Übung 02

1 Schnelleinstieg CSS

1.1 Syntax

Die grundlegende Syntax von Cascading Stylesheets (CSS) sieht wie folgt aus:
`Selektor {Eigenschaft: Wert;}` Es ist möglich, mit verschiedenen Selektoren demselben Element mehrfach Stileigenschaften zuzuweisen. Wird in einem Stylesheet mehrmals dieselbe Eigenschaft auf demselben Element verändert, so wird die zuletzt verarbeitete Anweisung dargestellt.

1.2 Stil-Eigenschaften

Die verschiedenen Anzeigeelemente haben individuelle Sätze von Eigenschaften, die in Stylesheets verändert werden können. Es folgen einige Beispiele:

```
h1 { color: red; }
div {
  width: 260px;
  margin: auto;
}
p {
  color: white;
  font-size: large;
  background-color: gray;
}
```

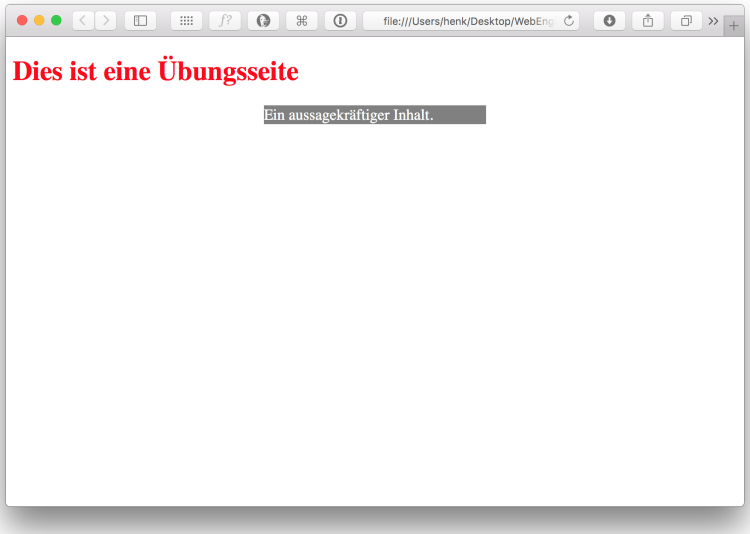

Abbildung 1: Darstellung des beispielhaften CSS

1.3 Stil-Eigenschaften

Die verschiedenen Anzeigeelemente haben individuelle Sätze von Eigenschaften, die in Stylesheets verändert werden können. Es folgen einige Beispiele:

```
h1 { color: red; }
div {
  width: 260px;
  margin: auto;
}
p {
  color: white;
  font-size: large;
  background-color: gray;
}
```

1.4 Einbettung in HTML

Es gibt verschiedene Möglichkeiten CSS in Webseiten einzubinden:

1.4.1 link

Beispiel:

```
<link rel="stylesheet" href="stylesheet.css">
```

Mithilfe des `link`-Tags können ein oder mehrere extern definierte Stylesheets in eine Webseite integriert werden. Dies erlaubt die höchste Wiederverwendbarkeit und sorgt für eine übersichtliche Trennung von Auszeichnung (HTML) und Darstellung.

1.4.2 zentrales style-Element

Beispiel:

```
<style>
  h1 { color: green; }
</style>
```

Das zentrale style-Element erlaubt es, direkt im HTML-Dokument Stilanpassungen vorzunehmen.

1.4.3 style-Attribut für einzelne Elemente

Beispiel:

```
<h1 style="color: green;">
  Dies ist eine Überschrift
</h1>
```

Viele HTML-Elemente besitzen ein `style`-Attribut, das es erlaubt, Stilanpassungen direkt am Element vorzunehmen. Diese `Inline-Styles` sind jedoch selten von Vorteil, da sie für jedes neue Element wieder definiert werden müssten und so den Wartungsaufwand erheblich erhöhen.

1.5 Selektoren/Kombinatoren

Selektoren dienen im Wesentlichen dazu, einer bestimmten Menge von Elementen Stileigenschaften zuzuweisen. Um einen möglichst präzisen Zugriff auf bestimmte Elemente zu erlauben, gibt es eine Vielfalt an verschiedenen Selektoren.

