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. Ffficient 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 proces
16. Developing a real-time Node.js application using Websockets
17. Automated testing (Unit tests)

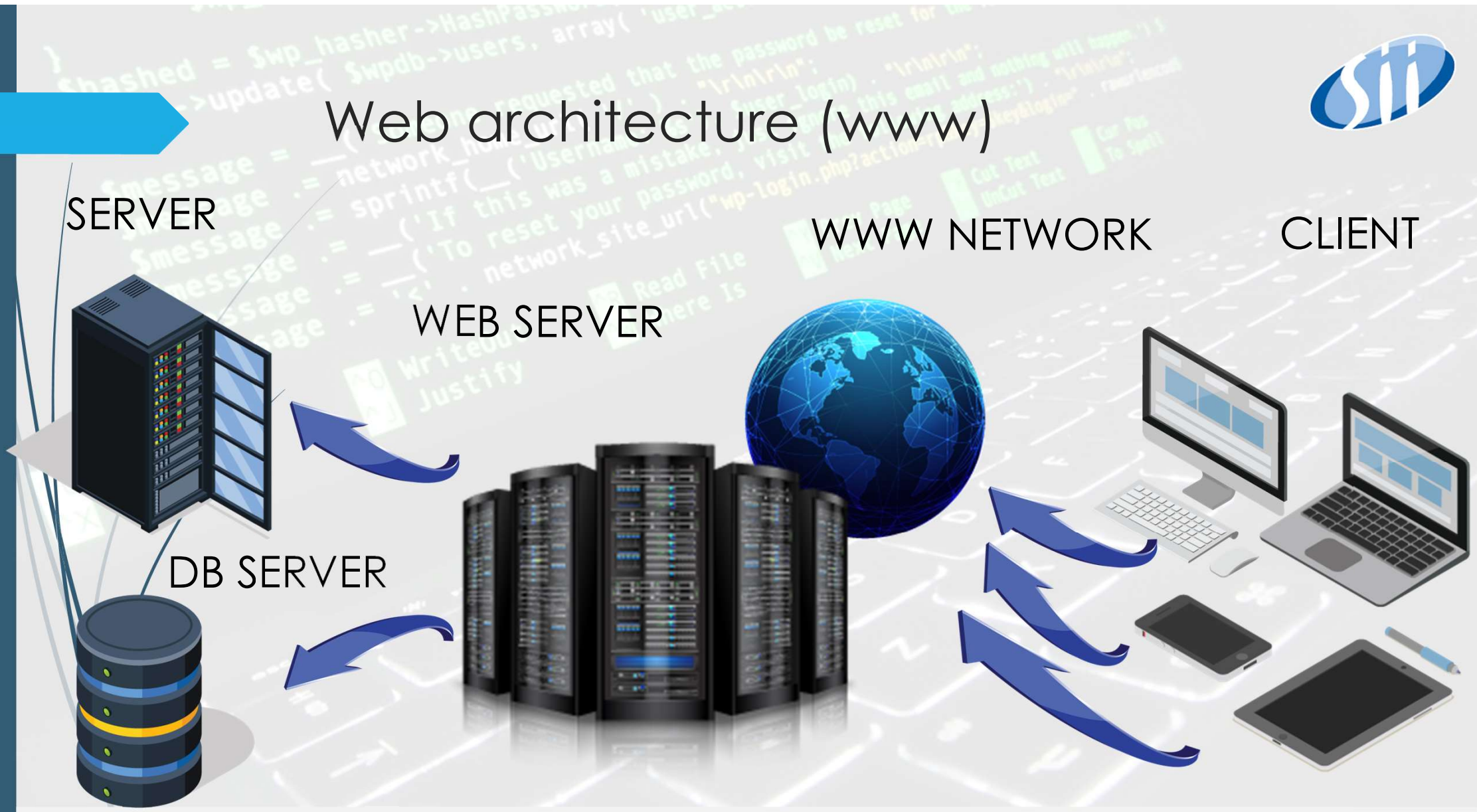
Module PLAN

- 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)



HOW A WEB SERVER WORKS?

WEB SERVER

CLIENT

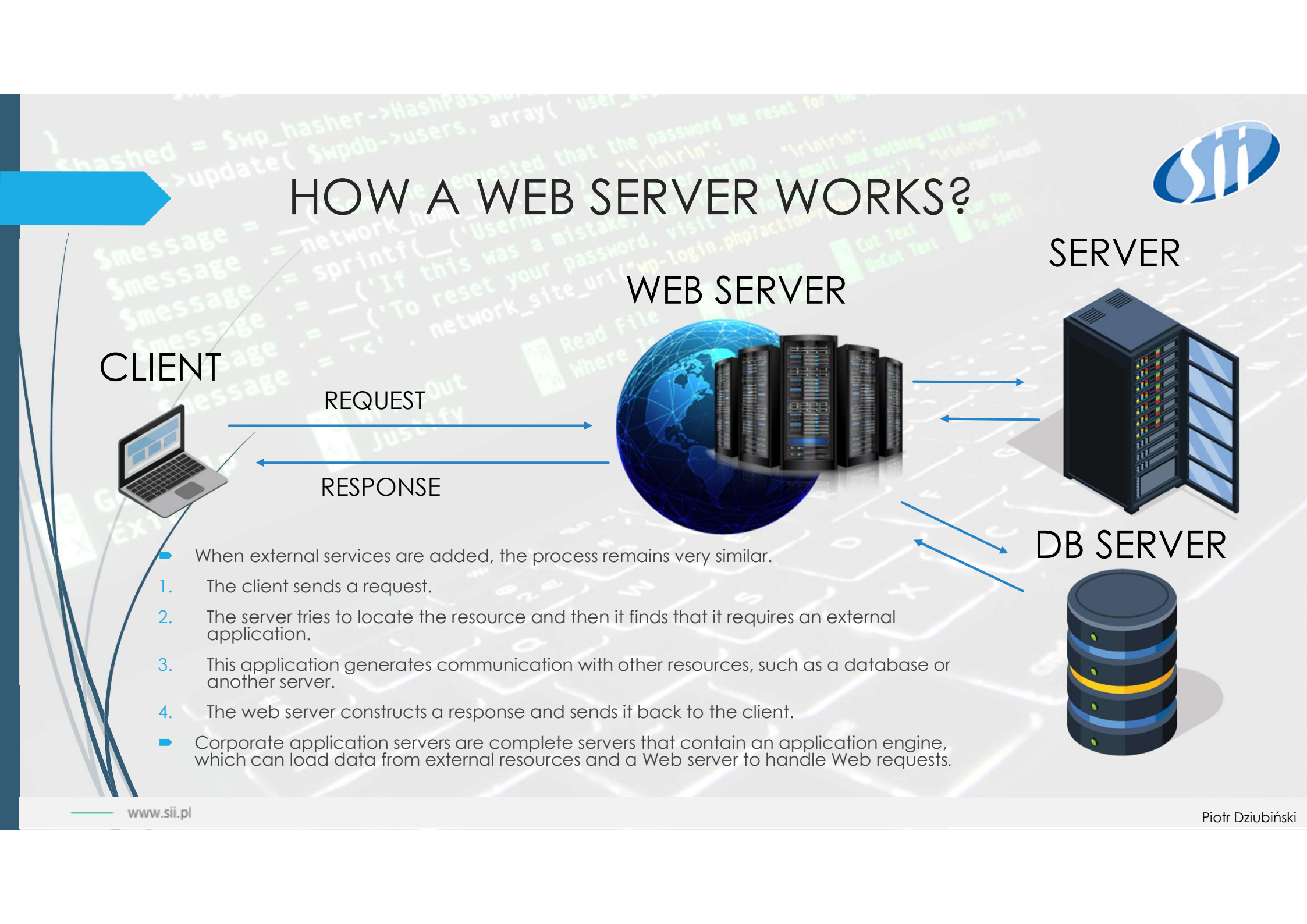
REQUEST

RESPONSE

- 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?



- 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 apllication
    - 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.

