# HTML, CSS, JAVASCRIPT

Dariusz Zarzycki

software
**development**
academy

# HTML

## HTML - HYPERTEXT MARKUP LANGUAGE

HTML is a markup language for creating web documents. HTML describes the meta-data, structure and contents of a web page.

# HTML - VERSIONS

HTML standard is evolving, some of the milestones:

- HTML 2.0 was published in 1995
- HTML 3.2 was published in 1997
- HTML 4.0 was published in 1997
- HTML 4.1 was published in 1999
- XHTML 1.0 (an XML based HTML) was published in 1999
- HTML5 was published in 2011

## HTML DOCUMENT

HTML document structure is similar to XML. We will refer to it's nodes as tags. It also consists of text nodes, doctype declaration and comments.

# HTML DOCUMENT - BASIC STRUCTURE

```
<!DOCTYPE html>
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

## DOM - DOCUMENT OBJECT MODEL

If a HTML document is being loaded it is represented by a tree, which nodes are HTML elements (created according to document tags). The root of the tree is the `<html>` element. It's children are `<head>` and `<body>` elements, and so on.

## BROWSER'S DEVELOPER TOOLS

Most browsers offer a set of developer tools. They provide information about the document DOM and each element. They also can be used to manipulate the document structure, provide a script console and much more.

To access developer tools (in most browsers) you can use ctrl+shift+i (or option+command+i) shortcut.

# DOCTYPE

Doctype is the first element of a HTML document. It is an information about what version of HTML document is written in.

HTML5 document type declaration:

```
<!DOCTYPE html>
```

# TAGS

Document elements are represented by tags.
Element's start is represented by start tag (`<tag>`)
and it's end by end tag (`</tag>`)

```
<tag>content...</tag>
```

If a tag has no content it can be closed immediately:

```
<tag />
```

Some tags do not require the ending tag.

# TAG ATTRIBUTE

HTML elements can be described by tag attributes. An attribute name and value are placed in the start tag.

```
<tag attribute-name="attribute-value">content...</tag>
```

Attribute specifies additional information about the element.

## HTML COMMENTS

HTML document can contain comments. They are not displayed by the browsers.

```
<!-- This is a comment -->
```

# <HEAD>

# <HEAD>

<head> tag contains the document meta-data (data that provides information about data). It can specify document's title, charset, keywords, viewport, etc.

# <TITLE>

<title> tag specifies document's title. It is required in a HTML document. The title is presented by the web browser. Web page search engines use the title tag content to present the page.

# <META>

<meta> tag can specify one of the multiple meta-data. It does not have content and does not require end tag.

```
<meta charset="UTF-8">
<meta name="description" content="Document description">
<meta name="author" content="John Smith">
<meta name="viewport" content="initial-scale=1.0">
```

# <LINK>

<link> tag defines a link to an external document. It is often used to use external stylesheets (more on the topic later).

```
<link rel="stylesheet" type="text/css" href="style.css">
```

# <BODY> TAG

Body tag defines contents of a web document. It contents describe HTML elements that are rendered by the web browser.

## HEADER TAGS

A <h1>, <h2>, ... <h6> tags are used to create headers (h1 is the most important one, h6 the least important one).

# HEADER TAGS EXAMPLE

```
<h1>This is important</h1>
<h2>This is not as important</h2>
<h6>This is still a header, but the least important one</h6>
```

Results with:

# THIS IS IMPORTANT

## THIS IS NOT AS IMPORTANT

### THIS IS STILL A HEADER, BUT THE LEAST IMPORTANT ONE

# <DIV> TAG

A div tag creates a block section in a document.

```
<div>First block</div>
<div>Second block</div>
<div>Third block</div>
```

Results with:

First block
Second block
Third block

# <P> TAG

A p tag creates a paragraph. By default paragraphs have a margin between each other.

```
<p>This text is a paragraph</p>
<p>This text is another paragraph</p>
```

Results with:

This text is a paragraph

This text is another paragraph

# <SPAN> TAG

A span tag is an inline tag - creates an element that will be (by default) displayed in a line.

```
This is a line of text <span>with a span element</span> inside
```

Results with:

This is a line of text with a span element inside.

# <A> TAG

An a tag represents a hyperlink. It's `href` attribute indicates the link's destination.

