“*The Internet is just a passing fad.* “

* Bill Gates 1995

Inhalt

[1 Einleitung 3](#_Toc152765830)

[2 Mentoren suche 4](#_Toc152765831)

[3 Theoretischer Teil 5](#_Toc152765832)

[3.1 Was ist Arduino 5](#_Toc152765833)

[3.1.1 Hardware 5](#_Toc152765834)

[3.1.2 Verschiedene Arduino Hardware. 5](#_Toc152765835)

[3.1.3 Software 6](#_Toc152765836)

[3.2 Wie benutze ich Arduino 7](#_Toc152765837)

[3.2.1 Wie funktioniert die Programmierung der Hardware 8](#_Toc152765838)

[3.2.2 Wie programmiere ich die Hardware 8](#_Toc152765839)

[3.3 Microcontroller 11](#_Toc152765840)

[3.3.1 Wie funktioniert ein Microcontroller 11](#_Toc152765841)

[3.3.2 Beispiele 11](#_Toc152765842)

[3.4 Wie baut man eine Website 12](#_Toc152765843)

[3.4.1 HTML (Hyper Text Markup Language) 12](#_Toc152765844)

[3.4.2 DOM (Document Object Model) 13](#_Toc152765845)

[3.4.3 CSS 14](#_Toc152765846)

[3.4.4 JavaScript 16](#_Toc152765847)

[3.4.5 Was sind Frameworks? 16](#_Toc152765848)

[3.5 Geschichte von Arduino 17](#_Toc152765849)

[3.6 Geschichte von HTML/CSS 17](#_Toc152765850)

[3.7 Geschichte von JavaScript 17](#_Toc152765851)

[3.7.1 Geschichte von Node.js 17](#_Toc152765852)

[3.7.2 ES6-ES13 17](#_Toc152765853)

[3.8 OOP (Objekt Orientierte Programmierung) 17](#_Toc152765854)

[4 Praktischer Teil 18](#_Toc152765855)

[4.1 Mein Coding-Partner Benno 18](#_Toc152765856)

[4.2 Was habe ich benutzt (Programmiersprachen, Frameworks etc.) 18](#_Toc152765857)

[4.3 Planung 18](#_Toc152765858)

[4.3.1 PAP (Programm Ablauf Plan) 18](#_Toc152765859)

[4.3.2 Neu Anfang 18](#_Toc152765860)

[4.4 Anfänge mit Board und Powerbase 18](#_Toc152765861)

[4.4.1 Wie funktioniert das Board 18](#_Toc152765862)

[4.4.2 Wie funktioniert die Powerbase 18](#_Toc152765863)

[4.5 Einzelne Components 18](#_Toc152765864)

[4.6 Text Editor 18](#_Toc152765865)

[4.6.1 Was ist eine IDE? 18](#_Toc152765866)

[4.6.2 IDE != Texteditor 18](#_Toc152765867)

[4.7 Debugging 18](#_Toc152765868)

[5 Fazit 18](#_Toc152765869)

[6 Quellen 18](#_Toc152765870)

[7 Abbildungsverzeichnis 19](#_Toc152765871)

# Einleitung

Anfangs hatte ich eigentlich vor etwas mit Musik zu machen, jedoch wusste ich nicht genau was. Als ich einen Freund gefragt hab hat er gemeint: „Wie wäre es mit einem Arduino in einer Website!“. Ich war von dieser Idee so begeistert das ich es direkt als Große Arbeit machen wollte, was aber „Arduino“ ist, erkläre ich in dieser Arbeit. Und das Internet stirbt nie, selbst wenn Bill Gates es prophezeit.

# Mentoren suche

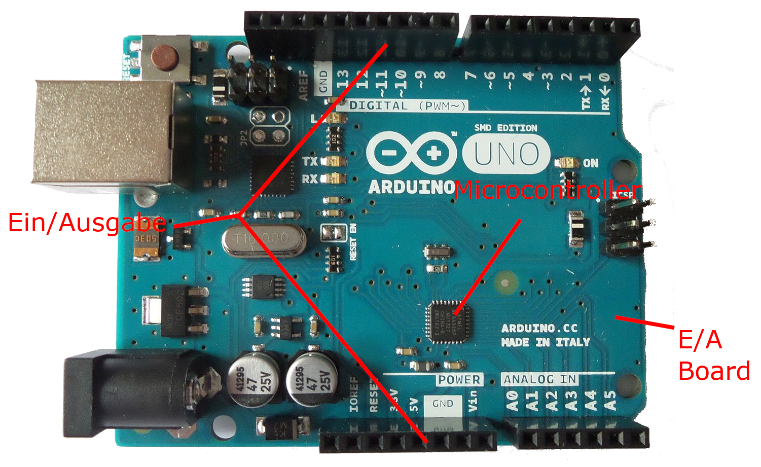
Für mich war von Anfang an klar das Jakob Ohage mein Mentor wird. Zugegebener Weise vor allem, weil er der Freund meiner Schwester ist und damit auch die Kommunikation einfacher wird, aber auch weil er meiner Meinung nach ein sehr guter Informatiker ist, da er auch ein Medien-Pädagoge ist und es für Ihn wichtig ist das junge Leute programmieren lernen.

