HTML & CSS

Course Material • Hamburg Coding School • January 2022

—

Outline

Introduction

Three Languages for Web Development

Almost all webpages are built using 3 languages:

- **HTML** for the structure and content of the website,
- **CSS** for the layout and design,
- **JavaScript** to animate, work with data and overall behavior of the site.

Coding Principles

- You don't need to remember every little rule (you can look that up). But you need to understand *how* it works.

- When learning to code, make heavy use of search engines like google.

- RTFM - 'read the frigging manual' 😉 - Most popular frameworks have excellent documentation. Use it!

IDE - Integrated Development Environment

Your IDE (a fancy name for a code editor) is your most important tool.

In this course, you can use any of these:

- **Visual Studio Code** https://code.visualstudio.com/
- **Atom** https://atom.io/
- **Sublime Text** https://www.sublimetext.com/

Chrome Developer Tools

You can look at the HTML and CSS of any website in the Chrome browser by using the Chrome Developer Tools. You can open them at:

- **View > Developer > Developer Tools**, or
- Right-click on the website and select **Inspect**.

We will make use of the Developer Tools throughout the course.

HTML

All Websites are Documents

A website is a document, just like any text file or Word document.

It has the file ending `*.html`

You can create a new file on your laptop and call it `mywebsite.html`. If you open it, it will open in your web browser.

Websites on the internet are not much more. They are HTML documents that are saved on a server, and can be opened in your browser from there.

What is HTML?

HTML stands for: **HyperText Markup Language**.

It is a *structural language*, that means it is used for declaring how the document is structured and which text is displayed.

It is **not** a *programming language*, like for example JavaScript. You cannot write logic with it, like with `if` and `else`.

> 💡 **History:** HTML was originally created for displaying academic papers and linking them to each other.

HTML Syntax

HTML describes the structure of a Web page, i.e. what is a text block, what is a headline, what is a link, and so on. For that, it uses *HTML elements*. HTML elements tell the browser how to display the content.

An HTML element always consists of an opening and a closing *tag*. A tag is written in pointy brackets. It looks like this:

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

The `<p>` tag tells the browser: here starts a text block.
Then there is the text, which the browser will show.
Then comes the closing tag: `</p>` It tells the browser: the text block ends here.

The enclosed text is called the *text content* ("This is a paragraph" in this example).

In addition to the text content, a tag can also have one or multiple **attributes**. This will not be shown in the browser, but is used for additional information like styles.

An attribute has a **key** and a **value**, of the form: `key="value"`.
We will see more about that later.

Element

```
<p class="important">This is a paragraph</p>
```

Start tag   Attribute   Value   Content   End tag

Image: http://desarrolloweb.dlsi.ua.es/cursos/2011/html5-css3/html-basics
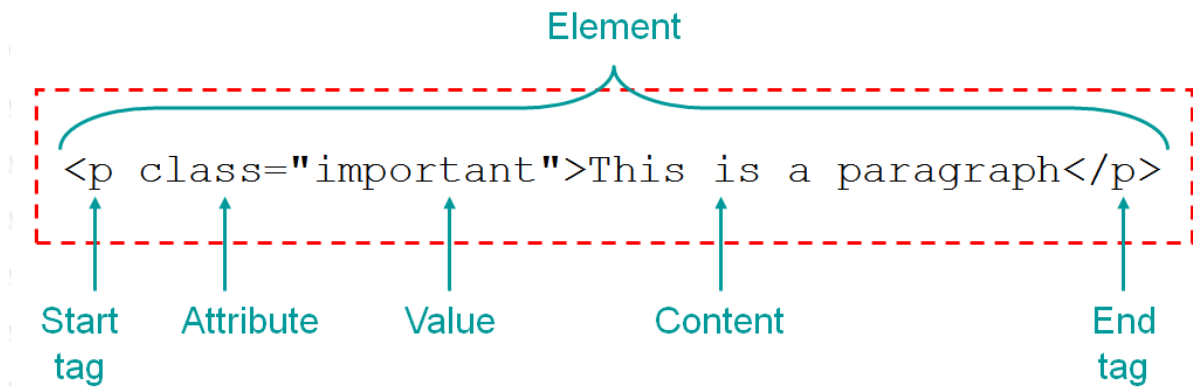
💡 **Good to know**: W3Schools has a list of all HTML tags. It is a great site to look up everything about HTML and CSS: https://www.w3schools.com/tags/default.asp

Text Blocks and Headlines

A paragraph in HTML looks like this:

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

Another text block is the `<div>` block, short for division.

```
<div>This is a text block</div>
```

The difference to a `<p>` block is that it doesn't have any space before or after.