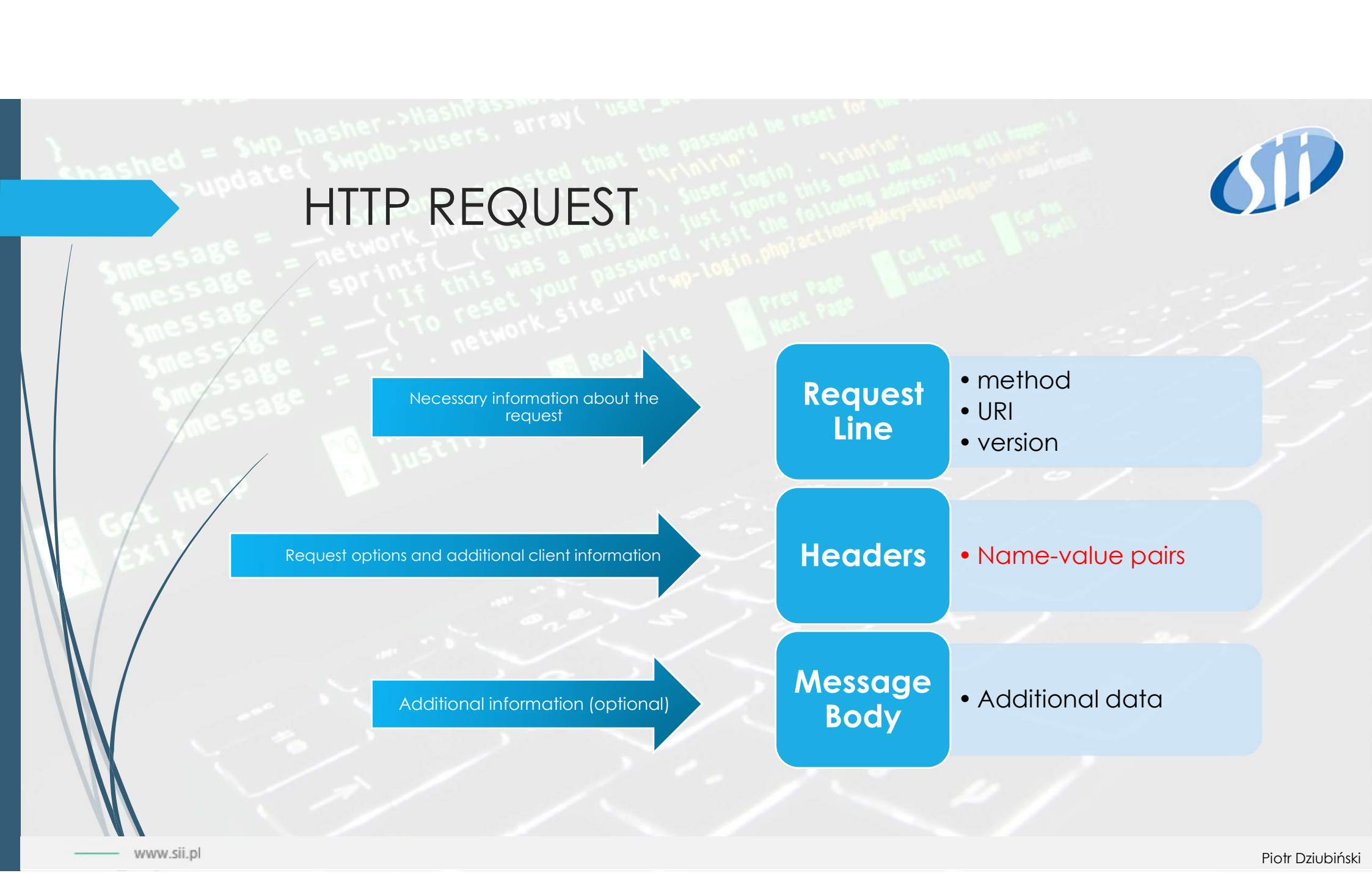
HTTP REQUEST

- 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

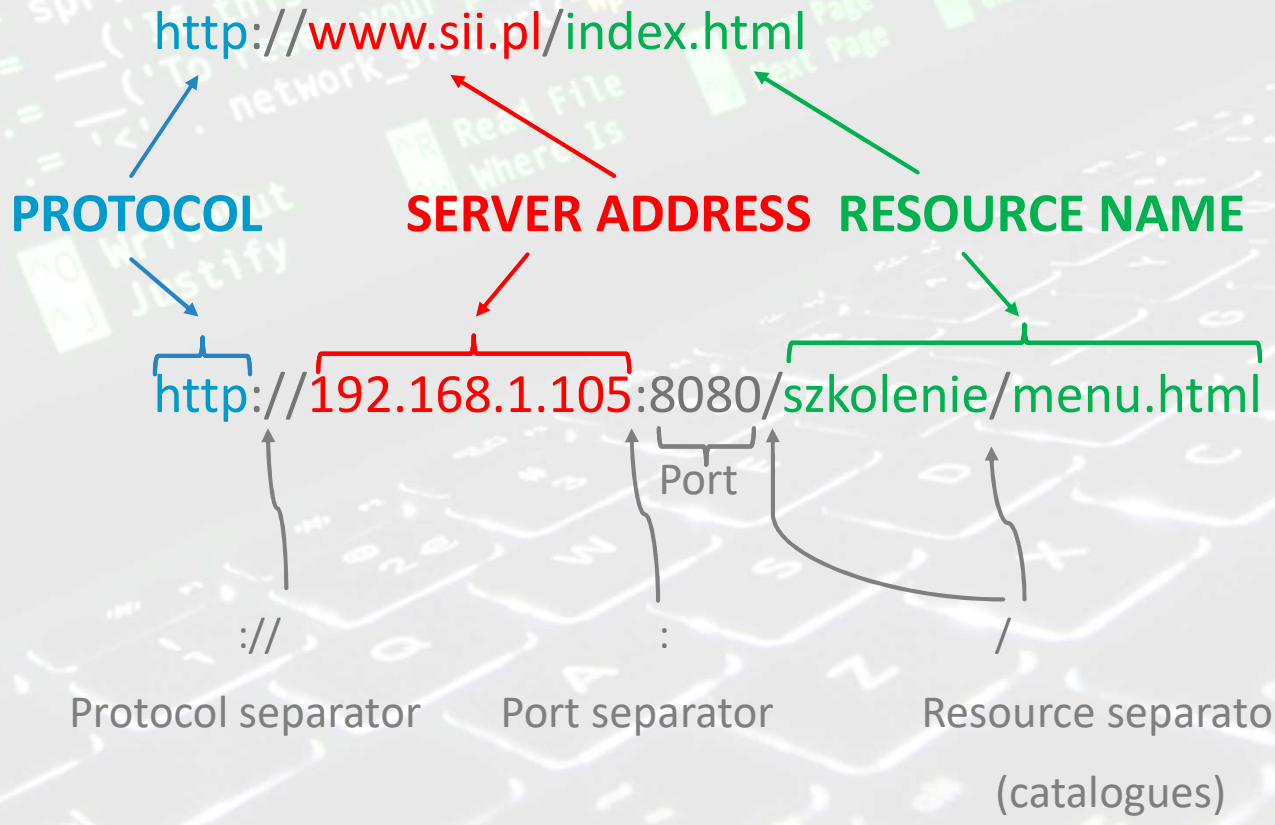
HTTP REQUEST

- 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.