# Theoretischer Teil

## Was ist Arduino

Arduino ist eine Physical-Computing-Plattform. „Physical-Computing“ ist ein Physisches System welches aus Hard- und Software besteht. Zusammen gefasst hast du ein Programm auf deinem Computer und irgendeine Form von Hardware (zum Beispiel ein Mini Computer). Mit diesen zwei Komponenten kannst du nun zum Beispiel Lämpchen leuchten lassen. Es ist aber auch möglich komplexere Sachen wie eine Gesichts Erkennung zu bauen, es gibt sogar Videos, in denen Leute „Smart-Glasses“ mit Arduino bauen.

### Hardware

Die Hardware kann sehr verschieden sein, es gibt Hardware speziell für Arduino angepasst, oder auch Hardware die einfach mit Arduino kompatibel ist. Eine Arduino kompatible Hardware besteht aus einem [Microcontroller](#_Microcontroller) einem E/A Board und Ein- und Ausgängen. Das klingt zwar recht kompliziert ist aber recht simpel. Ein E/A Board ist einfach nur eine Platine die Eingabe und Ausgabe (Input/Output) verwaltet, dadurch kann die Hardware mit anderen Komponenten und der Software arbeiten. Die Ein- und Ausgänge sind hier für Sachen wie Kabel oder um andere Hardware anzuschließen. Was genau ein Microcontroller ist erkläre ich später, aber kurzgesagt ist ein Microcontroller einfach ein Mini-Computer.

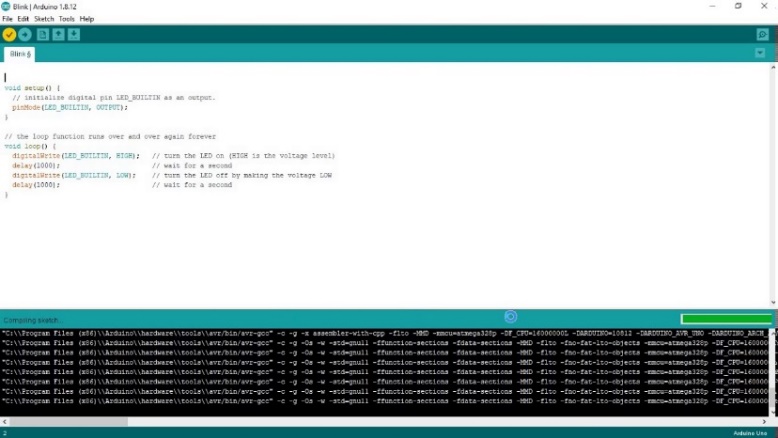
### Verschiedene Arduino Hardware.

|  |  |
| --- | --- |
| **Arduino Uno**  Die Arduino Uno ist die meist genutzte Arduino Hardware und wird auch von Arduino für die Privat-Nutzung empfohlen. |  |
| **Arduino Uno WiFi**  Ermöglicht es dir auch ein aufs Internet zuzugreifen, so etwas könnte man zum Beispiel für GPS brauchen. Der Preis ist dementsprechend teurer. | ARDUINO UNO WiFi REV2 |
| **Arduino Micro**  Die Micro Version ist sehr viel kleiner, hat aber ähnliche Möglichkeiten. Lustiger Weise ist die Micro Version nicht wirklich billiger | Arduino Micro |
| **Raspberry PI**  Es ist auch möglich eine PI als Arduino Hardware zu benutzen. Das ist auch meine persönliche Empfehlung, da eine PI auch für z.B. System Administration genutzt werden kann. | RASPBERRY PI 3B+: Raspberry Pi 3 B+, 4 x 1,4 GHz, 1 Go RAM, WiFi, BT ... |
| **Zusatz Hardware**  Für bestimmte Funktionen benötigt man Zusatz Hardware. Z.B. die Nano, die es ermöglicht Sensoren und Bluetooth zu benutzen! Hier gibt es auch wieder verschiedene Versionen. | Arduino Nano |