A headline looks like this:

```
<h1>This Is A Title</h1>
```

There are six different kinds of headlines, **h1** to **h6** (largest to smallest):

```
<h1>Largest Headline</h1>
<h2>Second Largest</h2>
<h3>Third Largest</h3>
<h4>Fourth Largest</h4>
<h5>Fifth Largest</h5>
<h6>Smallest Headline</h6>
```

Empty Elements

There are elements that are empty, meaning they don't have a closing tag.

An example is the **
** tag, the line break.

```
<p>Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed
diam nonumy eirmod tempor invidunt ut labore et dolore magna
aliquyam erat, sed diam voluptua.<br>
At vero eos et accusam et justo duo dolores et ea rebum. Stet
clita kasd gubergren, no sea takimata sanctus est Lorem ipsum
dolor sit amet.</p>
```

The browser will only display line breaks if you put a **
** tag into the text. The tag itself will not be shown.

> 💡 **Optional Knowledge**: In the strict version of HTML we need to close every tag, including empty ones. This means, a line break should actually look like: **
**

Links

Some long time ago, links on websites were called *hyperlinks*. Nowadays we don't call them like this anymore, but this is what gave HTML its name: **Hyper**Text Markup Language. Links are probably the most important feature of HTML.

With links, you can create a clickable link in your document that leads to another document.

If you have **mywebsite.html**, and another page called **about.html**, you can create a link in `mywebsite.html` that leads to `about.html` like this:

```
<a href="about.html">About</a>
```

The **href** attribute specifies the target document where the link should lead to. In this example, it leads to `about.html`.

Alternatively, you can also link to an external, public website by specifying the address.

```
<a href="https://hamburgcodingschool.com">Where I learned HTML</a>
```

The text in between the **<a>** tags is the text that the browser will show, the text that will be clickable. By default, links are displayed blue and underlined.

Nested Blocks

In HTML, tags are usually nested into each other. You might have a div, in which you have a headline and a text block, in which you have links.

```
<div>
  <h1>My First Website</h1>
  <p>
    Welcome to my first website. I'm happy that I learned how to
use HTML to write my own websites. Btw, this is
<a href="https://hamburgcodingschool.com">where I learned
HTML</a>.
  </p>
</div>
```

Note that we indent each line according to how it is nested, meaning we add whitespace at the beginning of the line: the deeper the nesting level the more.

> 💡 **Good to know**: In HTML, everything is a block. You have blocks of text, blocks of images, and blocks of blocks nested into each other. Every block is rectangular, just the sizes and their arrangement is different. You can imagine tags as boxes: you can put boxes into boxes into boxes...

Comments

In HTML, you can write **comments**. Comments will not be displayed in the browser. They are only visible in the source code.

```
<!-- Survey Form -->
<div class="container">...
```

Comments are often used by developers to put extra information into the HTML code, like hints about a block of HTML code. This makes it easier to quickly look over a document and understand what a code block is for.

Comments are also great for debugging HTML, because you can **comment out** lines of code, one at a time, to search for errors:

```
        </div>
      </div>
<!-- maybe one closing tag too many?
</div>
-->
```

💡 **Good to know**: In most editors you can use a shortcode for that: select the text you wish to comment out and press ⌘ + **/** on a Mac or **Ctrl** + **/** on a PC.

HTML Document Structure

Each website (= each HTML document) has a basic structure that looks like this:

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    ...
  </body>
</html>
```

It starts with the *document type declaration*: `<!DOCTYPE html>` that explains to the browser that the document is written in the newest version of HTML: **HTML5**.

The HTML document itself begins with the opening **`<html>`** tag and ends with the closing **`</html>`** tag.

Inside the html tags are the **`<head>`** and **`<body>`** tags.

The **`<head>`** element is a container for metadata about the HTML document. It typically defines the title of the document, styles, scripts, and other meta information. Metadata is not displayed.

Everything visible on the webpage is defined between the **`<body>`** tags.

> 💡 **Good to know**: Browsers are very patient, and try to display anything that you write in the code. Even if you leave out the **`<html>`**, **`<head>`** and **`<body>`** tags, it will still display your content. But it might not have the desired results.
>
> In general, the rule applies: ***Be kind to your browser, and your browser will be kind to you.*** Meaning: try to write well-formed HTML and stick to the document structure. Then you minimize the errors or unexpected results that you get from your browser.

Head

The **`<head>`** part of the HTML document contains metadata and other data that is not part of the document structure, but has some other function.

A typical head looks like this:

