Weather App Project SoSe22

TechAcademy e.V.

Summer Term 2022

Contents

Chapter 1

Introduction

Welcome to the Web Development program for the summer term of 2022! We are very excited to have you here. In these first paragraphs we want to briefly introduce our program and give you an overview of what to expect in the following months as well as lay out the content of this guideline. We will help and guide you along the way to the best of our abilities and hope we can share some of that fascination that got us into programming with you!

Over the next months, there will be a total of four Coding Meetups, spaced out evenly where you will have the ability to meet your peers, work together on our application and ask questions to your mentors. In preparation we partnered up with Udemy and thus you have free access to the ultimate Web Development Bootcamp course. Everything you will learn there, you can later use within your own project and solidify your knowledge. We will go over the course and establish a timeframe on how far you should have progressed before each meetup later on. Feel free to sprint ahead on your own though, we are only laying out the minimal requirements here.

In your project, you can work in groups of up to four team members. You can choose your teammates independently and we will not interfere with your arrangements. It is however beneficial, that all group members have approximately the same level of knowledge. We explicitly encourage you to collaborate with students from different university faculties. This not only allows you to get to know people from other faculties, but may also give you a whole new perspective on the project and tasks. When submitting your project please note: for a certificate, each person must submit the project individually. However, hand-ins may be similar within your group. You can get more information at our first Coding Meetup on **May 18, 2022.**

So.. what is our project? For this term, we have decided to let you create a weather-app. You've seen them countless times on various websites and may use your favorite one every day to check if you have to put on a raincoat or

sunscreen. But how does the data get to the website? How does it update with every refresh and how can one make look that all appealing to an audience? These are all questions that we will tackle in this semester.

This project guide was developed and written by the Web Development team at TechAcademy: Feron Basoeki, Sebastian Hönig, Burak Onat and Tilo Flasche. Special thanks also to Ben Lucht, Feron Basoeki and Tom Schwitzkowski who were responsible for the Web Development course in the past.

Chapter 2

What's web development?

In short, web developers build and maintain websites.
Web developers often work for clients who are trying to get their product or service onto the web. The work is typically very project focused and involves collaborating with a team that helps to coordinate the client's needs into the end product. The work could involve front-end, back-end, or full-stack web development.
In our course we will focus on front-end development. The front end is what you see when you open any website in your browser. At a basic level, front-end developers use HTML, CSS and JavaScript.

2.1 How to use this section

In the following we will briefly introduce HTML, CSS and JavaScript. Our intention here was not to go into every little detail there is (There have been whole books written about each of these!), but to give a broad overview of the different components and how they work together. The Udemy course will go over the concepts we outlay here and lots of additional content in more detail. Use this section as an appetizer and/or as a refresher to come back to once you've gone through the course.

2.2 HTML

HTML, short for Hypertext Markup Language, is the raw data that a webpage is built out of. It allows you to structure the website through sections, paragraphs, headings and lets you create text, links, cards, lists, and buttons.

Tags

The structure of a html file is very simple. Almost all elements are just pieces of content wrapped in opening and closing HTML tags. For example, you would write the title of a webpage like so:

```
<title>Document</title>
```

And while the current version of HTML (HTML5) contains 140 different tags, most webpages are made just out of a fraction of these. You've already seen the title tag, let's quickly take a look at some more of those common ones. They are also the ones that you will most likely be using during your project.
- `<html>`: Must enclose the entire rest of the code
- `<head>`: Contains the title, the link to the css (styling) file and various other information
- `<body>`: Contains the contents of a website
- `<p>`: Paragraph, simple text belongs here
- `<hX>`: X is a number between 1 (large heading) and 6 (small heading)
- `<div>`: Structure that groups content inside
- `<ul>`: Unordered list containing list items `<li>`
- `<a>`: Link tag, used to link from one website to the next
- `<img>`: An image in an HTML page
- `<form>`: To create an HTML form for user input
- `<table>`: Defines a table, containing rows `<tr>` and columns `<td>`
- `<button>`: Create a clickable button