### Software

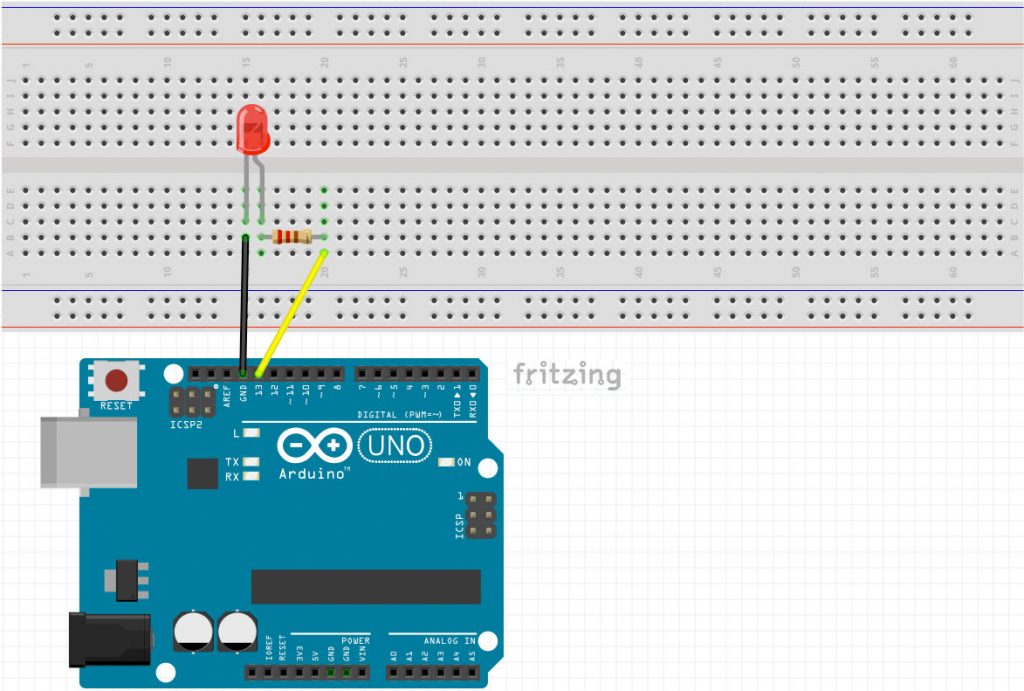
Die Software wird benötigt um die Hardware zu programmieren. Sie erlaubt es dir dann zum Beispiel, dass ein LED leuchtet. Um etwas auf die Hardware zu programmieren, benötigst du 2 Dinge: Ein Text Editor oder eine IDE und ein Kabel um dich mit der Hardware zu verbinden.

### Was ist eine IDE

Ein Integrated Development Environment (IDE) ist ein Programm in dem du programmieren kannst. Nicht zu verwechseln mit einem Text Editor. Ein Text Editor ist einfach nur ein Programm in dem ich Text Schreiben kann, z.B. Word, Notepad++. Das heißt ich kann mein Arduino Code auch in Word schreiben (wenn ich mich selbst foltern will). Eine IDE ist ein Programm was speziell fürs Programmieren angepasst wird. Zum Beispiel programmiere ich gerne in der „Programmiersprache“ C, ich könnte nun mit einem Text Editor wie Visual Studio Code arbeiten (ein Text Editor fürs Programmieren), aber ich könnte auch CLion verwenden. Ein Programm was nur auf C abgeschnitten ist, dadurch wird das Arbeiten leichter (in dem Fall ist es vermutlich schlauer Visual Studio Code zu benutzen, da CLion eine Lizenz braucht die mindestens 250€ pro Jahr kostet).

## Wie benutze ich Arduino

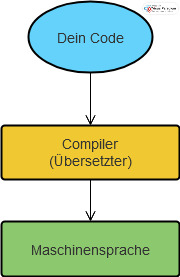
Mit der Hardware meiner Wahl und alles vorbereitet, kann ich jetzt eine kleine LED (Light Emitting Diode = Bauelement, das Licht ausstrahlt) leuchten lassen. Dafür brauche ich aber noch zwei Sachen: Offensichtlich ein LED und ein Steckbrett um mein LED und meine Kabel anzustecken.

Ich kann dann meine Hardware so aufbauen:  


Wichtig ist das es nicht notwendig ist es so aufzubauen, das ist nur ein Beispiel. Jedes Kabel gibt Strom, die „Strompins“ sind grün gekennzeichnet der Gelbe Bob ist ein Wiederstand, er sorgt dafür, dass die Stromspanne verringert wird und ein Wiederstand kann theoretisch auch den Strom leiten. Ein LED hat 2 Beine (wow!), beide repräsentieren einen Pol. Plus und Minus Pol, das wenn man sich anschaut wo die Kabel angeschlossen sind, dann sieht man, dass das gelbe Kabel am Board an einem Stecker mit der Aufschrift „13“ steckt und das schwarze Kabel am Board an einem Stecker mit der Aufschrift „GND“ steckt. Es ist auch wichtig das es genauso ist, denn die Stecker am Board repräsentieren auch „Pole“. Blöd gesagt Plus und Minus muss zu Minus. Ein „Minus Pol“ am Board ist einfach nur ein Stecker mit der Aufschrift „GND“, ein „Plus Pol“ am Board ist ein Stecker mit einer Zahl drauf (1-13), das liegt daran das man mit dem Plus pol später auch in der Software arbeitet.