```
<head>
  <title>Hamburg Coding School</title>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <meta name="last-modified" content='Mon, 06 Jan 2020 16:23:39 CET'>
  <meta name="description" content="Programmier-Kurse, die Spaß machen.">
  <meta property="og:title" content="Hamburg Coding School"/>
  <meta property="og:description" content="Programmier-Kurse, die Spaß machen."/>
  <meta property="og:image" content="https://hamburgcodingschool.com/class-201811.jpg"/>
  <meta name="twitter:card" content="summary_large_image">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1">
  <link rel="stylesheet" href="https://hamburgcodingschool.com/css/codingschool.css">
  <link rel="shortcut icon" href="/favicon.png">
  <style>
    h1 { color:red; }
    p { color:blue; }
  </style>
  <script>
    window.onload = function() {
      if (window.matchMedia("(max-width: 48em)").matches) {
        location.hash = "#content-start";
      }
    }
  </script>
</head>
```

`<title>`	Contains the title of the website = the name that is displayed in the browser tab.
`<meta>`	Includes meta information that is used for e.g. displaying snippets in search results or link previews in social media.
`<link>`	Includes CSS into the document, or loads the favicon (shortcut icon).
`<style>`	Defines local CSS styles.
`<script>`	Defines local JavaScript.

Metadata

The **`<meta>`** tags contain metadata: data that is not displayed on the page, but is machine parsable. Meta elements typically contain information about

- page description,
- keywords,
- author of the document,
- last modified or
- thumbnail image.

Metadata is mostly used by search engines or by social media to display snippets with title, description and a thumbnail image.

Viewport (optional knowledge)

Viewport is a special meta tag.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

HTML5 introduced a method to let web designers take control over the viewport, through the **<meta>** tag. The viewport is the user's visible area of a web page and varies with the device (mobile phone, tablet, computer screen).

You should include the **<meta>** viewport element in all your web pages.

The **viewport** element gives the browser instructions on how to control the page's dimensions and scaling:

- The **width=device-width** part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- The **initial-scale=1.0** part sets the initial zoom level when the page is first loaded.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

Without the viewport meta tag   With the viewport meta tag

Source: https://www.w3schools.com/css/css_rwd_viewport.asp

Text Formatting

HTML brings some tags for structuring and formatting your text.

<p>	A paragraph. Adds space before and after the text.
** **	A line break. Single tag without closing tag.
<div>	A box. Unlike <p>, <div> doesn't add space before or after the text.
<h1>	A headline. You can choose from <h1> (largest) to <h6> (smallest).
<b>	Makes the text bold. Inline tag: does not break the text.
<i>	Makes the text italic. Inline tag: does not break the text.
<u>	Underlines the text. Inline tag: does not break the text.
<small>	Makes the text smaller. Inline tag: does not break the text.

> 🤓 **Want to know more?** There are a lot more tags available. A good cheat sheet for your overview is this one: https://www.simplehtmlguide.com/cheatsheet.php

HTML Entities

Some characters will not render in HTML. A problem is, for example, if you want to use a > or a < in your text. Since HTML uses these characters for tags, how can we make it see it as text content instead? For this, we use HTML entities.

```
<p>The result is: a &gt; b</p>
```

The cryptic text **>** will be rendered as a > character in the text.

Similarly, we have a bunch of other HTML entities. Here is a selection of common ones:

>	>
<	<
&	&
"	"
'	'
€	€
©	©
** **	non-breaking space

Images

To include an image in your HTML, use the **<img>** tag:

```
<img src="path/to/picture.jpg" />
```

The path to the picture is the path relative to your HTML file.

To define height and width, use the **height** and **width** attributes:

```
<img src="path/to/picture.jpg" height="42" width="42" />
```

You can also define a value for the **alt** attribute. This is a text that is displayed if the image is broken. It is also useful for screen readers (if blind people read your website), and for google image search to list your image.

```
<img src="path/to/picture.jpg" alt="funny cat picture" />
```

Preparing pictures for your website

You can use images of various formats, e.g.:

- jpg
- png
- gif
- webp
- svg

If you prepare your picture, make sure it is not too big, because that means it loads much slower. If you want to display, for example, a profile picture that should be 500x500 pixels wide on your website, the image itself shouldn't be much larger than that.

Most retina displays use **double the amount of pixels**, so it is a good rule of thumb to make your images double the size of how you want them to include in your HTML document.

In the example of the 500x500 pixels profile picture, you need to prepare the image so that it is 1000x1000 pixels wide.

SVGs

SVG images are a little different. They are not a pixel-by-pixel image, like JPG and PNG, but they describe vectors and attributes of the image – they are a **vector image**.

Due to this characteristic, they are scalable in size. That means that they will always be focused, no matter how large you make them.

Lists

You can use HTML to make lists. There are two kinds of lists: unordered lists (with bullet points), and ordered lists (numbered).

Unordered List

```html
<ul>
   <li>HTML</li>
   <li>CSS</li>
   <li>JavaScript</li>
</ul>
```

- HTML
- CSS
- JavaScript

Ordered List

```html
<ol>
   <li>HTML</li>
   <li>CSS</li>
   <li>JavaScript</li>
</ol>
```

1. HTML
2. CSS
3. JavaScript

Different kind of bullet symbols

You can use different kinds of bullet point symbols with the attribute `style="list-style-type:<symbol>;"`.