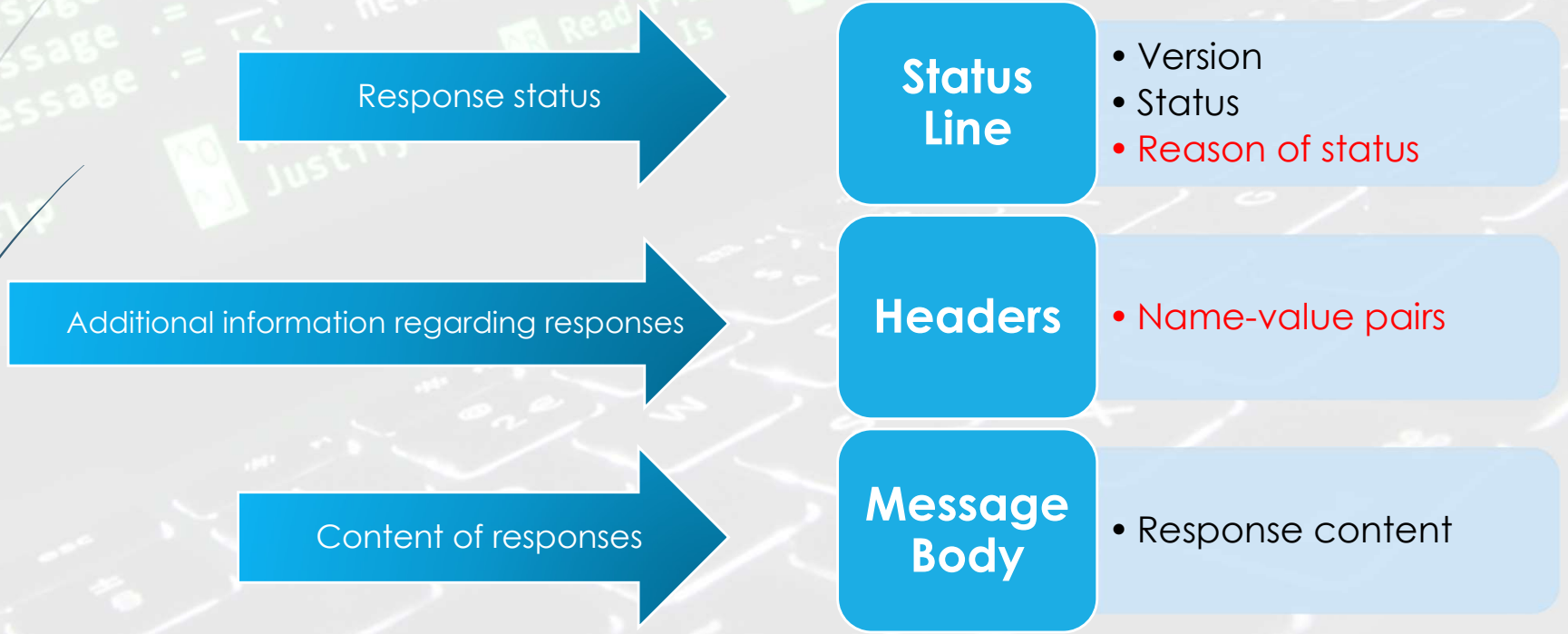
HTTP REQUEST



HTTP REQUEST: URL



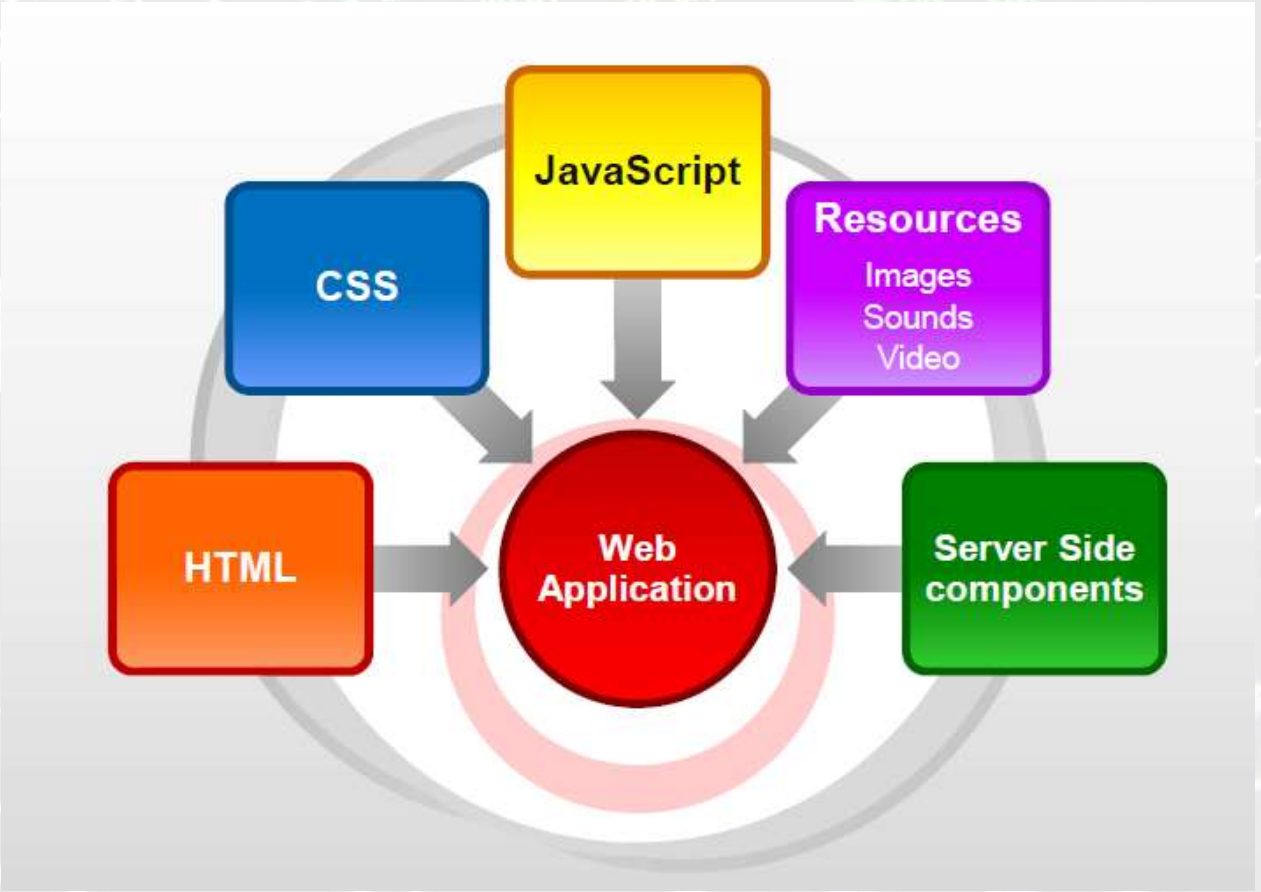
HTTP RESPONSE



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-sec10.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



Źródło: Oracle

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

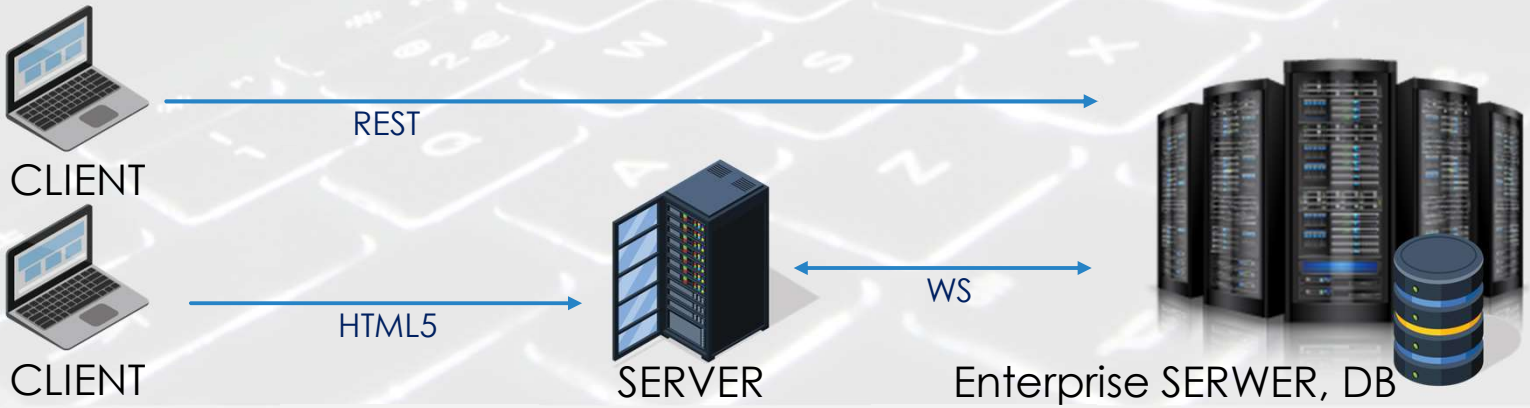
- A web application is a collection of files stored in a directory that can be aliased by a URI.
- Together, the files describe the user interface and contain the application logic that runs in a web browser.
- The user interface is defined using HTML. A consistent look and interface (colors, fonts, 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 **H**yper **T**ext **M**arkup **L**anguage
- You can use HTML to create your own website.

