representation of this number.
<code>toPrecision(x)</code>	Returns a string representing the number to a specified precision in fixed-point or exponential notation.
<code>toString()</code>	Returns a string representing the specified object.
<code>valueOf()</code>	Returns the primitive value of the specified object.



```
        $hashed = $wp_hasher->hashPassword($password);
        $wpdb->update( $wpdb->users, array( 'user_login' => $user_login, 'user_email' => $user_email ), array( 'user_pass' => $password ) );
        $message = __('Someone requested that the password be reset for the user %s on your site (%s).');
        $message .= sprintf(__('Username: %s'), $user_login);
        $message .= __("If this was a mistake, just ignore this email and nothing will happen.");
        $message .= __("To reset your password, visit the following address: ") . $wp_site_url;
        $message .= $network_home_url('') . "/wp-login.php?action=rp&key=$key&login=" . rawurlencode($user_email);
        $message .= "\r\n\r\n";
        $message .= $wp_email->get_footer();
        $wp_email->wp_email($message, $user_email);
    }
}
```





# Events

- ▶ HTML **events** are "things" that happen to HTML elements.
- ▶ When JavaScript is used in HTML pages, JavaScript can "react" on these events.
- ▶ A CSS pseudo-class is used to define a special state of an element. For example, it can be used to:
  - Style an element when a user mouses over it(hover)
  - Style visited and unvisited links differently (normal/visited)
  - Style an element when it gets focus
- HTML allows **event handler** attributes, with JavaScript code, to be added to HTML elements.



# Events

► Here is a list of some common HTML events:

- onmouseover
- onmouseout
- onclick
- ondblclick
- onkeydown
- onkeypress
- onkeyup

and others.



# Events - Escercise

## ► Excercise:

- Create a page of a simple calculator with basic arithmetic operations and history of operations (list)
- Use an HTML table or container <div> as a template for the calculator
- Use CSS to create visual support for the calculator
- Use JavaScript to handle calculations.
- Additionaly:
- Handle the keyboard with the methods onkeypressed, onkeydown, onkeyup



```
        $hashed = $wp_hasher->HashPassword($user['password']);
        $wpdb->update( $wpdb->users, array( 'user_pass' => $hashed ) );
    }

    $message = __('Someone requested that the password be reset for the');
    $message .= network_home_url( '/' ) . "\r\n\r\n";
    $message .= sprintf(__("Username: %s"), $user_login) . "\r\n\r\n";
    $message .= __('If this was a mistake, just ignore this email');
    $message .= "\r\n\r\n";
    $message .= __('To reset your password, visit the following address:');
    $message .= network_site_url("wp-login.php?action=rp&key=$key&login=" . rawurlencode($user_login));
}

function wp_email_password($message, $user_login, $key) {
    $message = __($message);
    $message .= "\r\n\r\n";
    $message .= $user_login;
    $message .= "\r\n\r\n";
    $message .= $key;
}

function wp_email_password_html($message, $user_login, $key) {
    $message = __($message);
    $message .= "\r\n\r\n";
    $message .= $user_login;
    $message .= "\r\n\r\n";
    $message .= $key;
}
```

Get Help  
Exit

WriteOut  
Justify

Read File  
Where Is

Prev Page  
Next Page

Cut Text  
UnCut Text

Car To  
To Spell



# Bootstrap

- ▶ Bootstrap 4 is the newest version of Bootstrap, which is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.
- ▶ Bootstrap 4 is completely free to download and use.
- ▶ Bootstrap is a free front-end framework for faster and easier web development
- ▶ Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- ▶ Bootstrap also gives you the ability to easily create responsive designs
- ▶ What is Responsive Web Design?

Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.



# Bootstrap

## ► Advantages of Bootstrap

- Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
- Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
- Mobile-first approach: In Bootstrap, mobile-first styles are part of the core framework
- Browser compatibility: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera)



# Bootstrap – where to get?

- ▶ If you don't want to download and host Bootstrap 4 yourself, you can include it from a CDN (Content Delivery Network).
- ▶ MaxCDN provides CDN support for Bootstrap's CSS and JavaScript. You must also include jQuery:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<title>Przykład Bootstrap</title>

<!-- Najnowszy skompilowany CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
<!-- biblioteka jQuery -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<!-- Popper JS -->
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></script>
<!-- Najnowsza skompilowana biblioteka JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>

</head>

<body>

</body>
</html>
```



# Bootstrap – where to get?

- ▶ Many users already have downloaded Bootstrap 4 from MaxCDN when visiting another site. As a result, it will be loaded from cache when they visit your site, which leads to faster loading time. Also, most CDN's will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.
- ▶ Why jQuery and Popper?
  - ▶ Bootstrap 4 use jQuery and Popper.js for JavaScript components (like modals, tooltips, popovers etc). However, if you just use the CSS part of Bootstrap, you don't need them.
  - ▶ Components that require jQuery i Popper.js:
    - Closable alerts
    - Buttons and checkboxes/radio buttons for toggling states
    - Carousel for slides, controls, and indicators
    - Collapse for toggling content
    - Dropdowns (also requires Popper.js for perfect positioning)
    - Modals (open and close)
    - Navbar (for collapsible menus)
    - Tooltips and popovers (also requires Popper.js for perfect positioning)
    - Scrollspy for scroll behavior and navigation updates



# Bootstrap – where to get?

- ▶ You can use Stackpath.
- ▶ The last version ://[www.bootstrapcdn.com/](https://www.bootstrapcdn.com/)

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<title>Przykład Bootstrap</title>


<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

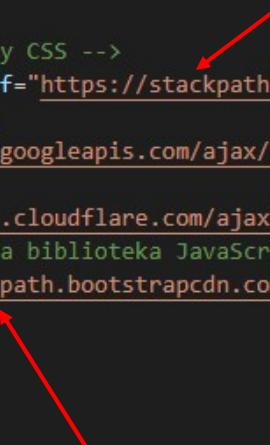
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>

</head>

<body>

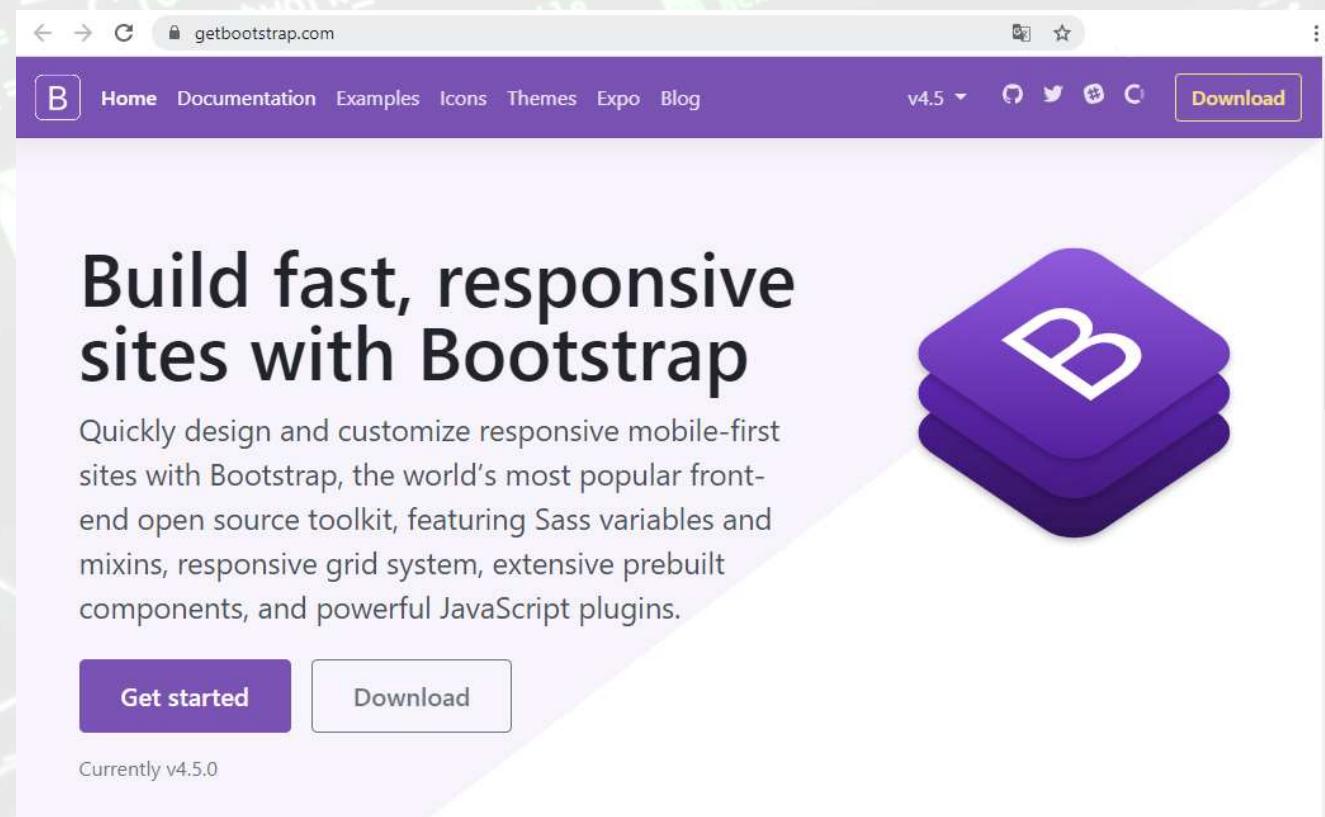
</body>
</html>
```

Two red arrows point to the URLs in the code: one to the CSS link at line 10 and another to the JS script at line 15.