```html
<ul style="list-style-type:circle;">
   <li>HTML</li>
   <li>CSS</li>
   <li>JavaScript</li>
</ul>
```

- HTML
- CSS
- JavaScript

This actually uses CSS, this means we are adding a style to the list.

These symbols exist:

disc	(default)	●	item
circle		○	item
square		■	item
none			item

Different kinds of numbering symbols

Similarly, you can use different kinds of symbols for your numbered list by adding a **type** attribute:

```
<ol type="A">
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ol>
```

A. HTML
B. CSS
C. JavaScript

These symbols exist:

type="1"	(default)	1. First 2. Second 3. Third
type="A"		A. First B. Second C. Third
type="a"		a. First b. Second c. Third
type="I"		I. First II. Second III. Third
type="i"		i. First ii. Second iii. Third

Tables

Tables are an important feature in HTML. You can create a table like this:

```html
<table>
  <tr>
    <th>Course</th>
    <th>Hours</th>
    <th>Teacher</th>
  </tr>
  <tr>
    <td>HTML and CSS</td>
    <td>27h</td>
    <td>Alex Löhn</td>
  </tr>
  <tr>
    <td>Learn to Code</td>
    <td>24h</td>
    <td>Helder Pereira</td>
  </tr>
  <tr>
    <td>JavaScript for Web</td>
    <td>24h</td>
    <td>Teresa Holfeld</td>
  </tr>
</table>
```

<table></table> Root element that defines the table.

<tr></tr>   Defines a row.

<th></th>   Defines a header cell.

<td></td>   Defines a cell.

Forms

If you want to create a website where the user should type in some data, e.g. login credentials, you need forms.

```html
<form action="/sign-in.php">
  First name:<br>
  <input type="text" name="firstname" value="Max"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mustermann"><br><br>
  <input type="submit" value="Submit">
</form>
```

<form></form>	Root element that defines the form.
action="..."	The program that is executed when the submit button is pressed.
<input ...>	An input field.
type="text"	Attribute for input, creates a text input field.
type="submit"	Attribute for input, creates a button.
name="..."	A name or id for an input field.
value="..."	The default value that is displayed in a field. In case of the submit button, it is the button label.

Form Input Groups

Form input groups are groups of input fields, e.g. a group of radio buttons, multiple-choice checkboxes, or a dropdown selection.

Radio Buttons

Radio buttons are round bullet buttons where you can select one of many options.

```html
<form>
  <input type="radio" name="role" value="student" checked> Student<br>
  <input type="radio" name="role" value="teacher"> Teacher<br>
  <input type="radio" name="role" value="administrative"> Administrative
</form>
```

type="radio"	Attribute for input, creates a radio button (for selecting one of many choices).
name="..."	Name for the input field group. All radio buttons that belong to the same group should have the same name.
value="..."	An ID for the item. This value is sent to the program where the data of the form is sent to when the user presses the submit button.

Checkboxes

Checkboxes are form inputs that the user can select or deselect independently from each other. They are useful for multiple-choice options.

```
<form>
  <input type="checkbox" name="course" value="html"> HTML & CSS<br>
  <input type="checkbox" name="course" value="l2c"> Learn to Code<br>
  <input type="checkbox" name="course" value="js4w"> JavaScript for Web<br>
</form>
```

type="checkbox"	Attribute for input, creates a checkbox.
name="..."	Name for the input field group. All checkboxes that belong to the same group should have the same name.
value="..."	An ID for the item. This value is sent to the program where the data of the form is sent to when the user presses the submit button.

Drop-down Selection

Many sign-up forms use a drop-down selection for the salutation, to know if they should address someone as Mr. or Mrs.

```
<form>
  <select name="salutation">
    <option value="mr">Mr.</option>
    <option value="mrs">Mrs.</option>
  </select>
</form>
```

`<select></select>`	Creates the selection group for the drop-down menu.
`<option></option>`	Defines one element in the drop-down.
`value="..."`	An ID for the item. This value is sent to the program where the data of the form is sent to when the user presses the submit button.

Other Input Elements

Textarea

A text area is a large input field for free text. You use it whenever the user needs to write more text that exceeds one line, e.g. a blog post.

```
<form>
  <textarea name="post" rows="10" cols="80">
    Your blog post
  </textarea>
</form>
```

Button

A button is an alternative to the `<input type="submit">` field.