```
<a href="http://google.com">Link to google.com</a>
```

Results with:

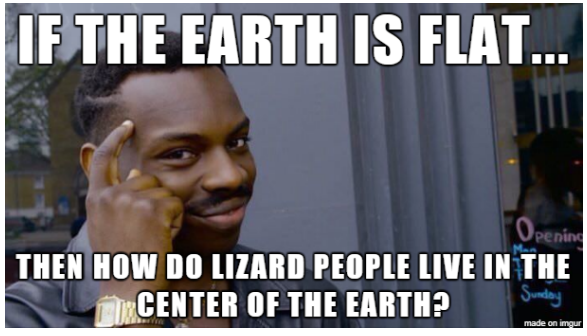
Link to google.com

# <IMG> TAG

`img` tag places an image on the page. The `src` attribute specifies location of an image (relative to the document or absolute).

```
<img src="img/meme.png">
```

Results with:



# <UL> TAG

`ul` tag creates an unordered list. List's element is created by the `li` tag.

```
<ul>
  <li>First element</li>
  <li>Second element</li>
</ul>
```

Results with:

- First element
- Second element

# <OL> TAG

`ol` tag creates an ordered list. It is similar to the `ul` tag, but the elements' bullets are replaced with numbers (or letters) indicating the order.

```
<ol>
  <li>First element</li>
  <li>Second element</li>
</ol>
```

Results with:

1. First element
2. Second element

# WHITESPACE CHARACTERS

Multiple whitespace characters do not influence the outcome of a HTML document. If we want to put multiple whitespace characters on a web page, we should use special characters. For a line break we can use the <br> tag.

```
First line<br>Second line
```

Results with:

First line
Second line

## SPECIAL CHARACTERS

To place any character in a HTML document we can use the `&xxxx` pattern, where `xxxx` is the name of the character of it's decimal or hexadecimal number.

- Non-breaking space: ` `
- > symbol: `>`
- < symbol: `<`

## <TABLE> TAG

`table` tag creates a table. It can contain header rows and body rows (contents).

# <TABLE> TAG EXAMPLE

```
<table>
  <thead>
    <tr>
      <th>First header</th>
      <th>Second header</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>First cell</td>
      <td>Second cell</td>
    </tr>
  </tbody>
</table>
```

Results with:

| First header | Second header |
|---|---|
| First cell | Second cell |

# `colspan` AND `rowspan` ATTRIBUTES

Table cells can be merged. Cell's expansion is defined by a `colspan` or a `rowspan` attribute.

```
<table>
  <tbody>
    <tr>
      <td>Cell 1A</td>
      <td colspan="2">Cell 1B</td>
    </tr>
    <tr>
      <td>Cell 2A</td>
    </tr>
    <tr>
      <td colspan="2">Cell 3A</td>
    </tr>
  </tbody>
</table>
```

# `colspan` AND `rowspan` ATTRIBUTES

Results with:

| | |
|---|---|
| Cell 1A | Cell 1B |
| Cell 2A | |
| Cell 3A contents | |

# EMPHASIZING TEXT TAGS

Text can be emphasized by using `em`, `strong` or `code` tags.

```
<em>Em tag contents</em><br>
<strong>Strong tag contents</strong><br>
<code>Code tag contents</code><br>
```

*Em tag contents*

**Strong tag contents**

`Code tag contents`

# HTML FORMS

# <FORM> TAG

Web pages often contain user forms. HTML defines multiple tags to create form elements.

Whole form is put in the <form> tag.

```
<form>
  ...form contents...
</form>
```

# TEXT INPUT ELEMENTS

```
<form>
  <!-- single-line text input-->
  <input type="text" value="Default value">
  <!-- multiple-line text input-->
  <textarea rows="3" cols="20">Default value</textarea>
</form>
```

Results with:

Default value

Default value

# INPUT LABEL

A <label> tag represents an input label. The `for` attribute's value refers to the `id` attribute's value of the input.

```
<form>
  <label for="some-input">Label for the input</label>
  <input type="text" value="some input" id="some-input">
</form>
```

Results with:

Label for the input some input

# BUTTONS

```
<form>
  <!-- a button that can contain html tags-->
  <button><strong>Displayed</strong> text</button>
  <!-- simple button-->
  <input type="button" value="Displayed text">
</form>
```

Results with:

**Displayed** text  Displayed text

# RADIO BUTTONS

Radio buttons is a set of buttons of which one can be pressed. Buttons in the same set have the same `name` attribute value.

