Tag 2 CSS: Farben und Textgestaltung

Mit CSS **Texte ansprechend gestaltet** und **Farben effizient einsetzt**. Dazu zählen Schriftarten, Größen, Textausrichtung und verschiedene Farbmodelle sowie der Umgang mit Hintergründen und Transparenz.

---

1. Textgestaltung

1.1 `font-family`

Mit `font-family` legst du die Schriftart fest. Es ist üblich, mehrere Schriftarten anzugeben, falls eine nicht geladen werden kann. Am Ende steht eine generische Familie wie `sans-serif`.

```
body {
  font-family: "Arial", "Helvetica", sans-serif;
}
```

> **Tipp:** Verwende Web-sichere Schriftarten oder Webfonts wie Google Fonts für maximale Kompatibilität.

Google Fonts einbinden

Du kannst kostenlose Webfonts über **Google Fonts** nutzen.
So geht's:

1. Schritt: Font-Link in den `<head>` der HTML-Datei einfügen

Beispiel mit der Schrift **Raleway**:

```
<link href="https://fonts.googleapis.com/css2?family=Raleway:wght@400;700&display=swap"
rel="stylesheet">
```

> Diesen Link findest du auf https://fonts.google.com, wenn du eine Schrift auswählst.

---

2. Schritt: Schrift im CSS mit `font-family` verwenden

```
body {
  font-family: "Raleway", sans-serif;
}
```

1.2 `font-size`

Die Schriftgröße kann in verschiedenen Maßeinheiten angegeben werden:

- **px (Pixel):** Feste Größe
- **em:** Relativ zur Schriftgröße des Elternelements
- **rem:** Relativ zur Schriftgröße des Root-Elements (`html`)
- **%:** Prozentual zum Elternelement

```
p {
  font-size: 16px;
}

h1 {
```

```
  font-size: 2em; /* doppelt so groß wie Elternelement */
}

body {
  font-size: 100%; /* entspricht meist 16px */
}
```

> **Hinweis:** `rem` wird oft für konsistentes responsives Design bevorzugt.

1.3 `font-style`

```
p {
  font-style: italic;
}
```

Verfügbare Werte:

- `normal` (Standard)
- `italic` (kursiv)
- `oblique` (schräg gestellt)

1.4 `font-weight`

```
h1 {
  font-weight: bold;
}
```

Alternativen:

- `normal`, `bold`, `lighter`
- Werte zwischen `100` (sehr dünn) und `900` (sehr fett)

> Verwende konsistente Gewichtsstufen (`400` für normal, `700` für fett) in Kombination mit Google Fonts oder Systemschriften.

1.5 `text-align`

```
p {
  text-align: justify;
}
```

Werte:

- `left`, `right`, `center`, `justify`

> `justify` sollte mit ausreichend Zeilenlänge verwendet werden, da es sonst unschön wirken kann.

1.6 `text-decoration`

```
a {
  text-decoration: underline;
}
```

Weitere Werte:

- `none`, `overline`, `line-through`

1.7 `text-shadow` (Textschatten)

Mit `text-shadow` kannst du Textschatten hinzufügen, um Texte hervorzuheben oder plastischer wirken zu lassen.

```
h1 {
  text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.4);
}
```

Syntax:

```
text-shadow: horizontal-offset vertical-offset blur-radius color;
```

- `2px 2px` = Schattenversatz
- `4px` = Weichzeichnung
- `rgba(...)` = Farbe inkl. Transparenz

> Verwende Textschatten sparsam. Für barrierefreie Kontraste sollte er **nicht** als einziger Stil zur Hervorhebung verwendet werden.

---

2. Farben und Farbsysteme

2.1 Hexadezimal (`#`)

```
body {
  background-color: #3498db;
}
```

Werte reichen von `#000000` (schwarz) bis `#ffffff` (weiß).

2.2 RGB (`rgb()`)

```
p {
  color: rgb(255, 0, 0); /* Rot */
}
```

RGB = Rot, Grün, Blau mit Werten von 0–255.