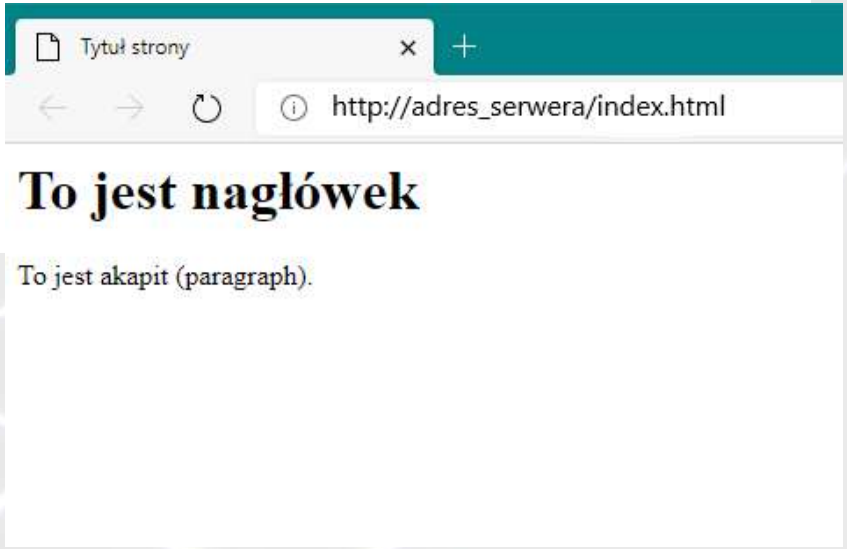
- This course is consistent with the latest HTML5 standard.

- **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 **H**yper **T**ext **M**arkup **L**anguage
- 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).

 </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
 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
<aside>	Defines content independently of page content
<audio>	Defines embedded audio content

HTML

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

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

HTML

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

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

HTML

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

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

HTML

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

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

HTML

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

Tag	Opis
<kbd>	Defines keyboard input
<label>	Defines a label for the <input> element
<legend>	Defines a signature for the <fieldset> element
	Defines a list element
<link>	Defines a relationship between a document and an external resource (most commonly used to link to style sheets)
<main>	Defines the main content of the document
<map>	Defines an image map
<mark>	Defines the selected/highlighted text
<meta>	Defines metadata about the HTML document
<meter>	Defines a scalar measurement over a familiar range (meter)
<nav>	Defines navigation links
<noframes>	Not supported in HTML5. Defines alternative content for users who do not support frames
<noscript>	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
<object>	Defines a container for an external application
	Defines an ordered list
<optgroup>	Defines a group of related options in the drop-down list
<option>	Defines an option in the drop-down list
<output>	Defines the result of a calculation
<p>	Defines paragraph
<param>	Defines a parameter for the object
<picture>	Defines a container for multiple image resources
<pre>	Defines pre-formatted text
<progress>	Represents the progress of the task
<q>	Defines short quote
<rp>	Specifies what to display in browsers that do not support ruby annotations
<rt>	Defines the explanation / pronunciation of characters (for East Asian typography)
<ruby>	Defines ruby annotation (for East Asian typography)

HTML

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

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

HTML

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

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

HTML

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

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

```
<img src="img_kurs.jpg">
```

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">`

```
<img src="img_kurs.jpg" width="800" height="600">
```

HTML attributes

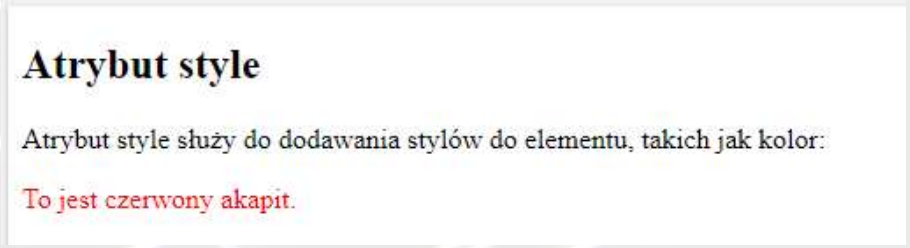
- The `alt` attribute
- The required `alt` attribute for the `<img>` tag specifies alternative text for the image if the image cannot be displayed for some reason. It could occure due to a slow connection or an error in the `src` attribute or due to usage of screen reader.
- Example

```
<img src="img_kur$.jpg" alt=" Logo kursu Front-end. ">
```

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>
```



HTML attributes

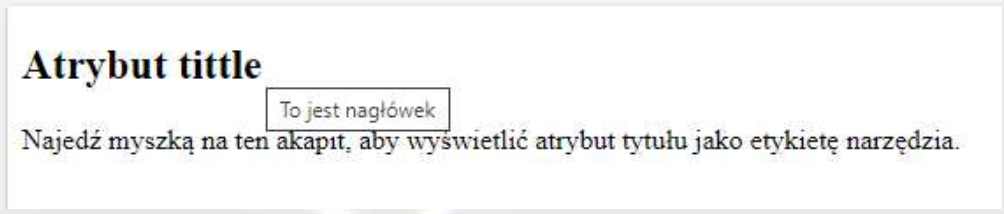
- You should always include `lang` attribute in the `<html>` tag to declare the language of the web page. The goal is to help search engines / browsers choose the display language and how they should present the symbols.
- Country codes can also be added to the language code in the `lang` attribute. 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>
```



HTML

- List of HTML tag attributes with their description

Attribute	Tag	Description
accept	<input>	Specifies the file types that the server accepts (only for type = "file")
accept-charset	<form>	Specifies the character encoding to be used when submitting the form
accesskey	All attributes	Specifies a hotkey to activate / concentrate an item
action	<form>	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	<area>, <img>, <input>	Specifies an alternative text when the original element does not display
async	<script>	Specifies if the script is prepared asynchronously (only for external scripts)
autocomplete	<form>, <input>	Determines whether the <form> or <input> element should have autocomplete turned on
autofocus	<button>, <input>, <select>, <textarea>	Specifies if the element should automatically focus when the page loads
autoplay	<audio>, <video>	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	`<button>, <fieldset>, <input>, <optgroup>, <option>, <select>, <textarea>`	Defines if the specified element / group of elements should be disabled
download	`<a>, <area>`	Specifies that the item will be downloaded when the user clicks the hyperlink
draggable	`All attributes`	Specifies whether the item can be dragged or not
enctype	`<form>`	Specifies how the form data is encoded when it is submitted to the server (only for method = "post")
for	`<label>, <output>`	Determines which elements of the form is associated with the label/calculation
form	`<button>, <fieldset>, <input>, <label>, <meter>, <object>, <output>, <select>, <textarea>`	Specifies the name of the form to which the element belongs
formaction	`<button>, <input>`	Specifies where to send the form data after submitting the form. Only for type = "submit"
headers	`<td>, <th>`	Specifies one or more header cells linked with this cell
height	`<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>`	Specifies the height of the element
hidden	`All attributes`	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	`<meter>`	Specifies a range that is considered to be a high value
href	`<a>, <area>, <base>, <link>`	Specifies the URL of the linked page
hreflang	`<a>, <area>, <link>`	Specifies the language of the linked document
http-equiv	`<meta>`	Provides the HTTP header for the information / value of the content attribute
id	`All attributes`	Specifies a unique identifier for the element
ismap	`<img>`	Specifies the image as a server-side image map
kind	`<track>`	Specifies the type of text path
label	`<track>, <option>, <optgroup>`	Defines the title of the text path
lang	`All attributes`	Defines the language of the content
list	`<input>`	Refers to a <datalist> element that contains predefined options for the <input> 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	<body>	Script to run when the web memory area is updated
onsubmit	<form>	Script to run after form submission
onsuspend	<audio>, <video>	Script (which run while media data is downloaded) is stopped before it is completely loaded for any reason
ontimeupdate	<audio>, <video>	Script to run when playing position changes (e.g., when user quickly scrolls to another media location)
ontoggle	<details>	Script to run when user opens or closes the <details> element
onunload	<body>	Script to run when page is unloaded (or browser window closed)
onvolumechange	<audio>, <video>	Script to run whenever video/audio volume changes
onwaiting	<audio>, <video>	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	<details>	Specifies that details should be visible (open) to the user
optimum	<meter>	Determines what value is the optimal value for the measure
pattern	<input>	Specifies the regular expression against which the value of the <input> element is checked

