Wissensfabrik
Mehr Wissen. Mehr Können. Mehr Zukunft.

# IT2School

**Gemeinsam IT entdecken**


# Modul E4 – Webseiten

## Erstellung von Webseiten

Eine Entwicklung von

OFFIS

In Kooperation mit

Carl von Ossietzky Universität Oldenburg

Im Auftrag der

Wissensfabrik
Mehr Wissen. Mehr Können. Mehr Zukunft.

# Inhalt

# 1 Webseiten

In diesem Modul lernen die Schülerinnen und Schüler *Markup-Languages* (ML) – im Deutschen auch als Auszeichnungs- oder Beschreibungssprachen[1] bezeichnet – exemplarisch an der *Hypertext Markup Language* (HTML) kennen, die das Gerüst von Webseiten ist. Dazu nutzen sie Online-Editoren, um in projektorientiertem Unterricht eigene einfache Projekte zu entwickeln.

Während der Nutzung der in diesem Erweiterungsmodul vorgestellten Online-Editoren[2] (Abs. 4.5) steht den Schülerinnen und Schülern eine Live-Vorschau der von ihnen erarbeiteten HTML-Seiten zur Verfügung. Dies bietet den Vorteil, dass sie nicht nur unter Anleitung, sondern auch experimentativ vorgehen können.

| | | |
|---|---|---|
| **Lernfeld/Cluster:** | IT spielend entdecken | |
| **Zielgruppe/Klassenstufe:** | | 4. bis 5. Klasse |
| | **X** | 6. bis 7. Klasse |
| | **X** | 8. bis 10. Klasse |
| | **X** | 11. bis 12. Klasse |
| **Geschätzter Zeitaufwand:** | 8 – 10 Stunden | |
| **Lernziele:** | • Kennenlernen von Markup-Languages<br>• Darstellen von Informationen mittels Markup-Languages<br>• Kennenlernen und Verwendung von Cascading Style Sheets (CSS)<br>• Kennenlernen von JavaScript als Scriptsprache zur clientseitigen Programmierung<br>• Programmierung einfacher Programme mittels JavaScript | |
| **Vorkenntnisse der Schülerinnen und Schüler:** | Keine | |
| **Vorkenntnisse der/des Lehrenden:** | Keine | |
| **Vorkenntnisse der Unternehmensvertreterin/des Unternehmensvertreters:** | Keine | |
| **Sonstige Voraussetzungen:** | Keine | |

[1] Auszeichnungs- bzw. Beschreibungssprachen unterscheiden sich von Programmiersprachen dahingehend, als dass sich mit letzteren Scripts oder Programme erzeugen lassen, mit ersteren hingegen Dokumente.

[2] Namensgebend für dieses Erweiterungsmodul war bei der Einführung von IT2School der einfach zu bedienende Online-Editor zur Erstellung von Webseiten – *Thimble* (engl. für „Fingerhut“). Zwar wird das Open-Source-Projekt seit 2019 von Mozilla leider nicht fortgesetzt, das Git-Repository ist aber noch aufrufbar: https://github.com/mozilla/thimble.mozilla.org

# 2 Warum gibt es das Modul?

In diesem Modul können sich Schülerinnen und Schüler kreativ entfalten und letztlich ein Produkt entsteht lassen, das starken Alltagsbezug hat und ihre Lebenswelt aufgreift. Die Nutzung des World Wide Webs und die Möglichkeit, sich online zu präsentieren - auf z.B. verschiedenen sozialen Netzwerken wie Facebook, Instagramm oder Ähnlichem - löst großes Interesse für die Gestaltung eigener Webseiten aus.

In diesem Modul können die Schülerinnen und Schüler hinter die Kulissen der Entstehung von Websites schauen. Dabei entwickeln sie ein Verständnis, wie Webseiten aufgebaut sind. Sie erfahren mehr über die Grundlagen und erlernen die Basiskenntnisse von HTML. Durch die Erstellung eigener Webseiten, erleben sie das Internet als etwas, dass sie aktiv mitgestalten können.

