CSS – Cascading Style Sheets

History, Concepts and Nutshell Examples

Author: Oliver Kerschbaumsteiner, h11801721

Seminar Paper

Submitted: December 16th, 2021

Vienna University of Economics and Business

Seminar Business Information Systems [0095]

Advisor: Univ. Prof. Mag. Dr. Rony Flatscher

Declaration of Authenticity

I assure:

- to have individually written, to not have used any other sources or tools than referenced and to not have used any other unauthorized tools for the writing of this seminar paper.

- to never have submitted this seminar paper topic to an advisor neither in this, nor in any foreign country.

- that this seminar paper matches the seminar paper reviewed by the advisor.

[illegible]

Date: December 16th, 2021 Signature

(Oliver Kerschbaumsteiner)

Table of Contents

1 Introduction 5

2 CSS – History and Development 5

2.1 Web Design Before Cascading Style Sheets 6

2.2 History of Cascading Style Sheets 7

2.3 Development Process of CSS 8

2.4 CSS Levels 9

3 Operating Principle of CSS 10

3.1 Concept 11

3.2 Box Model 11

3.3 Syntax 12

3.4 Selectors 13

3.4.1 Simple Selectors 13

3.4.2 Combinator Selectors 15

3.4.3 Pseudo-Class Selectors 16

3.4.4 Pseudo-Elements Selectors 16

3.4.5 Attribute Selectors 16

3.5 Ways to Insert CSS 18

3.5.1 Insert into HTML 18

3.5.2 Insert into XML 20

4 Example of Use 21

4.1 Using CSS Frameworks 21

4.1.1 Components and Advantages 21

4.1.2 Popular Frameworks 22

4.2 Nutshell Examples 23

4.2.1 General 24

4.2.2 Navigation Bar 24

4.2.3 Picture ... 27
4.2.4 Flexbox ... 28
4.2.5 Grid ... 31
4.2.6 Footer ... 33
5 Summary and Outlook ... 33
References ... 35
List of Figures ... 37
Appendix ... 39
HTML-Script ... 39
CSS-Code ... 44

1 Introduction

CSS, short for Cascading Style Sheets, is a style sheet language and besides HTML and JavaScript, one of the core languages of the world wide web. Therefore, it mainly is used in combination with HTML, but can also be applied to other XML-based documents. The purpose of CSS is the separation of the presentation specifications and the content itself.

The Norwegian computer scientist Håkon Wium Lie proposed the first version of CSS and over time the style sheet language became the standard of the World Wide Web. The development and maintenance are carried out by a working group of the World Wide Web Consortium (W3C).

This seminar paper is structured as follows. First, the history and development of CSS are examined. Furthermore, the basic concept, the syntax, and different forms of selectors are explained in detail. Also, the insertion of CSS files into HTML and XML documents is shown. Afterward, the usefulness of CSS frameworks is discussed, and some framework examples are given. Finally, a HTML document is styled with the help of a CSS file in a case study.

2 CSS – History and Development

The first chapter of this term paper is about the history and development of Cascading Style Sheets. First, there is a short historical description of the time before the introduction of CSS. After that, the history, and the technical development of CSS since its introduction is examined in more detail.

2.1 Web Design Before Cascading Style Sheets

Cascading Style Sheets is a style sheet language and is now considered as the standard in the field of website design. But there was a time before CSS and HTML existed. Before HTML was introduced, there already was SGML (Standard Generalized Markup Language), which was the base for HTML [JaLe97]. Before style sheet languages were considered, the layout of web pages was determined by the use of tables. Disadvantages, such as the high bandwidth requirements, led to the development of solutions for tableless web design. The first approach was called FOSI. FOSI documents were written in SGML itself. This was a logical approach because at this point many web developers were already familiar with HTML, which is a subset of SGML. Nevertheless, it was only considered as an interemistic solution. The actual standard should have become DSSSL (Document Style Semantics and Specification Language). This language is based on a subset of the functional programming language Scheme. Therefore, DSSSL is not only a style sheet language but more a programming language. However, there were two reasons why DSSSL was not able to establish itself as the standard in the long term. On the one hand, like other Scheme-based languages, it had too much parenthesis. On the other hand, the released specification included 210 separate designable properties, which may have intimidated web developers [Bloo17].

In 1990, Tim Berners-Lee created the first web browser which at first was called WorldWideWeb and later was renamed Nexus. He later launched the first website in August 1991, and also published a document that contained the first HTML tags in October 1991. The World Wide Web was created as a platform for publishing documents electronically. The problem was that there was no way to style the layout and the overall appearance of these HTML documents. Although the separation of the document’s structure and its presentation was a goal of HTML since its introduction, there was no standardized concept for styling websites. It was up to the different browsers, how they displayed the web pages. Therefore, every browser had its styling language, all of which were very similar. In 1993, the browser NCSA Mosaic was launched. This browser brought the web to the masses. In terms of styling, however, it offered fewer options than its predecessors. Web developers, on the other hand, demanded more control over how the web pages they wrote were presented. The Norwegian web pioneer Håkon Wium Lie recognized the need for a style sheet language for the web. In 1994, he published the *Cascading HTML Style Sheets* proposal, which should be the basic concept of CSS [Bos16].

2.2 History of Cascading Style Sheets

After the publishing of *Cascading HTML Style Sheets* by Håkon Wium Lie, Bert Bos responded to this proposal. Bos was working on a highly customizable browser at the time and decided to team up with Lie to work together on the style sheet language. With Bos' contribution, the style sheet language changed so that it was no longer only designed for HTML but was also compatible with other common markup languages. Therefore, Cascading HTML Style Sheets was renamed Cascading Style Sheets [Bos16].

In November 1994, the first proposal of CSS was presented at the Web Conference in Chicago. The proposal faced some opposition as many experts felt it was too simple to meet the requirements [Bos16].

In addition to CSS, there were about 10 other style sheet languages that were proposed at this time. DSSSL for example was one of them. The feature that distinguished CSS from all the other proposals was the approach, that the user, the author, and the technical possibilities of the display devices and the browser should all influence the design [Bos16].

The balance between the influence of users and authors on the presentation of web pages triggered a fundamental discussion at the next Web Conference in 1995. Web developers felt they should have the power to decide how the document is ultimately presented. Lie and Bos argued that the user should have the last word, as he is the one who has to process the presented impressions [Bos16].

Also in 1995, the World Wide Web Consortium (W3C), founded the year before, developed into an operative organization. Many companies joined the consortium and its influence increased. They organized workshops on a variety of topics, including one on style sheets. One of the participants of this workshop was Thomas Reardon. He was one of the developers of Microsoft’s Internet Explorer. He ensured the support of CSS in the upcoming versions of the browser. This was a milestone for style sheets. W3C established a working group called *HTML Editorial Review Board*. One of the goals of this working group was to make CSS a W3C Recommendation, which is equivalent to an area-wide standard. To do this, they had to convince Netscape, the other major commercial browser provider at the time, to include CSS into its browsers. Otherwise, there would have emerged several browsers which support different specifications instead of one standard [Bos16].

In December 1996, CSS1 became a W3C Recommendation. In 1997, W3C introduced a separate working group for CSS to develop features that were not yet implemented in the first version. In 1998, CSS2 became a W3C Recommendation. The working group, which consisted of 15 members in 1999, had 115 members in 2016. Their task is to develop new modules for CSS and also fix errors [Bos16].

2.3 Development Process of CSS

Since the first publication of a CSS specification, the style sheet language has been further developed. This task has since been taken over by the World Wide Web Consortium. The W3C is the main institution for standardization regarding the World Wide Web. To understand the development steps of CSS, it is necessary to know the process new technology has to go through. To become a W3C Recommendation, a recommendation-track document must go through three stages. These are intended to ensure the stability of the technology [AtEt20].

The first stage is called Working Draft (WD). It is the design phase of a new specification. The first official working draft is also called First Public Working Draft (FPWD). For the CSS working group, the publication of the FPWD would mean that the whole group has agreed to work on a new module. The end of this stage is sometimes called the Last Call Working Draft (LCWD) phase. At this point, the working group has resolved all known errors and cannot make further progress without external feedback through testing and implementations. It sets a deadline for any pending issues. The working group must consequently follow up on and address the feedback received. To demonstrate wide review and acceptance, a comment tracking document, the Disposition of Comments (DoC), is submitted along with an updated draft for the Director's approval [AtEt20].

The second stage is called Candidate Recommendation (CR). This phase is all about testing and implementing the specification. Many issues of the specification are revealed in this phase. Therefore, the Candidate Recommendation often changes at this stage. To exit the CR, the demonstration of two correct and independent implementations of each feature is required. The working group generates a test series and implementation reports. To move to the next stage, the W3C Advisory Committee must confirm the transition. This phase is called Proposed Recommendation (PR) [AtEt20].

The last stage is called Recommendation (REC). At this point, the W3C specification is in a completed state. The working group now only maintains an errata document. It also publishes updates to the document where errata are reintegrated into the specification [AtEt20].

2.4 CSS Levels

Different from other programming and script languages, CSS is not developed in versions but levels. Each level builds on the previous. Therefore, each higher level is a superset of any lower level. The behavior allowed for a higher-level feature is a subset of the allowed behavior in the lower levels. This means that if a higher CSS level is supported by a user agent, this also applies to the lower levels of this module [AtEt20].

CSS Level 1

The first level of CSS includes all the features of the CSS1 specification but uses the syntax of the CSS 2.1 specification [AtEt20].

CSS Level 2

When CSS Level 2 was introduced, the working group had not yet defined the Candidate Recommendation stage. Therefore, CSS2 became a W3C Recommendation without going through this stage. This led to more and more errors appearing in the specification over time. Instead of further expanding the already very comprehensive errata list, the working group decided to publish CSS Level 2 Revision 1 (CSS2.1). If a conflict arises between the two specifications, CSS 2.1 is considered the final definition [AtEt20].

After CSS2.1 became a CR, the CSS2 Recommendation became obsolete, although they were not on the same stability level. Some of the functions in CSS2 were removed from the CSS2.1 specification and were subsequently reclassified as Candidate Recommendations. It should be noted that many of these features have been incorporated into a working draft for CSS Level 3 [AtEt20].

CSS Level 3

The third level of CSS uses the CSS2.1 specification as a basis. The further developed modules add functionality respectively replace parts of the CSS2.1 specification. Special care is taken to ensure that the new CSS modules do not contradict the old specifications but complement them [AtEt20].

After reaching CSS Level 3, the modules are leveled independently. This means that for some modules Level 4 can be completed before Level 3 has been completed for others. Modules start at Level 1, if they have no equivalent in CSS Level 2, or Level 3, f there is a counterpart [AtEt20].

CSS Level 4 and beyond

CSS considered as a language has no level 4. However, individual modules can reach level 4 and higher [AtEt20].

3 Operating Principle of CSS

In this chapter, the functional principle of CSS is discussed. First, the basic concept is explained. Then the box-model of CSS and the syntax are introduced. In addition, the most important selectors and the possibilities of implementation are presented.

3.1 Concept

As mentioned before, CSS is the standard style sheet language for the World Wide Web. The purpose of Cascading Style Sheets is to define the presentation of websites. The presentation includes fonts, colors, layout, and the responsiveness on different screens. However, there are many more possibilities offered by the language [W3hc21].

One of the key features of CSS was also already mentioned before in chapter 2.2: The property that the language applies to any XML-based markup language. However, the use of CSS in conjunction with XML formats that have a Document Type Definition (DTD) offers the most functionality. Independent of the format, the elements can be described either in a separate style area of the document or in a separate document with values and attributes [ACPa21].

Regarding web pages, it makes sense to separate the HTML and the CSS documents. This makes it easier to maintain them. In addition, the style sheet can be applied to multiple websites and may be used in other environments, where the same design is needed. This division is called the separation of structure from the presentation [W3hc21].

3.2 Box Model

The CSS box model is one of the essential concepts of this style sheet language when it comes to design and layout. Every HTML or XML element that should be styled with CSS, gets encased in a box. This box consists of four parts: the content, the padding, the border, and the margin. (Figure 1) The content is the part of the box, where text and images to be displayed are located. The padding is a transparent area, that allows providing spacing around the content. The border surrounds both the content and the padding. The margin is also a transparent area, that provides spacing around the border [W3bm21].

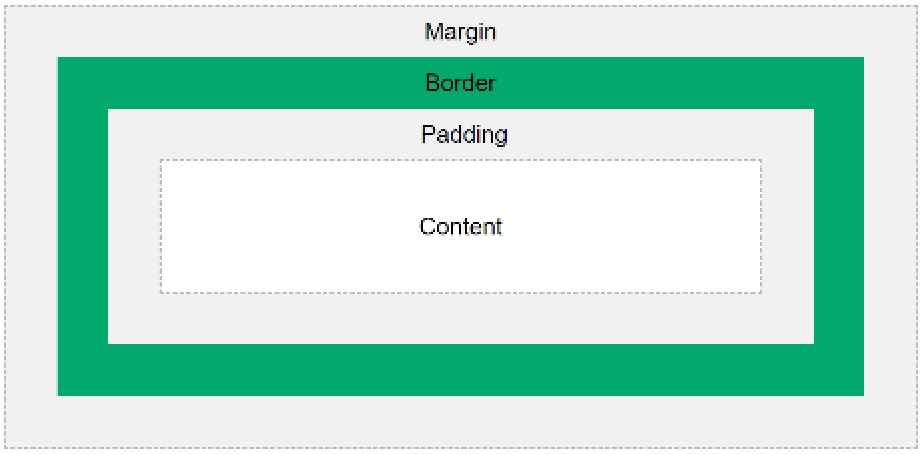


Figure 1: Box Model [W3bm21]

Especially when it comes to defining the height and width of specific elements, it is necessary to understand the CSS box model. If the height and width of an element are specified, this declaration only applies to the content area. To calculate the real size of the element, the padding, the border, and the margin must be added [W3bm21].

3.3 Syntax

If you look at the syntax of CSS, you will notice that each statement consists of two parts. The first part is the selector. A selector defines which HTML element should be styled. The second part is the declaration, which is defined by curly brackets. A declaration consists of one property and one value. The property and the value are separated by a colon. A declaration block may consist of more than one declaration. Multiple declarations in a block are separated by semicolons. In the example in figure 2, the selector is the h1-tag. The declaration block consists of two declarations. One defines the font color as blue, the other the font size as 12 pixels [W3sy21].

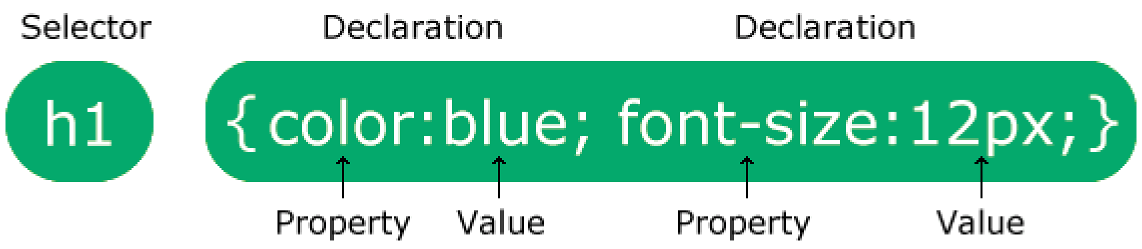


Figure 2: CSS-Syntax [W3sy21]

3.4 Selectors

As mentioned before, the selector “selects” the elements that are meant to be styled. In general, a distinction is made between five different categories of selectors. These categories are:

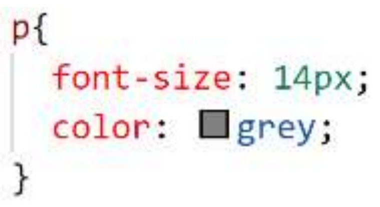
- Simple selectors
- Combinator selectors
- Pseudo-class selectors
- Pseudo-elements selectors
- Attribute selectors

Every selector category has its purpose [W3se21]. Some selectors are examined in more detail in this chapter.

3.4.1 Simple Selectors

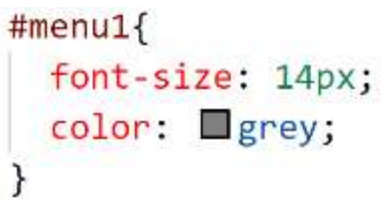
The first category is the simple selectors. There are three basic ways to select the HTML elements. [W3se21]

With the element selector, all HTML elements with the same tag are addressed [W3se21]. For example, all elements with the <p> tag are styled according to the same properties and values (Figure 3).



```
p{
  font-size: 14px;
  color: ■grey;
}
```

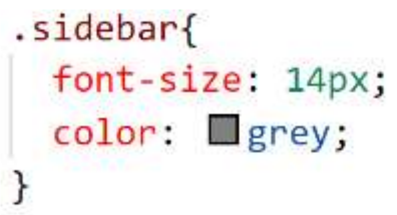
Figure 3: Simple Selector



```
#menu1{
  font-size: 14px;
  color: ■grey;
}
```

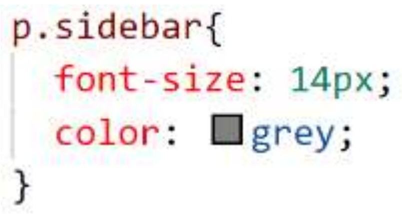
Figure 4: ID Selector

The id selector and the class selector are following the same principle as the element selector. The difference is that the elements are selected by the id attribute (Figure 4) respectively by the class of the HTML elements (Figure 5) [W3se21].



```css
.sidebar{
  font-size: 14px;
  color: grey;
}
```

Figure 5: Class Selector

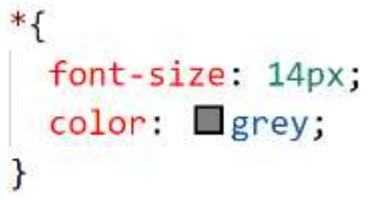


```css
p.sidebar{
  font-size: 14px;
  color: grey;
}
```

Figure 6: Simple and Class Selector combined

It is also possible to combine the element selector and the class selector to select only certain HTML elements with the said class (Figure 6) [W3se21].

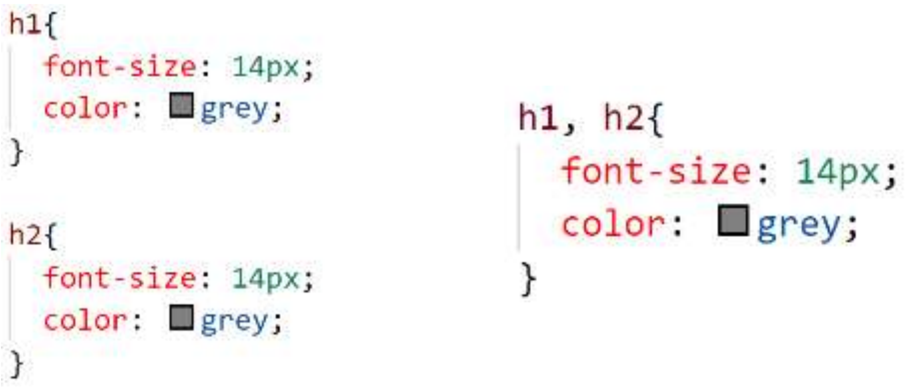
The universal selector is used when all elements of the HTML document are to be given the same properties and values (Figure 7) [W3se21].



```css
*{
  font-size: 14px;
  color: grey;
}
```

Figure 7: Universal Selector

If only certain HTML elements are to be assigned the same style attributes, one way would be to define the same properties and values for each element individually like in the example on the left. To minimize the amount of code, it would be better to use the group selector, like in the example on the right (Figure 8) [W3se21].



```css
h1{
  font-size: 14px;
  color: grey;
}

h2{
  font-size: 14px;
  color: grey;
}
```

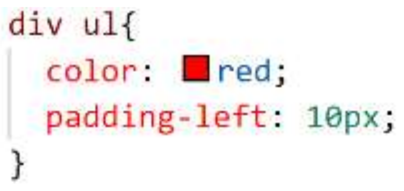
```css
h1, h2{
  font-size: 14px;
  color: grey;
}
```

Figure 8: Group Selector

3.4.2 Combinator Selectors

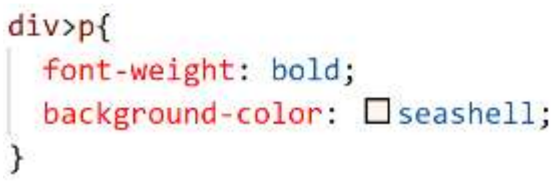
A CSS selector can consist of more than one simple selector by using a combinator. A combinator explains the relationship between the selectors. There are four different types of combinators [W3co21].

The first one is the descendant selector. It selects all elements that are descendants of a certain element. The simple selectors are combined with a space between them [W3co21]. In the example in figure 9, all <ul> elements inside a <div> element are selected



```
div ul{
  color: red;
  padding-left: 10px;
}
```

Figure 9: Descendant Selector

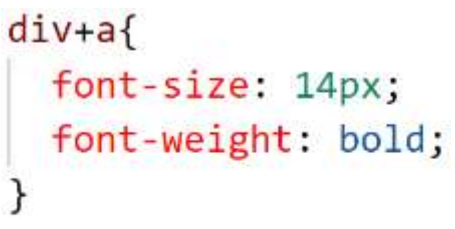


```
div>p{
  font-weight: bold;
  background-color: seashell;
}
```

Figure 10: Child Selector

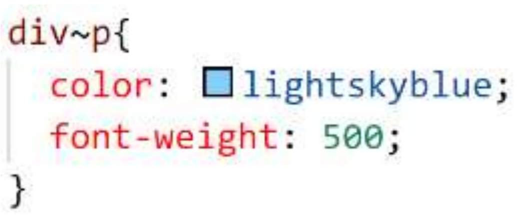
The second CSS selector that uses a combinator is the child selector. It uses a greater-than sign between the selectors, to match only the elements that are children of a certain element [W3co21]. The code in figure 10 styles all <p> elements that are children of a <div> element.

Another combinator selector is the adjacent sibling selector. Adjacent elements are elements that immediately follow another. Siblings are elements that have the same parent element. Therefore, this selector is used to select elements that are directly after another certain element and have the same parent element. The selectors are combined with a plus [W3co21]. In the example in figure 11 are only <a> elements selected, which directly follows <div> elements.



```
div+a{
  font-size: 14px;
  font-weight: bold;
}
```

Figure 11: Adjacent Sibling Selector



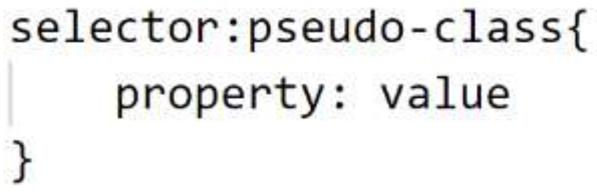
```
div~p{
  color: lightskyblue;
  font-weight: 500;
}
```

Figure 12: General Sibling Selector

The last one of the combinator selectors is the general sibling selector. It selects all elements that are the next siblings of a particular element. The selector is defined with a tilde between the selectors [W3co21]. In the example in figure 12, all <p> elements are selected, that are next siblings of <div> elements.

3.4.3Pseudo-Class Selectors

The pseudo-class selector, as the name suggests, applies the defined style specifications to special CSS pseudo-classes. these classes are there to define the style of certain states of an element. Use cases include changing the appearance of a button when the cursor moves over it or changing a link after it has been visited [W3pc21]. The syntax is shown in figure 13.

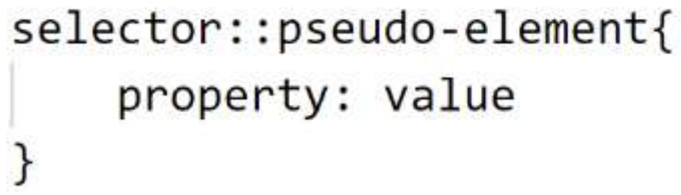


```
selector:pseudo-class{
    property: value
}
```

Figure 13: Pseudo-Class Selector [W3pc21]

3.4.4 Pseudo-Elements Selectors

The pseudo-element selector has similar functionality as the pseudo-class selector. Pseudo-elements are used to select and style certain parts of an element. They can be used for example to style the first letter of an element or to insert additional content before or after the content of an element [W3pe21]. Figure 14 illustrates the syntax.

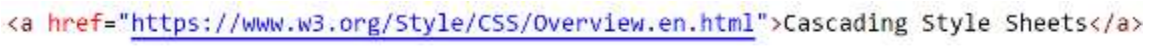


```
selector::pseudo-element{
    property: value
}
```

Figure 14: Pseudo-Element Selector [W3pe21]

3.4.5 Attribute Selectors

The last category of CSS selectors is the attribute selectors. An attribute provides additional information about an element and its syntax is usually a name/value pair. Figure 15 shows an example of a <a> HTML-tag, that defines a hyperlink, whereas ‘href’ is the attribute that defines the URL of the website to which the hyperlink leads [W3at21].

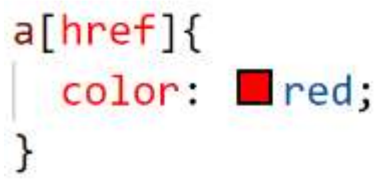


```
<a href="https://www.w3.org/Style/CSS/Overview.en.html">Cascading Style Sheets</a>
```

Figure 15: Defining a Hyperlink in HTML with Attribute ‘href’

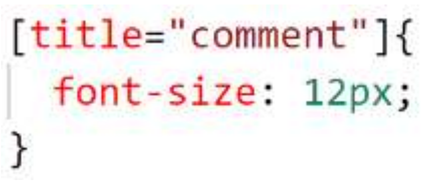
The attribute selector is used to address elements by their attributes. There are seven different ways this selector can be used [W3as21].

The simplest attribute selector selects the elements that have the specific attribute [W3as21]. An example of this selector is shown in figure 16, where all <a> elements with the attribute ‘href’ have a red font color.



```
a[href]{
  color: red;
}
```

Figure 16: Selecting a specific Attribute

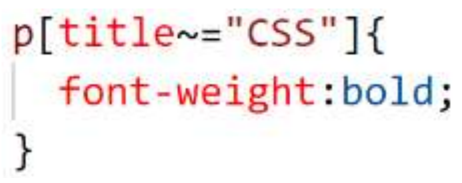


```
[title="comment"]{
  font-size: 12px;
}
```

Figure 17: Selecting by Attribute and the Value

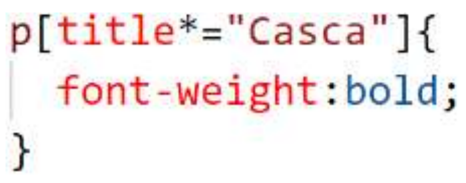
Another way to select an element by its attribute is to also specify its value [W3as21]. Figure 17 demonstrates how all elements with 'title' as attribute and ‘comment’ as value are designed in font-size 12 pixels.

With the next selector, it is possible to select elements where a certain word occurs in the value of the attribute [W3as21]. Figure 18 shows an example of how to style <p> elements where the value of a ‘title’ attribute contains the word ‘CSS’. The fourth option resembles the previous selector. The difference is that only a string and not a whole word must be included in the attribute value (Figure 19) [W3as21].



```
p[title~="CSS"]{
  font-weight:bold;
}
```

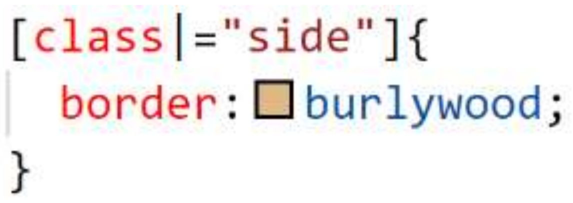
Figure 18: Attribute Value contains “CSS”



```
p[title*="Casca"]{
  font-weight:bold;
}
```

Figure 19: Attribute Value contains “Casca”

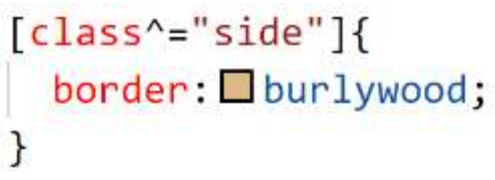
The next selector offers the possibility to address elements where the value of the attribute starts with a certain string. This string must be a whole word that either stands alone or is followed by a hyphen [W3as21]. The syntax is illustrated in figure 20.



```
[class|="side"]{
  border: burlywood;
}
```

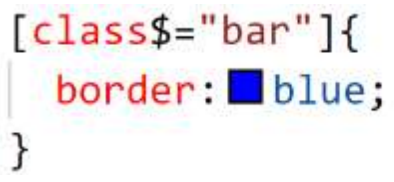
Figure 20: Select where Attribute Value starts with "side"

The selector shown in figure 21 is similar to the previous one. But instead of looking for a whole word at the beginning of an attribute value, the selector matches elements where the attribute value starts with a string that does not have to be a whole word. [W3as21]



```
[class^="side"]{
  border: burlywood;
}
```

Figure 21: Select where Value starts with “side”



```
[class$="bar"]{
  border: blue;
}
```

Figure 22: Select where Value ends with "bar"

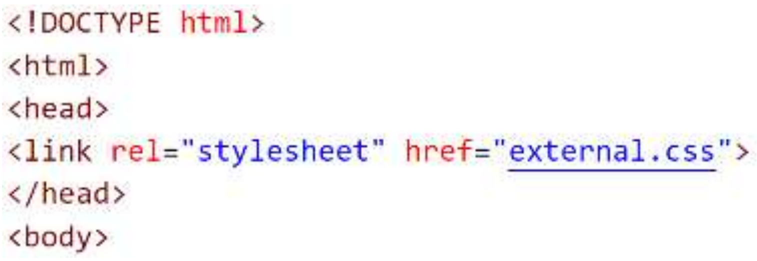
The last attribute selector follows the same principle as the previous one, with the difference that it selects elements with attribute values that end with a certain string (Figure 22) [W3as21].

3.5 Ways to Insert CSS

3.5.1 Insert into HTML

For the defined style specifications to be applied to the HTML document by the browser, CSS must be inserted into the document. There are three different ways to do this [W3ad21].

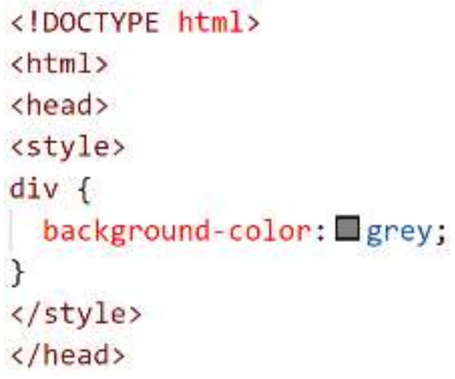
The first possibility is to insert CSS externally. This means, an external style sheet is used, which contains the CSS statements. This allows the same style sheet to be applied to several HTML documents, by including a reference to the style sheet inside of a <link> element inside the HTML header (Figure 23). This makes it possible to change several web pages by changing only one CSS file. This CSS file should only consist of CSS statements. It can be written in any text editor and is saved with the ‘.css’ extension [W3ad21].



```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="external.css">
</head>
<body>
```

Figure 23: Insert CSS external

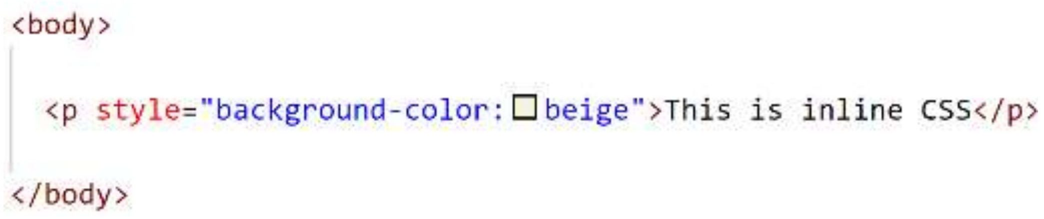
Another method to insert CSS is, by including the statements inside an <style> element directly in the header of the HTML document. This approach is called internal CSS. It is used, if one single webpage must have a unique style compared to the others (Figure 24) [W3ad21].



```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color:■grey;
}
</style>
</head>
```

Figure 24: Insert CSS internal

The last approach is called inline CSS. This method may be used when a single element is to be designed uniquely. The CSS statements are inserted directly into the start tag of the specific element, using the ‘style’ attribute (Figure 25). This type of insertion should be used sparingly and only in very special cases, as it contradicts the principle of separation of content and presentation [W3ad21].



```
<body>

  <p style="background-color:□beige">This is inline CSS</p>

</body>
```

Figure 25: Insert CSS inline

It is also possible to apply multiple style sheets to a single HTML document. If the same element with different properties appears in different style sheets, the style definition that appears last in the document is applied [W3ad21].

Apart from that, some rules define the order in which styles are applied:

- Inline style
- External and internal style sheets
- Browser default

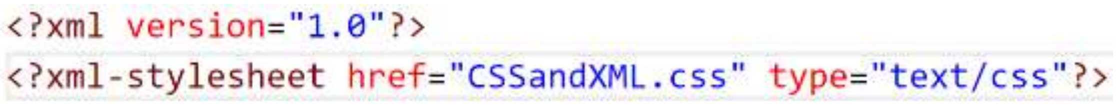
Following these rules, the styles will cascade into a new, virtual style sheet. This means, the inline style has the highest priority and will override both external and internal style sheets as well as the browser default settings [W3ad21].

3.5.2 Insert into XML

Due to the use of CSS for the design of web pages and web applications, it is largely used in conjunction with HTML. Nevertheless, as mentioned before, it is possible to apply CSS to all sorts of XML documents. However, there are some differences to be considered.

To understand these differences, the difference between HTML and XML must first be clarified. The first difference is the fundamental purpose of the two markup languages. While the task of HTML is to display content, the purpose of XML is to transport and store data. HTML is a markup language itself and therefore has predefined tags. XML on the other hand is basically a framework to define markup languages and therefore the tags can also be defined individually. By naming the XML tags individually, context and information about the content of the element can be made readable for humans [JP21], The problem is that the browser does not know the self-defined tags. The crucial difference is that the HTML elements are known to the browser and are already provided with simple formatting without CSS. The XML tags on the other side cannot be recognized because they are named individually. This means that more CSS properties must be taken into account than with HTML documents [ACPb21].

The inclusion of the CSS document in an XML file is similar to HTML. Figure 26 shows how a CSS file with the name ‘CSSandXML’ is inserted into a XML document. The insertion of the file must come right after the XML declaration [ACPb21].



```
<?xml version="1.0"?>
<?xml-stylesheet href="CSSandXML.css" type="text/css"?>
```

Figure 26: Insert CSS external into XML Document

4 Example of Use

Now that the basic concept, functionality, and syntax of CSS have been explained in the previous chapters, this chapter will illustrate practical applications.

4.1 Using CSS Frameworks

When it comes to developing a website or a web application from scratch, the process may be a very tedious and complex one. Not only does a considerable amount of written code accumulates, but this code must also be maintained and adapted while the website or web application evolves. For this reason, it can make sense to use CSS frameworks, even for smaller web development projects.

4.1.1 Components and Advantages

Frameworks are prefabricated sets of concepts, modules, and standardized specifications, which can be reapplied on different projects. Depending on the framework, a lot of different functions are provided. The use of a framework may reduce the amount of code noticeably. This can significantly reduce both the overall development time and the maintenance effort. In addition, the developer is relieved of many basic tasks and can concentrate on the more project-specific tasks. Basic components are listed and shortly described in the following section [ShPr18].

- ***Grids*** are systems that make it possible to structure the content of the web project horizontally and vertically. Moreover, they mostly add responsive design based on the screen or browser window size [ShPr18].
- ***Typography elements***, include all font-related designs, such as font styles, font colors, but also the alignment of the text.
- ***Cross-browser compatibility*** ensures, that all defined CSS statements are also compatible with the common browsers and are displayed correctly.
- ***Helper classes for positioning elements*** are used to avoid code repetition, by reusing classes that are made only for the purpose to position the elements [Aj14].

- ***Utility classes*** are self-descriptive classes, made only for one particular purpose. An example is the class '.bg-red' for a red background [Ba19].
- ***Navigational elements*** make it easier to create horizontal or vertical navigation bars, because of the given basic structure.
- ***Pre-processors*** are extended versions of CSS, that offer more functionality and attack the limitations of CSS. Examples are Syntactically Awesome Style Sheets (SASS) or 'Less'. Features of pre-processors are, for example, the possibility to use functions or to perform calculations [NdMu19].
- ***Media elements*** are predefined components like badges, tooltips, or comments [ShPr18].

Further advantages of frameworks are the ease of modification and expansion, existing documentation, a clean and consistent code structure [ShPr18].

4.1.2 Popular Frameworks

Bootstrap

The most popular framework in web design is Bootstrap. It applies the mobile-first approach, which means that the design is first designed for mobile devices. Thereafter, responsiveness adapts the design to other devices. Bootstrap provides a lot of extras like SASS variables and functions, a responsive grid system, prebuilt components, and JavaScript plugins. Due to the number of possibilities of Bootstrap, it is especially suitable for large web projects [ShPr18].

Foundation

Foundation was the first framework that enabled responsiveness. Like Bootstrap, it also follows the mobile-first approach and is similarly equipped. With the help of Foundation, you can customize web pages and apps as well as e-mails with ready-made components [ShPr18].

Materialize

Materialize is based on the Material Design philosophy of Google. This design language was developed by Google to unify the user experience across their products. Materialize can be used in two different forms. The standard version comes with CSS and JavaScript files, while the other version also contains SASS source code [ShPr18].

The frameworks described above have a very wide range of possibilities, which is why they are often too extensive and complex for web development beginners or very small and simple web projects. For this reason, three more lightweight frameworks are described below. These frameworks are intended to act as a starting point on which to build [ShPr18].

UIkit

UIkit is one of these simpler frameworks. Nevertheless, it contains a wide range of HTML, CSS, and JavaScript components and modules, with a high grade of customizability [ShPr18].

Milligram

Milligram is a very minimalistic framework, that focuses more on performance and high productivity than on a spectacular user interface. This is also reflected in the size of the neede files, which is only 2 kb when zipped [ShPr18].

Skeleton

Skeleton is also a very simple framework, consisting of only about 400 lines of code. Therefore, it only styles a few standard HTML elements. It follows a mobile-based philosophy. Advantages are that no installing or compiling is necessary and that it includes a grid system [ShPr18].

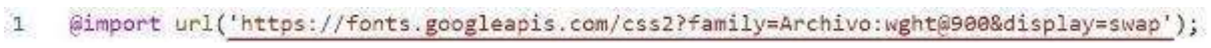
4.2 Nutshell Examples

In this chapter, some examples of CSS applications will be explained. The corresponding HTML document and the CSS code are attached in the appendix. Primarily, two concepts are explained to facilitate the arrangement of the various elements, namely Flexbox and Grid systems. In addition, some other styling features are shown. To simplify the description, the CSS file is divided into six parts: a general section, the navigation bar, the insertion of a picture, arranging elements with Flexbox, arranging elements with a Grid system, and the footer.

4.2.1 General

The first styling definitions apply to the entire document or to elements that occur repeatedly and should always look the same.

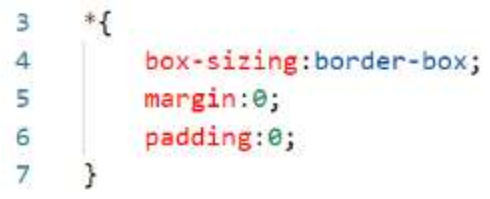
The first step in this example is to import a different font from Google Fonts. This is a web service of Google, where a huge amount of different fonts in different designs is provided. It would also be possible to embed the font in the HTML document. In the CSS file, this has to be the first statement (Figure 27).



```
@import url('https://fonts.googleapis.com/css2?family=Archivo:wght@900&display=swap');
```

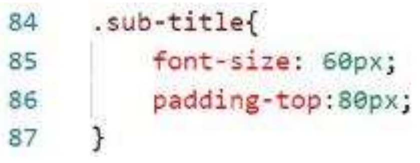
Figure 27: Importing a Font

Next, using the universal selector, some properties are specified that should apply to all elements (Figure 28). The property ‘box-sizing’ defines how elements behave when the height and width of the elements are specified. By using the value ‘border-box’, all padding and borders are included when entering height and width and the content field is reduced in size. The other two statements indicate that no padding or margin should be applied. In figure 29, the style of the subtitles is defined with a size of 60 pixels and padding on the top of 80 pixels.



```
*{
    box-sizing:border-box;
    margin:0;
    padding:0;
}
```

Figure 28: Universal Selector

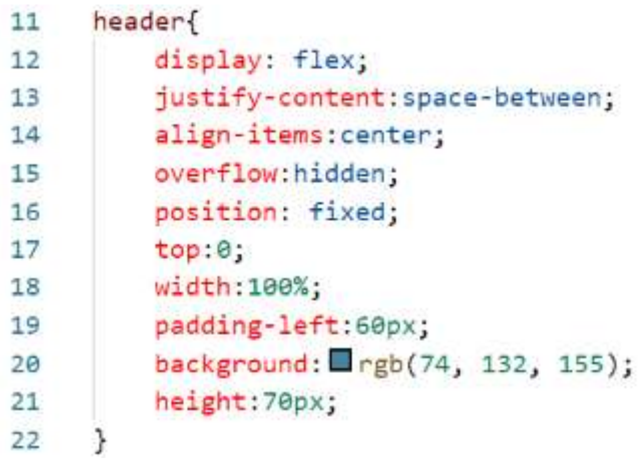


```
.sub-title{
    font-size: 60px;
    padding-top:80px;
}
```

Figure 29: Sub-Header definition

4.2.2 Navigation Bar

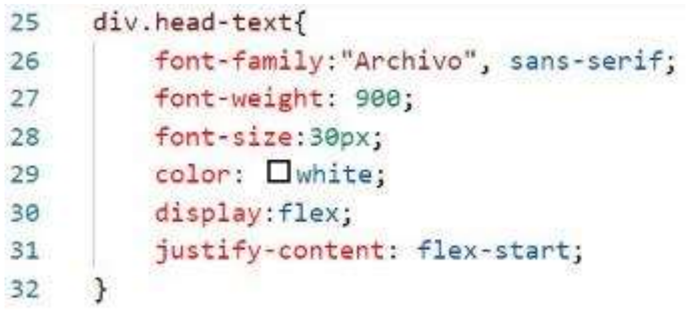
The first big part of the HTML document that has to be styled is the navigation bar. The header element (Figure 30), is the container of all elements that belong to the navigation bar at the top. The first three property-value pairs define the arrangement with Flexbox, which will be explained in detail later in this chapter. The statements ‘position: fixed’ and ‘top: 0’, specify that the navigation bar should be fixed at the top of the browser window. The statement ‘overflow: hidden’, specifies that the content should disappear behind the bar when scrolling. The width is set to 100%, which means that the element takes 100% of the width of the parent element. Furthermore, the height is set to 70 pixels, the left padding is set to 60 pixels and also the background color is defined.



```css
header{
    display: flex;
    justify-content:space-between;
    align-items:center;
    overflow:hidden;
    position: fixed;
    top:0;
    width:100%;
    padding-left:60px;
    background: rgb(74, 132, 155);
    height:70px;
}
```

Figure 30: Container for Navigation Bar Elements

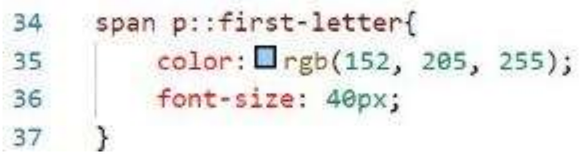
The next declaration styles the ‘Cascading Style Sheets’ text in the navigation bar. It includes the definition of the font family, the font-weight, and the font size and color. Furthermore, the element arrangement is again made with Flexbox (Figure 31)



```css
div.head-text{
    font-family:"Archivo", sans-serif;
    font-weight: 900;
    font-size:30px;
    color: white;
    display:flex;
    justify-content: flex-start;
}
```

Figure 31: Text in Navigation Bar

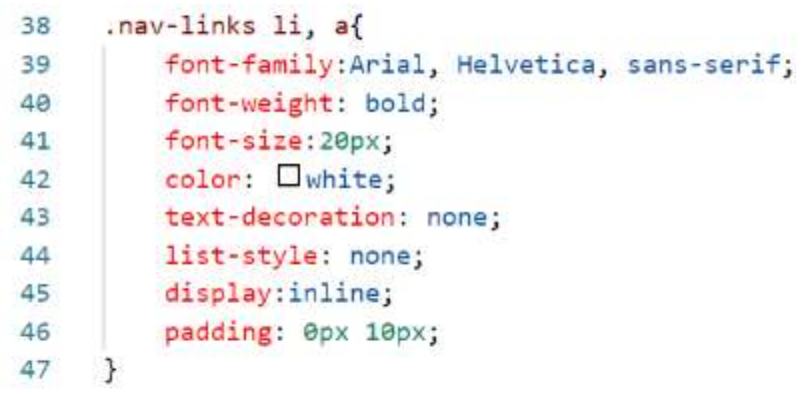
To style the first letters of the ‘Cascading Style Sheets’ in a different way, the <p> elements are spanned with a <span> tag. With the pseudo-element selector ‘first-letter’, the size and color can be adjusted among other things (Figure 32).



```css
span p::first-letter{
    color: rgb(152, 205, 255);
    font-size: 40px;
}
```

Figure 32: Pseudo-Element 'First-Letter’

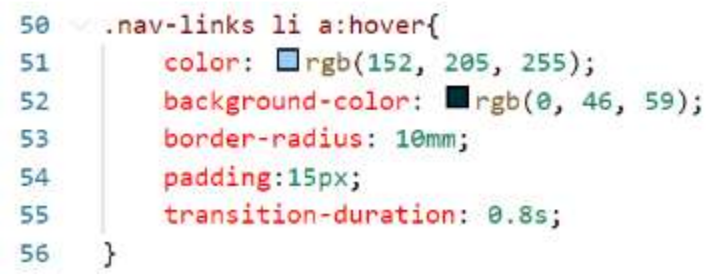
The next declaration defines the style of the <li> and <a> elements inside of elements with the class ‘.nav-links’. Besides the font properties described above, 'text-decoration: none' defines that hyperlinks should not be underlined. In addition, 'list-style: none" removes the bullets of the list elements. By using the declaration 'display: inline’ the elements are arranged side by side (Figure 33).



```css
.nav-links li, a{
    font-family:Arial, Helvetica, sans-serif;
    font-weight: bold;
    font-size:20px;
    color: white;
    text-decoration: none;
    list-style: none;
    display:inline;
    padding: 0px 10px;
}
```

Figure 33: Hyperlink-Elements in Navigation Bar

The declaration block in figure 34 specifies how the hyperlink elements in the navigation bar change when the cursor hovers over them. The background and font color change. With the property 'border-radius' the corners of the border are rounded. The ‘transition-duration’ property is used to define the time it takes to change from one defined state to another.



```css
.nav-links li a:hover{
    color: rgb(152, 205, 255);
    background-color: rgb(0, 46, 59);
    border-radius: 10mm;
    padding:15px;
    transition-duration: 0.8s;
}
```

Figure 34: Hover over Hyperlink-Elements in navigation Bar

Figure 35 shows the styled navigation bar, with the cursor over the ‘EXAMPLES’ hyperlink.

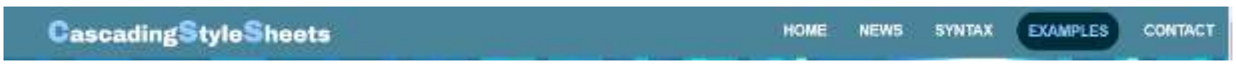
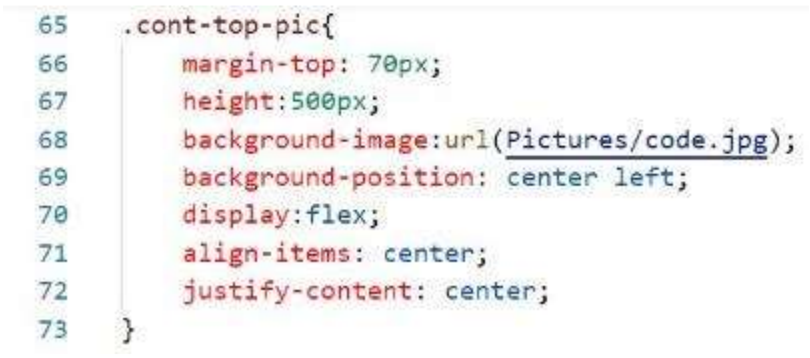


Figure 35: Finished Navigation Bar

4.2.3 Picture

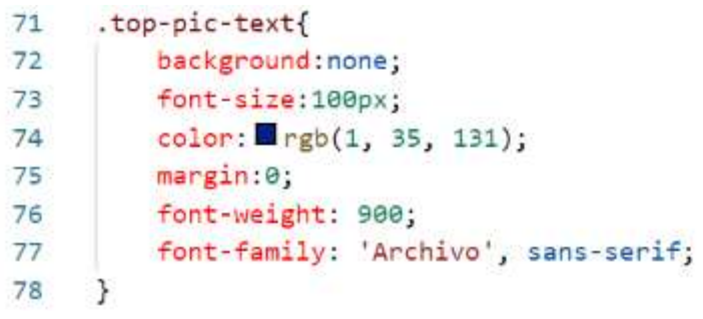
Now an image should be inserted directly after the navigation bar that runs across the entire browser window. One possible way would be to embed the picture in the HTML document. Since in this example a text is to be placed in the image, this variant is difficult to implement without first inserting the text into the original image. Therefore in the HTML document first an element with the class 'cont-top-pic' was created, in which an element with the class 'top-pic-text' is located, which contains the text. The declaration block for the container element contains again definitions for the arrangement of the child elements, using ‘display: flex’ and ‘align-items’ as well as ‘justify-content’ (see chapter 4.2.4). Furthermore, the top margin and the height of the element are set. Now to insert the picture as the background of the container element, the ‘background-image’ property with the URL of the picture as value is used. With ‘background-position’, the part of the image to be shown is selected (Figure 36).



```
.cont-top-pic{
    margin-top: 70px;
    height:500px;
    background-image:url(Pictures/code.jpg);
    background-position: center left;
    display:flex;
    align-items: center;
    justify-content: center;
}
```

Figure 36: Container with Background Picture

After that, only the element containing the text has to be styled (Figure 37).



```
.top-pic-text{
    background:none;
    font-size:100px;
    color:rgb(1, 35, 131);
    margin:0;
    font-weight: 900;
    font-family: 'Archivo', sans-serif;
}
```

Figure 37: Text in Picture

Figure 38 shows the inserted picture and text, that are positioned between the navigation bar and the main content.

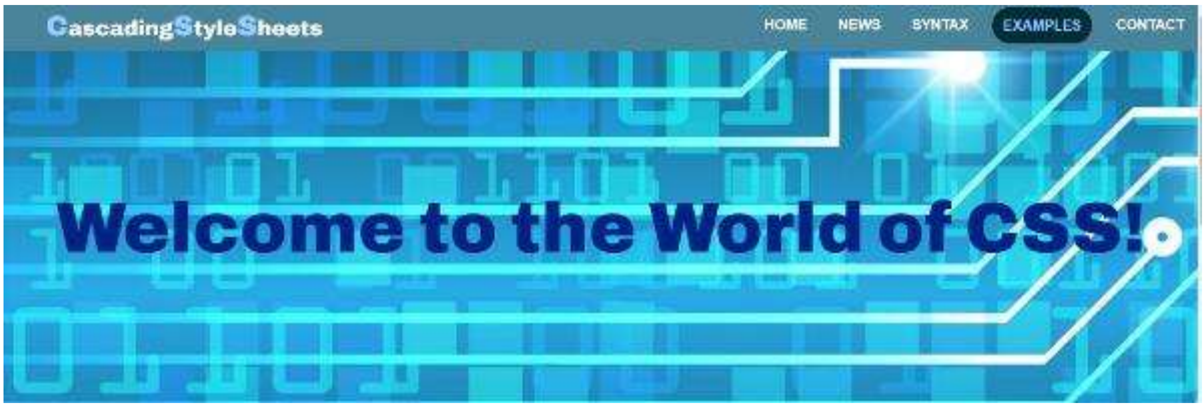
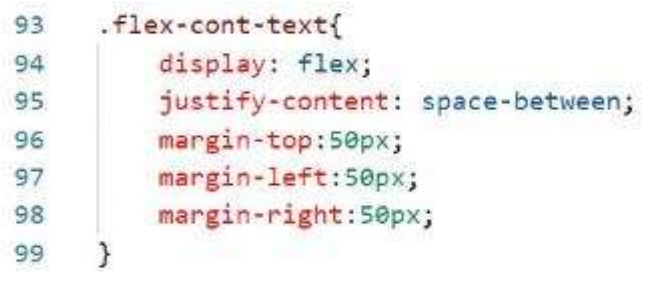


Figure 38: Text with Picture as Background

4.2.4 Flexbox

We continue with the next section of the CSS file, where the content of the HTML document is arranged using the Flexbox layout module. Flexbox is made for one-dimensional layout. This means it only can work on either columns or rows at the same time. It is designed for small-scaled layouts.

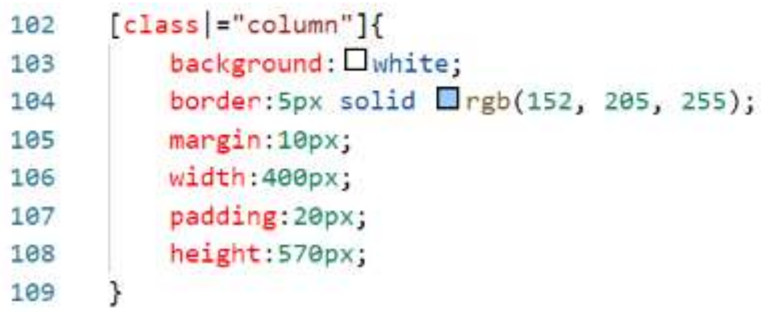
To use flexbox, first, a so-called ‘flex-container’ must be created, which contains the content that should be arranged. With the declaration 'display: flex', an element is defined as such a container. The justify-content property controls the horizontal alignment, while the align-content property controls the vertical alignment of the content. In this example are three columns to arrange. With the value ‘space-between’ of the justify-content property, the space between the elements is divided equally (Figure 39). Other possible values are ‘flex-start’, ‘flex-end’, ‘center’, or ‘space-around’.



```
.flex-cont-text{
    display: flex;
    justify-content: space-between;
    margin-top:50px;
    margin-left:50px;
    margin-right:50px;
}
```

Figure 39: Flex-Container

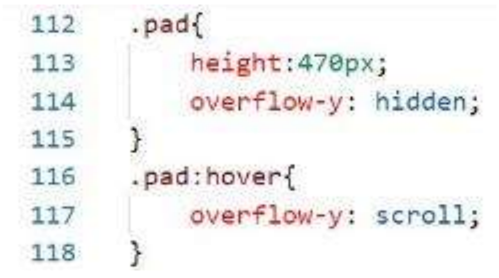
With an attribute selector, every element that starts with the string ‘column’ is styled like in figure 40. These are the elements that contain the content. Since the texts to be displayed are of different lengths, the height is set to 570 pixels so that the columns displayed have the same size.



```
[class|="column"]{
    background:□white;
    border:5px solid □rgb(152, 205, 255);
    margin:10px;
    width:400px;
    padding:20px;
    height:570px;
}
```

Figure 40: Textboxes in the Flex-Container

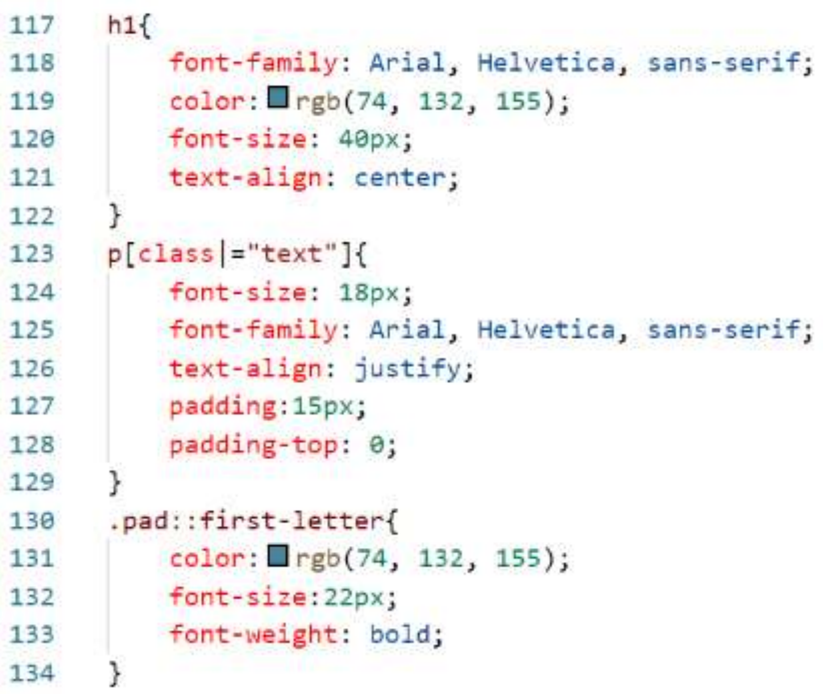
Since the text would now be displayed beyond the defined textbox, another element with the class 'pad' is spanned around the actual content. This element is set to a height of 470 pixels, to generate space between the textbox and the text itself. In addition, the overflowing text is hidden. Since the entire text is no longer readable, the pseudo-class selector 'hover' with the declaration 'overflow-y: scroll' determines that the element containing the text is provided with a scroll bar as soon as the cursor hovers over the element (Figure 41).



```
.pad{
    height:470px;
    overflow-y: hidden;
}
.pad:hover{
    overflow-y: scroll;
}
```

Figure 41: Text-Container and Hover-Effect

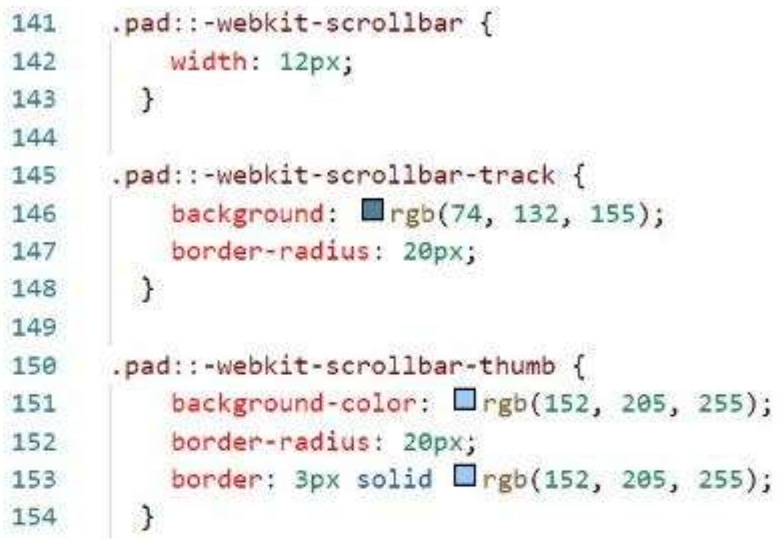
The following declaration blocks are defining the overall style of the content in the text boxes. First, the heading is styled by changing the default font family, color and size. Also, the heading is centered in the middle of the text box. Next, also a different size and font family are defined. Furthermore, the text alignment is set to justification. Lastly, the first letter of the three texts is given a different size and color, as well as being displayed in bold (Figure 42).



```css
h1{
    font-family: Arial, Helvetica, sans-serif;
    color: rgb(74, 132, 155);
    font-size: 40px;
    text-align: center;
}
p[class|="text"]{
    font-size: 18px;
    font-family: Arial, Helvetica, sans-serif;
    text-align: justify;
    padding:15px;
    padding-top: 0;
}
.pad::first-letter{
    color: rgb(74, 132, 155);
    font-size:22px;
    font-weight: bold;
}
```

Figure 42: General Styling of the Text

Because the displayed scrollbar does not match the overall design concept of the document, it is styled with the next declaration blocks. First, the general width of the scrollbar is set to 12 pixels. Second, the scrollbar track gets a different background and rounded corners. Finally, the scrollbar thump is given a different color and the same rounded corners as the scrollbar track (Figure 43).

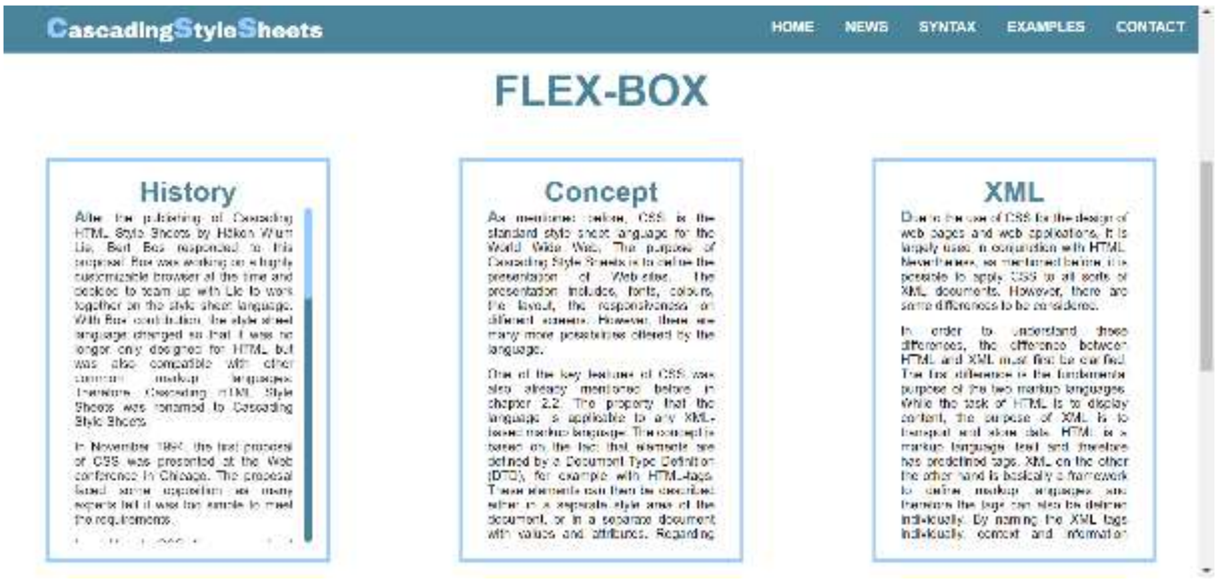


```css
.pad::-webkit-scrollbar {
    width: 12px;
  }

.pad::-webkit-scrollbar-track {
    background: rgb(74, 132, 155);
    border-radius: 20px;
  }

.pad::-webkit-scrollbar-thumb {
    background-color: rgb(152, 205, 255);
    border-radius: 20px;
    border: 3px solid rgb(152, 205, 255);
  }
```

Figure 43: Styling the Navigation Bar

The ‘::-webkit-scrollbar’ pseudo-element is no standard element. Therefore, it is not supported by every browser. However, among the most used browsers like Chrome, Safari, Opera, etc., only Firefox and older versions of Edge are not supporting this pseudo-element.

Figure 44 shows the content aligned with flexbox. The cursor hovers over the left text-box so that the scrollbar for the invisible text can also be seen. It can also be seen that the navigation bar is still visible even though the page is scrolled down.



FLEX-BOX

History

After the publishing of Cascading HTML Style Sheets by Håkon Wium Lie, Bert Bos responded to this proposal. Bos was working on a highly customizable browser at the time and decided to team up with Lie to work together on the style sheet language. With Bos contribution, the style sheet language changed so that it was no longer only designed for HTML but was also compatible with other common markup languages. Therefore Cascading HTML Style Sheets was renamed to Cascading Style Sheets.

In November 1994, the first proposal of CSS was presented at the Web conference in Chicago. The proposal faced some opposition as many experts felt it was too simple to meet the requirements.

Concept

As mentioned before, CSS is the standard style sheet language for the World Wide Web. The purpose of Cascading Style Sheets is to define the presentation of Web sites. The presentation includes, fonts, colours, the layout, the responsiveness on different screens. However, there are many more possibilities offered by the language.

One of the key features of CSS was also already mentioned before in chapter 2.2. The property that the language is applicable to any XML-based markup language. The concept is based on the fact that elements are defined by a Document Type Definition (DTD), for example with HTML-tags. These elements can then be described either in a separate style area of the document, or in a separate document with values and attributes. Regarding

XML

Due to the use of CSS for the design of web pages and web applications, it is largely used in conjunction with HTML. Nevertheless, as mentioned before, it is possible to apply CSS to all sorts of XML documents. However, there are some differences to be considered.

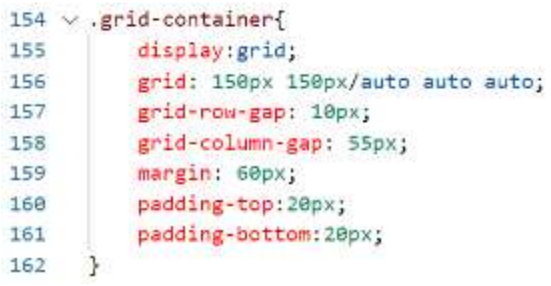
In order to understand these differences, the difference between HTML and XML must first be clarified. The first difference is the fundamental purpose of the two markup languages. While the task of HTML is to display content, the purpose of XML is to transport and store data. HTML is a markup language itself and therefore has predefined tags. XML on the other the other hand is basically a framework to define markup languages and therefore the tags can also be defined individually. By naming the XML tags individually, context and information

Figure 44: Content that is arranged by using Flexbox

4.2.5 Grid

Besides the Flexbox module, another layout module is used in this example, which is called Grid. Unlike Flexbox, the Grid is a two-dimensional system. This means that both, columns and rows, can be worked on at the same time. Another difference to flexbox is that the grid is less focused on content and more on the overall layout of, for example, a website. Therefore it is designed for large-scale layouts.

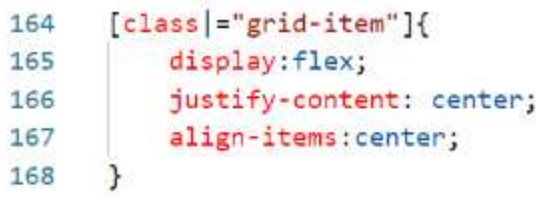
The first step, similar to the Flexbox module, is to define a grid container by using the declaration ‘display: grid’. With the ‘grid’ property and the corresponding values, a grid system is defined, which consists of two rows with a height of 150 pixels and three columns whose width is automatically defined. The properties ‘grid-row-gap’ and ‘grid-column-gap', are setting the spaces between the grid-items. Also, a margin of 60 pixels and a padding of 20 pixels on the top and the bottom is defined (Figure 45).



```css
.grid-container{
    display:grid;
    grid: 150px 150px/auto auto auto;
    grid-row-gap: 10px;
    grid-column-gap: 55px;
    margin: 60px;
    padding-top:20px;
    padding-bottom:20px;
}
```

Figure 45: Grid-Container

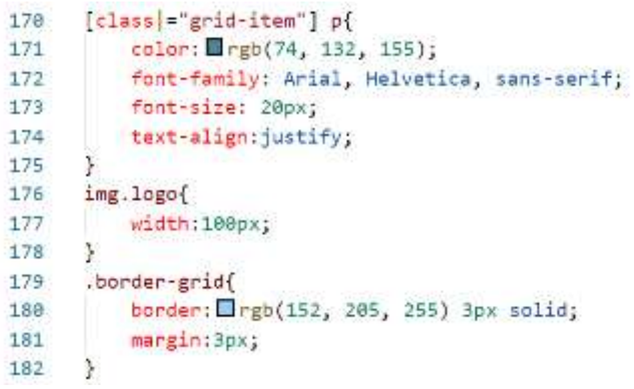
To align the contents of the grid items, they are defined as Flexbox-containers. The child elements of the grid items should be centered both horizontally and vertically (Figure 46).



```css
[class|="grid-item"]{
    display:flex;
    justify-content: center;
    align-items:center;
}
```

Figure 46: Grid Items

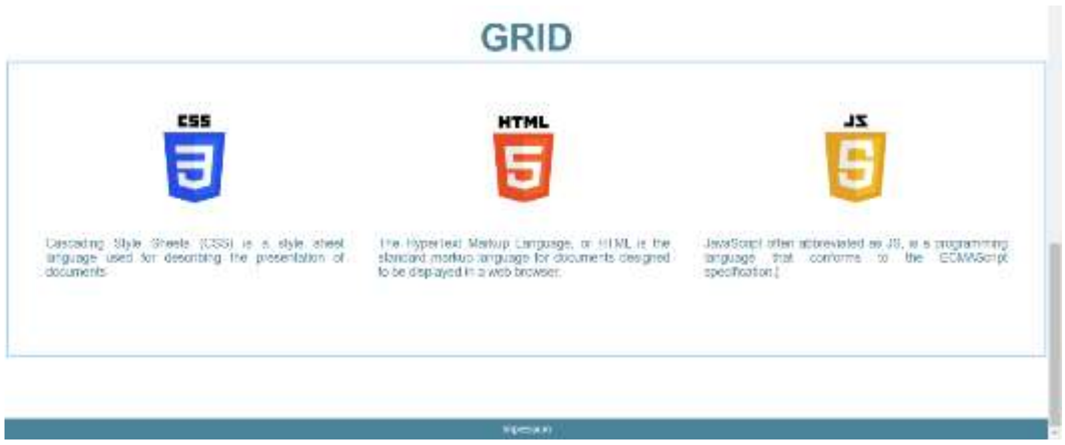
The last declaration blocks are again used for styling the content. The <p> elements that are contained in the grid elements get a different color, font family, and font size. Additionally, they are displayed in justification. The pictures which are embedded in the grid items should have a width of 100 pixels. Finally, a border is defined with an element that is spanned around the entire grid section (Figure 47).



```css
[class|="grid-item"] p{
    color: rgb(74, 132, 155);
    font-family: Arial, Helvetica, sans-serif;
    font-size: 20px;
    text-align:justify;
}
img.logo{
    width:100px;
}
.border-grid{
    border: rgb(152, 205, 255) 3px solid;
    margin:3px;
}
```

Figure 47: Content in the Grid Section

Figure 48 shows the content that was arranged using a grid system and the footer. The grid system consists of three columns with two rows. The first row contains the logos and the second row some text. With the grid, the content can be arranged very easily.



Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of documents.

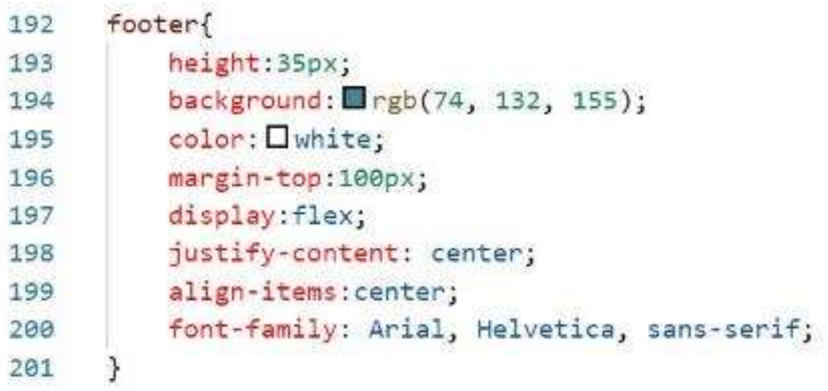
The Hypertext Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser.

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification.[

Figure 48: Content arranged with a Grid-System and Footer

4.2.6Footer

The last to be styled part of the HTML document is the footer. It is basically the same as the navigation bar, but more minimalistic (Figure 49).



```
footer{
    height:35px;
    background:rgb(74, 132, 155);
    color:white;
    margin-top:100px;
    display:flex;
    justify-content: center;
    align-items:center;
    font-family: Arial, Helvetica, sans-serif;
}
```

Figure 49: Footer Style Declarations

5 Summary and Outlook

CSS has evolved tremendously over the last 25 years. From being only a proposal for a standard style sheet language to becoming the standard for styling websites on the

World Wide Web. It should be emphasized that CSS can be applied not only to HTML but to all XML-based documents.

Despite the relatively simple syntax and the easy-to-understand box-model, CSS can be quite overwhelming because of the huge number of properties and values that are available. This is mainly due to the constant further development. Furthermore, keeping the CSS code neat and compact is challenging, especially for beginners.

CSS frameworks are one way to shorten the code by using predefined elements and classes. Nevertheless, beginners shouldn’t rely only on frameworks. However, for large web projects, these can be very useful and, at a certain level, necessary.

The further development of the individual modules takes place independently of each other. This leads to the fact that since CSS Level 3 no new versions of CSS as a whole are released. Individual modules may therefore be more advanced than others.

CSS is already a very rich style sheet language and with the development of pre-processors like SASS, even more, functionality is added. Therefore, even 25 years after its introduction, the development of CSS is far from complete.

References

[ACPa21] CSS (Cascading Style Sheets) lernen. (n.d.). a coding project.
from https://www.a-coding-project.de/ratgeber/css
Retrieved 22 October 2021

[ACPb21] XML gestalten mit CSS. (n.d.). a coding project.
from https://www.a-coding-project.de/ratgeber/xml/gestalten-mit-css.
Retrieved 23 October 2021

[Aj14] Ajmi, A. (2018, October 4). Using Helper Classes to DRY and Scale CSS. SitePoint.
from https://www.sitepoint.com/using-helper-classes-dry-scale-css/
Retrieved 15 November 2021

[AtEt20] Atkins Jr., T., Etemad, E. J., & Rivoal, F. (2020, December 22). CSS Snapshot 2020. World Wide Web Consortium (W3C).
from https://www.w3.org/TR/CSS/#css-levels.
Retrieved 21 October 2021

[Ba19] Barker, M. (2019, January 28). A Year of Utility Classes. CSS In Real Life.
from https://css-irl.info/a-year-of-utility-classes/.
Retrieved 15. November 2021

[Bloo17] Bloom, Z. (2018, August 29). The Languages Which Almost Became CSS. The Cloudflare Blog.
from https://blog.cloudflare.com/the-languages-which-almost-became-css/.
Retrieved 20 October 2021

[Bos16] Bos, B. (2017, June 16). 20 Years of CSS. World Wide Web Consortium (W3C)
from https://www.w3.org/Style/CSS20/.
Retrieved 20 October 2021

[JaLe97] Raggett, D., le Hors, A., & Jacobs, I. (1997, July 8). A brief SGML tutorial. World Wide Web Consortium (W3C).
from https://www.w3.org/TR/WD-html40-970708/intro/sgmltut.html. Retrieved 20 October 2021

[JP21] HTML vs XML. (n.d.). Javatpoint.
from https://www.javatpoint.com/html-vs-xml.
Retrieved 23 October 2021

[NdMu19] Ndia, J. G., Muketha, G. M., & Omieno, K. K. (2019). A SURVEY OF CASCADING STYLE SHEETS COMPLEXITY METRICS. International Journal of Software Engineering & Applications, 10(03), 21–33.

[ShPr18] Shenoy, A., & Prabhu, A. (2018). CSS Framework Alternatives: Explore Five Lightweight Alternatives to Bootstrap and Foundation with Project Examples (1st ed.). Apress.

[W3ad21] How to add CSS. (n.d.). W3C Schools.
from https://www.w3schools.com/Css/css_howto.asp.
Retrieved 23 October 2021

[W3as21] CSS Attribute Selector. (n.d.). W3C Schools.
from https://www.w3schools.com/Css/css_attribute_selectors.asp.
Retrieved 22 October 2021

[W3at21] HTML Attributes. (n.d.). W3C Schools.
from https://www.w3schools.com/htmL/html_attributes.asp.
Retrieved 22 October 2021

[W3bm21] CSS Box Model. (n.d.). W3C Schools.
from https://www.w3schools.com/css/css_boxmodel.asp.
Retrieved 22 October 2021

[W3co21] CSS Combinators. (n.d.). W3C Schools.
from https://www.w3schools.com/Css/css_combinators.asp.
Retrieved 22 October 2021

[W3hc21] HTML & CSS (2016). World Wide Web Consortium (W3C).
from https://www.w3.org/standards/webdesign/htmlcss.
Retrieved 21 October 2021

[W3pc21] CSS Pseudo-classes. (n.d.). W3C Schools.
from https://www.w3schools.com/Css/css_pseudo_classes.asp.
Retrieved 22 October 2021

[W3pe21] CSS Pseudo-elements. (n.d.). W3C Schools.
from https://www.w3schools.com/Css/css_pseudo_elements.asp.
Retrieved 22 October 2021

[W3se21] CSS Selectors. (n.d.). W3C Schools.
from https://www.w3schools.com/Css/css_selectors.asp.
Retrieved 22 October 2021

[W3sy21] CSS Syntax. (n.d.). W3C Schools.
from https://www.w3schools.com/Css/css_syntax.asp.
Retrieved 22 October 2021

List of Figures

Figure 1: Box Model [W3bm21] 12
Figure 2: CSS-Syntax [W3sy21] 13
Figure 3: Simple Selector 13
Figure 4: ID Selector 13
Figure 5: Class Selector 14
Figure 6: Simple and Class Selector combined 14
Figure 7: Universal Selector 14
Figure 8: Group Selector 14
Figure 9: Descendant Selector 15
Figure 10: Child Selector 15
Figure 11: Adjacent Sibling Selector 15
Figure 12:General Sibling Selector 15
Figure 13: Pseudo-Class Selector [W3pc21] 16
Figure 14: Pseudo-Element Selector [W3pe21] 16
Figure 15: Defining a Hyperlink in HTML with Attribute ‘href’ 16
Figure 16: Selecting a specific Attribute 17
Figure 17: Selecting by Attribute and the Value 17
Figure 18: Attribute Value contains “CSS” 17
Figure 19: Attribute Value contains “Casca” 17
Figure 20: Select where Attribute Value starts with "side" 17
Figure 21: Select where Value starts with “side” 18
Figure 22: Select where Value ends with "bar" 18
Figure 23: Insert CSS external 18
Figure 24: Insert CSS internal 19
Figure 25: Insert CSS inline 19
Figure 26: Insert CSS external into XML Document 20
Figure 27: Importing a Font 24
Figure 28: Universal Selector Figure 29: Sub-Header definition 24
Figure 30: Container for Navigation Bar Elements 25
Figure 31: Text in Navigation Bar 25
Figure 32: Pseudo-Element 'First-Letter’ 25
Figure 33: Hyperlink-Elements in Navigation Bar 26

Figure 34: Hover over Hyperlink-Elements in navigation Bar 26
Figure 35: Finished Navigation Bar 26
Figure 36: Container with Background Picture 27
Figure 37: Text in Picture... 27
Figure 38: Text with Picture as Background 28
Figure 39: Flex-Container 28
Figure 40: Textboxes in the Flex-Container.. 29
Figure 41: Text-Container and Hover-Effect 29
Figure 42: General Styling of the Text 30
Figure 43: Styling the Navigation Bar 30
Figure 44: Content that is arranged by using Flexbox 31
Figure 45: Grid-Container 32
Figure 46: Grid Items... 32
Figure 47: Content in the Grid Section 32
Figure 48: Content arranged with a Grid-System and Footer 33
Figure 49: Footer Style Declarations 33

Appendix

HTML-Script

```html
<!DOCTYPE html>
<html lang="de">
    <head>
        <meta charset="utf-8">
        <meta name="viewpoint" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" type="text/css" href="style.css" />
        <title>Cascading Style Sheets</title>
    </head>

    <body>
<!------------------------------NAVBAR----------------------------->
        <header>
            <div class="head-text">
                <span><p>Cascading</p></span>
                <span><p>Style</p></span>
                <span><p>Sheets</p></span>
            </div>
            <nav>
                <ul class="nav-links">
                    <li><a href="#">HOME</a></li>
                    <li><a href="#">NEWS</a></li>
                    <li><a href="#">SYNTAX</a></li>
                    <li><a href="#">EXAMPLES</a></li>
                    <li><a href="#">CONTACT</a></li>
                </ul>
            </nav>
        </header>
<!---------------------------MAIN CONTENT---------------------------->
        <div class="main">

            <!--Picture at the top--->
            <div class="cont-top-pic">
                <p class="top-pic-text">Welcome to the World of CSS!</p>
            </div>

            <!-------Sub title--------->
            <h1 class="sub-title">FLEX-BOX</h1>

            <!--Inserting three rows of text--->
            <div class="flex-cont-text">
                <div class="row-1">
            <h1>History</h1>
            <div class="pad">

                <p class="text-1">
                After the publishing of Cascading HTML Style
                Sheets by Håkon Wium Lie, Bert Bos responded to
                this proposal. Bos was working on a highly cus-
                tomizable browser at the time and decided to
                team up with Lie to work together on the style
                sheet language. With Bos' contribution, the
                style sheet language changed so that it was no
                longer only designed for HTML but was also com-
                patible with other common markup languages.
                Therefore, Cascading HTML Style Sheets was re-
                named Cascading Style Sheets.
                </p>

                <p class="text-1">
                In November 1994, the first proposal of CSS was
                presented at the Web conference in Chicago. The
                proposal faced some opposition as many experts
                felt it was too simple to meet the require-
                ments.
                </p>

                <p class="text-1">
                In addition to CSS, there were about 10 other
                style sheet languages that were proposed at
                this time. DSSSL for example was one of them.
                The feature that distinguished CSS from all the
                other proposals was the approach, that the
                user, the author, and the technical possibili-
                ties of the display devices and the browser
                should all influence the design.
                </p>

                <p class="text-1">
                The balance between the influence of users and
                authors on the presentation of web pages trig-
                gered a fundamental discussion at the next web
                conference in 1995. Web developers felt they
                should have the power to decide how the docu-
                ment is ultimately presented. Lie and Bos ar-
                gued that the user should have the last word,
                as he is the one who has to process the impres-
                sions presented.
                </p>

                <p class="text-1">
```

```html
                Also in 1995, the world wide web consortium
                (W3C), founded the year before, developed into
                an operative organisation. Many companies
                joined the consortium and its influence in-
                creased. They organised workshops on a variety
                of topics, including one on style sheets. One
                of the participants of this workshop was Thomas
                Reardon. He was one of the developers of Mi-
                crosoft’s Internet Explorer. He ensured the
                support of CSS…
                </p>

                <p class="text-1">
                In December 1996, CSS1 became a W3C Recommenda-
                tion. In 1997, W3C intro-duced a separate work-
                ing group for CSS to develop features for that
                were not yet implemented in the first version.
                In 1998, CSS2 became a W3C Recommendation. The
                working group, which consisted of 15 members in
                1999, had 115 members in 2016. Their task is it
                to develop new modules for CSS and also fix er-
                rors
                </p>

            </div>
        </div>
        <div class="row-2">
            <h1>Concept</h1>
            <div class="pad">
                <p class="text-2">
                As mentioned before, CSS is the standard style
                sheet language for the World Wide Web. The pur-
                pose of Cascading Style Sheets is to define the
                presentation of Web-sites. The presentation in-
                cludes, fonts, colours, the layout, the respon-
                siveness on different screens. However, there
                are many more possibilities offered by the lan-
                guage.
                </p>

                <p class="text-2">
                One of the key features of CSS was also already
                mentioned before in chapter 2.2: The property
                that the language is applicable to any XML-
                based markup language. The concept is based on
                the fact that elements are defined by a Docu-
                ment Type Definition (DTD), for example with
                HTML-tags. These elements can then be…
                </p>
            </div>
```

```
                </div>
                <div class="row-3">
                    <h1>XML</h1>
                    <div class="pad">
                        <p class="text-3">
                        Due to the use of CSS for the design of web
                        pages and web applications, it is largely used
                        in conjunction with HTML. Nevertheless, as men-
                        tioned before, it is possible to apply CSS to
                        all sorts of XML documents. However, there are
                        some differences to be considered.
                        </p>

                        <p class="text-3">
                        In order to understand these differences, the
                        difference between HTML and XML must first be
                        clarified. The first difference is the funda-
                        mental purpose of the two markup languages.
                        While the task of HTML is to display content,
                        the purpose of XML is to…
                        </p>
                    </div>
                </div>
            </div>


            <h1 class="sub-title">GRID</h1>
            <div class="border-grid">

                     <!---------------GRID------------------------->

                     <div class="grid-container">
                         <div class="grid-item-1">
                              <img class="logo" src="Pictures/css.png"
                              alt="">
                         </div>

                         <div class="grid-item-2">
                              <img class="logo" src="Pictures/html.png"
                              alt="">
                         </div>

                         <div class="grid-item-3">
                              <img class="logo" src="Pictures/JS.png"
                              alt="">
                         </div>

                         <div class="grid-item-4">
                             <p>
```

```html
                        Cascading Style Sheets (CSS) is a style
                        sheet language used for describing the
                        presentation of documents
                        </p>
                    </div>

                    <div class="grid-item-5">
                        <p>
                        The HyperText Markup Language, or HTML is
                        the standard markup language for docu-
                        ments designed to be displayed in a web
                        browser.
                        </p>
                    </div>

                    <div class="grid-item-6">
                        <p>
                        JavaScript often abbreviated as JS, is a
                        programming language that conforms to the
                        ECMAScript specification.
                        </p>
                    </div>
                </div>
            </div>
        </div>

        <footer>
            <p>Impressum</p>
        </footer>
    </body>
```

CSS-Code

```
@import url('https://fonts.googleapis.com/css2?family=Archivo:wght@900&dis-
play=swap');

*{
    box-sizing:border-box;
    margin:0;
    padding:0;}

/*--------------------HEADER-------------------------*/

header{
    display: flex;
    justify-content:space-between;
    align-items:center;
    overflow:hidden;
    position: fixed;
    top:0;
    width:100%;
    padding-left:60px;
    background:rgb(74, 132, 155);
    height:70px;}

div.head-text{
    font-family:"Archivo", sans-serif;
    font-weight: 900;
    font-size:30px;
    color: white;
    display:flex;
    justify-content: flex-start;}

div.head-text p::first-letter{
    color:rgb(152, 205, 255);
    font-size: 40px;}

.nav-links li, a{
    font-family:Arial, Helvetica, sans-serif;
    font-weight: bold;
    font-size:20px;
    color: white;
    text-decoration: none;
    list-style: none;
    display:inline;
    padding: 0px 10px;}
```

```css
.nav-links li a:hover{
    color: rgb(152, 205, 255);
    background-color: rgb(0, 46, 59);
    border-radius: 10mm;
    padding:15px;
    transition-duration: 0.8s;}

/*------------------------CONTENT---------------------*/

/*Picture*/

.cont-top-pic{
    margin-top: 70px;
    height:500px;
    background-image:url(Pictures/code.jpg);
    background-position: center left;
    display:flex;
    align-items: center;
    justify-content: center;}

.top-pic-text{
    background:none;
    font-size:100px;
    color:rgb(1, 35, 131);
    margin:0;
    font-weight: 900;
    font-family: 'Archivo', sans-serif;}

/*--------------Sub Header----------------*/
.sub-title{
    font-size: 60px;
    padding-top:80px;}


/*---------------FLEX-BOX-------------*/

/*Flex Container*/
.flex-cont-text{
    display: flex;
    justify-content: space-between;
    margin-top:50px;
    margin-left:50px;
    margin-right:50px;}
```

```css
/*Container for rows*/
[class|="row"]{
    background:white;
    border:5px solid rgb(152, 205, 255);
    margin:10px;
    width:400px;
    padding:20px;
    height:570px;}

/*Padding for scrollable text*/
.pad{
    height:470px;
    overflow-y: hidden;}

.pad:hover{
    overflow-y: scroll;}

h1{
    font-family: Arial, Helvetica, sans-serif;
    color:rgb(74, 132, 155);
    font-size: 40px;
    text-align: center;}

p[class|="text"]{
    font-size: 18px;
    font-family: Arial, Helvetica, sans-serif;
    text-align: justify;
    padding:15px;
    padding-top: 0;}

.pad::first-letter{
    color:rgb(74, 132, 155);
    font-size:22px;
    font-weight: bold;}

/*Scrollbar design*/
.pad::-webkit-scrollbar {
    width: 12px; }                          /* width of the entire scrollbar */

.pad::-webkit-scrollbar-track {
    background: rgb(74, 132, 155);            /* color of the tracking area */
    border-radius: 20px;}

.pad::-webkit-scrollbar-thumb {
    background-color: rgb(152, 205, 255);   /* color of the scroll thumb */
    border-radius: 20px;                    /* roundness of the scroll thumb */
    border: 3px solid rgb(152, 205, 255);} /* creates padding around scroll
thumb */
```

```css
/*-------------GRID----------*/
.grid-container{
    display:grid;
    grid: 150px 150px/auto auto auto;
    grid-row-gap: 10px;
    grid-column-gap: 55px;
    margin: 60px;
    padding-top:20px;
    padding-bottom:20px;}

[class|="grid-item"]{
    display:flex;
    justify-content: center;
    align-items:center;}

[class|="grid-item"] p{
    color:rgb(74, 132, 155);
    font-family: Arial, Helvetica, sans-serif;
    font-size: 20px;
    text-align:justify;}

img.logo{
    width:100px;}

.border-grid{
    border:rgb(152, 205, 255) 3px solid;
    margin:3px;}

/*----------------------------FOOTER----------------------------------*/
footer{
    height:35px;
    background:rgb(74, 132, 155);
    color:white;
    margin-top:100px;
    display:flex;
    justify-content: center;
    align-items:center;
    font-family: Arial, Helvetica, sans-serif;}
```