HTML

- List of HTML tag attributes with their description

Attribute	Tag	Description
placeholder	`<input>, <textarea>`	Specifies a brief indication describing the expected value of the element
poster	`<video>`	Specifies the image to be displayed while the video is downloading or until the user presses the play button
preload	`<audio>, <video>`	Specifies whether and how the author thinks the audio / video should be loaded when the page loads
readonly	`<input>, <textarea>`	Indicates that the item is read-only
rel	`<a>, <area>, <form>, <link>`	Specifies the relationship between the current document and the linked document
required	`<input>, <select>, <textarea>`	Specifies that the element must be completed before the form is submitted
reversed	`<ol>`	Specifies that the order of the list should be descending (9,8,7 ...)
rows	`<textarea>`	Defines the visible number of lines in the text field
rowspan	`<td>, <th>`	Specify the number of rows that a table cell should cover
sandbox	`<iframe>`	Enables an additional set of constraints for content in <frame>.
scope	`<th>`	Specifies whether the header cell is the header of a column, row, or group of columns or rows
selected	`<option>`	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	<area>	Defines the shape of the area
size	<input>, <select>	Specifies the width in characters (for <input>) or specifies the number of visible options (for <select>)
sizes	<img>, <link>, <source>	Specifies the size of the linked resource
span	<col>, <colgroup>	Specifies the number of columns to stretch
spellcheck	All attributes	Specifies whether the element should check spelling and grammar or not
src	<audio>, <embed>, <iframe>, <img>,<input>, <script>, <source>, <track>, <video>	Specifies the URL of the media file
srcdoc	<iframe>	Specifies the HTML content of the page to display in <frame>.
srclang	<track>	Specifies the language of the track text data (required if kind = "subtitles")
srcset	<img>, <source>	Specifies the URL of the image for use in different situations
start		Specifies the initial value of the ordered list
step	<input>	**Defines the legal number ranges for the input field**

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>	Specifies the element type
usemap	<img>, <object>	Defines an image as a client-side image map
value	<button>, <input>, , <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

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/

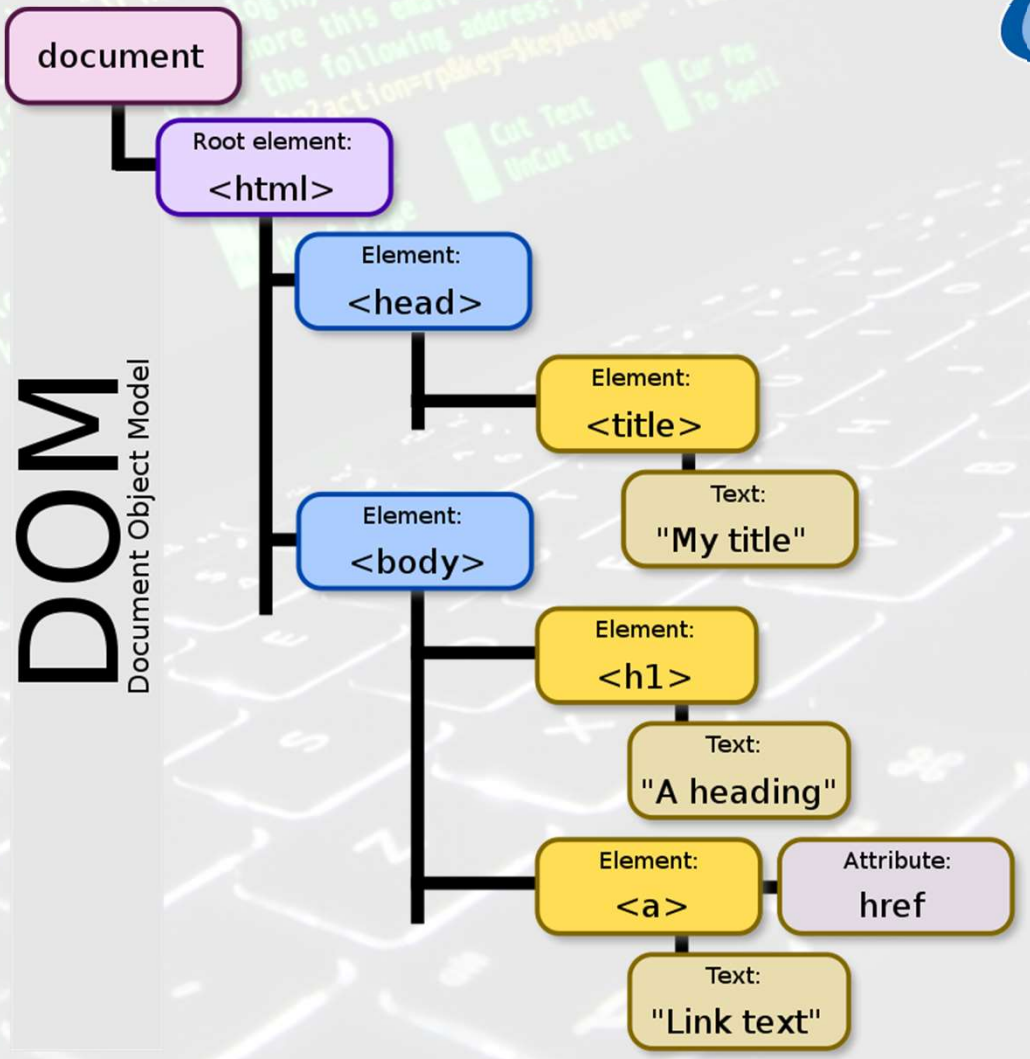
HTML DOM – Web browser

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

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>

    </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.