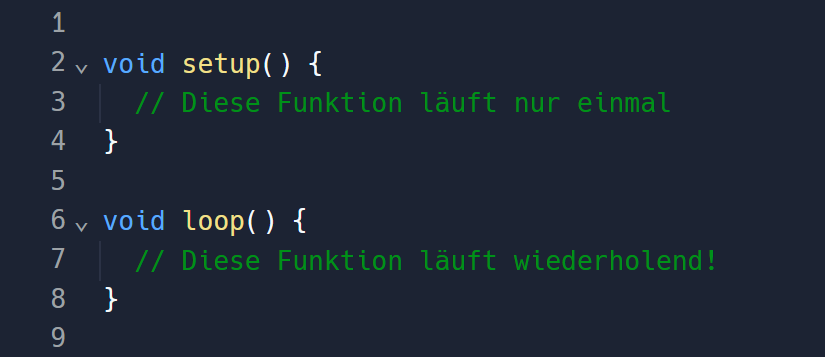
### Wie funktioniert die Programmierung der Hardware

Programmieren wird in Filmen und Serien zwar immer sehr kryptisch dargestellt, aber in Wirklichkeit ist Programmieren nur eine Anreihung von Schlagwörtern die dann in „Maschinensprache“ übersetzt werden. Maschinensprache beschreiben die „Einsen und Nullen“ die oft mit Programmieren assoziiert werden, in Wirklichkeit wird seit 1964 (Einführung der ersten Software die man als „Programmiersprache“ beschreiben kann) nicht mehr mit Maschinensprache gearbeitet. Die Programmiersprache die mit Arduino arbeitet basiert, oder eigentlich ist, ist C++. Zum verstehendes: C++ ist eine „low-level Sprache“, dass bedeutet das es für C++ nur wenige Schritte braucht um in Maschinensprache zu übersetzen. Allerdings wird eigentlich nur die „Arduino Language“ benutzt, oder C.



### Wie programmiere ich die Hardware

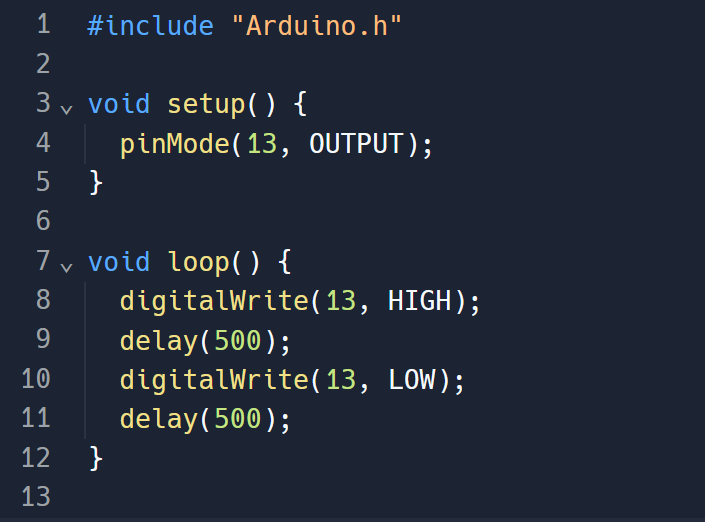
Um zu starten braucht man zwei „Funktionen“. Funktionen sind dafür da das Code strukturiert wird, Funktionen werden auch als Art „start-hilfe“ benötigt. Das heißt dein Programm braucht bestimmte Funktionen das es läuft. In Fall von Arduino sind es die „setup“ und „loop“ Funktionen. Die Setup Funktion wird beim Start des Programmes einmal ausgeführt, die loop Funktion hingegen wiederholt sich nach dem Start dauerhaft. In der Programmierung würde das dann so aussehen:



Ok, das sieht kryptisch aus, aber es ist eigentlich ganz einfach „void“ ist ein „Datentyp“, Daten können verschiedene Formen annehmen, zum Beispiel eine ganze Zahl (ein Integer), eine Zeichenkette (ein String) oder auch Wahrheitswerte (ein Boolean). Der void Datentyp ist nichts, der void Datentyp ist hat keine Eigenschaft und kann auch nicht etwas speichern. Da wäre nun die Frage: „Was bringt mir dann dieser Datentyp?“. Ganz einfach: normaler Weise, wenn du einer Funktion ein Datentyp übergibst musst du irgendetwas zurückgeben. Also z.B. du hast eine Integer Funktion, dann musst du eine Ganzzahl zurückgeben, was ich dann mit dieser Ganzzahl mache, ist mir überlassen. Im Fall von void ist nichts zum Zurückgeben da und daher auch nicht Notwendig.