To all of these tags you can add classes or id's. An id has to be unique. Why would we do this!? We will get to that once we learn about CSS.

```
<div class="small-text">
  <p id="first-line">Name</p>
  <p id="second-line">Tom</p>
</div>
```

Structure

Now that you know the majority of the relevant tags the only question left to answer is how do we actually write HTML code. For that it is relevant that all HTML documents have the same boilerplate-like structure that needs to be in place before anything useful can be done. The boilerplate looks something like this and is often easily accessible with a shortcut (so you don't have to type it all out everytime you create a new file):

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

All of the content, that appears on a webpage is written between the `<body>` tags.
So.. we know the basic structure and some HTML tags, but how and where do we write HTML Code? The answer is straight forward: We can write HTML Code anywhere! Even in a simple plain editor, as long as we save the code as .html we can view the results in the web-browser. But don't worry, there are more handy choices that we will look at in the Tools Section. Please note here, that we should always save the homepage of our websites as index.html. This is because web servers will by default look for an index.html page when users land on our websites - and not having one will cause big problems.

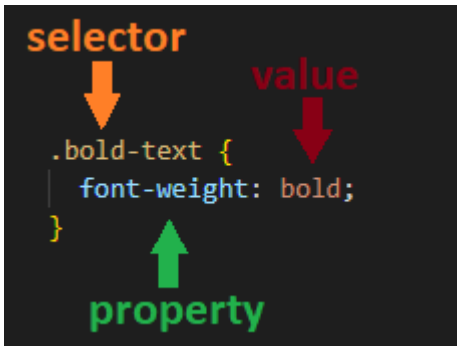
Interesting reads
- The most used HTML tags (8 million website analyzed)

2.3 CSS

Now we have a structure for the contents of a website, but to be frank: that looks horrible. The next step is to make that structure look good with some style, which is exactly what CSS, short for Cascading Style Sheets, is for. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

Syntax

The CSS Syntax is as straight forward as it was the case with HTML. It is a rule-based syntax and every rule is made up of a selector and a semi-colon separated list of property:value pairs.



Let's go over how selectors work so you can get a feel for fit. If the goal is to style all elements of the same type in a certain way, simply reference the tag.

```
h1 {
  color: black;
}
```

If you want to style multiple elements, that may be of different types, you use dot-notation and add a class to the element in HTML.

```
.bold-text {
  font-weight: bold
}
```

If you only want to style one element, you use the hashtag-notation and add an id to the element in HTML.

```
#header {
  font-size: x-large;
}
```

Common properties

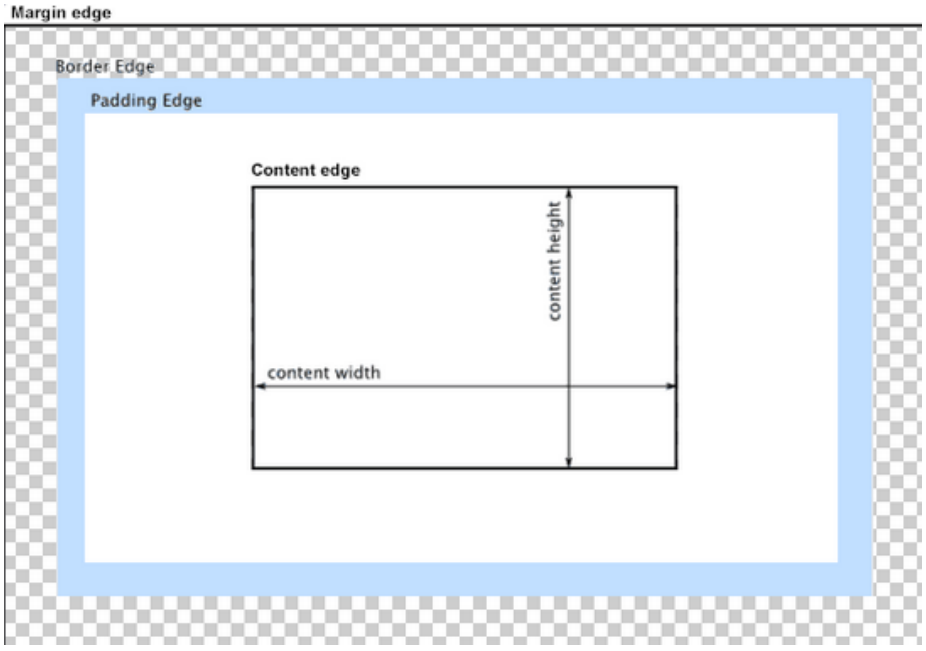
Now, let's check out the properties that you will encounter most often.