# Bootstrap – dołączanie do strony

- If you want to download and host Bootstrap 4 yourself, go to <https://getbootstrap.com>, and follow the instructions there.



The screenshot shows the official Bootstrap website at [getbootstrap.com](https://getbootstrap.com). The page features a purple header with the Bootstrap logo and navigation links for Home, Documentation, Examples, Icons, Themes, Expo, and Blog. A "Download" button is also present. The main content area has a light purple background with the text "Build fast, responsive sites with Bootstrap". It describes Bootstrap as a mobile-first toolkit with Sass variables, mixins, a responsive grid system, prebuilt components, and JavaScript plugins. Two prominent purple rounded squares with a large white letter "B" are displayed. At the bottom, there are "Get started" and "Download" buttons, and a note stating "Currently v4.5.0".



# Bootstrap

- Desktop browsers:

## Supported browsers

Mac	Yes	N/A	Yes	Yes	Yes
Windows	Yes	Yes (IE10+)	Yes	No	Yes

- Mobile devices:

## Supported browsers

					Android Browser & WebView
Android	Yes	N/A	Yes	Yes	(Android 5.0+)
iOS	Yes	Yes	Yes	Yes	N/A
Windows 10 Mobile	N/A	Yes	N/A	N/A	N/A



# Bootstrap

- ▶ Unofficially, Bootstrap should look and behave well enough in Chromium and Chrome for Linux, Firefox for Linux, and Internet Explorer 9, though they are not officially supported.

## Supported browsers

Linux					
Yes	Yes (IE9+)	Yes	N/A	Yes	

- ▶ For a list of some of the browser bugs that Bootstrap has to grapple with, see our Wall of browser bugs.

<https://getbootstrap.com/docs/4.5/browser-bugs/>



# Bootstrap

- ▶ Bootstrap 4 is designed to be responsive to mobile devices. Mobile-first styles are part of the core framework.
- ▶ To ensure proper rendering and touch zooming, add the following `<meta>` tag inside the `<head>` element

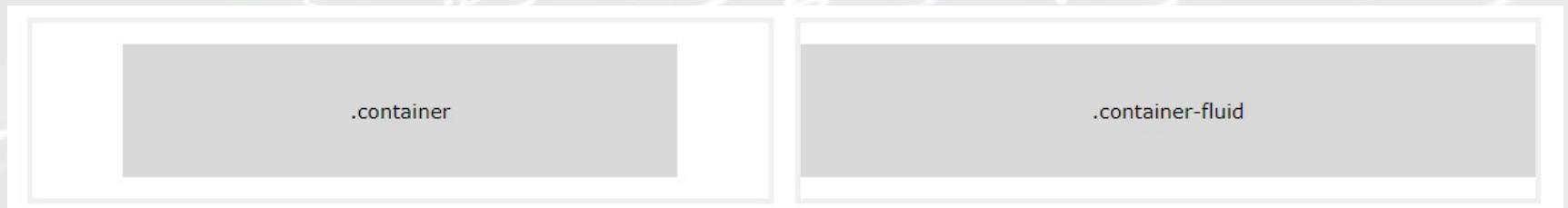
```
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

- ▶ The `width=device-width` part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- ▶ The `initial-scale=1` part sets the initial zoom level when the page is first loaded by the browser.



# Bootstrap

- Bootstrap 4 also requires a containing element to wrap site contents.
- There are two container classes to choose from:
- The `.container` class provides a responsive **fixed width container**
- The `.container-fluid` class provides a **full width container**, spanning the entire width of the viewport





# Bootstrap

## Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Bootstrap Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
    <h3>Pierwsza strona z użyciem Bootstrap</h3>
    <p>To część wewnętrz kontenera z przypisaną klasą .container</p>
    <p>Klasa .container class zapewnia responsywny kontener o stałej szerokości (fixed-width)</p>
</div>

</body>
```

## Pierwsza strona z użyciem Bootstrap

To część wewnętrz kontenera z przypisaną klasą .container

Klasa .container class zapewnia responsywny kontener o stałej szerokości (fixed-width)



# Bootstrap

## ► Container classes:

Class	Extra small $<576px$	Small $\geq 576px$	Medium $\geq 768px$	Large $\geq 992px$	Extra large $\geq 1200px$
.container	100%	540px	720px	960px	1140px
.container-fluid	100%	100%	100%	100%	100%
.container-sm	100%	540px	720px	960px	1140px
.container-md	100%	100%	720px	960px	1140px
.container-lg	100%	100%	100%	960px	1140px
.container-xl	100%	100%	100%	100%	1140px



# Bootstrap

```
<div class="container p-3 my-3 border">  
| <h1>strona z użyciem Bootstrap</h1>  
| <p>Ten kontener ma ramkę (border) oraz dodatkowy odstęp (padding) i marginesy (margins).</p>  
</div>  
  
<div class="container p-3 my-3 bg-dark text-white">  
| <h1>strona z użyciem Bootstrap</h1>  
| <p>Ten kontener ma ciemne tło, biały tekst oraz dodatkowy odstęp (padding) i marginesy  
</div>  
  
<div class="container p-3 my-3 bg-primary text-white">  
| <h1>strona z użyciem Bootstrap</h1>  
| <p>Ten kontener ma niebieskie tło, biały tekst oraz dodatkowy odstęp (padding) i margin  
</div>
```

## strona z użyciem Bootstrap

Ten kontener ma ramkę (border) oraz dodatkowy odstęp (padding) i marginesy (margins).

## strona z użyciem Bootstrap

Ten kontener ma ciemne tło, biały tekst oraz dodatkowy odstęp (padding) i marginesy (margins).

## strona z użyciem Bootstrap

Ten kontener ma niebieskie tło, biały tekst oraz dodatkowy odstęp (padding) i marginesy (margins).

- ▶ Borders and colors



# Bootstrap – Grid system

- Bootstrap's grid system is built with flexbox and allows up to 12 columns across the page.
- If you do not want to use all 12 columns individually, you can group the columns together to create wider columns:

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1				
span 4				span 4				span 4							
span 4				span 8											
span 6						span 6									
span 12															

- The grid system is responsive, and the columns will re-arrange automatically depending on the screen size.
- Make sure that the sum adds up to 12 or fewer (it is not required that you use all 12 available columns).



# Bootstrap – Grid system

- ▶ Grid Classes
- ▶ The Bootstrap 4 grid system has five classes:
  - ▶ `.col-*` (extra small devices - screen width less than 576px)
  - ▶ `.col-sm-*` (small devices - screen width equal to or greater than 576px)
  - ▶ `.col-md-*` (medium devices - screen width equal to or greater than 768px)
  - ▶ `.col-lg-*` (large devices - screen width equal to or greater than 992px)
  - ▶ `.col-xl-*` (xlarge devices - screen width equal to or greater than 1200px)
- ▶ The classes above can be combined to create more dynamic and flexible layouts.
- ▶ Tip: Each class scales up, so if you wish to set the same widths for sm and md, you only need to specify sm.