*Wenn du etwas zurück geben wollen würdest dann machst du das mit dem „return“ Stichwort.*

Nachdem man der Funktion den Datentyp gegeben hat, kommt der Name der Funktion und dann zwei Klammern, das ist wichtig, weil es die Funktion zur Funktion macht, Sie ist für Parameter da, also eine Art Weiterführung von Daten (also nicht die Parameter aus dem Mathe Unterricht). Dann kommen die geschweiften Klammern die dafür da sind, dass die Arduino Language erkennt wann die Funktion endet und wann anfängt. Die zwei Slashes in der Funktion sind Kommentare, diese sind für Strukturierung da und werden von dem Programm ignoriert, das ist wichtig falls mal andere Leute deinen Code sehen wollen und davor nicht erst mal sich 2 Stunden damit beschäftigen müssen was die Zeile denn jetzt genau macht.

Nehmen wir jetzt unser aufgebautes Beispiel von vorher, wir versuchen jetzt das LED zum Leuchten zu bringen. Dafür müssen wir wieder zwei Sachen machen, wir müssen das LED für Arduino konfigurieren und anschließend dauerhaft das LED zum Leuchten bringen. Warum dauerhaft? Weil wir direkt ein Blink Effekt Programmieren, das heißt das LED geht an und nach einer Zeit wieder aus und dass für immer Fortlaufend. Im Code würde das dann so aussehen:

In Setup ist die „pinMode“ Funktion, warum eine Funktion? So funktionieren Funktionen auch. Du schreibst sie dir ähnlich wie die loop und setup Funktionen (mit anderen Namen natürlich) und dann kannst du sie per *nameDerFunktion()*aufrufen. In unserem Fall müssen wir sogar der Funktion noch 2 „Parameter“ übergeben. Parameter sind Daten, die wir in diese Klammern reinschreiben die Funktion kann dann mit diesen Daten arbeiten. *pinMode* nimmt zwei Parameter der Pin und ob der Pin ein Input oder ein Output ist. Der Pin ist die Zahl auf dem unser „Plus-Kabel“ angesteckt ist. Output ist so etwas wie ein LED oder ein Lautsprecher ein Input wäre so etwas wie ein Knopf oder ein Mikrofon. Wir haben unser Plus-Kabel an der Stelle 13 konfiguriert und ein LED ist ein Output. Jetzt schreiben wir in unsere loop Funktion „*digitalWrite*()“. Das digital ist egal wichtig ist das *write*, dadurch wird auf den Pin zugegriffen und mit „HIGH“ oder „LOW“ wird die Stromstärke eingestellt. HIGH sind 5V. Daher auch der Wiederstand bei 5V würde vermutlich die LED nach einer Zeit einen Kurzschluss kriegen.

Die delay() Funktion macht immer Pausen zur nächsten Operation, sie nimmt die Dauer der Zeit als Millisekunden in den Parameter.

Das Programm sollte nun die LED leuchten lassen, 500 Millisekunden (eine halbe Sekunde) wartet dann wieder das LED ausschaltet, wieder wartet und dass für immer vorlaufend.

## Microcontroller

### Wie funktioniert ein Microcontroller

### Beispiele

#### Raspberry PI

#### Eigener Computer

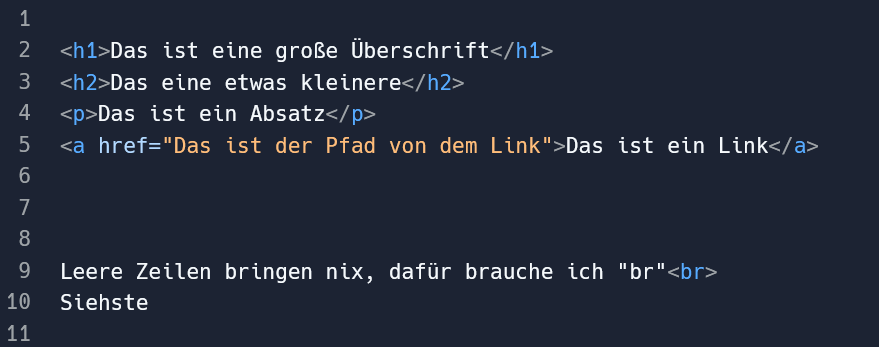
## Wie baut man eine Website

Eine Website zu bauen klingt zwar wie ein großer Schritt von einem Lämpchen blinken zu lassen, in Wirklichkeit ist es aber andersrum. Eine Website besteht hauptsächlich aus HTML/CSS und optional JavaScript. HTML/CSS sind keine Programmiersprachen, sondern *Markup Languages.* JavaScript ist tatsächlich eine Programmiersprache, zählt aber zu den einfacheren.

Eine Website zu bauen ist ähnlich wie ein Haus zu bauen. Du braust erst irgendein Grundgerüst, dein HTML, danach muss du dein Haus verkleiden, dein CSS. Jetzt hättest du theoretisch schon ein Haus, das Haus ist nur ein Haus aus dem Mittelalter, es hat keine Glühlampen oder Heizungen, dass alles macht dann JavaScript (Die Logik der Website). Jetzt gibt es noch viele andere komplizierte Schritte die man machen kann, aber das muss man alles nicht verstehen.