```
<form>
  ...
  <button type="reset">Reset fields</button>
</form>
```

The main difference to the `<input type="submit">` field is that you have more type options:

`<button type="submit">`	Submits the form. Submit is the default option for a <button>, so you can leave that attribute off.
`<button type="reset">`	Resets all form fields.
`<button type="button">`	A click button for a different action that you define yourself (e.g. with JavaScript).

🤓 **Want to know more?** You can learn about more input types here: https://www.w3schools.com/html/html_form_input_types.asp

Action

Every form should define a form action with the **action="..."** attribute. The action attribute specifies where to send the form-data when a form is submitted.

```html
<form action="/sign-in.php">
```

In this example, the data that the user typed into the form fields is sent to the sign-in.php program. It is assumed that the file containing the PHP code exists and is able to handle the data.

In the course **JavaScript for Web** you will learn to build your own programs in JavaScript, which can be used as receiver of form input.

HTML5 Forms (optional knowledge)

HTML5 is a new version of HTML. Here, a lot of helpful tags and attributes were added to add more functionality to forms, for example:

- Email input type
- Phone number input type
- Date and time input type
- Max length for text input
- Placeholder
- Label
- Built-in form validation
- Email validation
- Required input fields

You can learn more here:

- https://www.html5rocks.com/en/tutorials/forms/html5forms/
- https://www.w3schools.com/html/html_form_attributes.asp

Block and Inline Elements

Remember that **<div>** tags were breaking the text, while **<small>** tags were not?

There are two ways in which HTML elements behave in the flow on a line.
It is called the **display type**.

The two options are: **block** and **inline**.

A **block-level element** always starts on a new line.
Some block level elements are:

```
<article>      <blockquote>        <div>         <fieldset>      <form>        <h1>-<h6>

<header>        <footer>       <ul> <ol> <li>      <p>          <pre>          <table>
```

An *inline element* does not start on a new line.
Some frequently used inline elements are:

```
<a>            <img>          <button>       <small>      <b> <i> <u>      <label>

<span>         <strong>       <textarea>     <select>
```

🤓 **Optional knowledge: `<img>`** elements are actually not *inline*, but *inline-block*. This is a combination of both. The element behaves like an inline element, but it is also a block, meaning that you can set both height and width.

Comparison of inline, inline-block and block:

Lorem ipsum inline inline dolor sit amet, inline-block inline-block consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad block block minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis

From: https://www.w3schools.com/css/tryit.asp?filename=trycss_inline-block_span1

CSS

CSS Syntax

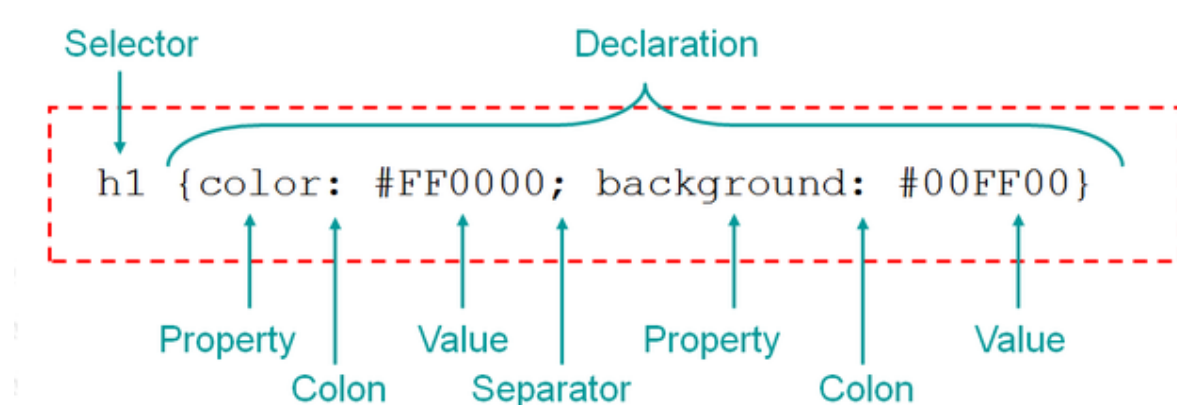**CSS** stands for **Cascading Style Sheets** and describes how HTML elements are to be displayed in the browser.

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**: { }

Comments

Just as in HTML **comments** are used to explain the code and may help when you edit the source code at a later date and are ignored by browsers but the syntax is a bit different. A CSS comment starts with **/*** and ends with ***/**. Comments can span multiple lines:

```css
.logo {
  height: 32px;
  width: 32px;
/*  margin: 20px;
  Padding: 5px; */
  cursor: pointer;
}
```

Colors

Colors in CSS are defined like this:

```css
.logo {
  color: red; /* text color */
  background-color: #44F25A; /* background color */
}
```

Color values can be specified in four ways: as **_color name_**, **_hex color_**, **_rgb color_**, or **_hsl color_**.

Color Names

CSS has some predefined names for colors, e.g. **red**, **blue**, **orange** or **tomato**.

📚 You can look up the complete list here:
https://www.w3schools.com/colors/colors_names.asp

Hex Colors

Hex colors follow the pattern: **#rrggbb**   (red, green, blue)

The values are hexadecimal values between **00** and **FF**.

📚 Read more: https://www.w3schools.com/colors/colors_hexadecimal.asp

RGB Colors

Here, colors are defined as: **rgb(r, g, b)**   (red, green, blue)

The values are numbers between **0** and **255**.

📚 Read more: https://www.w3schools.com/colors/colors_rgb.asp

HSL Colors

This is a different color space where colors are defined by hue, saturation and lightness.

They follow the pattern: **hsl(h, s%, l%)**   (hue, saturation, lightness)

Hue is a number between **0** and **360** (degrees on the color wheel).

Saturation and lightness are a percentage (between 0 and 100).

📚 Read more: https://www.w3schools.com/colors/colors_hsl.asp

> 💡 **Note:** RGB and HSL colors are not used very often, but you will see hex colors all the time. It is a good idea to study hex codes and know the most common ones.

CSS Selectors

In CSS, a **selector** is the element before the opening bracket **{** that defines to which element(s) the style will be applied.

Tags

Tag selectors look like this:

```css
/* Applies to all <a> elements. */
a {
  color: blue;
}


/* Applies to all <p> elements. */
p {
  font-size: 12px;
}
```

Classes

To select a class, you use the dot (**.**):

```css
/* Applies to all HTML elements that specify class="blue". */
.blue {
  color: blue;
}


/* All HTML elements that specify class="small-text". */
.small-text {
  font-size: 9px;
}
```

IDs

To select HTML tags with a certain id, use the hash (#):

```css
/* Applies only to the element with id="application-submit". */
#application-submit {
  color: blue;
  font-weight: bold;
  text-transform: uppercase;
}
```

All Elements

The selector for all elements is the asterisk (*):

```css
/* Applies to all HTML elements. */
* {
  font-family: Arial, Helvetica, sans-serif;
}
```

Combining Selectors

Instead of defining the same style for multiple selectors, you can separate them by comma (,).

```css
/* Applies to HTML elements of both class="profile" and
class="teacher". */
.profile, .teacher {
  font-weight: bold;
  font-family: Arial, Helvetica, sans-serif;
  margin: 12px;
}
```

💡 **Learn more about selectors and combining them:**
https://www.w3schools.com/CSSref/css_selectors.asp

The BEM Naming Convention (optional knowledge)

Sometimes it is difficult to give good class names. One strategy is the BEM Naming Convention that aims to bring consistency into class names.

BEM - Block Element Modifier

The pattern works like this:

```
block__element--modifier
```

The **block** is the larger section that you want to display, e.g. a form.

The **element** is an element of it, e.g. a button.

The **modifier** specifies a variant of this element, e.g. a highlighted button. It is optional.

In the HTML, this would look like this:

```
<button class="form__button--highlighted">Submit</button>
```

And in the CSS, this would look like:

```
.form__button--highlighted {
  ...
}
```

> 💡 **Take a deeper dive:** http://getbem.com/naming/

Units

For specifying a length, CSS provides a selection of units you can choose from.

You usually use them for width, height, margin, padding or font-size.

Length values are written as a number followed by the unit:

```css
.wrapper {
  font-size: 1em;
  width: 200px;
}
```

There are two types of units: ***absolute*** and ***relative***.

Absolute units:

- **px**   pixels
- **pt**   points
- **pc**   picas (1pc = 12 pt)
- **cm**   centimeters
- **mm**   millimeters
- **in**   inches

In web development, you usually use **px** for height and width, and **pt** for font-size.

> 💡 **Note:** Pixel values (**px**) are heavily used, but their actual size on the screen depends on the screen of the device. It is good practice to test your website on multiple devices and to make the website responsive (see section "Responsiveness").

Relative units:

- **em**   Relative to the element's font size (2em = 2 times the font size)
- **rem**   Relative to the font size of the root element
- **vw**   Relative to 1% of the viewport width
- **vh**   Relative to 1% of the height of the viewport
- **vmin** Relative to 1% of viewport's smaller dimension
- **vmax** Relative to 1% of viewport's larger dimension
- **%**   Relative to the parent element
- **ex**   Relative to the x-height of the element's font (rarely used)
- **ch**   Relative to width of the **0** character (rarely used)

In web development, you will use em, rem, vw and % a lot. The other values you will rarely encounter.

The ***viewport*** is the size of the window in the browser.

> 💡 **Tip:** It is good practice to use **em** and **rem** a lot. It helps create good responsive websites. **vw** is also a useful unit, but has to be tested especially on small devices.

Including CSS

There are three ways of inserting a style sheet:

- Inline CSS
- Internal CSS
- External CSS

Inline CSS

CSS is defined in the **style** attribute of an HTML tag.