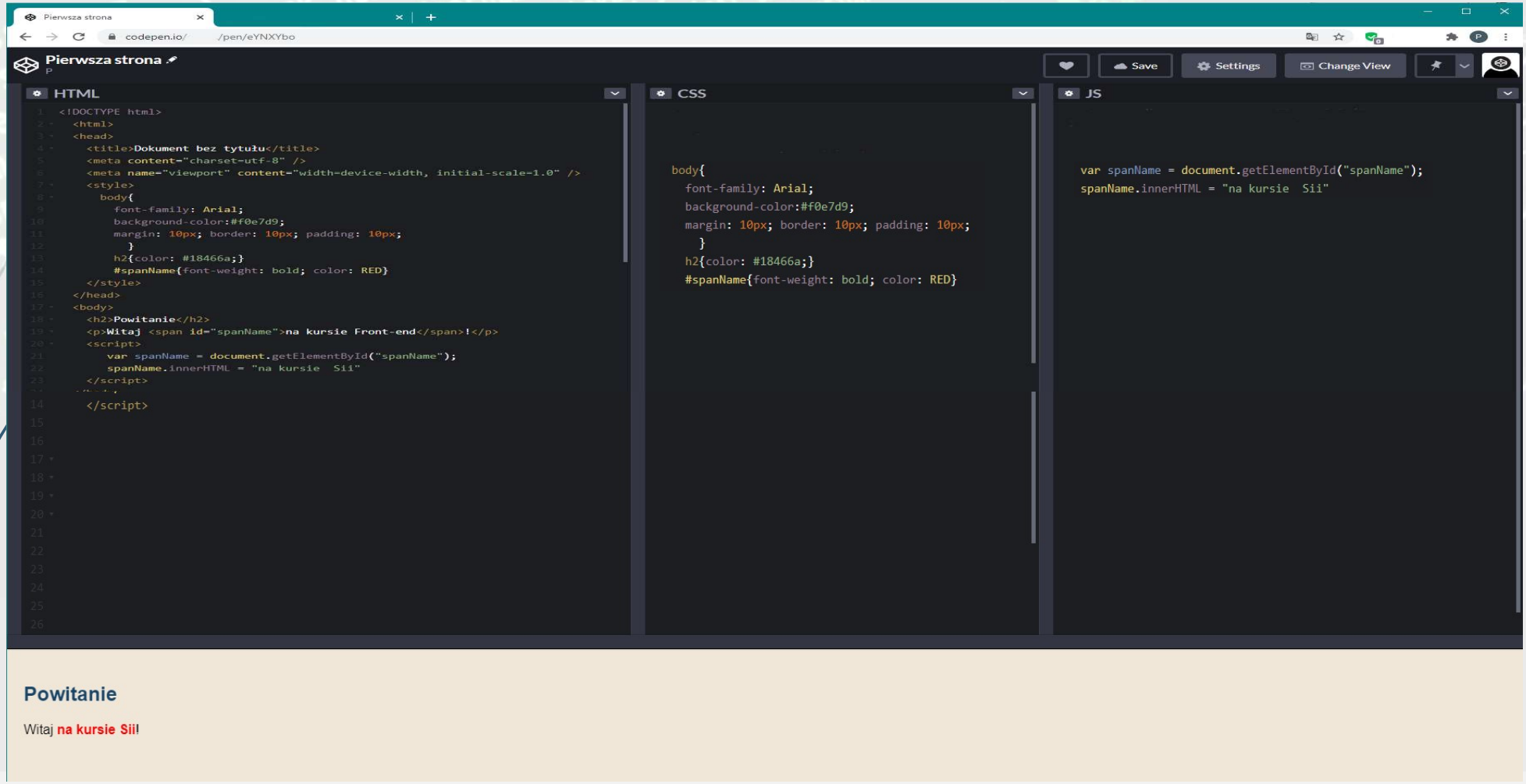
```
<style>
  body{
    font-family: Arial;
    background-color:#f0e7d9;
    margin: 10px; border: 10px; padding: 10px;
      }
    h2{color: #18466a;}
    #spanName{font-weight: bold; color: RED}
</style>
```

HTML JavaScript (Embedded JS)

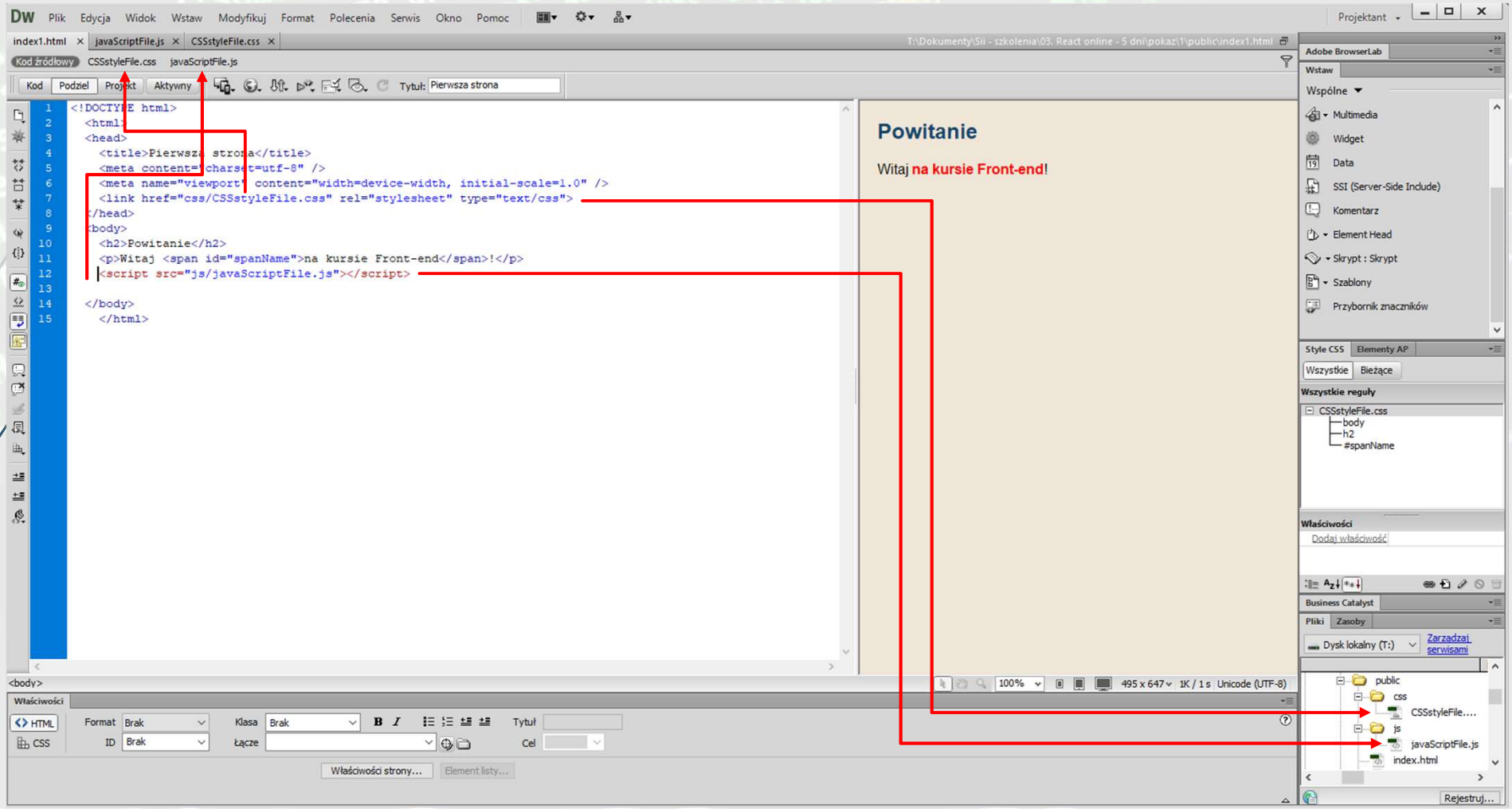
- The <script> element either contains script statements, or it points to an external script file through the src attribute.

```
<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>
```