- `width`: specifies the width of the element's content area
- `height`: specifies the height of the element's content area
- `color`: defines the color of text, can be named, rgb(a) or hex value
- `background-color`: sets the background color

- `border`: specifies what kind of border to display
- `outline`: similar to border, but within the elements "box"
- `font-size`: Sets size of font (small, large..)
- `font-weight`: Sets weight of font (bold, italic..)
- `font-family`: Sets font style (arial, times…)
- `margin`: distance from the border of one element to the border of the other element
- `padding`: distance from border to the content of the element

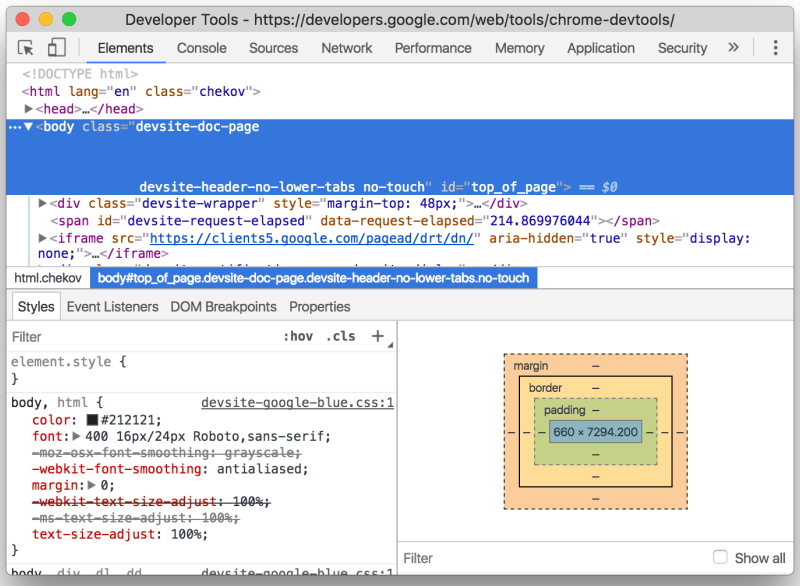
Box Model

CSS is based on the so-called box model: every element is a box. The content inside the element is a box, surrounded by the padding, surrounded by the border and surrounded by the margin.



This schematic can also be found in every modern web browser. Simply open your favorite website in a new window, right click and then select „inspect". This will open the „developer tools" and allow you to check out the content of a page. With these developer tools, you can also alter the content of a website

directly in the browser! However beware, these changes will not be saved and a reload will get rid of them.