```
<p style="color: grey; font-size: 12px;">Lorem ipsum</p>
```

Internal CSS

The CSS is defined in the **<head>** part of the HTML document.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML and CSS Course</title>
    <style>
      body {
        background-color: gold;
      }
      h1 {
        color: #3c3c3c;
      }
    </style>
  </head>
  <body>
    <h1>HTML and CSS Course</h1>
    <p>In this course, we are going to learn ...</p>
    ...
  </body>
</html>
```

External CSS

In external CSS, the styles are defined in its own file and then included via a **`<link>`** element in the **`<head>`** of the HTML file.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML and CSS Course</title>
    <link rel="stylesheet" type="text/css" href="courses.css">
  </head>
  <body>
    <h1>HTML and CSS Course</h1>
    <p>In this course, we are going to learn ...</p>
    ...
  </body>
</html>
```

Specificity Rules

In CSS, some styles can overwrite others. For this, CSS applies its specificity rules.

Rule 1: From Top to Bottom

CSS applies its rules from top to bottom. This means that styles that appear later in the style sheet overwrite the ones that appeared earlier.

Rule 2: IDs over Classes over Tags over Universal

If multiple contradicting styles apply to the same element, the one with the most importance is choses. The importance hierarchy looks like this:

```
id > classes > tag > universal selector (*)
```

Rule 3: Inline CSS over Internal CSS over External CSS

Similarly, if there are styles defined in multiple ways, the more important ones overwrite the less important ones. The importance hierarchy here looks like this:

```
inline CSS > internal CSS > external CSS
```

Rule 4: !important trumps everything

If you need to hack your CSS so that your style overwrites the others regardless of the importance hierarchy, you can use **!important**.

```
.highlighted {
  background-color: yellow !important;
}
```
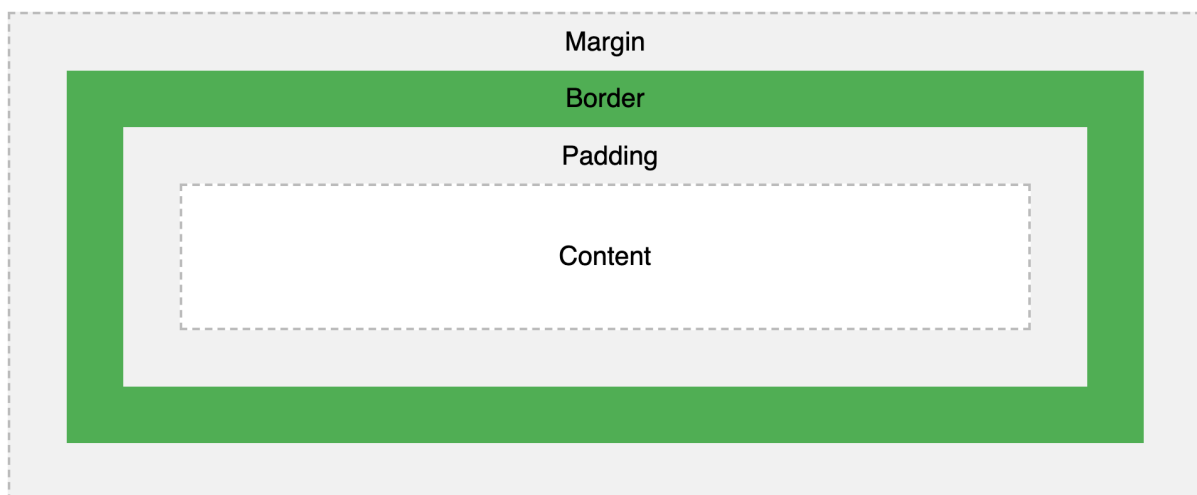
> ⚠️ **Attention: !important** overwrites everything but is to be used wisely and only in exceptional cases.

The 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 ***margin***, ***border***, ***padding***, and the actual ***content***.

The image below illustrates the box model:

(https://www.w3schools.com/css/css_boxmodel.asp)

Explanation of the different parts:

- *Margin* - Clears an area outside the border. The margin is transparent.
- *Border* - A border that goes around the padding and content.
- *Padding* - Clears an area around the content. The padding is transparent.
- *Content* - The content of the box, where text and images appear.

The box model allows us to add a border around elements, and to define space between elements.

Responsiveness

A website is *responsive* if it is suitable for all screen sizes, even for small mobile screens.

For making a website responsive, we can use *media queries*.

Media queries are styles wrapped in a `@media` block that defines for which screen size this style applies.

```
// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) { ... }

// Medium devices (tablets, 768px and up)
@media (min-width: 768px) { ... }

// Large devices (desktops, 992px and up)
@media (min-width: 992px) { ... }

// Extra large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }
```

The style without the media query is the smallest and the default style. The style inside the media query is the one for the larger screens and will overwrite the default style if the screen where the website is displayed is larger than specified.

CSS Flexbox (optional knowledge)

Flexbox is a set of CSS properties that help to align items in columns on a website.

In flexbox, the outer box is called the *container*.

The inner boxes, the columns, if you will, are called the *items*.

To make it work, you put this style on the container element:

```
.container {
  display: flex;
}
```

The flex direction specifies if the items will be aligned in a row or as a column (one below the other). This is specified on the container element as well:

```
.container {
  display: flex;
  flex-direction: row;
}
```

Possible values are: **row**, **column**, **row-reverse**, **column-reverse**.

> 📚 **Learn more:** You can specify much more for the container and items. A good cheat sheet with visual aids is this:
> https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Bootstrap

Bootstrap is a library for CSS, HTML and JavaScript. It provides a lot of functionality, and predefined CSS styles that make a website look nice.

https://getbootstrap.com/

Glossary

`HTML`	HyperText Markup Language
`CSS`	Cascading Style Sheets
`RTFM`	Read The Frigging Manual 😉
`IDE`	Integrated Development Environment, fancy name for a code editor like Visual Studio Code
`Developer Tools`	Feature of the Chrome browser to look at HTML, CSS and other things.
`syntax`	The structure of a programming language or markup language. Tells you how to write in that language, e.g. in HTML you put tags in < and >.
`tag`	In HTML, a tag is an HTML element, e.g. <p>. It is the one in these brackets: < >.
`paragraph`	A paragraph is a text block.
`hyperlinks`	A link that leads from one website to another.
`metadata`	Data or information about the document. Any data that is not the content itself. In HTML, useful metadata are the author or a description.
`viewport`	A special metadata tag that helps to display content for mobile screens.
`entity`	Pieces of text that are used to display reserved characters (like "**<**") or characters you don't find on your keyboard (like "æ" or "**Ω**"). The more frequently used have easy to remember names like **♠** for ♠ and **€** for **€**. They always start with an ampersand and end with a semicolon.
`Lorem ipsum ...`	Dummy text.

SVG	An image format where the image is not saved in pixels, but with vectors. Vector images don't have a resolution and can be enhanced without limits.
Unordered list	A list with bullet points.
Ordered list	A list with numbers.
input	A form field
radio button	A group of items with round buttons where you can select only one.
block-level element	An HTML element that is displayed as a block, meaning it will break surrounding text.
inline element	An HTML element that is displayed inline, meaning it will not break surrounding text.
selector	In CSS: the item that specifies the tag, class or id to which the style should be applied. In this example, the .blue class:

```
.blue {
  color: blue;
}
```

hex color	Color value in the format: `#rrggbb`
rgb color	Color value in the format: `rgb(r, g, b)`
hsl color	Color value in the format: `hsl(hue, saturation%, lightness%)`
BEM	Block - Element - Modifier A naming convention for CSS classes.
specificity	The order of importance in which CSS styles are processed and overwrite each other.
CSS box model	The box model is a model that shows where margin, border, padding and content of an HTML element are located.

`responsive`	A website is responsive if it works well on both large and small screens.

`media query`	A media query is a special CSS element that defines for which screen size a style should be applied. Of the format:
	`@media (min-width: 576px) {`
	`   ...`
	`}`

`Flexbox`	A set of CSS properties for arranging items in a container in a certain way. Useful for implementing responsive websites.
`Bootstrap`	A set of libraries for HTML, CSS and JavaScript. https://getbootstrap.com/

Useful Links

HTML

W3Schools: https://www.w3schools.com/html/
Many of this handout's examples stem from here. Thank you, w3schools! 🙏

MDN: https://developer.mozilla.org/de/docs/Web/HTML

SelfHTML: https://www.selfhtml.org/

Boilerplates: https://amp.dev/boilerplate/

Overview of the most used HTML elements with short explanations:
https://www.advancedwebranking.com/html/

Placeholder Bilder: http://placekitten.com/, https://www.fillmurray.com

CSS

W3Schools: https://www.w3schools.com/css/

To exercise and test your CSS Knowledge: https://flukeout.github.io/

BEM Naming Convention: http://getbem.com/naming/

Flexbox: http://flexbox.help/

Flexbox Generator: https://loading.io/flexbox/

Learn flexbox by playing: http://flexboxfroggy.com/

The complete guide to flexbox: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Bootstrap: https://getbootstrap.com

Browser support: https://caniuse.com/