1.5.1 Typselektoren

Der Typselektor wählt alle Elemente eines Typs aus:

```
<style>
  p { color: red; }
</style>

<div>
  <p>
    Ein Absatz <!-- rot -->
  </p>
</div>

<p>
  Ein weiterer Absatz <!-- rot -->
</p>
```

1.5.2 Klassenselektor und ID-Selektor

Der Klassenselektor wird durch `.` und der ID-Selekltor durch `#` zur Auswahl aller Elemente denen mit dem `class`- oder `id`-Attribut eine bestimmte Klasse/ID zugewiesen wurde genutzt.

```
<style>
  .content-pane { background-color: red; } /* Klassenselektor */
  #special { background-color: blue; } /* ID-Selektor */
</style>

<!-- rot -->
<div class="content-pane">
  ...
</div>
<!-- blau -->
<div id="special">
  ...
</div>
<!-- neutral -->
<div>
  ...
</div>
```

1.5.3 Kindselektor

Der Kindselektor `a > b` Wählt ein Element `b` nur dann aus, wenn es ein Kindelement von `a` ist.

```
<style>
  div > p { color: red; } /* färbt den Text aller
```

```
                              p-Elemente innerhalb
                              eines div-Elements rot */
</style>

<div>
  <p>Ein Absatz</p> <!-- rot -->
</div>

<p>Ein weiterer Absatz</p> <!-- neutral -->
```

1.5.4 Nachfahrenselektor

Ähnlich zum Kindselektor, wählt der Nachfahrenselektor `a b` jedoch auch weiter entfernte Nachfahren.

```
<style>
  div a { font-weight: bold; } /* Wählt ein Element a
                                  auch dann aus, wenn
                                  es in einem anderen
                                  Element enthalten
                                  ist, solange es ein
                                  Nachfahre eines
                                  div-Elements ist. */
</style>

<div>
  <!-- Der Link wird fett dargestellt -->
  <a href="_blank">Ein Link</a>
  <p>
    <!-- Dieser Link wird ebenfalls fett dargestellt -->
    <a href="_blank">Noch ein Link</a>
  </p>
</div>
```

1.5.5 Universalselektor

Der Universalselektor `*` selektiert zunächst alle Elemente:

```
<style>
  * { color: red; }
</style>

<h1>Eine Überschrift</h1> <!-- rot -->
<p>Ein Absatz.</p> <!-- rot -->
```

Jedoch kann der Universalselektor auch mit anderen Selektoren kombiniert werden, z.B. dem Nachfahrenselektor:

```
<style>
  div * { color: red; }
```

```
</style>

<p>Ein Absatz</p> <!-- nicht rot -->
<div>
  <h2>Eine Überschrift</h2> <!-- rot -->
  <p>Ein Absatz</p> <!-- rot -->
</div>
```

1.5.6 Nachbarselektor

Der Nachbarselektor `a + b` wählt Element `b` nur dann aus, wenn es direkter nachfolgender Nachbar von `a` ist.

```
<style>
  h1 + p { font-weight: bold }
  p + p { font-style: italic }
</style>

<h1>Der Nachbarselektor</h1>
<p>Erster Absatz.</p> <!-- bold -->
<p>Zweiter Absatz.</p> <!-- italic -->
<p>Dritter Absatz.</p> <!-- italic -->
<div>neutrales Element</div>
<p>Vierter Absatz</p> <!-- neutral -->
```

1.5.7 Geschwisterselektor

Der Geschwisterselektor `a ~ b` selektiert alle Elemente `b`, die auf derselben Ebene wie `a` sind und auf `a` folgen.

```
<style>
  h1 ~ p {font-weight: bold;}
</style>

<p>Erster Absatz.</p>
<h1>Der Geschwisterselektor</h1>
<p>Zweiter Absatz.</p> <!-- bold -->
<hr>
<p>Dritter Absatz.</p> <!-- bold -->
```

2 CSS-Beispiel

Mithilfe von CSS können wir den Stil eines HTML-Dokumentes bestimmen und festlegen wie einzelne HTML-Elemente angezeigt werden. Für einen Einblick in die Benutzung von CSS erstellen wir eine Webseite, wie in folgender Abbildung dargestellt.