# Bootstrap – Z-index

- ▶ Several Bootstrap components utilize z-index, the CSS property that helps control layout by providing a third axis to arrange content. We utilize a default z-index scale in Bootstrap that's been designed to properly layer navigation, tooltips and popovers, modals, and more.
- ▶ These higher values start at an arbitrary number, high and specific enough to ideally avoid conflicts. We need a standard set of these across our layered components—tooltips, popovers, navbars, dropdowns, modals—so we can be reasonably consistent in the behaviors. There's no reason we couldn't have used 100+ or 500+.
- ▶ We don't encourage customization of these individual values; should you change one, you likely need to change them all.

```
$zindex-dropdown:      1000! default;  
$zindex-sticky:        1020! default;  
$zindex-fixed:         1030! default;  
$zindex-modal-backdrop: 1040! default;  
$zindex-modal:          1050! default;  
$zindex-popover:        1060! default;  
$zindex-tooltip:         1070! default;
```

- ▶ To handle overlapping borders within components (e.g., buttons and inputs in input groups), we use low single digit z-index values of 1, 2, and 3 for default, hover, and active states. On hover/focus/active, we bring a particular element to the forefront with a higher z-index value to show their border over the sibling elements.



# Bootstrap – Examples

- Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content.
- It's built with flexbox and is fully responsive.
- Below is an example and an in-depth look at how the grid comes together.

```
<div class="container p-3 my-3 border">
  <div class="row">
    <div class="col-sm">
      Jedna z trzech kolumn
    </div>
    <div class="col-sm">
      Jedna z trzech kolumn
    </div>
    <div class="col-sm">
      Jedna z trzech kolumn
    </div>
  </div>
```

Jedna z trzech kolumn

Jedna z trzech kolumn

Jedna z trzech kolumn



## Bootstrap – Examples

- ▶ The above example creates three equal-width columns on small, medium, large, and extra large devices using our predefined grid classes. Those columns are centered in the page with the parent .container.
- ▶ Containers provide a means to center and horizontally pad your site's contents. Use `.container` for a responsive pixel width or `.container-fluid` for width: 100% across all viewport and device sizes.
- ▶ Rows are wrappers for columns. Each column has horizontal padding (called a gutter) for controlling the space between them.
- ▶ This padding is then counteracted on the rows with negative margins. This way, all the content in your columns is visually aligned down the left side.
- ▶ In a grid layout, content must be placed within columns and only columns may be immediate children of rows.
- ▶ Thanks to flexbox, grid columns without a specified width will automatically layout as equal width columns. For example, four instances of `.col-sm` will each automatically be 25% wide from the small breakpoint and up. See the auto-layout columns section for more examples.



## Bootstrap – Examples

- ▶ Column classes indicate the number of columns you'd like to use out of the possible 12 per row. So, if you want three equal-width columns across, you can use `.col-4`.
- ▶ Column widths are set in **percentages**, so they're always fluid and sized relative to their parent element.
- ▶ Columns have horizontal padding to create the gutters between individual columns, however, you can remove the margin from rows and padding from columns with `.no-gutters` on the `.row`.
- ▶ To make the grid responsive, there are five grid breakpoints, one for each responsive breakpoint: all breakpoints (extra small), small, medium, large, and extra large.
- ▶ Grid breakpoints are based on minimum width media queries, meaning they apply to that one breakpoint and all those above it (e.g., `.col-sm-4` applies to small, medium, large, and extra large devices, but not the first xs breakpoint).
- ▶ You can use predefined grid classes (like `.col-4`) or Sass mixins for more semantic markup.



# Bootstrap – Examples

```
<div class="container p-3 my-3 border">
  <div class="row border">
    <div class="col bg-secondary text-white border">
      1 z 2
    </div>
    <div class="col bg-secondary text-white border">
      2 z 2
    </div>
  </div>
  <div class="row border">
    <div class="col bg-secondary text-white border">
      1 z 3
    </div>
    <div class="col bg-secondary text-white border">
      2 z 3
    </div>
    <div class="col bg-secondary text-white border">
      3 z 3
    </div>
  </div>
</div>
```





# Bootstrap – examples

- ▶ Equal-width
- ▶ Equal-width columns can be broken into multiple lines, but there was a Safari flexbox bug that prevented this from working without an explicit flex-basis or border. There are workarounds for older browser versions, but they shouldn't be necessary if you're up-to-date.

```
<div class="container p-3 my-3 border">
  <div class="row border">
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="w-100"></div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
  </div>
</div>
```

kolumna	kolumna
kolumna	kolumna



# Bootstrap – examples

- ▶ Setting one column width
- ▶ Auto-layout for flexbox grid columns also means you can set the width of one column and have the sibling columns automatically resize around it. You may use predefined grid classes (as shown below), grid mixins, or inline widths. Note that the other columns will resize no matter the width of the center column.

```
<div class="container p-3 my-3 border">
  <div class="row border">
    <div class="col bg-secondary text-white border">1 z 3</div>
    <div class="col-6 bg-secondary text-white border">2 z 3 (szersza)</div>
    <div class="col bg-secondary text-white border">3 z 3</div>
  </div>
  <div class="row border">
    <div class="col bg-secondary text-white border">1 z 3</div>
    <div class="col-5 bg-secondary text-white border">2 z 3 (szersza)</div>
    <div class="col bg-secondary text-white border">3 z 3</div>
  </div>
</div>
```

1 z 3	2 z 3 (szersza)	3 z 3
1 z 3	2 z 3 (szersza)	3 z 3



# Bootstrap – examples

- ▶ Variable width content
- ▶ Use `col-{breakpoint}-auto` classes to size columns based on the natural width of their content.

```
<div class="container p-3 my-3 border">
  <div class="row justify-content-md-center border">
    <div class="col-lg-2 bg-secondary text-white border">1 z 3</div>
    <div class="col-md-auto bg-secondary text-white border">Zmienna szerokość</div>
    <div class="col-lg-2 bg-secondary text-white border">3 z 3</div>
  </div>
  <div class="row border">
    <div class="col bg-secondary text-white border">1 z 3</div>
    <div class="col-md-auto bg-secondary text-white border">Zmienna szerokość</div>
    <div class="col-lg-2 bg-secondary text-white border">3 z 3</div>
  </div>
</div>
```

1 z 3
Zmienna szerokość
3 z 3
1 z 3
Zmienna szerokość
3 z 3

1 z 3	1 z 3	Zmienna szerokość	3 z 3	Zmienna szerokość	3 z 3
1 z 3					



## Bootstrap – examples

- ▶ Bootstrap's grid includes five tiers of predefined classes for building complex responsive layouts.
- ▶ Customize the size of your columns on extra small, small, medium, large, or extra large devices however you see fit.
- ▶ For grids that are the same from the smallest of devices to the largest, use the `.col` and `.col-*` classes.
- ▶ Specify a numbered class when you need a particularly sized column; otherwise, feel free to stick to `.col`.

```
<div class="container p-3 my-3 border">
  <div class="row border">
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
  </div>
  <div class="row border">
    <div class="col-8 bg-secondary text-white border">kolumna col-8</div>
    <div class="col-4 bg-secondary text-white border">kolumna col-4</div>
  </div>
</div>
```

kolumna	kolumna	kolumna	kolumna
kolumna col-8			kolumna col-4



# Bootstrap – przykłady

- ▶ Mix&match
- ▶ Don't want your columns to simply stack in some grid tiers? Use a combination of different classes for each tier as needed. See the example below for a better idea of how it all works.

```
<div class="container p-3 my-3 border">
  <!-- Ułóż kolumny w telefonie komórkowym, tworząc jedną na całej szerokości, a drugą na połowę -->
  <div class="row border">
    <div class="col-md-8 bg-secondary text-white border">kolumna .col-md-8</div>
    <div class="col-6 col-md-4 bg-secondary text-white border">kolumna .col-6 .col-md-4</div>
  </div>
  <!-- Kolumny zaczynają się od szerokości 50% na urządzeniach mobilnych i zwiększają się do 33,3% na komputerach -->
  <div class="row border">
    <div class="col-6 col-md-4 bg-secondary text-white border">kolumna .col-6 .col-md-4</div>
    <div class="col-6 col-md-4 bg-secondary text-white border">kolumna .col-6 .col-md-4</div>
    <div class="col-6 col-md-4 bg-secondary text-white border">kolumna .col-6 .col-md-4</div>
  </div>
  <!-- Kolumny mają zawsze szerokość 50% na urządzeniach mobilnych i stacjonarnych -->
  <div class="row border">
    <div class="col-6 bg-secondary text-white border">kolumna .col-6</div>
    <div class="col-6 bg-secondary text-white border">kolumna .col-6</div>
  </div>
</div>
```

kolumna .col-md-8	kolumna .col-6 .col-md-4	
kolumna .col-6 .col-md-4	kolumna .col-6 .col-md-4	kolumna .col-6 .col-md-4
kolumna .col-6		kolumna .col-6



# Bootstrap

- ▶ Klasy **row-cols**
- ▶ Use the responsive **.row-cols-\*** classes to quickly set the number of columns that best render your content and layout. Whereas normal **.col-\*** classes apply to the individual columns (e.g., **.col-md-4**), the row columns classes are set on the parent **.row** as a default for contained columns.
- ▶ With **.row-cols-auto** you can give the columns their natural width.
- ▶ Use these row columns classes to quickly create basic grid layouts or to control your card layouts and override when needed at the column level.