### HTML (Hyper Text Markup Language)

*Hyper Text Markup Language* ist wie man im Namen sieht eine “Markup Language”, dass bedeutet das es keine Programmiersprache ist, sondern einfach eine Art Text der in Typen unterteilt wird. Daher braucht es auch keine „Tokens“ (Stichwörter) zum auswendig lernen, sondern „Tags“. Z.B. gibt es den Tag „<a>“ man öffnet einen Tag immer mit einem „<“ und schließt ihn immer mit einem „>“, Alles was ich danach schreibe ist vom Typ „a“ (einen Link) bis ich den Tag wieder schließe, dass mache ich mit: </a>, quasi der gleiche Tag nur, dass ich ein „/“ vor dem Typen setze.

Wenn man nun eine Website nur mit programmiert, sieht das so aus:

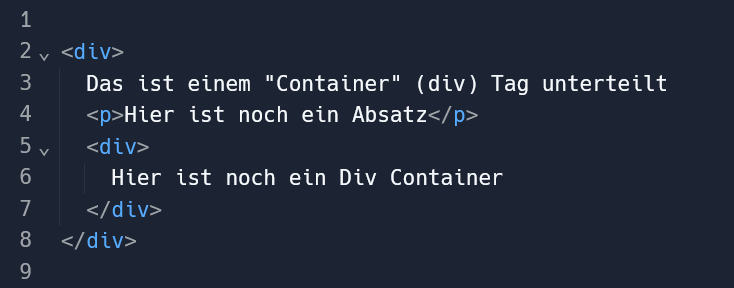
Die Seite sieht dann so aus:

Wie man sieht ist der Code sehr verschieden zu dem von dem Arduino Programm. Es gibt auch Tags wie „br“ die keine Schließung benötigen, einfach da man ja in einen Text Umbruch nichts reinschreiben kann. Das Ding ist man kann auch Tags an Tags unterordnen, dafür sorgt das DOM.

### DOM (Document Object Model)

Das DOM sorgt dafür das ich Tags untereinander anordnen kann, quasi wie eine Hierarchie, dass bringt z.B. das man einem Tag sagt, dass alle Tags unter ihm die Schriftgröße 24 haben. Oder auch für Formatierung Zwecke zum Beispiel der „<nav>“ Tag, „nav“ steht für „Navigator Bar“, das sind diese oberen Zeilen die man auf Webseiten immer sieht.

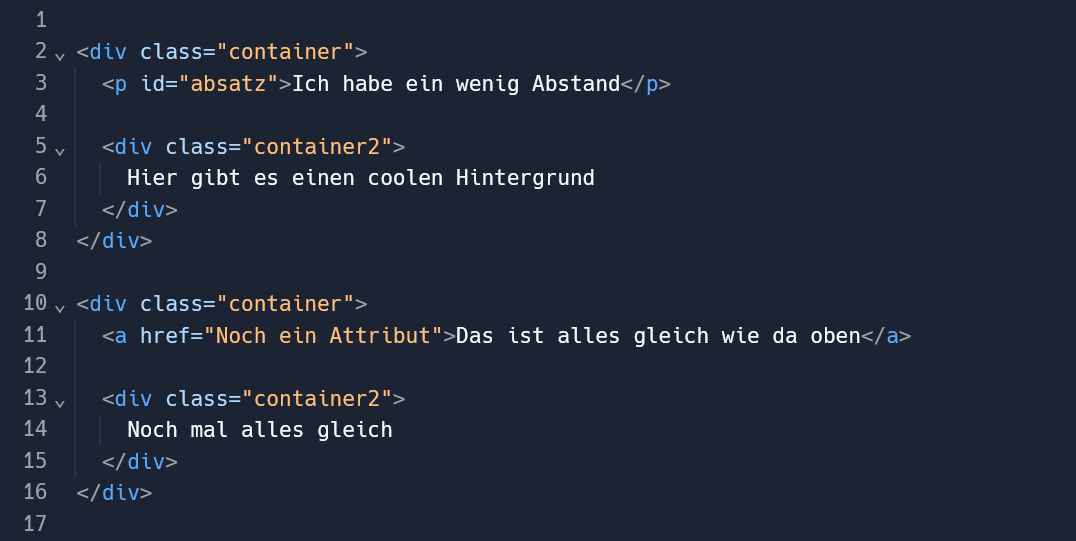
Das Programmieren so einer Unterordnung von Tags ist relativ einfach:



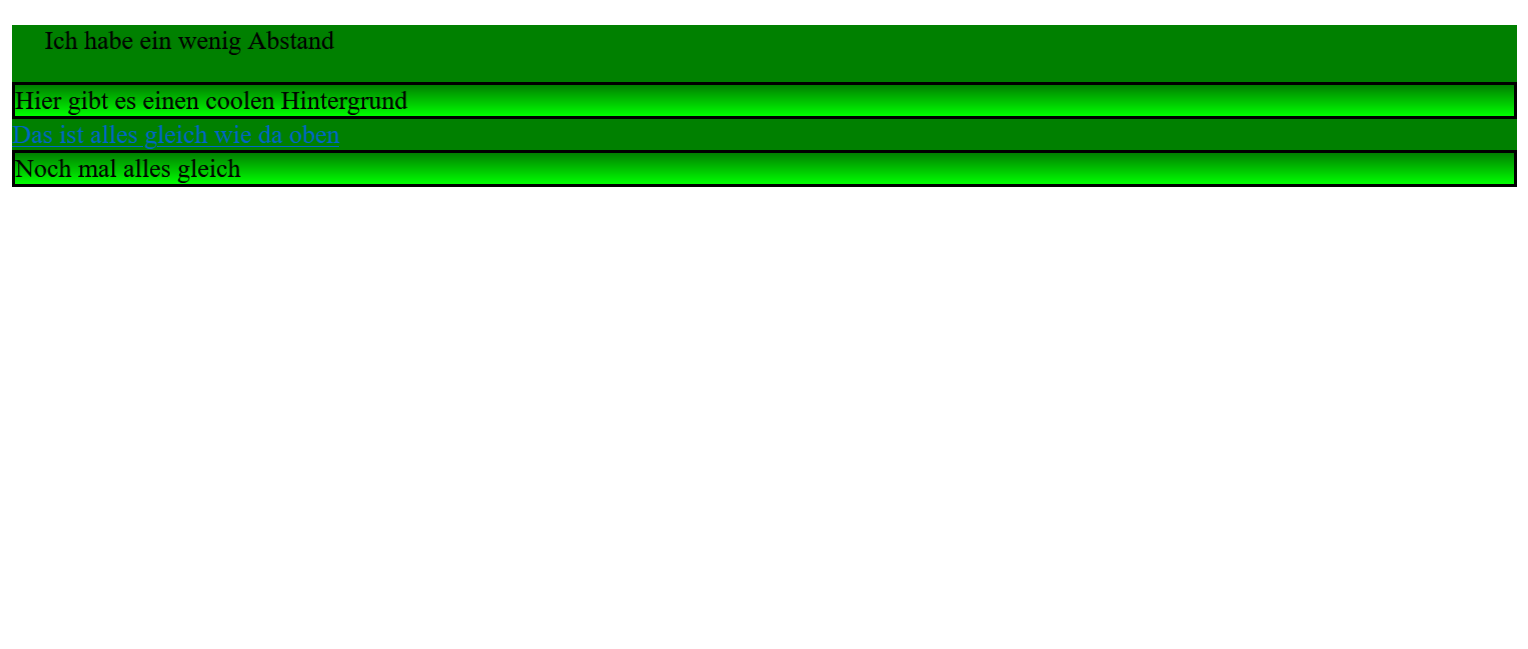
Wie man sieht öffnet man den Div Container schreibt alles rein und schließt ihn wieder ein paar Zeilen später. Diese Abstände von dem inneren des Div Containers sind nicht notwendig, aber empfehlenswert, da es sonst sehr schnell schwer zu Lesen wird.

### CSS

Die Website ist nun ziemlich unschön, um das zu lösen benutzt man CSS. Bevor man das machen kann muss man „Attribute“ verstehen. Ein Tag kann verschiedene Attribute haben, der „<a>“ Tag, hatte sogar schon ein Attribut, dass „href“ Attribut. Damit entweder HTML was damit anfangen oder der CSS Code. Jeder Tag kann einen „class“ oder „id“ Attribut haben. Damit kann man dann mit den CSS Code arbeiten.

So würde dann eine HTML/CSS Seite aussehen:

Das CSS würde dann so aussehen:

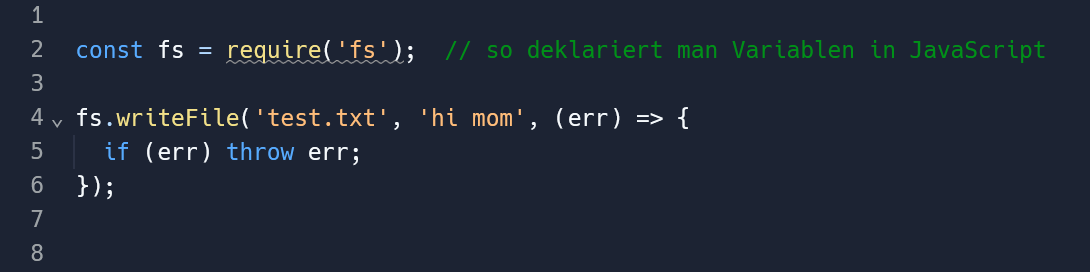
Die Seite würde dann so aussehen:

Im CSS Code wird immer ein Punkt oder ein Kreuzchen benutzt um auf die class oder id Attribute zuzugreifen. Das Punkt und den Namen der Klasse greift man auf alle Tags mit dem class Attribut mit dem passenden Namen zu, das Kreuzchen macht das gleiche nur mit IDs. Hier gilt wieder das Konzept mit den Geschwungen Klammern. Die ganzen Begriffe muss man auswendig lernen, aber zum Großteil sind sie reicht selbsterklärend.