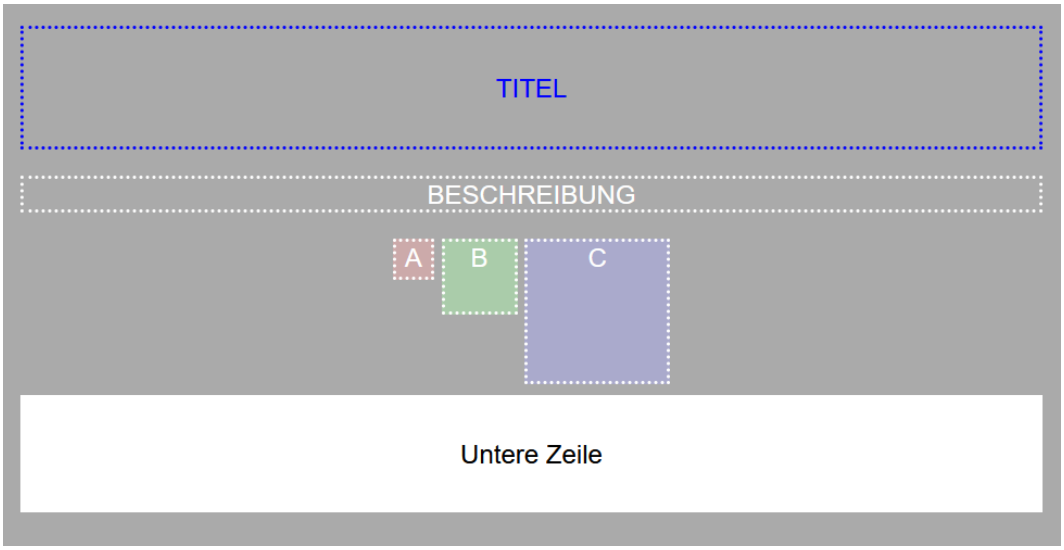

Abbildung 2: Screenshot der verschachtelten `div`-Elemente.

Hierbei erstellen wir schrittweise das Styling, um die gewünschte Darstellung zu erreichen. Dabei nutzen wir den gezeigten `body`-Block, bestehend aus einem `header`-, `main`- und `footer`-Elemente.

Listing 1

```
<body>
  <header>
     <p id="titel">TITEL</p>
     <p>BESCHREIBUNG</p>
  </header>
  <main>
    <div>
        A
    </div>
    <div>
        B
    </div>
    <div>
        C
    </div>
  </main>
  <footer class="deco">
    Untere Zeile
  </footer>
</body>
```

Bevor wir mit der Stilisierung der Webseite mithilfe von CSS beginnen können, benötigt unser HTML-Dokument noch einen `header`-Block. In dem `header`-Block können wir anschließend unser CSS-Dokument verlinken oder, wie in diesem Beispiel, den Stil direkt innerhalb des HTML-Dokumentes erstellen.

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="utf-8">
      <title>Eine CSS Beispielseite</title>
  <style type="text/css">
    <!-- ... -->
  </style>
</head>
<body>
  <!-- ... -->
</body>
</html>
```

Zuerst wollen wir für das gesamte Dokument eine serifenlose Schriftart in 18pt und weiß definieren. Zusätzlich zentrieren wir den Text für das gesamte Dokument in der Mitte. Da wir diese Änderungen für das gesamte Dokument deklarieren wollen, nutzen wir den Universal-Selektor `*`. Dieser Selektor wählt alle Elemente innerhalb des HTML-Dokumentes aus.

```
* {
  font-size: 18pt;
  font-family: sans-serif;
  color: white;
  text-align: center;
}
```

Desweitern soll der Hintergrund des `body`-Elements die Farbe grau `#aaa` erhalten. Hierfür nutzen wir den dazugehörigen Element-Selektor.

```
body {
  background-color: #aaa;
}
```

Alle direkt anhängenden Elemente des `body`-Elements sollen einen Abstand von 10 Pixel zueinander besitzen. Alle Elemente einer Verschachtlung tiefer erhalten einen gestrichelten Rahmen. Um den Abstand für direkt anhängende Elemente des `body`-Elements zu deklarieren, wählen wir diese mithilfe eines `child`-Selektors > aus und ändern das `margin`-Attribut. Analog wählen wir alle Elemente einer Verschachtlung tiefer mit einer weiteren Iteration des `child`-Selektors aus und ändern das `border-style`-Attribut auf `dotted`.

```
body > * {
  margin: 10px;
}

body > * > * {
  border-style: dotted;
}
```

Betrachten wir als nächstes unseren Titel und die Fußzeile. Der Titel wird in einer blauen Textfarbe dargestellt. Die Fußzeile besitzt einen weißen Hintergrund mit schwarzer Textfarbe und soll einen inneren Seitenabstand von 40px erhalten. In unserem HTML-Dokument

ist bereits ein `titel`-Anker definiert, sowie eine Klasse für das `footer`-Element. Beide Elemente können nun durch den zugehörigen Selektor eindeutig ausgewählt werden. Für den `titel`-Anker nutzen wir den `id`-Selektor `#` und für das `footer`-Element den `class`-Selektor `.`. Um den inneren Seitenabstand von 40px einzustellen, nutzen wir das `padding`-Attribut.