Style.css

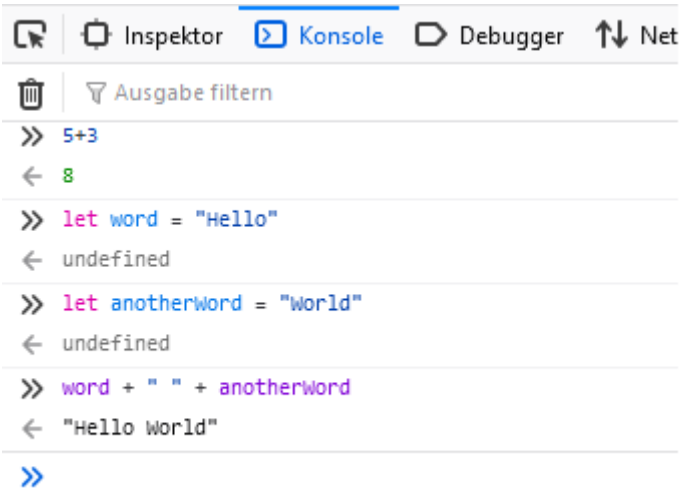
Great, now you know alot about CSS, but how can we combine it with our HTML code? There are several way's that you'll come across, but the most common way is to create a new file, by convention often named style.css, and put all our css code there. Afterwards we only have to link our created file in the head section of the html page and we are all set. For that to work, these files have to be in the same folder.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>...</body>
</html>
```

Interesting reads:
- The Box Model - test your box model skills with exercises
- The Box Model on MDN (check out the sidebar aswell!)

2.4 JavaScript

Let's dive in to JavaScript now! The JavaScript code that we will be writing during this semester will be run via the browser. As described in the last part on CSS, you can open the developer tools of your browser. There you'll also find the console. You can type any JavaScript command you like into the console, run it, and the browser will interpret the code for you. Let's try a few examples:



Code samples

Let's go over some more code. Feel free to type along in your console.

```
/* Variables */
let a = "hello"; /* Variables are simply storage containers */
a = "bye";       /* You use let to declare them if you expect the value to change */
const size = 10; /* You use const if you dont expect the value to change */

/* Primitive data types (boolean, string, number) */
let isHouseEmpty = false;             /* This is a boolean value and can either be true or false
let ourOrganisation = "Tech Academy"; /* This is a string value */
let participants = 95  ;              /* This is a number value */
```

```
/* Compound data types (arrays, objects) */
let myArray = [1,2,3,4,5,6,7,8,9,10];                    /* This is an array, consisting
let myObject = {name: "Sebastian", position: "Tutor"}; /* This is an object, to displa

/* Console.log */
console.log("Hello World!"); /* With console.log() we can print directly to the consol

/* If statement */
let grade = 1.7; /* We declare a variable "grade" and store the value 1.7 in it */
if (grade < 2.0) { /* Our code checks if grade is less than 2.0 */
  console.log("Wow, impressive!"); /* If that's the case 'Wow, impressive!' is the outp
}
else {
  console.log("Don't worry!"); /* If not, 'Dont't worry!' is the output */
}

/* For loop */
let myNumbers = [1,2,3,4,5];
for (let number of myNumbers) {
  console.log(number); /* Every number of myNumbers will be printed in a seperate line
}

/* While loop */
let a = 90;
while (a >= 50) {
  a -= 10;  /* Our variable a will decrement by 10 as long as a is larger than 50 */
}
```

The DOM

Now I bet you're asking how does this all link up to HTML and CSS? The DOM, or document object, is where it all comes together. The DOM is the most important object in JavaScript - it's the JavaScript representation of the structure of our HTML and is present in the variable document, when we load the page. Through the dom we can search through the document object,

```
const header = document.getElementById("header");
```

add event listeners, that specify an action everytime the element is clicked on

```
const button = document.getElementById("login-button");
button.addEventListener("click", () => {
  /* check if the user's credentials were valid */
})
```

and even change the style of our elements dynamically!

```
const button = document.getElementById("login-button");
button.style.color = "red";
```

Script.js

Obviously it doesn't make sense to keep going in the browser as none of the code will persist. Similar to CSS, the most common approach is to add a new file, often named script.js and to place a script tag at the end of our HTML code to link to that file. The updated boilerplate will look like this:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <!-- All the HTML code comes before the script tag -->
    <script src="script.js"></script>
  </body>
</html>
```

Want more practice? As a perfect supplement for the lessons in the Udemy course we recommend this website as it will allow you to practice JS hands on. After all, programming is not learned through tutorials, but by applying it yourself. We also highly encourage you to code along the instructor, Colt Steele, as this will further increase learning-speed.

Interesting reads:
- The History of JavaScript
- A (different) introduction to JavaScript
- Functions in JavaScript

Chapter 3

Tools

3.1 IDE

Now that you know the basics of Web Development you are ready to go and write some code!