2.3 HSL (`hsl()`)

```
p {
  color: hsl(120, 100%, 50%); /* Grün */
}
```

- **Hue** (Farbwinkel 0–360°)
- **Saturation** (Sättigung %)
- **Lightness** (Helligkeit %)

> HSL ist oft lesbarer und logischer beim Design mit harmonischen Farbabstufungen.

2.4 Transparenz mit Alpha-Wert (RGBA / HSLA)

```
div {
  background-color: rgba(52, 152, 219, 0.5);
}
```

Alpha = Deckkraft (0 = durchsichtig, 1 = undurchsichtig)

> Transparenz kann visuell sehr wirksam sein, sollte aber im Kontext von Lesbarkeit und Hintergrundkontrast geprüft werden.

---

3. Hintergrundgestaltung

3.1 `background-color`

```
body {
  background-color: #f0f0f0;
}
```

3.2 `background-image`

```
body {
  background-image: url('hintergrund.jpg');
}
```

> Bildpfade müssen korrekt und erreichbar sein (relative oder absolute Pfade).

3.3 `background-repeat`

```
body {
  background-repeat: no-repeat;
}
```

Werte:

- `repeat`, `no-repeat`, `repeat-x`, `repeat-y`

3.4 `background-size`

```
body {
  background-size: cover;
}
```

Werte:

- `cover`: füllt Fläche aus, kann beschneiden
- `contain`: alles sichtbar, aber evtl. leerer Raum

3.5 `background-position`

```
body {
  background-position: center;
}
```

Kombinierbar z. B. mit `top left`, `center center`, `bottom right`

3.6 `background-attachement`

```
body {
  background-attachment: fixed;
}
```

Werte:

- `scroll`: Bild scrollt mit dem Inhalt (Standard)
- `fixed`: Bild bleibt fest im Viewport (Parallax-Effekt möglich)
- `local`: Bild scrollt nur mit scrollbarem Inhalt eines Elements

> Achtung: `background-attachement: fixed;` kann auf mobilen Geräten eingeschränkt oder gar nicht unterstützt sein.

3.7 `background` Shorthand - Kurschreibweise

```
body {
  background: #f0f0f0 url('hintergrund.jpg') no-repeat center / cover fixed;
}
```

Diese Zeile ersetzt alle obigen Eigenschaften:

Bestandteil	Wirkung
`#f0f0f0`	Hintergrundfarbe
`url('hintergrund.jpg')`	Bilddatei
`no-repeat`	Keine Wiederholung
`center`	Position
`/ cover`	Größe
`fixed`	Attachment (Scrollverhalten)

> Die Angabe von Position und Größe muss mit / getrennt werden: `center / cover`

3.8 Farbverläufe (Gradients)

```
body {
  background: linear-gradient(45deg, #3498db, #2ecc71);
}
```

Alternativen:

- `radial-gradient()`

4. Übungsaufgaben – Farben und Text

Aufgabe 1: Schriftarten und Größen

Erstelle einen Absatz mit:

- einer Schriftart deiner Wahl
- einer festen Größe in `px`
- einer alternativen Version mit `em` oder `rem`

Aufgabe 2: Textgestaltung

Formatier mehrere Absätze:

- einer soll fett (`font-weight`),
- einer kursiv (`font-style`),
- einer unterstrichen sein (`text-decoration`)

Aufgabe 3: Textausrichtung

Erstelle drei Boxen mit unterschiedlichem Text und wende je eine Ausrichtung (`left`, `center`, `right`) an. Nutze `text-align`.

Aufgabe 4: Farbformate vergleichen

Erstelle drei Absätze mit jeweils:

- einer Farbe in Hex (`#`)
- einer Farbe in RGB (`rgb()`)
- einer Farbe in HSL (`hsl()`)

Aufgabe 5: Transparente Hintergründe

Erstelle ein Element mit `rgba()` oder `hsla()`, das halbtransparent ist. Füge darunter ein Bild oder einen farbigen Hintergrund ein.

Aufgabe 6: Gradient erstellen

Erstelle ein `<div>` mit einem Farbverlauf von Blau zu Grün. Verwende `linear-gradient()`.

Aufgabe 7: Textschatten kreativ einsetzen

Erstelle eine große Überschrift (`<h1>`) mit einem deutlichen Textschatten. Teste verschiedene Versatz- und Weichzeichnungswerte.