```
#titel{
  color: blue;
  padding: 40px;
}

.deco {
  background-color: white;
  color: black;
  padding: 40px;
}
```

Schauen wir uns nun den `main`-Block an. Die `div`s innerhalb des `main`-Blocks sind nebeneinander angeordnet. Diese Anordnung können wir mithilfe des `display`-Attributs und der Eigenschaft `inline-block` beeinflussen. Auch hier nutzen wir den `child`-Selektor um die `div`s innerhalb des `main`-Blocks auszuwählen.

```
main > div {
  display: inline-block;
}
```

Nun setzten wir noch die Größenvorgaben (`width` und `height`) für die `div`s innerhalb des `main`-Blocks von 32px, 64px und 128px sowie die Hintergrundfarben `#caa`, `#aca` und `#aac` um. Hierfür nutzen wir den `nth-child(an+b)`-Selektor um die einzelnen Elemente anhand ihres Index auszuwählen.

```
main > div:nth-child(1) {
  width: 32px;
  height: 32px;
  background-color: #caa;
}
main > div:nth-child(2) {
  width: 64px;
  height: 64px;
  background-color: #aca;
}
main > div:nth-child(3) {
  width: 128px;
      height: 128px;
  background-color: #aac;
}
```

Zuletzt wollen wir noch den Hintergrund eines `div`s innerhalb des `main`-Blocks mithilfe von CSS temporär schwarz färben, sobald der Mauszeiger darüber liegt. Um dies umzusetzen, nutzen wir neben dem `child`-Selektor ebenfalls den `hover`-Selektor, der ausgelöst wird, sobald sich der Mauszeiger über dem ausgewählten Element befindet.

```
main > div:hover {
  background-color: black;
}
```

Somit sieht unser vollständiges HTML-Dokument wie folgt aus:

Listing 2

```
<!DOCTYPE html>
<html lang="de">

<head>
  <meta charset="utf-8">
  <title>Übungstest</title>
  <style type="text/css">
    * {
      font-size: 18pt;
      font-family: sans-serif;
      color: white;
      text-align: center;
    }

    body {
      background-color: #aaa;
    }

  body > * {
    margin: 10px;
  }

  body > * > * {
      border-style: dotted;
  }

  #titel {
      color: blue;
      padding: 40px;
  }

  .deco {
    background-color: white;
    color: black;
      padding: 40px;
  }

  main > div{
      display: inline-block;
  }

    main > div:nth-child(1) {
      width: 32px;
      height: 32px;
      background-color: #caa;
    }

    main > div:nth-child(2) {
      width: 64px;
      height: 64px;
      background-color: #aca;
    }
```

```
    main > div:nth-child(3) {
      width: 128px;
      height: 128px;
      background-color: #aac;
    }

    main > div:hover {
      background-color: black;
    }
  </style>
</head>

<body>
  <header>
  <p id="titel">TITEL</p>
  <p>BESCHREIBUNG</p>
  </header>
  <main>
    <div>
      A
    </div>
    <div>
      B
    </div>
  <div>
      C
    </div>
  </main>
  <footer class="deco">
  Untere Zeile
  </footer>
</body>