# 3 Ziele des Moduls

- Kennenlernen von Markup-Languages (im speziellen HTML)
- Darstellen von Informationen mittels Markup-Languages (im speziellen mit HTML)
- Kennenlernen und Verwendung von Cascading Style Sheets (CSS)
- Kennenlernen von JavaScript als Scriptsprache zur clientseitigen Programmierung
- Programmierung einfacher Programme mittels JavaScript

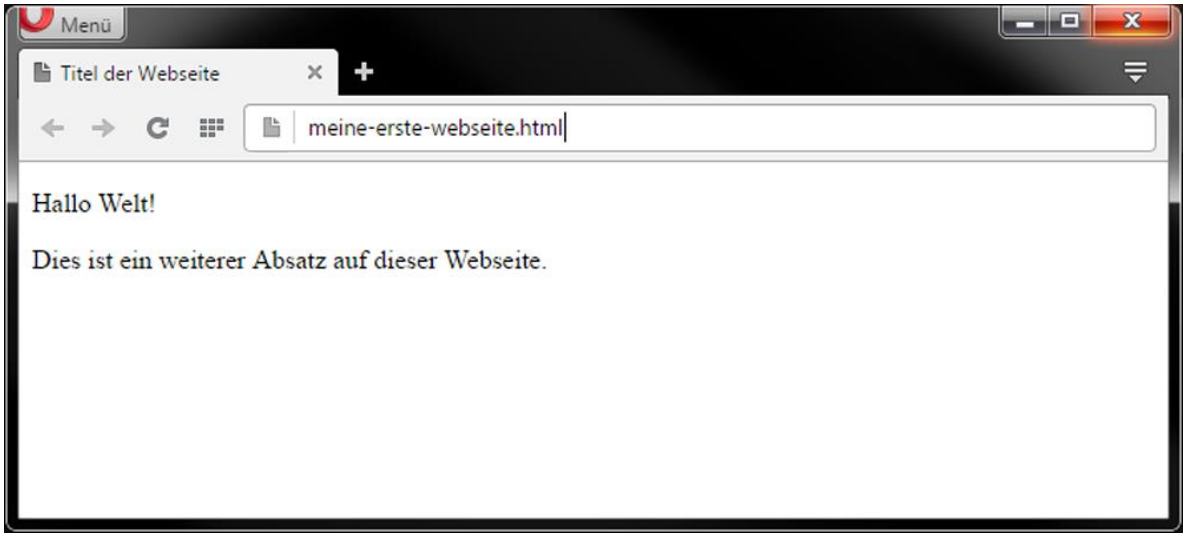
# 4 Inhalte des Moduls

## 4.1 Hypertext-Markup-Language

Die *Hypertext-Markup-Language* (kurz HTML) ist eine besondere Form der textbasierten Auszeichnungsprachen zur Strukturierung von Texten inklusive *Hyperlinks*, Bildern sowie anderen Inhalten. Grundlage des *World Wide Webs* bilden die Webseiten in Form von HTML-Dokumenten, die auf einen festen Standard des *World Wide Web Consortiums* (W3C) basieren. Ähnlich zu anderen Auszeichnungssprachen (Markup-Laguages) wie *XML* gliedert sich ein HTML-Dokument durch verschiedene Tags. Dem folgenden Beispiel ist die Grundstruktur einer Webseite zu entnehmen. Sie gliedert sich in einen *Head-* und *Body-*Bereich.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titel der Webseite</title>
  </head>
  <body>
    <p>Hallo Welt!</p>
    <p>Dies ist ein weiterer Absatz auf dieser
       Webseite.</p>
  </body>
</html>
```

Erkennbar sollte sein, dass der Head-Bereich der Webseite durch das `<head>` … `</head>` eingegrenzt wird sowie der Body-Bereich durch das `<body>` … `</body>`. Im Head-Bereich stehen die wichtigen Meta-Informationen, wie beispielsweise der Titel der Webseite, der auch im Browser steht oder Angaben zum Autor. Im Body-Bereich steht dann der eigentliche Text der Webseite. Wie das obige HTML-Dokument im Browser dargestellt werden würde, zeigt folgendes Bild.




Für die Schülerinnen und Schüler kann es auch von Interesse sein, den Quellcode einer bekannten Webseite wie Google, Facebook oder Youtube zu öffnen. Dies ist mit jedem Browser mittels eines simplen Rechtsklicks möglich und zeigt, wie komplex und für den Menschen unleserlich dieser Quellcode werden kann.

Weitere Anlaufstellen, um sich tiefergehend mit HTML zu befassen und einzuarbeiten, finden Sie im Abschnitt „Literatur und Links“.

## 4.2 Cascading Style Sheets

Die Cascading Style Sheets (CSS-Dokumente), die in die HTML-Dokumente eingebunden werden können, dienen zur Formatierung einzelner und zur Gruppierung von Tags. Mit ihnen können zum Beispiel Schrift, Farbe, Position usw. beschrieben werden.

Der folgende Auszug zeigt ein Beispiel zur Formatierung von Absätzen (p Tags), bei dem die Schriftgröße auf 14px und Schriftfarbe auf rot gesetzt wird.