But there is still a question left to answer: Where do programmers write their code?

As we have already pointed out, you could start programming in the default text editor of your operating system, however take our word when we say: Don't. You will regret it. Most developers use an Integrated Development Environment (IDE) to write code because it comes with a lot of handy features that help developers code faster and cleaner. Once you get used to an IDE, you will never look back. Let's look at some reasons they're so handy.

1. Graphical User Interface
The GUI of an IDE has been designed in a way that allows you to see your project directory, editor and terminal all at once. This makes it very easy to navigate between the different html, css and javascript files.

2. Autocomplete
If the IDE knows your programming language it can anticipate what you are going to type next and thus allow for faster development.

3. Syntax Highlighting
An IDE that knows the syntax of your programming language can provide visual cues. Keywords, words that have special meaning like if, for or while in JavaScript, are highlighted with different colors.

4. Debugging
No one can avoid to write code without errors. IDEs provide hints while coding to prevent errors before compilation and help you with debugging afterwards.

Some popular IDEs for Web Development are Visual Studio Code, Vim and Sublime Text 3. In the Udemy course your instructor will use Visual Studio Code. Feel free to check it out here.

For this course and your project, you will use the online code editor CodeSandbox. We will set up the environment together in the first Meetup. This IDE is based on the same platform as Visual Studio Code, but offers some additional functionalities. The user interface has been designed in a way that allows you to see your file directory, editor, and your executed program all at once. CodeSandbox allows you to create teams that can not only access the same files but also make it possible to have live coding session together. To sign up you only need a free Github account, which you can create in a matter of seconds.

3.2 Udemy

For this course you will be working with the The Web Development Bootcamp 2022 on Udemy.

This course is filled with a ton of content. For our project we are only focusing on the front-end so you can naturally ignore (for now) the later half of the course, which exclusively deals with the back-end. We encourage you to do all the exercises you will encounter in between. You are going to learn a lot by doing the exercises and programming on your project will become far simpler.

The following sections should leave you with enough knowledge to complete your project:

Section Name	Duration (h)
1. Course Orientation	0.5
2. An Introduction to Web Development	0.75
3. HTML: The Essentials	1
4. HTML: Next Steps & Semantics	1
5. HTML: Forms & Tables	1
6. CSS: The Very Basics	1
7. CSS: The World of CSS Selectors	1.25
8. The CSS Box Model	1
10. Responsive CSS & Flexbox	1.25
14. JavaScript Basics!	1
15. JavaScript String and More	0.75
16. JavaScript Decision Making	1.25
17. JavaScript Arrays	1
18. JavaScript Object Literals	0.5
19. JavaScript Repeating Stuff With Loops	1.5
20. NEW: Introducing Functions	0.5
22. Callbacks & Array Methods	1.5

Section Name	Duration (h)
24. Introducing The World Of The DOM	1.75
25. The Missing Piece: DOM Events	1.75
27. Async JavaScript: Oh Boy!	1.5
28. AJAX and APIs	1.5
29. Prototypes, Classes & OOP	1.5

3.3 API

For this semester project you will also need to learn about APIs. But what exactly is an API? An API (Application Programming Interface) allows different devices to connect to each other and send, update and request data from one to the other. You can think of it as a waiter in a restaurant. The waiter takes and understands your order, delivers it to the kitchen and comes back with the requested order. APIs function in a similar way.

How do we get the data?

Every API is unique but if you understand how to obtain output from one API you will know how to get data from all. For the most part, APIs are accessed through URLs, and the specifics of how to query these URLs change based on the specific service you are using. For example, the OpenWeatherMap API, that we will be using in our project, has several types of data that you can request. To get the current weather in London, type in the following URL into your browser.

```
api.openweathermap.org/data/2.5/weather?q=London
```

You will get an error message, prompting you to use an API key. Go ahead and create an account to obtain an API key from their free tier. Once the key has been activated, which can take up to 2 hours, try making a new request with the city of your choice AND the API key passed in as query string parameters, like the example above.

```
api.openweathermap.org/data/2.5/weather?q=London&APPID=YOUR_API_KEY_HERE
```

Congrats, you have made your first API request!