# Bootstrap

## ► row-cols

```
<div class="container p-3 my-3 border">
  <!-- 2 kolumny w rzędzie -->
  <div class="row row-cols-2 border">
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
  </div>
  <!-- 3 kolumny w rzędzie -->
  <div class="row row-cols-3 border">
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
  </div>
</div>
```

kolumna	kolumna	
kolumna	kolumna	
kolumna	kolumna	kolumna
kolumna		



# Bootstrap

► row-cols

```
<div class="container p-3 my-3 border">
  <!-- 4 kolumny w rzędzie -->
  <div class="row row-cols-4 border">
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
  </div>
  <!-- 1, 2 (mobilne) lub 4 (komputery) kolumny w rzędzie -->
  <div class="row row-cols-1 row-cols-sm-2 row-cols-md-4 border">
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
    <div class="col bg-secondary text-white border">kolumna</div>
  </div>
</div>
```

kolumna	kolumna	kolumna	kolumna
kolumna	kolumna	kolumna	kolumna

kolumna	kolumna	kolumna	kolumna
kolumna			



# Bootstrap

- ▶ `align-items-*`
- ▶ Use flexbox alignment utilities to vertically and horizontally align columns.

```
<div class="container">
  <div class="row align-items-start">
    <div class="col">Kolumna</div>
    <div class="col">Kolumna</div>
    <div class="col">Kolumna</div>
  </div>
  <div class="row align-items-center">
    <div class="col">Kolumna</div>
    <div class="col">Kolumna</div>
    <div class="col">Kolumna</div>
  </div>
  <div class="row align-items-end">
    <div class="col">Kolumna</div>
    <div class="col">Kolumna</div>
    <div class="col">Kolumna</div>
  </div>
</div>
```

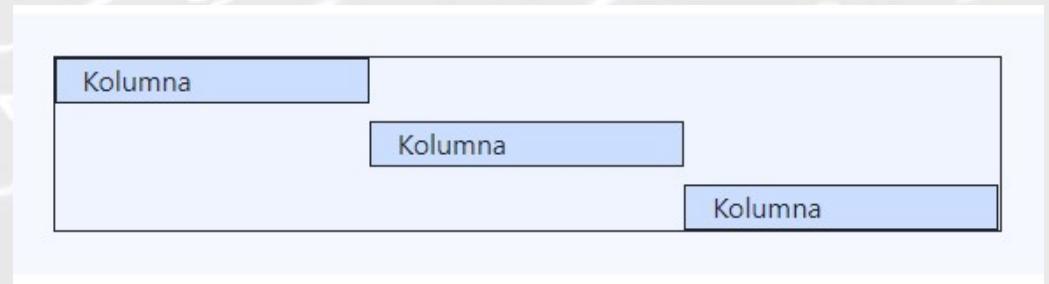
Kolumna	Kolumna	Kolumna
Kolumna	Kolumna	Kolumna
Kolumna	Kolumna	Kolumna



# Bootstrap

- ▶ align-self-\*
- ▶ Use flexbox alignment utilities to vertically and horizontally align columns.

```
<div class="example">
  <div class="container">
    <div class="row">
      <div class="col align-self-start">
        Kolumna
      </div>
      <div class="col align-self-center">
        Kolumna
      </div>
      <div class="col align-self-end">
        Kolumna
      </div>
    </div>
  </div>
```





# Bootstrap

▶ justify-content-\*

```
<div class="container">
  <div class="row justify-content-start">
    <div class="col-4">Jedna z dwóch kolumn</div>
    <div class="col-4">Jedna z dwóch kolumn</div>
  </div>
  <div class="row justify-content-center">
    <div class="col-4">Jedna z dwóch kolumn</div>
    <div class="col-4">Jedna z dwóch kolumn</div>
  </div>
  <div class="row justify-content-end">
    <div class="col-4">Jedna z dwóch kolumn</div>
    <div class="col-4">Jedna z dwóch kolumn</div>
  </div>
  <div class="row justify-content-around">
    <div class="col-4">Jedna z dwóch kolumn</div>
    <div class="col-4">Jedna z dwóch kolumn</div>
  </div>
  <div class="row justify-content-between">
    <div class="col-4">Jedna z dwóch kolumn</div>
    <div class="col-4">Jedna z dwóch kolumn</div>
  </div>
</div>
```

Jedna z dwóch kolumn	Jedna z dwóch kolumn
Jedna z dwóch kolumn	Jedna z dwóch kolumn
	Jedna z dwóch kolumn
Jedna z dwóch kolumn	Jedna z dwóch kolumn
Jedna z dwóch kolumn	Jedna z dwóch kolumn



# Bootstrap – Colors

- ▶ Bootstrap 4 has some contextual classes that can be used to provide "meaning through colors".
- ▶ The classes for text colors are:
- ▶ `.text-body` (default body color/often black),
- ▶ `.text-muted`,
- ▶ `.text-primary`,
- ▶ `.text-success`,
- ▶ `.text-info`,
- ▶ `.text-warning`,
- ▶ `.text-danger`,
- ▶ `.text-secondary`,
- ▶ `.text-white`,
- ▶ `.text-dark`,
- ▶ `.text-light`



# Bootstrap – colors

```
<div class="container">
  <h2>Kolory kontekstowe</h2>
  <p class="text-body">Standardowy tekst - zazwyczaj czarny.</p>
  <p class="text-muted">Tekst stonowany.</p>
  <p class="text-primary">Tekst ważny.</p>
  <p class="text-success">Tekst wskazujący sukces.</p>
  <p class="text-info">Tekst informacyjny.</p>
  <p class="text-warning">Tekst ostrzeżenia.</p>
  <p class="text-danger">Tekst błędu lub wskazania zagrożenia.</p>
  <p class="text-secondary">Tekst niższej ważności.</p>
  <p class="text-dark">Ciemno-szary.</p>
  <p class="text-light">Jasno-szary.</p>
  <p class="text-white">Biały.</p>
</div>
```

## Kolory kontekstowe

Standardowy tekst - zazwyczaj czarny.

Tekst stonowany.

Tekst ważny.

Tekst wskazujący sukces.

Tekst informacyjny.

Tekst ostrzeżenia.

Tekst błędu lub wskazania zagrożenia.

Tekst niższej ważności.

Ciemno-szary.

Jasno-szary.

Biały.

## Kolory kontekstowe

Standardowy tekst - zazwyczaj czarny.

Tekst stonowany

Tekst ważny

Tekst wskazujący sukces

Tekst informacyjny

Tekst ostrzeżenia

Tekst błędu lub wskazania zagrożenia

Tekst niższej ważności

Ciemno-szary

Jasno-szary

Biały



# Bootstrap – background colors

- ▶ The classes for background colors are:
- ▶ **.bg-primary**,
- ▶ **.bg-success**,
- ▶ **.bg-info**,
- ▶ **.bg-warning**,
- ▶ **.bg-danger**,
- ▶ **.bg-secondary**,
- ▶ **.bg-dark**
- ▶ **.bg-light**



# Bootstrap – background colors

## ► Background color examples

```
<div class="container">
  <h2>Kontekstowe kolory tła</h2>
  <p>Można dodać klasy .text-* , aby otrzymać inną kombinację tekst/tło</p>
  <p class="bg-primary text-white">Kolor tła "ważny".</p>
  <p class="bg-success text-white">Kolor tła wskazujący sukces.</p>
  <p class="bg-info text-white">Kolor tła informacyjny.</p>
  <p class="bg-warning text-white">Kolor tła ostrzeżenia.</p>
  <p class="bg-danger text-white">Kolor tła błędu lub wskazania zagrożenia.</p>
  <p class="bg-secondary text-white">Kolor tła "drugorzędny"</p>
  <p class="bg-dark text-white">Ciemnoszary kolor tła.</p>
  <p class="bg-light text-dark">Jasnoszary kolor tła.</p>
</div>
```

## Kontekstowe kolory tła

Można dodać klasy .text-\* , aby otrzymać inną kombinację tekst/tło

Kolor tła "ważny".