```
<form>
  <label for="option-1">Option 1
    <input type="radio" name="set-of-options" id="option-1">
  </label>
  <label for="option-2">Option 2
    <input type="radio" name="set-of-options" id="option-2">
  </label>
</form>
```

Results with:

Option 1 Option 2

# CHECKBOX

```
<form>
  <label for="some-checkbox">
    <input type="checkbox" id="some-checkbox"> Checkbox descri
  </label>
</form>
```

Results with:

☐ Checkbox description

# DROPDOWN

Dropdowns are created by the <select> tags.

```
<form>
  <label for="some-select">A dropdown</label>
  <select id="some-select">
    <option>Option 1</option>
    <option selected>Option 2
    <option disabled>Option 3
  </select>
</form>
```

Results with:

A dropdown Option 2

# CSS - CASCADING STYLE SHEETS

## CSS - CASCADING STYLE SHEETS

Cascading Style Sheets is a language used to define the style of a HTML document.

## STYLING HTML DOCUMENT

CSS can be applied to a HTML document in three ways:

- refering to an external style sheet document
- defined in a <style> tag in the head part of the document
- an element can be styled by it's `style` attribute

# REFERENCE TO EXTERNAL STYLE SHEET

To refer to an external style sheet the <link> tag can be used.

```
<head>
  ...
  <link rel="stylesheet" type="text/css" href="style.css">
  ...
</head>
```

# CSS STYLE SHEET DOCUMENT

Style sheet describes the set of rules that will be applied to style the document. Each rule starts with a CSS selector, followed by a set of properties applied according to the selector.

```
selector1 {
  property1: some-value;
  property2: some-value;
  ...
}

selector2 {
  ...
}
```

# CSS SELECTORS

## CSS SELECTORS

A CSS selector defines a pattern to select elements to be styled.

# ID SELECTOR

To select an element by it's `id` attribute value (the value of `id` attribute has to be unique!) use #id-attribute-value pattern.

## HTML:

```
<button id="my-element">Text</button>
```

## CSS:

```
#my-element {
  color: red;
}
```

## Results with:

Text

# CLASS SELECTOR

To select elements by their `class` attribute value (multiple elements can have the same `class` attribute value) use .class-attribute-value pattern.

HTML:

```
<button class="my-class">Text</button>
<div class="my-class">Text</div>
```

CSS:

```
.my-class {
  color: red;
}
```

Results with:

Text

Text

# TAG SELECTOR

To select elements by their `tag` name use tag-name pattern.

## HTML:

```
<button>Text</button>
<button>Text</button>
```

## CSS:

```
button {
  color: red;
}
```

## Results with:

Text Text

# DESCENDANT SELECTOR

CSS selector can be combined. The following example:

```
#my-element div {
  ...
}
```

will select all div elements that are descendants of an element with "my-element" id.

# CHILD SELECTOR

Selector:

```
#my-element>div { {
  ...
}
```

selects all div elements that are children of an element with "my-element" id.

# NEXT ELEMENT SELECTOR

Selector:

```
label+p { {
  ...
}
```

selects all p elements that are placed immediately after label elements.

# COMBINED SELECTOR

Selector:

```
selector1, selector2 { {
  ...
}
```

selects all elements selected by selector1 and all elements selected by selector2.

# ATTRIBUTE BASED SELECTOR

Elements can be selected by their attributes' values.

Selector:

```
[name=abc] { {
  ...
}
```

selects all elements with attribute name with value "abc".

# CSS PROPERTIES

# COLOR PROPERTIES

`color` properties sets the text color. `background-color` sets the background color of an element. The most popular patterns of color value are: english name, hexadecimal rgb, decimal based rgb or decimal based rgba (color with opacity).

```
button {
  color: red;
  background-color: brown;
}
```

Results with:

Text

# WIDTH AND HEIGHT

If element's size is required to be set, you can use height and width properties. The value's units can be: cm, mm, in (inches), px, pt (1/72 of an inch), pc (12pt) or %.

```
button {
  width: 100px;
  height: 50px;
}
```

Results with:

Text

# FONT STYLING

Font can be styled in multiple ways.