So how do we actually get the data from an API into our code?

In JavaScript we use the Fetch API to make our API call. You will learn more about how to use it in section 28 of the Udemy course which will also go over

JSON, the text format of the weather data.
To provide you with a template for later on and as a reminder, this is one way of how to deal with fetch:

```
async function exampleAPICall() {
  try {
    const response = await fetch(
      "https://api.openweathermap.org/data/2.5/weather?q=London&AAPID=YOUR_API_KEY_HERE
    );
    const data = await response.json();
    /* your code here to work with the data */
  } catch (error) {
    console.log(error);
  }
}
```

Chapter 4

Project

4.1 Project introduction

As mentioned before, this semester you will work on a weather website. Use everything you've learned in the Udemy course to create a weather forecast site using the weather API openweathermap.org, that we have looked into in the API Section. To complete this course, you are asked to meet the defined requirements. We will lay out the minimal requirements as well as give some inspiration for further development, make this project your own! Use this chapter as a rough guideline on how far you should progress everytime we meet up.

4.2 Minimal Requirements

1. Search Bar and Button

You should be able to look up any city name and by the press of a button/the enter key display the weather info of that city. Look into HTML forms!

2. API Call

In order to get the data for the weather we require you to use the Open Weather Map API. Check out their documentation here.

3. City Information

The name of the city, the current time, temperature and other city-specific information should be displayed in an appropriate section.

4. Forecast Information

Your website should have another section displaying a daily forecast for the next 7 days.

4.3 Curriculum

In the next months we will have 4+1 meetups. The following is intended to give you some direction on what we think would be a good working-pace for your project. We won't spoil much here, but instead give some hints and make you aware of problems you may encounter.

04.05.2022 - Web-Dev Coding Introduction

We will go over the basics of HTML and build your first website. Please remember to bring your own laptop!

18.05.2022 - 1. Coding Meetup

Udemy Course

You should have watched Section 1-5 by now. Optional: 6-8 and 10.

Action

1. Make a sketch, what your future weather website should look like and with annotations how some elements behave once you interact with them. If you need more inspiration, search for weather websites in the web. Think about what you would do differently and how these could be improved.

2. Set up a blank index.html document and try to model your sketch from Step 1 with HTML.
3. Try to group content that belongs together into sections. Dont use divs everywhere, remember this from the Udemy course and give your layout some meaning! Instead of getting weather data from the API, make up the data for a city of your choice.
4. If you've gotten to the CSS part of the course, create a style.css file and add some simple styling for the general layout (remember to link it within the HTML file).

01.06.2022 - 2. Coding Meetup

Udemy Course

You should have watched Section 6-8 and 10 by now.

Action

1. Now its time to apply your CSS knowledge. If you haven't already, add a style.css file and link it within the HTML document.
2. Before you can style elements, you will have to add selectors to your HTML file.
3. Remember Flexbox? Once you get some practice, you can easily control the layout of a page with it.

15.06.2022 - 3. Coding Meetup

Udemy Course

By now, you should be done with all our recommended sections. More specifically sections 14-19, 22, 24, 25 and 27-29.

Action

1. If you haven't already, go over the API Section now.
2. In the script.js file, write the functions that hit the API. You're going to want functions that can take a location and return the weather data for that location. For now, just console.log() the information.
3. Write the functions that process the JSON data you're getting from the API and return an object with only the data you require for your app.
4. Connect the search bar you've created to the function. Still just console.log() the result for now.
5. Now that you have a pretty good grip on the API, remove the "fake data" we added before and display the weather information on your website!

29.06.2022 - 4. Coding Meetup

Udemy Course

By now, you should be done with all our recommended sections.

Action

1. Finish up any left-over tasks from before.
2. Make this project your own! There are many additional features you can add. Here are some ideas:

- Check out Font Awesome
- Toggle options between Celcius/Fahrenheit
- Change background based on weather/time
- Add a search history with the Local storage API
- Ability to hide the forecast/search history with keyframes
- When loading the page, display the weather for the user's location if he gives permission. Use the Geolocation API
- Surprise us! Be creative and truly make this project your own!