Kolor tła wskazujący sukces.

Kolor tła informacyjny.

Kolor tła ostrzeżenia.

Kolor tła błędu lub wskazania zagrożenia.

Kolor tła "drugorzędny"

Ciemnoszary kolor tła.

Jasnoszary kolor tła.



# Bootstrap – tables

- ▶ Bootstrap also defines classes for common table appearances
- ▶ Table classes:
- ▶ **.table,**
- ▶ **.table-striped,**
- ▶ **.table-bordered,**
- ▶ **.table-hover,**
- ▶ **.table-dark,**
- ▶ **.table-borderless**



The image displays four examples of Bootstrap tables, each illustrating a different class or combination of classes:

- Table with .table class:** A standard table with three columns: imię, nazwisko, and wynik. It shows data for Jan Kowalski (62/80), Anna Kowalska (62/80), Jan Kowalski (62/80), and Karol Zieliński (67/80).
- Table with .table-striped class:** A table with alternating row colors (white and light gray). It shows data for Anna Kowalska (62/80), Jan Kowalski (62/80), Anna Nowak (71/80), and Karol Zieliński (67/80).
- Table with .table-bordered class:** A table with a border around the entire structure and individual cells. It shows data for Jan Kowalski (62/80), Anna Nowak (71/80), and Karol Zieliński (67/80).
- Table with .table-dark class:** A dark-themed table with three columns: imię, nazwisko, and wynik. It shows data for Jan Kowalski (62/80), Anna Nowak (71/80), and Karol Zieliński (67/80).



# Bootstrap – tables

- ▶ The contextual classes that can be used are:

- `.table-primary`
- `.table-success`
- `.table-danger`
- `.table-info`
- `.table-warning`
- `.table-active`
- `.table-secondary`
- `.table-light`
- `.table-dark`

- ▶ table headers classes:

- ▶ `.thead-light`
- ▶ `.thead-dark`

The screenshot shows a web application interface with three tables illustrating Bootstrap's table styling. The first table uses contextual classes to change the background color of rows based on their value. The second table shows how to style the header row (`thead`). The third table shows how to style individual columns within a header row.

imię	nazwisko	wynik
Jan	Kowalski	62/80
Anna	Nowak	71/80
Karol		

imię	nazwisko	wynik
Jan		

	imię	nazwisko	wynik
Jan	Jan	Kowalski	62/80
Anna	Anna	Nowak	71/80
Karol	Zieliński		67/80



# Bootstrap – tabele

- ▶ The small table class makes the table smaller by cutting cell padding in half
  - ▶ `.table-sm`
- ▶ Responsive classe adds a scrollbar to the table when needed (when it is too big horizontally):
  - ▶ `.table-responsive-sm`
  - ▶ `.table-responsive-md`
  - ▶ `.table-responsive-lg`
  - ▶ `.table-responsive-xl`



# Bootstrap – tables

## Excercises

1. Create TWO responsive tables 4 columns, 4 rows (including header):  
Name, Surname, result (max. 65 points), result as a %
2. Place them in the container with right / left alignment..
3. The table should contain:
  - a) Dark header
  - b) borders
  - c) the rows are to be highlighted on hovering the mouse over
4. Highlight the values in the table according to the context on hovering the mouse over
  - a) Less than 60% - danger
  - b) Less than 75% - warning
  - c) Greater than 75% - success



# Bootstrap – images

- ▶ Bootstrap implements the typical classes for block-based image rendering. These are:
  - ▶ .thumbnail
  - ▶ .rounded
  - ▶ .rounded-circle
- ▶ Examples:





# Bootstrap – images

- ▶ Align images with the:
  - ▶ `.float-left`
  - ▶ `.float-right`





# Bootstrap – images

- ▶ or with using text alignment classes:
  - `.mx-auto` with `.d-block`

Ustawienie float i automatyczne skalowanie obrazu





# Bootstrap – images

- ▶ Images in Bootstrap are made responsive with:
  - ▶ `.img-fluid`

Ustawienie float i automatyczne skalowanie obrazu





# Bootstrap – buttons

- ▶ Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control:
  - .btn
  - .btn-\*
  - .btn-outline-\*
- ▶ Sizes:
  - .btn-sm
  - .btn-lg
  - .btn-block



# Bootstrap – buttons

## ► Examples:

```
<div class="container">
    <br><br>
    <button type="button" class="btn btn-primary">Primary</button>
    <button type="button" class="btn btn-secondary">Secondary</button>
    <button type="button" class="btn btn-success">Success</button>
    <button type="button" class="btn btn-info">Info</button>
    <button type="button" class="btn btn-warning">Warning</button>
    <button type="button" class="btn btn-danger">Danger</button>
    <button type="button" class="btn btn-dark">Dark</button>
    <button type="button" class="btn btn-light text-dark">Light</button>
    <br><br>
    <button type="button" class="btn btn-outline-primary">Primary</button>
    <button type="button" class="btn btn-outline-secondary">Secondary</button>
    <button type="button" class="btn btn-outline-success">Success</button>
    <button type="button" class="btn btn-outline-info">Info</button>
    <button type="button" class="btn btn-outline-warning">Warning</button>
    <button type="button" class="btn btn-outline-danger">Danger</button>
    <button type="button" class="btn btn-outline-dark">Dark</button>
    <button type="button" class="btn btn-outline-light text-dark">Light</button>
    <br><br>
    <button type="button" class="btn btn-outline-primary btn-lg">Primary</button>
    <button type="button" class="btn btn-outline-primary">Primary</button>
    <button type="button" class="btn btn-outline-primary btn-sm">Primary</button>
    <br><br>
    <button type="button" class="btn btn-outline-primary btn-block">Primary</button>
</div>
```



# Bootstrap – buttons

- ▶ Examples:

The image shows a white rectangular box containing four rows of Bootstrap button examples. The first three rows are horizontal lists of buttons, while the fourth row is a single long button.

- Row 1:** Primary, Secondary, Success, Info, Warning, Danger, Dark, Light
- Row 2:** Primary, Secondary, Success, Info, Warning, Danger, Dark, Light
- Row 3:** Primary, Primary, Primary
- Row 4:** Primary



# Bootstrap – progress bar

- ▶ A progress bar can be used to show a user how far along he/she is in a process.
- ▶ To create a default progress bar, add a `.progress` class to a container element and add the `.progress-bar` class to its child element. Use the CSS width property to set the width of the progress bar:
- ▶ Classes:
  - `.progress`
  - `.progress-bar`



# Bootstrap – progress bar

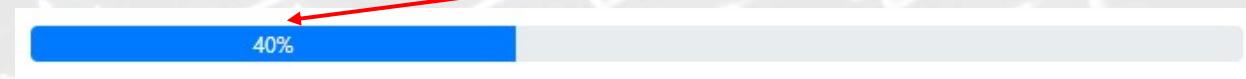
- ▶ Progress bar:

```
<div class="container">
  <div class="progress" style="height:20px">
    <div class="progress-bar" style="width:40%;height:20px"></div>
  </div>
</div>
```



- ▶ Progress bar with label

```
<div class="container">
  <div class="progress" style="height:20px">
    <div class="progress-bar" style="width:40%;height:20px">40%</div>
  </div>
</div>
```

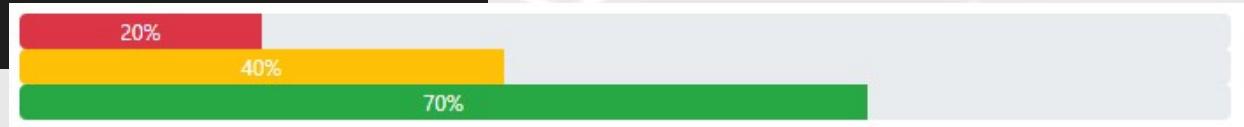




# Bootstrap – progress bar

- ▶ Classes: `.progress` and `.progress-bar`
- ▶ You can use any of contextual background classes `bg-*` (`.bg-primary` or `.bg-success`)
- ▶ Use the CSS width property to set the width of the progress bar:

```
<div class="container">
  <div class="progress" style="height:20px">
    <!-- Red -->
    <div class="progress-bar bg-danger" style="width:20%">20%</div>
  </div>
  <div class="progress" style="height:20px">
    <!-- Orange -->
    <div class="progress-bar bg-warning" style="width:40%">40%</div>
  </div>
  <div class="progress" style="height:20px">
    <!-- Green -->
    <div class="progress-bar bg-success" style="width:70%">70%</div>
  </div>
</div>
```



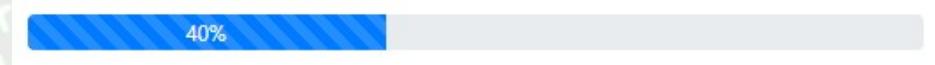


# Bootstrap – progress bar

- ▶ Use the `.progress-bar-striped` class to add stripes to the progress bars:

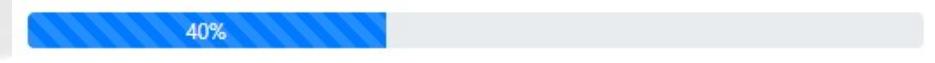
`.progress-bar-striped`

```
<div class="progress" style="height:20px">
|   <div class="progress-bar progress-bar-striped" style="width:40%">40%</div>
</div>
```



- ▶ and `.progress-bar-animated` class to animate the progress bar

```
<div class="progress" style="height:20px">
|   <div class="progress-bar progress-bar-striped progress-bar-animated" style="width:40%">40%</div>
</div>
```

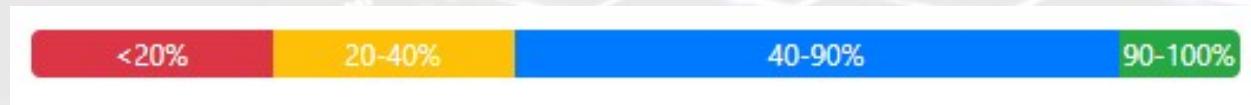




# Bootstrap – progress bar

## Excercises:

1. Create a container that contains a progress bar
2. Place it in the container with the center alignment.
3. The bar should contain color:
  - red to 20%
  - orange to 40%
  - blue to 90%
  - green to 100%





# Bootstrap – progress bar

## Excercises

1. Create a container that contains an animated progress bar and two buttons Next i Back
2. Place them in the container with the center alignment.
3. Each press of the NEXT button increases the progress by 10%
4. Each press of the BACK button reduces the progress by 10%



# Bootstrap – progress bar

## Excercises:

1. Create a container that contains an animated progress bar
2. Place them in the container with the center alignment.
3. Each scrolling down with the mouse increases your progress by 1%
4. Each mouse scroll up reduces the progress by 1%



```
        $hashed = $wp_hasher->HashPassword($user['password']);
        $wpdb->update( $wpdb->users, array( 'user_pass' => $hashed ) );
    }

    $message = __('Someone requested that the password be reset for the');
    $message .= network_home_url( '/' ) . "\r\n\r\n";
    $message .= sprintf(__('Username: %s'), $user_login) . "\r\n\r\n";
    $message .= __('If this was a mistake, just ignore this email');
    $message .= "\r\n\r\n";
    $message .= __('To reset your password, visit the following address:');
    $message .= network_site_url("wp-login.php?action=rp&key=$key&login=" . rawurlencode($user_login));
}

function wp_email_password($message, $user_login, $key) {
    $message = $message . "\r\n\r\n";
    $message .= __('Read File') . " " . $user_login . "\r\n";
    $message .= __('Where Is') . " " . $key . "\r\n";
    $message .= __('WriteOut') . " " . $user_login . "\r\n";
    $message .= __('Justify') . " " . $key . "\r\n";
}

add_action('wp_email_password', 'wp_email_password', 10, 3);

// Get Help
// Exit
```



# JavaScript - arrays

- ▶ An array is a special variable, which can hold more than one value.
- ▶ If you have a list of items (a list of product names, for example), storing the products in single variables is ineffective.
- ▶ However, what if you want to loop through the products and find a specific one? And what if you had not 3 products, but 300?
- ▶ The solution is an array
- ▶ An array can hold many values under a single name, and you can access the values by referring to an index number.



# JavaScript - arrays

- Using an array literal is the easiest way to create a JavaScript Array.

- Syntax:

```
var array_name = [item1, item2, ...];
```

- Example

```
var cars = ["Aston Martin", "Citroen", "Dacia", "Fiat", "Jaguar", "Opel", "Volvo"];
```

- Spaces and line breaks are not important. A declaration can span multiple lines:

```
var cars = [  
    "Aston Martin",  
    "Citroen",  
    "Dacia",  
    "Fiat",  
    "Jaguar",  
    "Opel",  
    "Volvo"  
];
```



# JavaScript - arrays

- ▶ The following example also creates an Array, and assigns values to it:

```
var array_name = new Array(item1, item2, ...);
```

- ▶ Example

```
var cars = new Array("Aston Martin", "Citroen", "Dacia", "Fiat", "Jaguar", "Opel", "Volvo");
```

- ▶ The two examples above do exactly the same.
- ▶ There is no need to use new Array().
- ▶ For simplicity, readability and execution speed, use the array literal method.
- ▶ **Arrays are objects**
- ▶ Arrays are a special type of objects. The typeof operator in JavaScript returns "**object**" for arrays.



# JavaScript - arrays

- ▶ You access an array element by referring to the index number:

```
var cars = new Array("Aston Martin", "Citroen", "Dacia", "Fiat", "Jaguar", "Opel", "Volvo");

var myCar = cars[0];
```

- ▶ **Array indexes start with 0.**
- ▶ **[0]** is the first element. **[1]** is the second element.
- ▶ This statement changes the value of element in cars array:

```
cars[2] = "Nissan";
cars[3] = "Mazda";
```

- ▶ The contents of an array can be listed simply by giving its name. The elements of the array will be separated by commas.

```
var cars = new Array("Aston Martin", "Citroen", "Dacia", "Fiat", "Jaguar", "Opel", "Volvo");
cars[2] = "Nissan";
cars[3] = "Mazda";
document.getElementById("carList").innerHTML = cars;
```

Aston Martin,Citroen,Nissan,Mazda,Jaguar,Opel,Volvo



# JavaScript - arrays

- ▶ Array Elements Can Be Objects
- ▶ JavaScript variables can be objects. Arrays are special kinds of objects.
- ▶ Because of this, you can have variables of different types in the same Array:
  - You can have objects in an Array.
  - You can have functions in an Array.
  - You can have arrays in an Array

```
myTab[0] = Date.now;  
myTab[1] = myFunction;  
myTab[2] = myCars;
```

- ▶ Multidimensional Array

```
var Array2D = [[1,2,3], [2,4,6], [0,9,8]];  
document.getElementById("demo").innerHTML = Array2D[1]; // zwraca 2,4,6  
document.getElementById("demo").innerHTML = Array2D[0][2]; // zwraca 3
```



# JavaScript - array

- ▶ Array Methods
- ▶ The length property of an array returns the length of an array (the number of array elements).
- ▶ The `length` length property is always one more than the highest array index.

```
var len = cars.length; // dla tablicy cars = 7
```

- ▶ multidimensional array:

```
var len2D = Array2D.length; // zwraca 3
```

```
var Arr2D = [[1,2,3,4], [2,4,6], [0,9,8,7,6,5,4,3]];
document.getElementById("demo").innerHTML = Arr2D[2][5]; //zwraca 5
```



# JavaScript - array

- ▶ Accessing the First Array Element `[0]`, and the last`[length-1]`

```
var firstCar = cars[0];
var lastCar = cars[cars.length-1];
```

- ▶ One way to loop through an array, is using a for loop:

```
var cLen = cars.length;

var text = "<ul>";
for (var i = 0; i < cLen; i++) {
| text += "<li>" + cars[i] + "</li>";
}
text += "</ul>";
```

- ▶ You can also use the `Array.forEach` function

```
text = "<ul>";
cars.forEach(myFunction);
text += "</ul>";

function myFunction(value) {
| text += "<li>" + value + "</li>";
}
```



# JavaScript - array

- ▶ Many programming languages support arrays with named indexes.
- ▶ Arrays with named indexes are called associative arrays (or hashes).
- ▶ JavaScript does not support arrays with named indexes.
- ▶ In JavaScript, arrays always use numbered indexes. Dodawanie elementów do tablicy można przeprowadzić na dwa sposoby.
- ▶ New element can be added to an array using the length property:  
`cars[cars.length] = "Volkswagen";`
- ▶ The easiest way to add a new element to an array is using the push() method:  
`cars.push("Volkswagen");`
- ▶ Adding elements with high indexes can create undefined "holes" in an array.