### JavaScript

JavaScript ist die Sprache die die Logik implementiert. Man benötigt ebenfalls die Attribute um mit JavaScript zu arbeiten. Der Code sieht sehr verscheiden zu der Arduino Language aus, dass liegt daran das JavaScript eine Dynamische Sprache ist, dass bedeutet, dass JavaScript man keine Datentypen, wie void, braucht, sondern Sie automatisch kategorisiert werden. Das hat den Vorteil, dass es einfacher wird, der Nachteil ist das das Programm langsamer wird. JavaScript benötigt auch keine Start-Hilfe ich kann einfach loslegen. Der Syntax (Syntax = Wie man programmiert) ist zwar anderes, aber würde den Rahmen sprengen.

#### Node.js

Node.js ist eine „Laufzeitumgebung“, das heißt das node.js ermöglicht viele andere Tools zur Verfügung zu stellen, zum Beispiel „Fes“ (File System), damit kann man auf Dateien im Computer arbeiten (was teilweise gegen die EU-Richtlinien verstoßen kann). Wenn ich node.js konfiguriert habe kann ich in meinen JavaScript Code (zum Beispiel) folgendes schreiben: 

Man erstellt eine Variable und weist ihr „fs“ zu. *require()* wird benötigt um auf die node.js „Modules“ zuzugreifen. Jetzt benutzt man fs*.writeFile()* das Punkt unterscheidet das man nun eine Art Unterfunktion haben will, da Funktionen auch eigene Funktionen haben können. *writeFile()* braucht drei Parameter, wie die von mir angelegte Datei heißen soll, was der Inhalt sein soll, und eine „Exception“, blöd gesagt einfach nur wenn der Computer bemerkt das ein Fehler auftritt das er das Programm beendet mit einer Fehlermeldung.

#### Bun: das bessere Node.js

Bun wird vermutlich in der Zukunft benutzt da es sehr angenehm zu benutzen ist und jetzt schon viele Leute die Jahrelang node.js benutzt haben auf Bun wechseln. Bun wurde in dieser Arbeit trotzdem nicht verwendet, da Bun zu dem Start der Arbeit noch nicht wirklich existiert hat.

### Was sind Frameworks?

Frameworks sind Erweiterungen für Programmiersprachen, zum Beispiel ermöglicht das JavaScript-Framework „WebGL“ 3D Grafiken weitaus einfacher zu implementieren.

## Geschichte von Arduino

## Geschichte von HTML/CSS

## Geschichte von JavaScript

### Geschichte von Node.js

### ES6-ES13

## OOP (Objekt Orientierte Programmierung)

Ein wichtiges Programmierkonzept ist die Objekt Orientierte Programmierung, sie sorgt dafür, dass eine Variable auch ein Objekt von einer bestimmten Klasse sein kann. Das bedeutet das ich einer Variable ein Objekt zuweisen kann und nur aufgrund seiner Klasse hat es automatische zahlreiche Eigenschaften.

Der Vorteil ist das es sehr Menschennah ist, also eine sehr einfache Lehrkurve hat, zudem wird das „debugging“ (debugging = Fehler ausmerzen) einfacher. Der Nachteil ist das OOP auch oft unübersichtlich werden kann, deswegen ist Sorgfalt bei der Objekt Orientierten Programmierung sehr wichtig.

# Praktischer Teil

## Mein Coding-Partner Benno

## Was habe ich benutzt (Programmiersprachen, Frameworks etc.)

|  |  |
| --- | --- |
| **HTML**  Für den Grundaufbau der Website | Html Logo Png Transparent Background , Free Transparent Clipart ... |
| **CSS**  Für das verschönern der Website | Logo de CSS: la historia y el significado del logotipo, la marca y el ... |
| **JavaScript**  Für die Logik der Website | JavaScript logo and symbol, meaning, history, PNG |
| **Node.js**  Für Zahlreiche Features | Node Js Logo Png | Images and Photos finder |
| **Visual Studio Code**  Als Text-Editor | Visual Studio Code Logo PNG Transparent & SVG Vector - Freebie Supply |

## Planung

### PAP (Programm Ablauf Plan)

### Neu Anfang

## Anfänge mit Board und Powerbase

### Wie funktioniert das Board

### Wie funktioniert die Powerbase

## Einzelne Components

## Text Editor

### Was ist eine IDE?

### IDE != Texteditor

## Debugging

# Fazit

# Quellen

<https://www.arduino.cc/>

<https://de.wikipedia.org/wiki/Physical_Computing>

<https://programmingwiki.de/Objektorientierung_in_JavaScript/Weiterf%C3%BChrendes/Vor-_und_Nachteile_der_OOP>

# Abbildungsverzeichnis