Chapter 5

Additional Resources

5.1 Git and GitHub

What's Git? Git is like an epic save button for all your code, files and directories. In fancy language it's also called a version control system.
Why is the save so epic? Well, a "normal" save, for example in a word file, records all of the words in that file at the moment you hit the save button. You are only ever given one record of the file, such as essay.doc, unless you make duplicate copies (which is difficult to remember to do and keep track of).
However, a save in Git records differences in the files and folders AND keeps a historical record of each save. This let's you review how your project grows and easily go back to any state from the past. If you connect it to a network, such as GitHub, it allows for collaboration among work colleagues, open-source projects and showcases your skills to the world!
We encourage you to sign up to github here and download git here as well as check out this video for the ultimate git crash course.

5.2 JSON

JSON (JavaScript Object Notation) is a text format for storing and transporting data. It's based on the notation of objects in JavaScript and easy to understand. We will deal with JSON data when we fetch from the openweather API in this course. As everything, it's pretty simple to understand once we get into the doing.

```
{
   "book": [
```

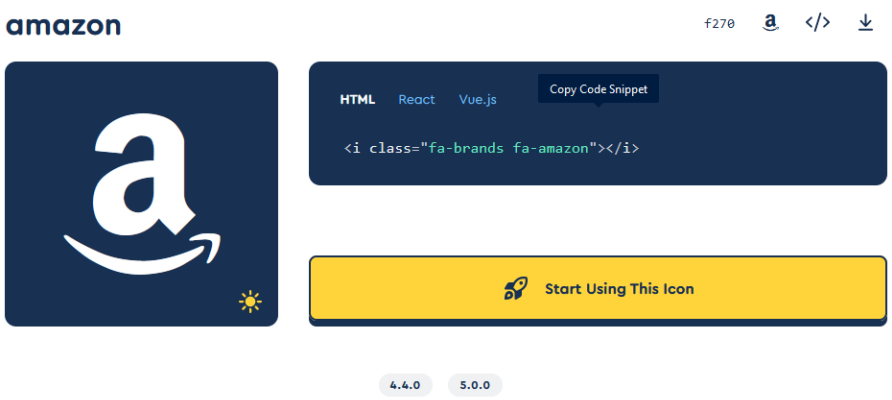
```
      {
         "id":"01",
         "title": "HTML5 and CSS3",
         "edition": 3,
         "author": "Jürgen Wolf",
         "read": true,
      },

      {
         "id":"02",
         "title": "JavaScript &JQuery",
         "edition": 2,
         "author": "Jon Duckett",
         "read": false,
      }
   ]
}
```

As you can see, data is represented by comma-separated name:value pairs. Every curly brace holds an object, and if you want to bundle multiple objects together you can use arrays, that are represented by square brackets and separated by comma. The value can hold different data types, in the example above and most commonly used are strings, numbers, boolean values or other objects. To extract the information stored in the JSON data, we can access the values much like with JavaScript objects. Let's assume that we have stored the example-json in a variable called data, thus:

```
let firstId = data.book[0].id /*This would yield 01 */
let jsAuthor = data.book[1].author /* This would result in John Duckett */
```

5.3 Font Awesome

Font Awesome has a library of awesome icons, that are simply integrated in your project and can make quite the difference optically. Take a look here to check out the icons. With a click on any of the icons you get shown the respective html code, that you can just copy and paste into your file.



There's just one more quick thing to do before you can use them in your project, you have to register using this link and add your personal script-tag, that you will get after the registration in the head section of your html file.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
    <link rel="stylesheet" href="style.css" />
    <script
      src="https://kit.fontawesome.com/YOUR_CODE.js"
      crossorigin="anonymous"
    ></script>
  </head>
  <body>
    <!-- All the HTML code comes before the script tag -->
    <script src="script.js"></script>
  </body>
</html>
```