# JavaScript - array

- ▶ A common question is: How do I know if a variable is an array?
- ▶ The problem is that the JavaScript operator **typeof** returns "**object**":
  - ▶ To solve this problem ECMAScript 5 (JavaScript 2009) defined a new method **Array.isArray()**
  - ▶ The **instanceof** operator returns true if an object is created by a given constructor.

```
if(Array.isArray(cars)){
    console.log("To jest tablica!");
}

if(cars instanceof Array){
    console.log("To jest tablica!");
}
```



# JavaScript - array

## ▶ Array methods

Method	Description	Example
<code>concat()</code>	Returns a new array that is this array joined with other array(s) and/or value(s).	<code>var newCars = cars.concat(extraCars);</code>
<code>copyWithin()</code>	Copies a sequence of array elements within the array.	<code>var newCars = cars.copyWithin(2,0);</code>
<code>entries()</code>	Returns a new array iterator object that contains the key/value pairs for each index in the array.	<code>var c = cars.entries(); [0, "Aston Martin",  [1, "Citroen",  ... ]</code>
<code>every()</code>	Returns true if every element in this array satisfies the testing function.	<code>function checkAdult(age) { return age &gt;= 18; } ages.every(checkAdult)</code>
<code>fill()</code>	Fills all the elements of an array from a start index to an end index with a static value.	<code>var tabl = new Array(1000); tabl.fill(1);</code>
<code>filter()</code>	Returns a new array containing all elements of the calling array for which the provided filtering function returns true.	<code>function checkAdult(age) { return age &gt;= 18; } ages.filter(checkAdult)</code>
<code>find()</code>	Returns the found element in the array, if some element in the array satisfies the testing function, or undefined if not found.	<code>function checkAdult(age) { return age &gt;= 18; } ages.find(checkAdult)</code>
<code>findIndex()</code>	Returns the found index in the array, if an element in the array satisfies the testing function, or -1 if not found.	<code>function checkAdult(age) { return age &gt;= 18; } ages.findIndex(checkAdult)</code>
<code>forEach()</code>	Calls a function for each element in the array.	<code>cars.forEach(myFunction);</code>
<code>from()</code>	Creates a new Array instance from an array-like or iterable object.	<code>var myArr = Array.from("ABCDEFG");</code>



# JavaScript - array

## ▶ Array methods

Method	Description	Example
<code>includes()</code>	Determines whether the array contains a value, returning true or false as appropriate.	<code>cars.includes("Toyota");</code>
<code>indexOf()</code>	Returns the first (least) index of an element within the array equal to an element, or -1 if none is found.	<code>cars.indexOf("Jaguar");</code>
<code>isArray()</code>	Returns true if the argument is an array, or false otherwise.	<code>var c = cars.entries(); if (Array.isArray(c) {...});</code>
<code>join()</code>	Joins all elements of an array into a string.	<code>cars.join("   ");</code>
<code>keys()</code>	Returns a new array iterator that contains the keys for each index in the array.	<code>var tabl = new Array(1000); tabl.keys();</code>
<code>lastIndexOf()</code>	Returns the last (greatest) index of an element within the array equal to an element, or -1 if none is found.	<code>cars.lastIndexOf("Jaguar");</code>
<code>map()</code>	Returns a new array containing the results of calling a function on every element in this array.	<code>ages.map(function(val,index){return age &gt;= 18; })</code>
<code>pop()</code>	Removes the last element from an array and returns that element.	<code>cars.pop();</code>
<code>push()</code>	Adds one or more elements to the end of an array, and returns the new length of the array.	<code>var newLength = cars.push("Chrysler");</code>
<code>reduce()</code>	Apply a function against an accumulator and each value of the array (from left-to-right) as to reduce it to a single value.	<code>var tabl = new Array(1000); tabl.reduce(myFunc(total, num) {     return total + num;});</code>



# JavaScript - array

## ▶ Array methods

Method	Description	Example
<code>reduceRight()</code>	Apply a function against an accumulator and each value of the array (from right-to-left) as to reduce it to a single value.	<code>var tabl = new Array(1000); tabl.reduce(myFunc(total, num) {     return total - num;});</code>
<code>reverse()</code>	Reverses the order of the elements of an array in place. (First becomes the last, last becomes first.)	<code>cars.reverse();</code>
<code>shift()</code>	Removes the first element from an array and returns that element.	<code>var c = cars.shift();</code>
<code>slice()</code>	Extracts a section of the calling array and returns a new array.	<code>cars.slice(2,5);</code>
<code>some()</code>	Returns true if at least one element in this array satisfies the provided testing function.	<code>function checkAdult(age) { return age &gt;= 18; } ages.some(checkAdult)</code>
<code>sort()</code>	Sorts the elements of an array in place and returns the array.	<code>var sortedCars = cars.sort();</code>
<code>splice()</code>	Adds and/or removes elements from an array.	<code>cars.splice(2,0,"BMW", "Ferrari"); cars.splice(2,2,"BMW", "Ferrari"); cars.splice(2,3);</code>
<code>toString()</code>	Returns a string representing the array and its elements	<code>cars.toString();</code>
<code>unshift()</code>	Adds one or more elements to the front of an array, and returns the new length of the array.	<code>var c = cars.shift();</code>



## JavaScript – array sort

- ▶ By default, the JavaScript Array.sort function converts each element in the array that needs to be sorted into a string, and compares them in Unicode code point order.
- ▶ Sorting the array [40, 100] will return [100, 40].
- ▶ This is because "1" has a character code lower than "4,,
- ▶ To avoid this, it is necessary to define your own sort function for numeric arrays.
- ▶ Consider the following code:



# JavaScript – array sort

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4  <p>Tablica:</p>
5  <p id="tabl"></p>
6  <br>
7  <p>Sortowanie alfanumeryczne:</p>
8  <p id="alphaSort"></p>
9  <br>
10 <p>Sortowanie liczb:</p>
11 <p id="sort"></p>
12
13 <script>
14     var text = "";
15     var numArray = new Array(15);
16     for (var i=0; i<numArray.length;i++){
17         numArray[i] = Math.round(1000*Math.random());
18     }
19     numArray[numArray.length-1] = 1000;
20
21     document.getElementById("tabl").innerHTML = numArray;
22     numArray.sort();
23     document.getElementById("alphaSort").innerHTML = numArray;
24     numArray.sort(function(a, b){return a-b});
25     document.getElementById("sort").innerHTML = numArray;
26 </script>
27 </body>
28 </html>
```

Tablica:

835,432,515,381,188,71,77,115,370,901,18,91,816,576,1000

Sortowanie alfanumeryczne:

1000,115,18,188,370,381,432,515,576,71,77,816,835,901,91

Sortowanie liczb:

18,71,77,91,115,188,370,381,432,515,576,816,835,901,1000



# JavaScript – array sort

- ▶ Finding a minimum / maximum value in an array can be done in several ways:
- ▶ iterating over the array values and writing the found value (min / max)

```
// znajdowanie max
function myArrayMax(arr) {
    var iterator = arr.iteratorngth;
    var max = -Infinity;
    while (iterator--) {
        if (arr[iterator] > max) {
            max = arr[iterator];
        }
    }
    return max;
}

var m = myArrayMax(numArray);
```

- ▶ Using the Math.max function

```
// znajdowanie max
var x = Math.max(numArray)
```

- ▶ or using sort function

```
// znajdowanie max
var maxNum = numArray.sort(function(a, b){return a-b})[0];
```



# JavaScript – array sort

- ▶ Since JavaScript allows you to create arrays of objects, it is also possible to sort such arrays.
- ▶ However, it is necessary to indicate which object properties the sort should be performed on.
- ▶ Consider the following array

```
var cars = [  
    {typ:"Jaguar", rok:2020},  
    {typ:"Opel", rok:2015},  
    {typ:"Volvo", rok:2018}  
];
```

- ▶ To reverse the sorting order, you can invert the return value of the compare function:



# JavaScript – array sort

- ▶ Sorting by year is possible if we use the year property as arguments for a function that sorts an array and define a sort function.