```
P {
  font-color: red;
  font-size: 14px;
}
```

Für eine weitere Vertiefung mit diesem Thema sei ebenfalls auf den Abschnitt *Literatur und Links* verwiesen.

## 4.3 Beispielhaftes HTML/CSS-Projekt

Als Minimalbeispiel schauen wir uns im Folgenden den Quelltext [3] der fiktiven Seite „We <3 IT2School“ (siehe rechter Screenshot) etwas genauer an:


[3] Um besser Bezug nehmen zu können, sind hier Zeilennummern vorangestellt (grau).

```
1  <!doctype html>
2  <html lang="de-DE">
3
4    <head>
5       <meta charset="utf-8">
6       <title>We love IT2School!</title>
7       <link rel="stylesheet" href="style.css">
8    </head>
9
10   <body>
11
12   <div id="box">
13         <h1>We ♥ IT2School!</h1>
14         <img style="width:75%;"
            src="https://thispersondoesnotexist.com/image" />
15         <h2>Würde diese Person existieren, dann würde sie IT2School
               sicher genauso lieben wie wir.</h2>
16         <p>Moment mal: Wieso "würde"? Die oben abgebildete Person
           existiert nämlich gar nicht. Sie enstammt einem
           Algorithmus, der gelernt hat, aus Unmengen vorhandener
           Bilddateien künstlich Bilder von Personen zu erzeugen. Das
           Tool <a href="https://thispersondoesnotexist.com">
           thispersondoesnotexist</a> und noch viel mehr lernst du in
           den KI-Modulen von <a href="https://www.wissensfabrik.de/
           it2school/">IT2School</a> kennen!</p>
17   </div>
18
19       <img style="width:25%;" src="https://www.wissensfabrik.de/wp-
         content/themes/wissensfabrik/images/logo.svg"/>
20
21   </body>
22 </html>
```

Im Head-Bereich (Z. 4-9) wird zunächst die verwendete Zeichencodierung (UTF-8) angegeben. Dadurch wird sichergestellt, dass die Zeichen im Dokument richtig interpretiert werden[4]. Anschließend folgt die Angabe des Titels der Seite, der bspw. von manchen Browsern in der Fensterleiste übernommen wird (Hinweis: Das meint hier nicht die Überschrift der eigentlichen Webseite, die in Z. 14 angegeben ist!).

Im Body-Bereich (Z. 11-39) folgt dann zunächst eine Aufteilung (engl. „division“) mittels eines `<div>`-Tags. Solche „Tags“ (engl. für „Etikett“) sind Behälter für HTML-Elemente, die sich wiederum mit CSS gestalten (vgl. Abs. 4.2) oder mit JS manipulieren (vgl. folgender Abs. 4.4) lassen. Eigenschaften zu `box`, `h1`, `h2`, `p` und `a`[5] werden im CSS spezifiziert.

Erwähnenswert ist die Einbettung eines zufällig generierten Bildes einer nicht-realen Person in Z. 14 von der Seite `thispersondoesnotexist.com` sowie das Logo der Wissensfabrik, das über Z. 19 eingebettet wird.

Hinweis: Im Basismodul KI-B1 lernt ihr weitere Beispiele von Künstlicher Intelligenz (KI) kennen!

[4] Warum es wichtig ist, sich mit der Zeichencodierung zu beschäftigen, wird bspw. unter https://www.w3.org/International/questions/qa-what-is-encoding anschaulich erklärt. Relevanz erfährt die Zeichencodierung im vorliegenden Dokument bspw. bei den Umlauten oder auch dem ♥ in Z. 14.

[5] Mit `<a>`-Tags werden üblicherweise Hyperlinks bezeichnet, `<p>` wird für Absätze (engl. „paragraph“) und `<h>` für Überschriften (engl. „heading“) verwendet.

Im digitalen Archiv findet sich das HTML-Dokument, das entweder zur Ansicht in einem Browser (bspw. Firefox) oder zur Bearbeitung in einem Texteditor (Bsp. werden im folgenden Abs. 4.5 gegeben, grundsätzlich eignet sich jedoch auch ein systemseitig vorinstallierter Texteditor wie Wordpad) geöffnet werden kann.

Dort wird das CSS über direkt in den Header eingebaut:

```
<!doctype html>
<html lang="de-DE">

  <head>
     <meta charset="utf-8">
     <title>We love IT2School!</title>
     <style type="text/css">
       * { font-family: Arial, sans-serif; }
        html, body { min-height:90%; min-width:90%; }
        body { background:rgb(237, 244, 250); text-align:center;}
        #box { padding:20px; margin:30px auto; max-width:70%; text-
               align:center; border-radius:20px; background-
              color:#fff; color:rgb(111,111,111);}
        a { color:#f5a700; }
        p { margin-top:10px; }
        h1 { color:#f5a700; font-size: 60px;}
        h2 { color:rgb(111, 111, 111); font-size: 30px;}
     </style>
  </head>

  <body>

       […]

  </body>
</html>
```

Insbesondere bei größeren Webseiten-Projekten ist es üblich, die CSS-Datei(en) stattdessen separat anzulegen (Suffix „.css“) und im HTML-Dokument zu verlinken. Das funktioniert wie folgt:

```
<!doctype html>
<html lang="de-DE">

  <head>
     <meta charset="utf-8">
     <title>We love IT2School!</title>
     <link rel="stylesheet" href="style.css">
 </head>

  <body>

       […]

  </body>
</html>
```

Das eigentliche HTML-Dokument lässt sich so übersichtlicher gestalten. Außerdem ermöglicht dieses Vorgehen, dass dasselbe CSS redundanzfrei auch in anderen HTML-Dokumenten verwendet werden kann.

Die Lesbarkeit von CSS wird durch Zeilenumbrüche und Tabulatoren massiv erleichtert. Online stehen frei zugängliche Tools zur Verfügung, bspw. unter https://www.freeformatter.com/css-beautifier.html (für HTML: https://www.freeformatter.com/html-formatter.html). Das verwendete CSS lässt sich so wie folgt strukturieren.

```
* {

	font-family: Arial, sans-serif;
}
html, body {
	min-height: 90%;
	min-width: 90%;
}
body {
	background: rgb(237, 244, 250);
	text-align: center;
}
#box {
	padding: 20px;
	margin: 30px auto;
	max-width: 70%;
	text-align: center;
	border-radius: 20px;
	background-color: #fff;
	color: rgb(111, 111, 111);
}
a {
	color: #f5a700;
}
p {
	margin-top: 10px;
}
h1 {
	color: #f5a700;
	font-size: 60px;
}
h2 {
	color: rgb(111, 111, 111);
	font-size: 30px;
}
```

## 4.4 JavaScript

JavaScript ist von der Programmiersprache Java abzugrenzen. Es bestehen nur wenige Gemeinsamkeiten. Bei JavaScript handelt es sich um eine Scriptsprache, das bedeutet, dass die Programme/Scripte und Befehle nicht (wie bei Java, C++ und andere) bereits vorher in Maschinencode vorhanden sind. Bei einer Scriptsprache werden diese beim Lesen/Aufruf mittels Browser von einem Interpreten ausgeführt. Dies hat den Vorteil, dass die Scripte clientseitig ausgeführt werden.

Mit JavaScript lassen sich dynamische HTML-Dokumente erzeugen. So lassen sich durch einen Knopfdruck auf einer Webseite bestimmte Formatierung verändern, Berechnungen ausführen oder bestehendes HTML und CSS erweitern. Verwendet werden können viele bekannte Elemente aus anderen Programmiersprachen wie Kontrollstrukturen (if-else), Schleifen, Variablen usw.

Das folgende Beispiel zeigt wie der String `"Hallo Welt"` in einer Variablen gespeichert und mittels `alert`-Befehl als Dialogbox im Browser ausgegeben wird.

```
var variable = "Hallo Welt";
alert(variable);
```

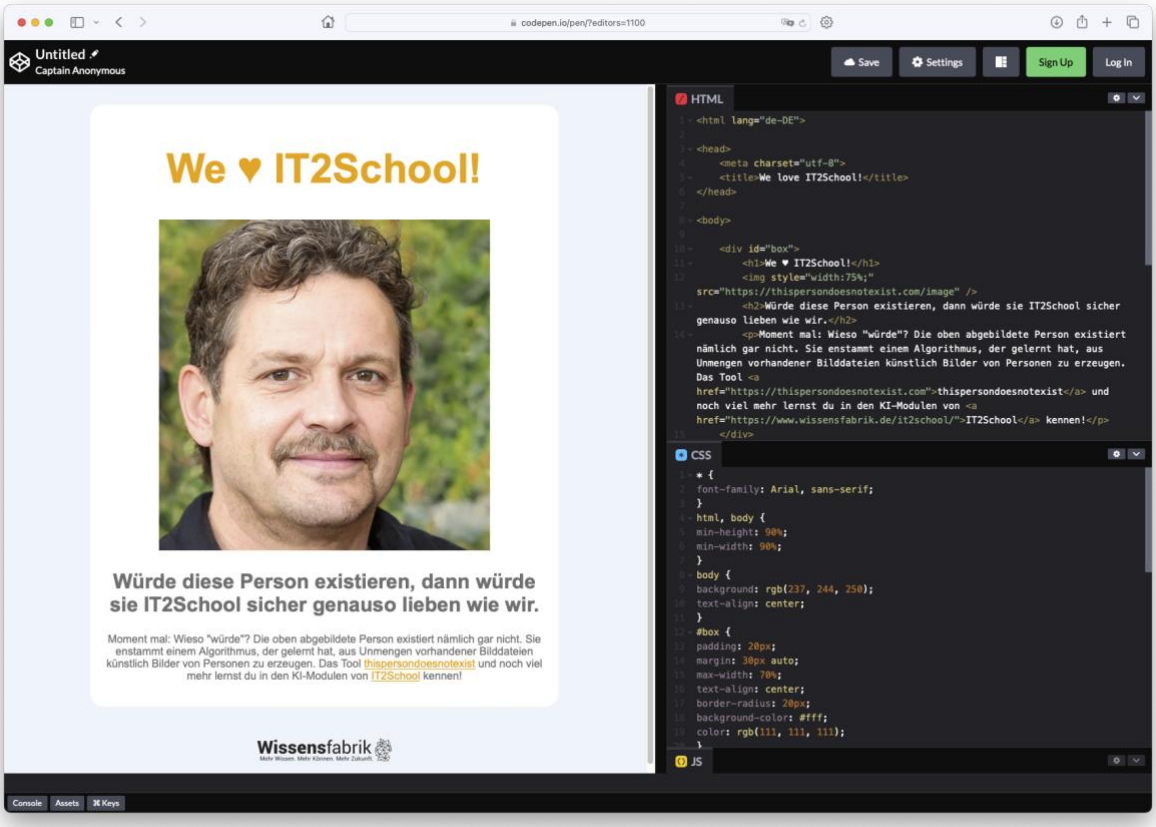
Für eine weitere Vertiefung in dieses Thema sei ebenfalls auf den Abschnitt *Literatur und Links* verwiesen.

## 4.5 Vorstellung verschiedener Editoren

Kern dieses Erweiterungsmoduls ist die Arbeit mit Online-Quelltext-Editoren zur Entwicklung einfacher, eigener Webseiten. Um eine Vorschau in Echtzeit zu ermöglichen, hat man bei all den drei im Folgenden vorgestellten Editoren sowohl den Quelltext als auch die Webseite gleichzeitig im Blick. Wenn man also etwas im Quelltext ändert, sieht man direkt die Veränderung.

### 4.5.1 Codepen.io

Der Editor **Codepen.io** ist unter dem Link https://codepen.io/pen/ zu finden und in einem beliebigen Browser einsetzbar. Auf der Startseite findet sich die Möglichkeit, sich mit Nutzerdaten anzumelden und Projekte online abzulegen. Alternativ kann man den Dienst auch ohne Konto nutzen, indem man auf „Start Coding“ klickt oder direkt https://codepen.io/pen/ aufruft.




Startet man den Editor so ist er wie folgt aufgebaut: Im oberen Bereich finden sich drei Textfelder: Eines für den HTML-Code, einen für das CSS und einen für JS. Mit einem Klick auf die entsprechende Schaltfläche in der oberen rechten Ecke lassen sich Editoren und Vorschaufenster auch nebeneinander anordnen (wie im Screenshot).

Unser Beispielprojekt kommt ohne JS aus, daher können wir den dritten Editor zur besseren Übersichtlichkeit minimieren (über einen Klick auf den entsprechenden Pfeil und eine Auswahl des „minimize“-Befehls). So entsteht ein insgesamt **sehr übersichtlicher Editor**.

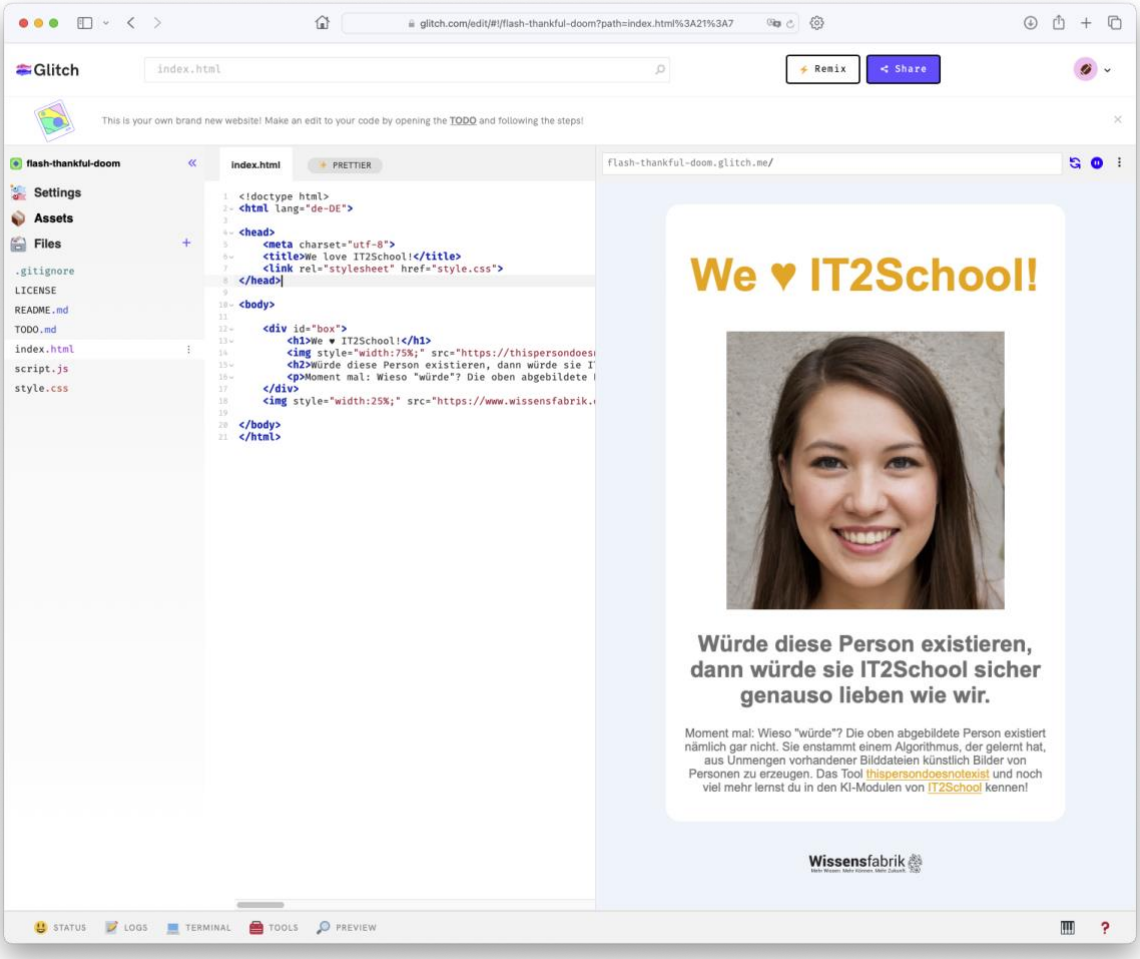
Ein weiterer Vorteil von Codepen.io ist, dass eine **Registrierung nicht zwingend notwendig** ist. Jedoch ist es empfehlenswert, einen eigenen Account anlegen. Dies ermöglicht es,

Projekte online zu speichern und später an ihnen weiterzuarbeiten, ohne die Dokumente lokal zwischenspeichern zu müssen.

Ein Nachteil kann sich aus datenschutzrechtlicher Sicht daraus ergeben, dass der Dienst **nicht in Deutschland gehostet** wird und das Tool **nur mit vorhandener Internetverbindung nutzbar** ist.

### 4.5.2 Glitch

Nachdem Thimble 2019 eingestellt wurde, wurde **Glitch** von Mozilla als Möglichkeit zur Migrierung von Thimble-Projekten angegeben. Wie auch Codepen.io ist Glitch online in den gängigen Browsern einsetzbar: https://glitch.com/ Im Gegensatz zu Codepen.io ist bei Glitch jedoch die Anlegung eines Nutzerkontos oder der Login über einen vorhandenen Facebook-, GitHub- oder Google-Account unumgänglich. Zu empfehlen ist das Anlegen eines Kontos über die Verknüpfung mit einer E-Mail-Adresse, an die zur Verifikation ein Code geschickt wird. Ggf. eignen sich hierfür Wegwerf-E-Mail-Adressen, deren Nutzung mit den Schülerinnen und Schülern in diesem Zusammenhang diskutiert werden könnte.



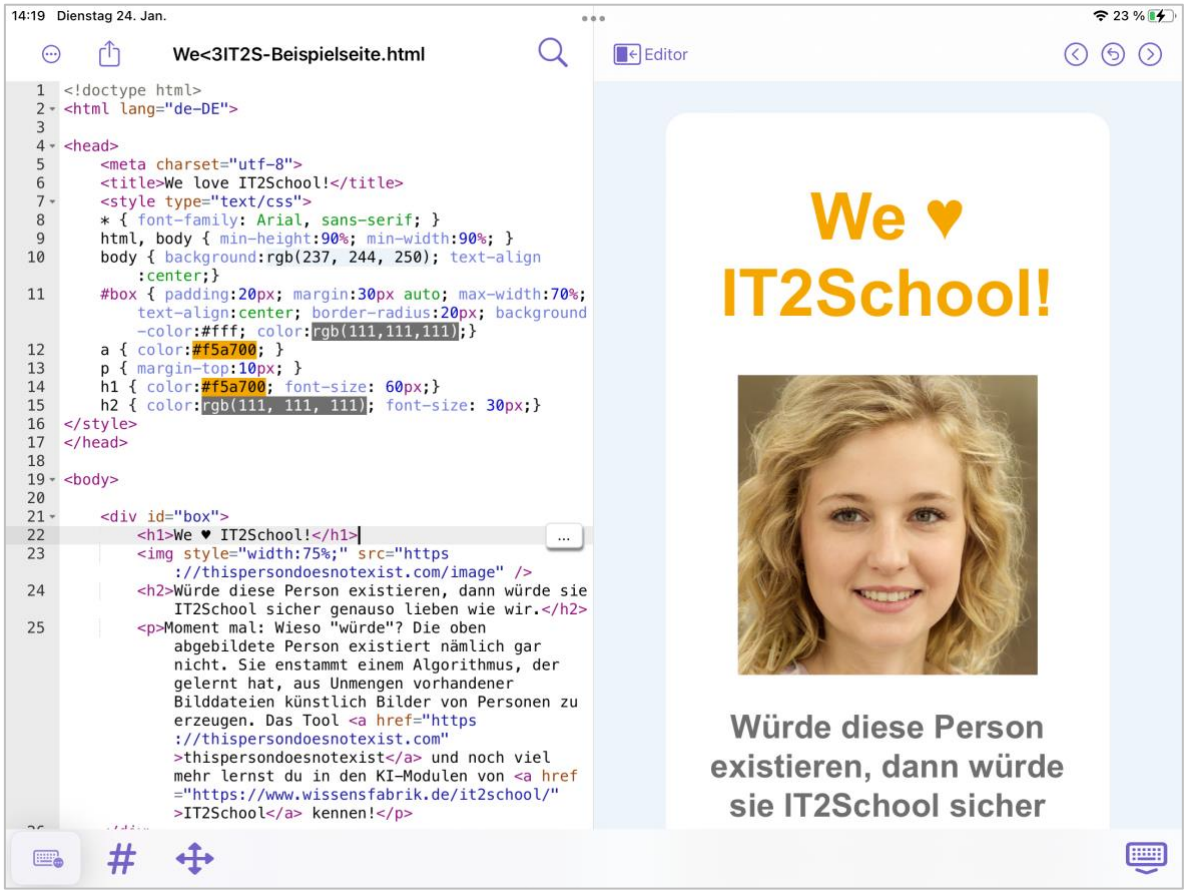

Nachdem der Prozess des Anlegens eines Nutzerkontos und der Login gemeistert sind, lässt sich in Glitch ein neues Webseiten-Projekt anlegen. Neben den bekannten HTML-, CSS- und JS-Dateien werden auch zwei MD-Dateien angelegt, die zu vernachlässigen sind. Das Projekt ist initial mit Beispielcode gefüllt, der durch eigene Projekte ersetzt werden kann. Über einen

Klick auf „Preview“ lässt sich das Vorschaufenster entweder als Panel im aktuellen Fenster ergänzen oder in einem separaten Fenster öffnen.

Ein Vorteil von Glitch ist die **integrierte „Prettier“-Funktion** (siehe Abs. 4.3). Ein Nachteil kann sich wie bei Codepen.io aus der Tatsache ergeben, dass der Dienst nicht in Deutschland gehostet wird. Anders als bei Codepen.io ist bei Glitch das **Anlegen eines Nutzerkontos verpflichtend**.

### 4.5.3 HTML Editor (iPad & Mac App)

iPads finden in Schulen zunehmend Verbreitung. Exemplarisch für die zahlreichen Möglichkeiten soll hier der **HTML Editor** (kostenlos als iPad App sowie auch als Mac App verfügbar) vorgestellt werden.




Ein Vorteil dieser und vergleichbarer Apps ist, dass sie **ohne Anlegen eines Kontos** funktionieren und **nativ auf den Geräten** laufen, d. h. die Inhalte der Webseiten-Projekte bleiben bei vielen dieser Apps auf dem Gerät.

# 5 Unterrichtliche Umsetzung

| Unterrichtsszenarien | Kurze Zusammenfassung |
|---|---|
| Einstieg | Präsentation einer Webseite per Beamer, Betrachtung des Aufbaus und des Quellcodes, Tags (<html> </html>) |
| Vertiefung | Umsetzung kleinerer Aufgaben und Übungen (z.B. von Mozilla Teaching Activities oder Lehrer-Online.de) |
| Vertiefung | Umsetzung einer eigenen Webseite |
| Präsentation | Präsentation und Veröffentlichung im Netz der enstandenen Webseiten |

# 6 Literatur und Links

- HTMLing.net – **Online-Kurs für Kinder**: http://www.htmling.net/
- Informatik-Zentrale - Erste HTML-Befehle:http://www.informatikzentrale.de/html-erste-befehle.html
- **Inf-Schule – das elektronische Schulbuch**: http://www.inf-schule.de/information/informationsdarstellunginternet
- Self-HTML – **HTML-Wiki** zum Nachschlagen, Austauschen und Lernen https://wiki.selfhtml.org/