

# Praktikum 2: Markup und HTML

---

## Ziele

In diesem Praktikum soll das Thema Markup mit HTML und anderen Auszeichnungssprachen abgeschlossen werden.

### Markup:

- Formulare in HTML5
- Versuche mit SVG und MathML

### Stylesheets:

- Stylesheet einbinden
- Textformate

Am Ende der Praktikumsbeschreibung hat es einige Verweise zu Informationsquellen über HTML und CSS. In verschiedenen Aufgaben dieses Praktikums sind die benötigten HTML-Elemente mit ihren Attributen sowie die gesuchten CSS-Eigenschaften und ihre Verwendung nicht in der Aufgabe angegeben. Sehen Sie in diesen Fällen einfach in die angegebenen Online-Quellen.

## Abgabe

Bitte alle Aufgaben zippen und in Moodle hochladen

## Aufgabe 1: Formulare im Web

In dieser Aufgabe sollen Sie einige Versuche mit Formularen machen. Sie finden im Praktikumsordner eine ZIP-Datei, in der das Gerüst für die Webseiten bereits vorhanden ist und sie nur noch den Inhalt ergänzen müssen. Für jede Aufgabe ist eine separate HTML-Seite vorbereitet. Entpacken Sie das ZIP-Archiv in ein Verzeichnis in Ihrem lokalen Dateisystem und passen Sie dann die entsprechende Datei an. Für die Formatierung ist jeweils das Stylesheet `css/styles.css` eingebunden.

### Aufgabe 1a: GET-Methode

In der Datei `get-post.html` finden Sie ein Formular mit folgendem Inhalt:



The screenshot shows a web form titled "GET versus POST". On the left, there is a sidebar with three links: "GET vs POST" (highlighted), "Registrierung", and "HTML5". The main content area is titled "Login mit GET" and contains two input fields labeled "Benutzername:" and "Passwort:", followed by a button labeled "Anmelden".

FORMULAR MIT GET-METHODE

- Passen Sie das Formular so an, dass die eingegebenen Daten beim Klick auf Anmelden mit Der GET-Methode an das Script <https://postman-echo.com/get> auf dem Server übertragen werden. Das Script gibt alle vom Browser übermittelten Daten auf einer Webseite aus.
- Lesen Sie die Seite <https://postman-echo.com> durch. Testen Sie den Service mit Hilfe der console und curl.
- Experimentieren Sie betreffend der Codierung mit der Seite <https://www.url-encode-decode.com/>. Codieren und decodieren Sie einige Umlaute.

#### Fragen für die Abgabe:

- Wie werden die Felder an den Server übermittelt? Wie sind Umlaute codiert? Warum?
- Wozu dient das fieldset -Element? Wie wird der «Titel» eines Fieldsets definiert?
- Was geschieht, wenn Sie ein Label mit der Maus anklicken?
- Wie werden die label -Elemente mit den Eingabefeldern verknüpft?
- Wozu dient das name -Attribut? Kann es weggelassen werden?
- Wie kann man Eingabewerte in den Feldern vordefinieren?
- Studieren Sie die vom serverseitigen Script ausgegebenen Environment-Variablen. Welche Informationen kann man diesen entnehmen? Wird auch ausgegeben, von welchem Computer die Formulardaten geschickt wurden?
- Analysieren Sie im Browser mit Hilfe der Entwicklertools – Tab Network den Request und den Response: Wozu dienen die Angaben: user-agent, accept, host, content-length, content-type, scheme, method, status

## Aufgabe 1b: Post-Methode

---

Ergänzen Sie die Datei get-post.html um ein zweites Formular mit dem gleichen Inhalt wie bei Aufgabe 1a. Diesmal soll das Formular jedoch mit HTTP-POST übermittelt werden:

**GET versus POST**

GET vs POST

Registrierung

HTML5

Login mit GET

Benutzername:

Passwort:

Anmelden

Login mit POST

Benutzername:

Passwort:

Anmelden

FORMULAR MIT POST-METHODE

Beachten Sie, dass `id`-Attribute eindeutig sein müssen, d.h. pro Datei nur einmal vorkommen dürfen. Passen Sie deshalb die Attributwerte im zweiten Formular an.

### Fragen für die Abgabe

- Was ist der Unterschied bei der Übertragung der Daten? Verwenden Sie die Browser-Entwicklertools, um den Datenaustausch zwischen dem Browser und Server zu beobachten.
- Würden Sie für dieses Formular GET oder POST bevorzugen? Warum?

## Aufgabe 1c: Umfangreicheres Formular

In der mitgelieferten Datei `register.html` ist bereits ein Teil eines Registrierungsformulars implementiert, das am Schluss so aussehen soll:

### REGISTRIERUNGSFORMULAR

- Ergänzen Sie noch die Radio-Buttons für das Geschlecht. In diesem Fall kommt das `label`-Element nach dem `input`-Element. Die etwas umständliche und nicht so schöne Konstruktion `<legend><span>...</span></legend>` erleichtert die Darstellung per CSS.
- Ergänzen Sie das Auswahlménü (Tipp: `select`-Element). Es soll Mehrfachauswahl zulassen. Die Kurse sollen die Werte `course1` bis `course4` haben.

### Fragen für Abgabe:

- Wie werden mehrere Werte bei einer Mehrfachselektion übertragen? Verwenden Sie die Entwicklertools des Browsers. Hinweis: Das PHP-Skript, das die Daten entgegen nimmt, liefert alle ausgewählten Elemente in einem Array, wenn das `name`-Attribut auf `[]` endet, z.B.: `name="course[]"`.
- Ergänzen Sie noch ein `input`-Feld vom Typ `hidden` und geben diesem ein `name`- und ein `value`-Attribut. Kommt der Wert ebenfalls am Server an? Wozu sind Felder vom Typ `hidden` gut und zu welchem Zweck können sie verwendet werden?

## Aufgabe 1d: HTML5 Möglichkeiten in Formularen

Kopieren Sie das Formular von Aufgabe 1c in die Datei `html5.html` und erweitern Sie es um einige der neuen HTML-Features. Die CSS-Datei ist bereits so vorbereitet, dass an der Darstellung angezeigt wird, ob ein Feld einen zum vorgegebenen Typ akzeptablen Inhalt hat:

The screenshot shows a web form titled "Registrierung". On the left is a sidebar with three links: "GET vs POST", "Registrierung" (highlighted), and "HTML5". The main form is divided into two sections:

- Logindaten:** Contains three text input fields labeled "Benutzername:", "Passwort:", and "Bestätigen:".
- Benutzerdaten:** Contains several fields:
  - "Name:" with the value "Peter Muster".
  - "E-Mail:" with the value "peter@muster.com".
  - "Geburtsdag:" with a date picker showing "tt.mm.jjjj".
  - "Anz. Teilnehmer:" with a numeric input field containing the value "1".
  - "Geschlecht:" with two radio buttons labeled "männlich" and "weiblich".
  - "Kurse:" with a dropdown menu showing "C++ für Fortgeschrittene" and "Stricken für Hacker".
  - "Kommentar:" with a large text area.
  - A "Registrieren" button at the bottom.

### FORMULARELEMENTE IN HTML5

- Verwenden Sie die neuen Eingabetypen wie `email`, `number`, etc. Fügen Sie ausserdem das zusätzliche Feld «Anzahl Teilnehmer» ein. Eine gute Übersicht über die neuen Formularelemente finden Sie unter <http://www.webkrauts.de/2009/10/01/tipphilfen>.

### Fragen für Abgabe:

- Wie können die Formularfelder validiert werden?
- Wie können Sie Pflichtfelder definieren?
- Wird das Formular bei Validierungsfehlern übermittelt?
- Wie können Sie die Felder mit einem Musterinhalt belegen, der entfernt wird, sobald Sie das Feld anwählen?
- Wie können Sie den Fokus/Cursor nach dem Laden der Seite automatisch in einem bestimmten Feld positionieren? (ohne JavaScript)

## Aufgabe 2: Versuche mit SVG und MathML

---

Im Verzeichnis `svg_mathml` des Praktikumsarchivs hat es zwei SVG-Dateien, `resistor.svg` und `todo.svg`. Mit diesen beiden Grafiken sollen nun ein paar Versuche gemacht werden:

Sehen Sie sich den Aufbau von `resistor.svg` an. Aus welchen SVG-Grundelementen ist die Grafik aufgebaut? Testen Sie mit verschiedenen Browsern, ob Sie die Grafik direkt laden können. Ändern Sie die Farbe mit der das Rechteck ausgefüllt ist und laden Sie die Grafik erneut.

Erstellen Sie eine einfache HTML5-Datei und betten Sie den Code der Grafik `resistor.svg` direkt in die HTML-Datei ein. Lassen Sie dabei die XML-Deklaration und die Namespace-Angaben weg, und verwenden das SVG-Element in der Form

```
<svg style="width:700px;height:400px"> .
```

Klappt die Anzeige der Grafik in allen Browsern?

Die Grafik `todo.svg` könnte als Logo der in einem früheren Praktikum erstellten To-Do-Liste dienen. Laden Sie die Grafik in den Browser und skalieren Sie die Ausgabe. Auch bei deutlicher Vergrößerung sollte die Qualität dieser Scalable Vector Graphic nicht leiden.

Probieren Sie, ob Sie die ToDo-Grafik im Online-SVG-Editor bearbeiten können:

<https://github.com/SVG-Edit/svgedit>.

Zum Bearbeiten von SVG-Grafiken gibt es zahlreiche Werkzeuge – viele Grafikprogramme können SVG einlesen und exportieren. Ein speziell auf SVG zugeschnittenes Programm ist Inkscape (<http://www.inkscape.org>, Open Source).

Nun nehmen wir uns noch mathematische Formeln vor, für welche die XML-Anwendung MathML vorgesehen ist. In der Datei `math.html` ist eine solche Formel integriert, die Formel zum Lösen quadratischer Gleichungen.

- Testen Sie mit verschiedenen Browsern, ob die Formel angezeigt wird. Mindestens in aktuellen Firefox-Browsern (sowie im Safari Browser - MacOS) sollte es funktionieren.
- Sehen Sie sich den Quellcode von `math.html` an. Können Sie die einzelnen Elemente der Formel identifizieren? Ändern Sie die Formel leicht ab und überprüfen Sie, ob die Anzeige wie erwartet ausfällt.
- Da nicht alle Browser mit MathML umgehen können, gibt es JavaScript-Bibliotheken, die diese Lücke füllen. In `math.html` sind zwei noch auskommentierte `script`-Elemente enthalten, die die MathJax-Bibliothek laden. Probieren Sie beide Varianten (einzeln, nicht beide zusammen) in verschiedenen Browsern aus. In der ersten Variante wird die Formel für die Darstellung in SVG umgewandelt, in der zweiten wird HTML und CSS für die Darstellung verwendet.

## Aufgabe 3: Einbinden einer CSS Datei

Als Beispielseite dient eine Seite, die in einem Tutorial auf [tuts+](https://tutsplus.com) verwendet wird: Ein Weblog über Blumen. Sie finden das Beispiel im Verzeichnis `flower_blog` im Praktikumsarchiv. Wie die Startseite fertig gestaltet aussehen soll, können Sie auf dem Bild auf der letzten Seite dieser Praktikumsbeschreibung sehen. Ohne CSS sieht die Seite natürlich weniger schön aus.

Sehen Sie sich den Aufbau der HTML-Datei an. Gibt es noch Verbesserungspotential?

Hier einige Hinweise:

- Sollte die Navigation zum Header gehören?
- Könnte das `main`-Element eingesetzt werden?

Die Fragen sind nicht einfach und eindeutig zu beantworten. Letztlich entscheidet man als Autor der Seite, welche Rolle die einzelnen Seitenelemente auf der Seite spielen sollen. Sehen Sie sich auch die Verwendung der Elemente `article`, `section`, `aside`, `footer` und `time` an.

Beispiel Seite:

<https://code.tutsplus.com/tutorials/html-5-and-css-3-the-techniques-youll-soon-be-using--net-5708>



# Anhang: Referenzen

## HTML

- HTML 4.01 Specification (W3C)  
<https://www.w3.org/TR/html4/cover.html>
- HTML 5.1 Recommendation (W3C)  
<https://www.w3.org/TR/html/>
- 
- HTML Living Standard (WHAT WG)  
<https://html.spec.whatwg.org/multipage/>
- HTML-Tutorial und -Referenz (W3Schools)  
<http://www.w3schools.com/html/default.asp>  
<http://www.w3schools.com/tags/default.asp>

## CSS

- CSS-Spezifikationen (W3C)  
<http://www.w3.org/Style/CSS/current-work>
- CSS Referenz (MDN web docs)  
[https://developer.mozilla.org/de/docs/CSS\\_Referenz](https://developer.mozilla.org/de/docs/CSS_Referenz)
- CSS-Tutorial und -Referenz (W3Schools) (mit Möglichkeiten zum Ausprobieren)  
<http://www.w3schools.com/css/default.asp>  
<http://www.w3schools.com/cssref/default.asp>

## Verschiedene Dokumentationen

- Webtechnologien für Entwickler (MDN web docs)  
<https://developer.mozilla.org/de/docs/Web>
- DevDocs – multiple API documentations  
<https://devdocs.io>