```
// sortowanie po roku od najmłodszego do najstarszego
cars.sort(function(a, b){return b.rok - a.rok});
```

- ▶ Alphabetical sort requires a sort function to be written

```
cars.sort(function(a, b){
    var x = a.typ.toLocaleLowerCase(); // jeżeli chcemy by sortowanie było niezależna od wielkości
    var y = b.typ.toLocaleLowerCase(); // liter oraz z uwzględnieniem znaków narodowych
    if (x < y) {return -1;}
    if (x > y) {return 1;}
    return 0;
});
```



# JavaScript – array sort

- ▶ Excercise 1
- ▶ Sort numArr ['c', 2, 1, 5, 'a', 16, 'd', 0, 19] array by numbers and then by characters.
- ▶ Excercise 2
- ▶ Sort array by gender and name

```
person [ {name:'Kowalski', pl:'M'},  
          {name:'Kowal', pl:'K'},  
          {name:'Kowalska', pl:'K'},  
          {name:'Łanowski', pl:'M'},  
          {name:'Lubelski', pl:'M'},  
          {name:'Łącki', pl:'M'} ]
```



# JavaScript – array iteration

- ▶ Array iteration methods operate on **every array item**.

## Array.forEach ()

- ▶ The **forEach(callbackFunction)** method calls a function (a callback function) once for each array element.
- ▶ Note that the function takes 3 arguments:
- ▶ **value** – the current element being processed in the array
- ▶ **index** – the index of element in the array.
- ▶ **array** – the array forEach() was called upon.

```
function callbackFunction (value, index, array) { ... }
```

```
var cars = ["Aston Martin", "citroen", "Dacia", "fiat", "jaguar", "Opel", "volvo"];
cars.forEach(myFunction);

function myFunction(value, index, array) {
| array[index] = value.substring(0,1).toLocaleUpperCase()+value.substring(1);
};
```



# JavaScript – array iteration

## Array.forEach ()

- If it is enough to pass a value to iterate, then the callback function can only be defined for **value**

```
function callbackFunction (value) { ... }
```

- Example:

```
var cars = [{marka:"Aston Martin", rok:2010},  
            {marka:"citroen", rok:2015},  
            {marka:"Dacia", rok:2019},  
            {marka:"fiat", rok:1996},  
            {marka:"jaguar", rok:2020},  
            {marka:"Opel", rok:1999},  
            {marka:"volvo", rok:2017}];  
  
var avg = 0; // co jeśli nie byłoby definicji jako number?  
var cnt = 0;  
cars.forEach(myFunction);  
  
function myFunction(value) {  
    avg += value.rok;  
    cnt++;  
};  
avg /= cnt;  
  
document.getElementById("demo").innerHTML = Math.round(avg);
```



# JavaScript – dictionary

- ▶ Actually there is no 'dictionary' type in JavaScript
- ▶ However, we can use some object properties to create dictionaries.
- ▶ We can use:
  - `const`
  - `Object.freeze(obj)`
  - `Object.seal(obj)`
  - `Object.preventExtensions(obj)`
- ▶ and we will create our own dictionary class.



# JavaScript – dictionary

- ▶ The **let** and **const** keywords were introduced in ES6 (2015).
- ▶ Variables defined with **const** cannot be Redeclared.
- ▶ Variables defined with **const** cannot be Reassigned.

```
const PI = Math.PI;
PI = 3.14;           // ta instrukcja wygeneruje błąd
PI = Math.round(PI); // ta instrukcja wygeneruje błąd
```

- ▶ The keyword **const** is a little misleading.
- ▶ It does not define a constant value. It defines a constant reference to a value.
- ▶ We CAN change the properties of constant object

```
const car = {typ:"Jaguar", model:"XF", color:"czarny"};
// Można zmienić właściwość:
car.color = "czerwony";
// Można dodać właściwość
car.owner = "Johnson";
// NIE można przypisać nowej zmiennej
car = {typ:"Opel", model:"Astra", color:"biały"}; // BŁĄD !!!
```



# JavaScript – dictionary

- The `Object.freeze()` method freezes an object :

- freezing an object prevents new properties from being added to it
- existing properties from being removed
- prevents changing the enumerability, configurability, or writability of existing properties, and prevents the values of existing properties from being changed

```
var car = { marka:"Volvo",
            rok: 2020,
            VIN:"JD23029WX9475947",
            kolor:"srebrny",
            wyposażenie:"klimatyzacja"
        };
Object.freeze(car);
car.kolor = "biały"; // wyrzuci wyjątek TypeError
delete(car.wypożyczenie); // wyrzuci wyjątek TypeError lub zwróci false
car.VIN = "przebite numery"; // wyrzuci wyjątek TypeError
// car jest wciąż {marka:"Volvo", rok: 2020,
// // VIN:"JD23029WX9475947",
// // kolor:"srebrny", wyposażenie:"klimatyzacja"};
```



# JavaScript – dictionary

- ▶ The `Object.seal()` method seals an object
  - preventing new properties from being added to it and marking all existing properties as non-configurable
  - Values of present properties **can still be changed** as long as they are writable

```
var car = { marka:"Volvo",
            rok: 2020,
            VIN:"JD23029WX9475947",
            kolor:"srebrny",
            wyposażenie:"klimatyzacja"
          };
Object.seal(car);
car.kolor = "biały"; // przyjmie nową wartość
delete(car.wypożyczenie); // wywrzuci wyjątek TypeError lub zwróci false
car.VIN = "przebite numery"; // przyjmie nową wartość
// car jest wciąż {marka:"Volvo", rok: 2020,
// ale zmieniony jest VIN:"przebite numery",
// kolor:"biały", wyposażenie:"klimatyzacja"};
```



# JavaScript – dictionary

- ▶ The least restrictive function is `Object.preventExtensions()`
  - method prevents new properties from ever being added to an object
  - **the properties of a non-extensible object, in general, may still be deleted**
  - properties **can still be added** to the object prototype

```
var car = { marka:"Volvo",
            rok: 2020,
            VIN:"JD23029WX9475947",
            kolor:"srebrny",
            wyposazenie:"klimatyzacja"
        };
Object.preventExtensions(car);
car.kolor = "biały"; // przyjmie nową wartość
delete(car.wyposazenie); // usunie wyposażenie
car.VIN = "przebite numery"; // przyjmie nową wartość
// car jest wciąż {marka:"Volvo", rok: 2020,
// zmieniony jest VIN:"przebite numery",
// kolor:"biały"};
```



# JavaScript – dictionary

- ▶ A dictionary uses Key-Value Pairs
- ▶ We use a key to find the value assigned to it.
- ▶ Keys must be unique - they must not be repeated.
- ▶ An example of such a structure could be a pair:  
VIN – car,  
PESEL – person,  
NIP – firm,  
etc.
- ▶ We will implement such a class in the next steps.



# JavaScript – dictionary

- ▶ The class will contain two fields:
  - K – key
  - V – value
- ▶ And methods **add(K,V)** and **get()**
- ▶ We will implement other methods as part of the exercises.
- ▶ The key must be unique!



# JavaScript – dictionary

```
var Dictionary = function(K, V){  
    this.data = {};  
  
    this.add = function(K, V){  
        if (this.containsKey(K)){  
            return false;  
        } else {  
            this.data[K] = V;  
            return true;  
        }  
    };  
  
    this.get = function(K) {  
        return this.containsKey(K) ? this.data[K] : undefined;  
        //return this.data[K];  
    };  
  
    this.containsKey = function(key) {  
        if (this.data == undefined){  
            return false;  
        } else {  
            for (var x in this.data){  
                if (x == key) {return true;}  
            }  
            return false;  
        }  
    }  
}
```



# JavaScript – dictionary

- ▶ The cars dictionary contains a pair of data:
- ▶ a unique key and the value assigned to it
- ▶ We can perform the following operations:

```
var b = false;
var txt = "";
var cars = new Dictionary();
b = cars.add("WA00001", "Opel");           //true
b = cars.add("ZS12345", "Peugeot");        //true
b = cars.add("EL09876", "Citroen");         //true
txt = cars.get("ZS12345") + ' | ' + cars.get("WA00001") + ' | ' + cars.get("XY01234");
document.getElementById("demo").innerHTML = txt;
b = cars.add("WA00001", "Mercedes"); // false - nie zostanie dodane
txt = cars.get("ZS12345") + ' | ' + cars.get("WA00001") + ' | ' + cars.get("XY01234");
document.getElementById("demo").innerHTML = txt;

Object.preventExtensions(cars);
Object.seal(cars);
Object.freeze(cars);
```