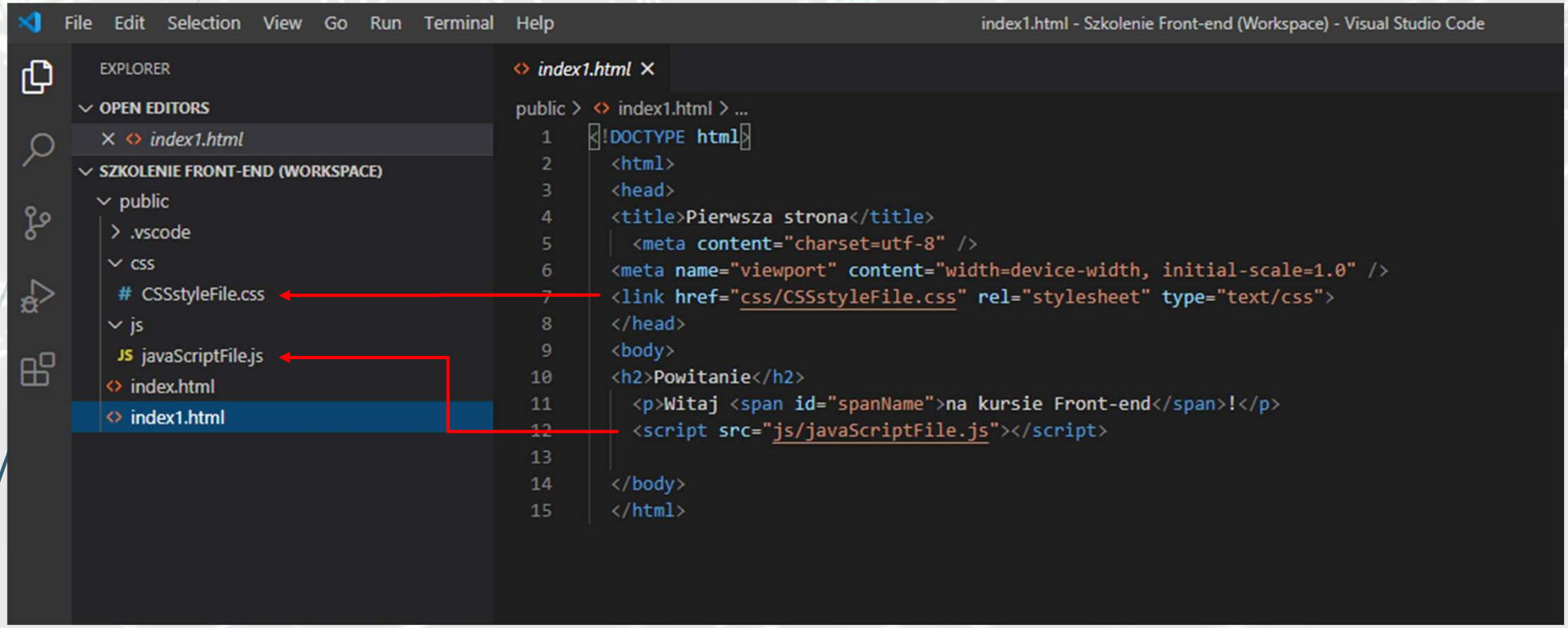
On-line codepen.io editor



Adobe Dreamweaver



Visual Studio Code



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

```html
<!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

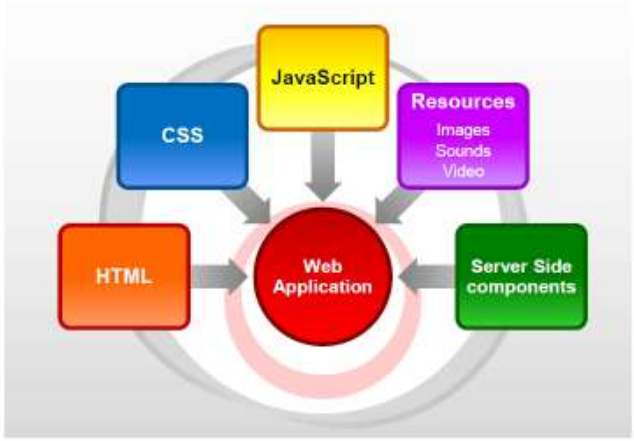
©Kurs Front-end

HTML – simple examples

```html
<!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>Nagłowek</h3>
  </div>
  <div id="mainPanel">
  <img src="img/WebApp.PNG" width="357" height="253" alt="Architektura aplikacji Web">
  <hr>
  <p>podpis pod obrazem</p>
 </div>
  <div id="footer">&copy;Kurs Front-end, 2020</div>
</div>
</body>
</html>
```

Kontener z zawartością strony

Nagłowek



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

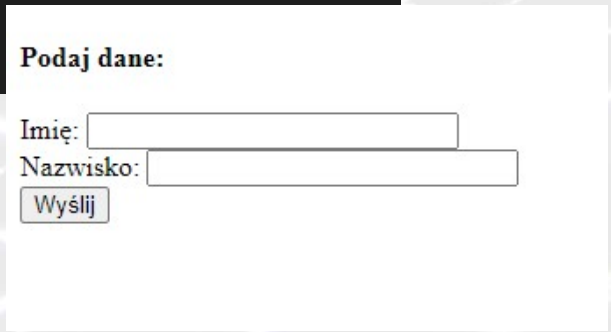
- pozycja 1
- pozycja 2

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:     <input name="imie" type="text" size="25" maxlength="30"></label><br>
<label>Nazwisko: <input name="nazwisko" type="text" size="25" maxlength="30"></label><br>
<input name="OK" type="button" value="Wyślij">
</form>
</body>
</html>
```



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

Wyślij

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>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 dolnośląskie

Wyślij

HTML+CSS – simple examples

```html
<!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>
```

```css
p
{
    text-align:left;
    color:rgb(153, 0, 0);
    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
}
```

Lista nieuporządkowana

- pozycja 1
- pozycja 2

Lista uporządkowana

1. pozycja 1
2. pozycja 2

HTML+CSS – simple examples

```html
<!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>
```

```css
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;}
```

Lista nieuporządkowana

- pozycja 1
- pozycja 2

Lista uporządkowana

1. pozycja 1
2. pozycja 2

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:

- 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":

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

```html
<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:

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

```html
<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:

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

```html
<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:

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

```html
<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):

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

```html
<h3>Nagłowek h3</h3>
<h4>Nagłowek h4</h4>
<p>Tekst akapitu</p>
```

Nagłowek h3

Nagłowek 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

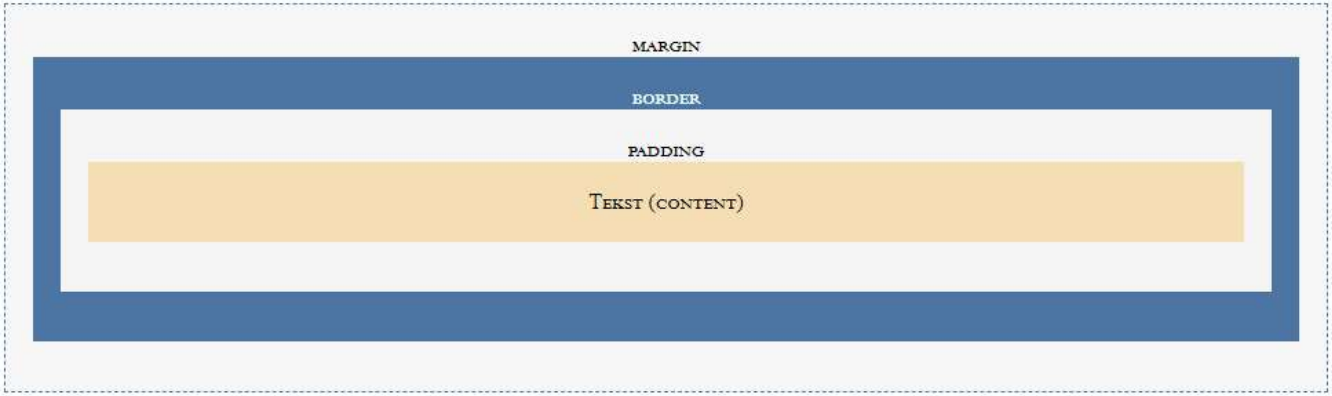
- Examples:

```css
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;
}
```

```html
<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>
```

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:

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

- This 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

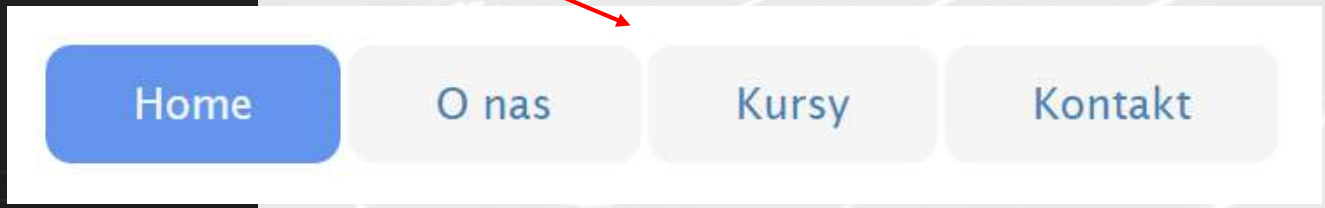
- Hiding an element can be done by setting the display property to none (`display:none`).
- The element will be hidden, and the page will be displayed as if the element **is not there**:.

- `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:

```html
<!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 U
    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	Zieliński	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	Zieliński	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 50px

```
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 – Tabeles

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

```css
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 – Tabeles

- 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:

```css
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:

```css
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:

```css
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

- Excersise:
  - 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.

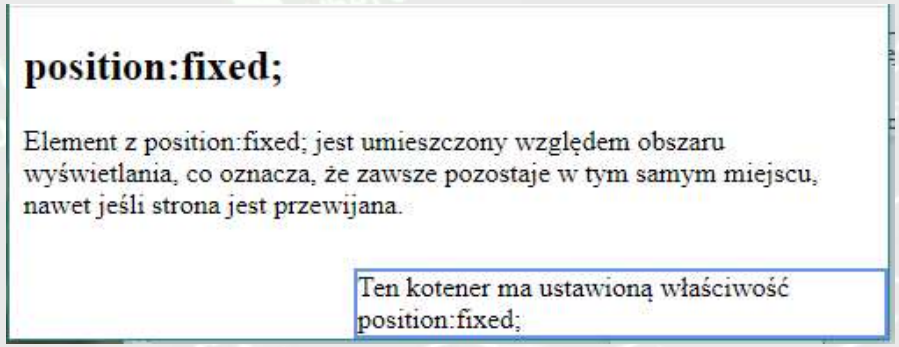
CSS – Position

- 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;`



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:

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

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

Aby usunąć przyleganie (stickyness) 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 repudiandae 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 repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

Jestem elementem "lepkim"!

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

Aby usunąć przyleganie (stickyness) 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 repudiandae 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 repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

ski

CSS – z-index

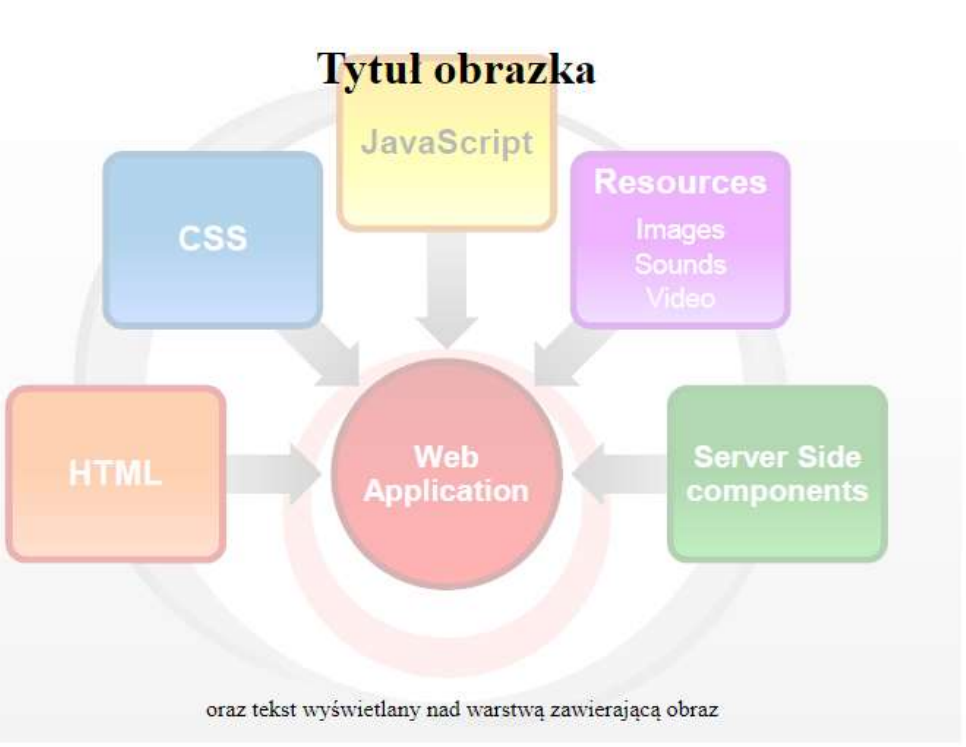
- The `z-index` property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

- **`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>
  <img src="img/WebApp.PNG">
  <p style="position: absolute; top:460px; left:200px;">
    oraz tekst wyświetlany nad warstwą zawierającą obraz</p>
</div>

</body>
</html>
```



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 flo `float` at 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

JavaScript

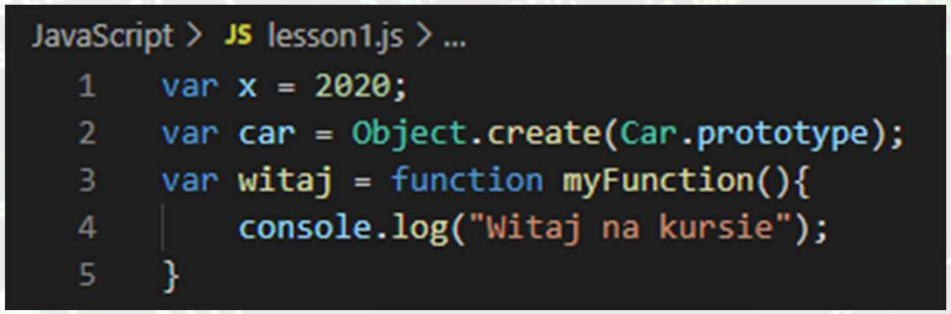
- JavaScript (JS) was initially created to “make web pages alive”.
- The programs in this language are called scripts. They can be written right in a web page’s HTML and run automatically as the page loads.
- Scripts are provided and executed as plain text. They don’t need special preparation or compilation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



- 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;
```

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
```

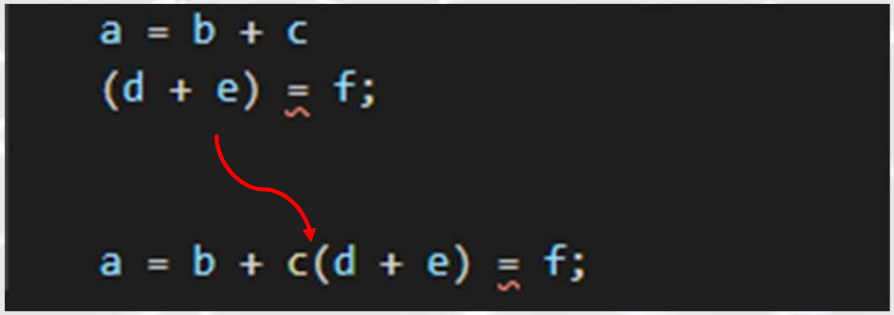
JavaScript

- 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.



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_tez_jest_zmienna
_inna_zmienna
$zmienna
MojaZmienna
```

JavaScript – słowa zastrzeżone

- 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`.

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

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
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

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
==	equal to
===	equal value and **equal type**
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	mniejsze lub równe
?	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	Descritpion
&&	AND
||	OR
!	NOT

Operator	Description
typeof	checks the type of object.
instanceof	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	++	Postfix Increment	i++
18	--	Postfix Decrement	i--
17	++	Prefix Increment	++i
17	--	Prefix Decrement	--i
17	!	Logical not	!(x==y)
17	typeof	Type	typeof x

JavaScript - Operator Precedence Values

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

JavaScript - Operator Precedence Values

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

JavaScript - Operator Precedence Values

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

JavaScript - Operator Precedence Values

Value	Operator	Description	Example
3	+=	Assignment	x += y
3	/=	Assignment	x /= y
3	-=	Assignment	x -= y
3	*=	Assignment	x *= y
3	%=	Assignment	x %= y
3	<<=	Assignment	x <<= y
3	>>=	Assignment	x >>= y
3	>>>=	Assignment	x >>>= y
3	&=	Assignment	x &= y
3	^=	Assignment	x ^= y
3	|=	Assignment	x |= y

JavaScript - Operator Precedence Values

Value	Operator	Description	Example
2	**yield**	Pause Function	yield x
1	**,**	Comma	5 , 6

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 – *instrukcje*

- 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;
```