```
button {
  font-family: "Courier-New";
  font-size: 25px;
  text-decoration: underline;
  font-weight: bold;
  font-style: italic;
}
```

Results with:

***Text***

# MARGINS

Elements can have inner and outer margins. Inner margin is defined by `padding` property, outer margin by `margin` property.

```
button {
  padding: 10px;
  margin: 20px;
}
```

Results with:

Text Text

# BORDER

Elements can also have borders. Border property can be defined by three values at once: thickness, style and color.

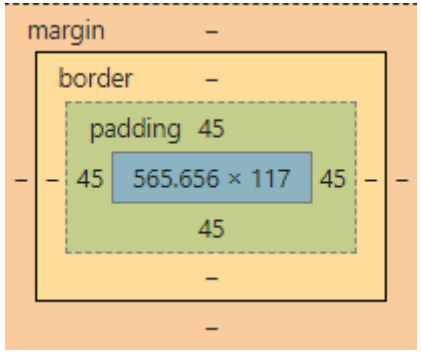
```
button {
  border: 3px dashed blue;
}
```

Results with:

Text

# ELEMENTS SIZE

HTML element is put in a box which size is: element size + padding + margin + border. This is often referred to as the "Box model" term.



# DISPLAY PROPERTY

`display` property defines how an element is displayed.

Some of the values:

- `display: block;` - an element starts on a new line and takes up the whole width.
- `display: inline;` - element is a part of a line. `height` and `width` properties have no effect.
- `display: inline-block;` - element is a part of a line, but it's width and height can be set.
- `display: none;` - element is not displayed at all.

# CSS PSEUDO-CLASSES

## PSEUDO-CLASSES

A pseudo-class is used to select an element according to it's state or it's placement. It can be used to select n-th elements, focused elements, active links, etc.

# :HOVER PSEUDO-CLASS

:hover pseudo-class selects an element under the mouse cursor.

```
button:hover {
  color: red;
}
```

The result of the example is when a mouse is over a button it's color is red.

# :NTH-CHILD PSEUDO-CLASS

:nth-child pseudo-class selects elements according to their order.

```
<button>button1</button>
<button>button2</button>
<button>button3</button>
<button>button4</button>
```

```
button:nth-child(2n) {
  color: red;
}
```

Results with:

button1 button2 button3 button4

# PSEUDO-ELEMENT :BEFORE

HTML elements can be also created by CSS. A selector that ends with :before will create elements (pseudo-elements) and put them as the first child in each element selected by the selector.

```
<button>button1</button>
<button>button2</button>
```

```
button:before {
  content: "abc";
  color: red;
}
```

Results with:

abcbutton1 abcbutton2

# PSEUDO-ELEMENT :AFTER

Pseudo-element `:after` does the same thing `:after`, but it places the new element at the end of the tag.

```
<button>button1</button>
<button>button2</button>
```

```
button:after {
  content: "abc";
  color: red;
}
```

Results with:

button1abc button2abc

# @MEDIA RULE

`@media` rule is used to apply different styles for different screen sizes/devices.

```
@media (max-width: 500px) {
  p {
    ...
  }
}
```

Example above results with styling <p> tags differently, when screen's width is less than or equal to 500px.

# <STYLE> TAG

CSS rules can be applied using <style> tag placed in the <head> tag.

```
  <style>
selector1 {
  attribute1: value;
  attribute2: value;
}

...
  </style>
```

# INLINE STYLE

HTML element can be styled by specifying css properties directly in the style attribute of the tag.

```
<div style="color: red; background-color: blue">Text</div>
```

# JAVASCRIPT

# JAVASCRIPT

JavaScript is programming language that makes web pages interactive.

# <SCRIPT> TAG

Scripts are put in a <script> tag.

```
<script>...script...</script>
```

<script> tag can also be used to load script from an external file. The closure of script tag is required.

```
<script src="script.js"></script>
```

# DECLARING VARIABLES

A variable can be declared using `var` keyword. JavaScript is weakly typed, it means that we don't declare variable's type.

```
var x;
```

# BASIC TYPES

Strings are put in the double or single quotes.

```
var text1 = "abc";
var text2 = 'abc';
```

Numbers literals are put directly.

```
var number1 = 123;
var text2 = 5.01234;
```

Boolean literals are put directly.