</html>
```

3 Aufgaben

Eine andere Art und Weise Elemente innerhalb einer Webseite anzuordnen, ist CSS-Grid. CSS-Grid liefert ein auf einem Grid basierendes Layout und vereinfacht das Erstellen von Webdesigns mithilfe von Spalten und Zeilen.

Aufgabe 1

Vervollständigen Sie die CSS Grid-Definitionen in der beiliegenden `grid.css`, sodass die Webseite `index.html` so dargestellt wird, wie in folgender Abbildung.

Nutzen Sie folgende Funktionen des Grid-Moduls:

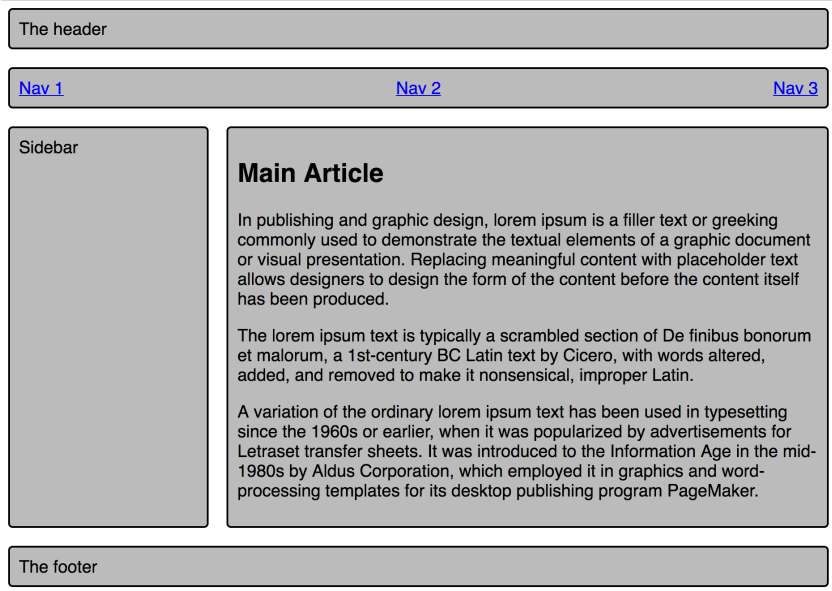

Abbildung 3: Darstellung der Webseite mit Grid-Definition

- `display: grid`
  Macht aus einem HTML-Element einen Grid-Container. Diese Eigenschaft sollte auf das Element angewandt werden, das die eigentlichen Grid-Elemente beinhaltet.

- `grid-template-areas: <value>`
  Der hier zu setzende String definiert die Aufteilung von Grid-Elementen anhand der ihnen zugewiesenen `grid-area`-Attribute. Der String kann mehrzeilig sein um eine Grid-Struktur mit mehren `rows` zu definieren. Jede durch ein Leerzeichen getrennte Zelle im String beschreibt den Inhalt der entsprechenden `column` des Grids. Die Zeilen und Spalten des Strings müssen eine Rechteckform ergeben. Wiederholungen von Elementnamen über Zeilen oder Spalten bewirken, dass eine Grid Area die jeweiligen Bereiche überdeckt.

- `grid-template-columns: <value>`
  Definiert die Bemaßung der Spalten eines Grids. Es können bekannte Größeneinheiten wie `px` oder `%` genutzt werden, wie auch das neue Schlagwort `fr` mit einem vorangestellten Faktor oder `auto`, um eine betreffende Spalte an anderen Bemaßungen auszurichten.

  Beispiele:

  ```
  grid-template-columns: 100px, auto, 100px;
  ```

100px	dynamische Breite	100px

```
grid-template-columns: 1fr, 2fr, 1fr;
```

1fr	2fr	1fr

- `grid-gap: <value>`
  Definiert den Abstand zwischen Grid-Elementen.

Aufgabe 2

Wie im innenliegenden CSS der HTML-Datei erkennbar, gibt es verschiedene Darstellungsoptionen, abhängig von der Bildschirmbreite in Pixeln (siehe Media Query `@media`). Führen Sie auch für die Grid-Definition in grid.css eine solche Unterscheidung ein, sodass erst ab einer horizontalen Auflösung von 500px das Grid wie zuvor definiert angezeigt wird. Bei geringerer Auflösung sollen alle Elemente untereinander angezeigt werden, siehe folgende Abbildung.

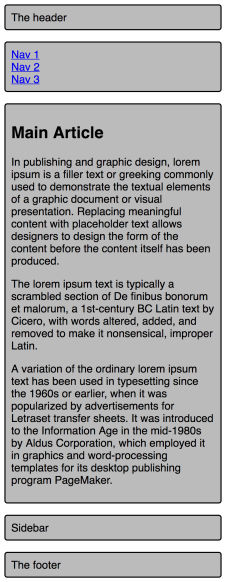

Abbildung 4: Darstellung der Webseite bei geringer horizontaler Auflösung

Lösungs zu den Aufgabes

Lösung 1

Listing 3

```
.head {
    grid-area: header;
}
.content {
    grid-area: content;
}
.nav {
    grid-area: nav;
}
.side {
    grid-area: sidebar;
}
.footer {
    grid-area: footer;
}
.wrapper {
    display: grid;
    grid-gap: 20px;
    grid-template-areas:
    "header"
    "nav"
    "content"
    "sidebar"
    "footer";
}
```

Lösung 2

Listing 4

```
@media (min-width: 500px) {
    .wrapper {
        grid-template-columns: 1fr 3fr;
        grid-template-areas:
        "header  header"
        "nav     nav"
        "sidebar content"
        "footer  footer";
    }
}
```