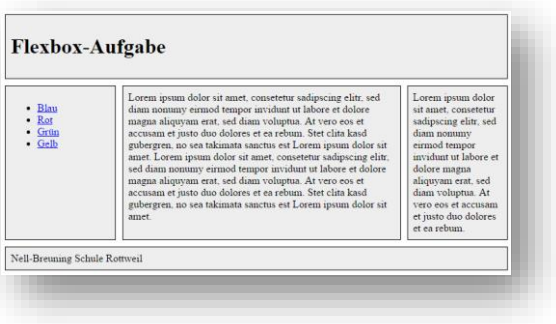
L2_2.3 Modularität / Navigation

Komplexere Webseiten verfügen in der Regel über eine Navigation, die den Besucher einen schnellen Zugriff auf einzelne Inhalte ermöglicht. Dabei werden immer nur Teile einer Seite durch neue Inhalte ersetzt. Seitenelemente wie Kopf-/Fußzeile oder Seitenleisten bleiben oft unverändert.



Natürlich können wir den darzustellenden Text und die Grafiken direkt in unsere HTML-Seite integrieren. Das Ergebnis könnte wie folgt aussehen. Die eigentlichen sichtbaren Inhalte, sind durch blaue Kästen eingerahmt.

```
<html>

  <head>
    <title>Das ist die blaue Seite</title>
        <link rel="stylesheet" href="stylesheet.css">
  </head>

  <body>

        <header>
                <h1>Flexbox-Aufgabe</h1>
        </header>

        <div id="maincontainer">

                <nav>
                        <ul>
                                <li><a href="blau.html">Blau</a></li>
                                <li><a href="rot.html">Rot</a></li>
                                <li><a href="gruen.html">Grün</a></li>
                                <li><a href="gelb.html">Gelb</a></li>
                        </ul>
                </nav>

                <main>
                        Lorem ipsum dolor sit amet, consetetur sadipscing elitr, …
                </main>

                <section>
                        Lorem ipsum dolor sit amet, consetetur sadipscing elitr…
                </section>

        </div>

        <footer>
                Nell-Breuning Schule Rottweil
        </footer>

  </body>

</html>
```

Man kann sich leicht vorstellen, dass der Umfang einer solchen Seite schnell sehr anwachsen kann. Damit kann die Pflege und Wartung sehr mühsam werden. Ein weiteres Problem ist die Navigation auf eine andere Seite des Internetauftritts. Stellen wir über einen Verweis (<a>-Element) eine Anfrage an den Server, so ersetzt die vom Server gelieferte Seite die aktuell im Browser befindliche Seite. Somit muss der Server immer eine komplette Seite, also mit Kopf-/Fußzeile, …, mit identischer Struktur liefern. Beispielsweise hat jede vom Server ausgelieferte Seite den gleichen Kopfbereich, dessen Inhalt sich nicht ändern wird. Es entsteht eine große Redundanz. Möchte man den Inhalt des Kopfbereiches ändern, so muss dies folglich in zahlreichen einzelnen Seiten passieren.

Wir können ein Teil dieser Probleme lösen, indem die Seite modularisiert, also in kleine Teilbereiche aufgeteilt wird. Die Inhalte werden in kleine Portionen aufgeteilt. In einem ersten Schritt wird jeder Inhalt in einer eigenen HTML- bzw. PHP-Datei gespeichert. Wir haben also nicht nur eine HTML- oder PHP-Datei, sondern mehrere.

kopf.html

```
<h1>Flexbox-Aufgabe</h1>
```

fusszeile.html

```
Nell-Breuning Schule Rottweil
```

inhalt.html

```
Lorem ipsum dolor sit amet,
consetetur sadipscing elitr,
…
```

seite.html

```
Lorem ipsum dolor sit amet,
consetetur sadipscing elitr,
…
```

navi.html

```
<ul>
    <li><a href="#">Blau</a></li>
    <li><a href="#">Rot</a></li>
    <li><a href="#">Grün</a></li>
    <li><a href="#">Gelb</a></li>
</ul>
```

Index.html

```
<html>
  <head>
    <title>Das ist die blaue Seite</title>
    <link rel="stylesheet" href="stylesheet.css">
  </head>
  <body>
    <header>
    </header>
    <div id="maincontainer">
        <nav>
        </nav>
        <main>
        </main>
        <section>
        </section>
    </div>
    <footer>
    </footer>
  </body>
</html>
```

Die Inhalte werden aus der Datei „index.html“ entfernt und in separaten Dateien gespeichert. Von der Datei „index.html“ bleibt lediglich das Gerüst übrig.

Nun können die Dateien separat gewartet werden. Man könnte die Wartung auch auf mehrere Personen aufteilen, die für bestimmte Seiten bzw. Inhalte zuständig sind. Keiner dieser Personen müsste Kenntnisse über den Aufbau und die Anordnung der Inhalte haben. Außerdem ist bei einem solchen Aufbau der Austausch von Inhalten sehr einfach möglich.

Es stellt sich nun die Frage, wie wir unsere Seite „index.html“, die das Gerüst aufspannt, mit den nun separat gespeicherten Inhalten verknüpfen können. Hierzu benötigen wir eine PHP-Anweisung:

```
include 'dateiname';
```

Triff der PHP-Interpreter auf eine solche Anweisung, dann wird der Inhalte der angegebenen Datei genau an dieser Stelle platziert. In unserem konkreten Beispiel könnte die „index.php“ nun wie folgt aussehen:

```
<html>
  <head>
    <title>Das ist die blaue Seite</title>
      <link rel="stylesheet" href="stylesheet.css">
  </head>

  <body>

      <header>
          <?php include 'kopf.html' ?>
      </header>

      <div id="maincontainer">

          <nav>
              <?php include navi.html' ?>
          </nav>

          <main>
              <?php include 'inhalt.html' ?>
          </main>

          <section>
              <?php include 'seite.html' ?>
          </section>

      </div>

      <footer>
          <?php include 'fusszeile.html' ?>
      </footer>

  </body>
</html>
```

Achtung: Eine Datei „index.html“ muss nun auf .php enden (-> index.php). Ansonsten werden PHP-Blöcke und damit die include-Anweisungen nicht erkannt!

Seitenavigation

Typischerweise ist eine Navigation wie folgt aufgebaut. Eine ungeordnete Liste (<ul>) stellt die einzelnen Navigationseinträge als Listenelement (<li>) zur Verfügung. Jedes Listenelement besteht dabei aus einem Verweis auf ein anderes Ziel (<a>). Das Attribut „href“ gibt dabei das Zieldokument an.

```
<ul>
  <li><a href="blau.html">Blau</a></li>
  <li><a href="rot.html">Rot</a></li>
  <li><a href="gruen.html">Grün</a></li>
  <li><a href="gelb.html">Gelb</a></li>
</ul>
```

Wird ein Verweis durch Klick ausgelöst, wird die aktuell im Browser sichtbare Seite verworfen und durch das neue Zieldokument ersetzt. Es gibt verschiedene Lösungsansätze, um dieses Verhalten zu umgehen. Wir werden jedoch einen anderen Ansatz wählen.

Wir nehmen das Beispiel von oben auf. Aktiviert man einen Navigationseintrag, so soll lediglich der Inhaltsbereich durch das neue Zieldokument ersetzt werden. Die Idee ist folgende: Die Datei „index.php“ bildet das Grundgerüst und baut die Seite zusammen (`include`). Wir rufen nun stets die Datei „index.php“ auf und übermitteln dieser eine Information, welche Datei im Inhaltsbereich eingefügt werden soll. Diese Information können wir dem Verweis an den Namen des Zieldokumentes, durch ein Fragezeichen getrennt, anhängen. Wir sprechen auch von einem Parameter, denken Sie an den Abschnitt zum Aufbau einer URL (1.1.2). Ein Parameter ist wie folgt aufgebaut:

```
Schlüssel = Wert
```

Konkret sieht dies bei uns wie folgt aus:

```
<ul>
  <li><a href="index.php?i=blau">Blau</a></li>
  <li><a href="index.php?i=rot">Rot</a></li>
  <li><a href="index.php?i=gruen">Grün</a></li>
  <li><a href="index.php?i=gelb">Gelb</a></li>
</ul>
```

Wir rufen also immer die, für den Zusammenbau der Seite zuständige Datei „index.php“. Mit Hilfe eines Parameters liefern wir die Information, welcher Inhalt geladen werden soll (z.B. i=blau). Wir haben uns hier für den Schlüssel „**i**“ und einem Wert entschieden, der dem Dateinamen des einzubindenden Zieldokuments entspricht (**blau**.html). Es stellt sich nun die Frage, wie wir diese Information in der PHP-Datei auswerten können.

Da es ich um den gleichen Mechanismus handelt, wie auch ein Formular die Daten überträgt, können wir den Wert in unserer PHP-Seite auf gleiche Weise, also über das assoziative Array „$_GET“ ermitteln:

```
$wert = $_GET['schlüssel']
```

Konkret sieht dies bei der URL „index.html?i=blau“ so aus:

```
$wert = $_GET['i']
```

In der Variablen „$wert“ steht nun der Wert „blau“. Wie können wir diesen Mechanismus nun für das Laden der Inhalte nutzen? In der „index.php“-Datei werten wir den Wert des Schlüssels „i“ aus und ermitteln mit Hilfe einer Verzweigung die zu „inkludierende“ Datei:

```
<nav>
      <?php include 'navi.html' ?>
</nav>

<main>
      <?php
        $w = $_GET['i'];

        if($w == "blau"){
            #include 'blau.html';
        }
        if($w == "rot"){
            #include 'rot.html';
        }
        […]
      ?>
</main>

<section>
      <?php include 'seite.html' ?>
</section>
```

Erklärung:
Der obige Ausschnitt der Datei „index.php“ zeigt die Bereiche für die Navigation (<nav>), den Inhaltsbereich (<main>) und die rechte Seitenleiste (<section>). Sowohl im Navigationsbereich als auch in der rechten Seitenleiste werden immer dieselben Dateien „inkludiert“. Lediglich im Inhaltsbereich ist eine Fallunterscheidung zu erkennen. In einem ersten Schritt wird dort er übergebene Parameter mit dem Schlüssel „i“ ausgewertet und in der Variablen $w gespeichert. In Abhängigkeit des Wertes von $w wird die gewünschte Zieldatei eingebunden.

Beispiel:
Der Verweise `<a href="index.php?i=blau">` hat zur Folge, dass der Inhaltsbereich mit der Datei „blau.html“ bestückt wird. Der Verweis `<a href="index.php?i=rot">` würde die Datei „rot.html“ in den Inhaltsbereich laden.

Hinweis:
Man könnte den übermittelten Parameter auch als „Dateiname“ interpretieren und direkt in der „include“-Anweisung nutzen:

```
$w = $_GET['i'];
#include $w .'.html';
```