```
var boolean1 = true;
var boolean2 = false;
```

## CONSOLE.LOG

To print value in the browsers console `console.log` method can be used.

```
console.log("this will be printed in the console");
```

# BASIC OPERATORS

Most of the common operators work similarly to Java operators.

```
console.log(1 + 2); // prints 3
console.log(1 % 2); // prints 1
console.log("abc" + "def"); // prints "abcdef"
console.log(true && false); // prints false
```

# == VS ===

In JavaScript there are two comparison operators. If we want to compare values we use == operator. If we want to compare values and types we use ===.

```
console.log(5 == 5); // prints true
console.log("5" == 5); // prints true
console.log("5" === 5); // prints false (types differ)
```

# ARRAYS

An array is created by using brackets - elements are put inside of them.

```
var array = [1, 4, 7];
```

An array's element and length are referred to just like it is done in Java.

```
console.log(array[2]); // prints 7
console.log(array.length); // prints 3
```

To add element to array we use the push method

```
array.push(9);
```

# OBJECTS

Objects are created using JSON (JavaScript Object Notation).

```
var object = {
  property1: "value",
  property2: 123,
  property3: false
}
```

Object's property can be accessed in two ways:

```
console.log(object["property1"]);
```

or

```
console.log(object.property1);
```

# FUNCTIONS

Functions a declared as follows:

```
function functionName(firstParameterName, secondParameterName)
  // instructions
  return "some value";
}
```

Function is called the standard way:

```
functionName(firstParameterValue, secondParameterValue);
```

# FUNCTION EXAMPLE

```
function add(a, b) {
  return a + b;
}

console.log(add(5, 7)); // prints 12
```

# OTHER JS AND JAVA SIMILARITIES

There are multiple other syntax similarities between these languages (and many other C-family programming languages).

```
for(var i = 0; i < 10; i++) {
  ...
}

while(condition) {
  ...
}
```

```
if(condition) {
  ...
} else ...

switch(x) {
  case X:
    ...
}
```

## GETTING A DOM ELEMENT

The `document` object represents the whole DOM. It can be used to get a particular element or a set of elements.

```
var element = document.getElementById("some-id");
var elements = document.getElementsByClassName("some-class");
```

# ACCESSING ELEMENT'S PROPERTIES

Element's inner HTML is referred by innerHTML property.

```
var innerHTML = element.innerHTML;
element.innerHTML = "<div>New inner HTML</div>";
```

To get the list of properties simply print the element.

```
console.log(element);
```

# JQUERY

# JQUERY

jQuery is a library that makes programming in JavaScript easier. It simplifies the code a lot.

## ADDING JQUERY TO THE DOCUMENT

jQuery can be added by downloading the library (http://jquery.com/download/), and adding:

```
<script src="jQuery.version.js"></script>
```

where the "version" is downloaded version. It is important to load the jQuery before the script that uses it.

# JQUERY SELECTORS

One of the most powerful jQuery's tools are selector methods. The method is called $ (just the dollar sign). The argument can be any css selector:

```
var $divElements = $("div");
// prints a collection of all div elements
console.log($divElements);
```

# JQUERY ELEMENT METHODS

We can easily modify DOM with the usage of jQuery, for example:

```
var $element = $("#some-id");
// sets the new inner html
$element.html("<div>new html</div>");
// sets the inner text
$element.text("new text");
// applies css rules
$element.css({color: "blue", "background-color": "red"});
// sets the attribute value
$element.attr("id", "new-id");
// sets the attribute value
$element.addClass("class-name");
```

# AJAX

AJAX (Asynchronous JavaScript and XML) - a set of techniques used to send requests asynchronously (in the background) by the web application to the server and perform actions when the response is retrieved.

# JQUERY AJAX METHOD

jQuery offers a set of useful methods to perform AJAX requests. The most universal one is the `$.ajax` method.

```
$.ajax({
  url: "....",
  method: "GET",
  success: function(response) {
    console.log("Success! Server responsed with: " + response)
  }
});
```

A GET method request will be send to a location specified by url property. If response's status code indicates that request was performed correctly, then the method passed as the value of the success property is called.

# FURTHER READING

- Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages
- Head First JavaScript Programming: A Brain-Friendly Guide
- HTML and CSS: Design and Build Websites
- JavaScript and JQuery: Interactive Front-End Web Development
- Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics