# Tag 3 CSS: Web-Layout mit Floating-Elementen & Media Queries

`float` und `media queries` sind zwei klassische Techniken um Webseiten zu layouten und responsive zu gestalten . Auch wenn heute moderne Methoden wie **Flexbox** und **Grid** dominieren, solltest du Float und Media Queries kennen und verstehen, da sie immer noch in vielen Projekten vorkommen.

---

## 1. Floating-Elemente

### a) Was ist `float`?

`float` lässt ein Element zur linken oder rechten Seite fließen. Ursprünglich für Textumfluss bei Bildern gedacht, später oft für Layouts verwendet.

```
<img src="bild.jpg" style="float:left; margin-right:10px;">
<p>Text fließt um das Bild herum...</p>
```

> Float nimmt das Element aus dem normalen Dokumentenfluss – das kann zu Problemen führen.

---

### b) Werte von `float`

- `left`: Fließt nach links
- `right`: Fließt nach rechts
- `none`: Kein Float (Standard)
- `inherit`: Erbt Float-Wert vom Elternteil

---

### c) Float für mehrspaltige Layouts

```
<div class="left" style="float: left; width: 30%;">Links</div>
<div class="right" style="float: right; width: 70%;">Rechts</div>
```

> Zwei Spalten nebeneinander durch gegensätzliches Floaten und feste Breiten

---

### d) Float-Clearing

Gefloatete Elemente werden vom Eltern-Container ignoriert.

**Lösung:** `clearfix`-Hack

```
.clearfix::after {
  content: "";
  display: table;
  clear: both;
}
```

Oder mit `overflow: auto` auf dem Container:

```
.container {
  overflow: auto;
}
```

### e) Nachteile von Float

- Kein vertikales Zentrieren
- Umständliches Clearing
- Fragile Struktur

> Heute `float` nur noch für Textumfluss verwenden. Für Layouts: `Flexbox` oder `Grid`.

---

## 2. Media Queries

### a) Was sind Media Queries?

Media Queries passen das Layout an Geräteeigenschaften an (Breite, Höhe, Auflösung). Basis für **Responsive Design**.

### b) Syntax

```
@media (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

### c) Varianten

- **Langform:** `@media screen and (max-width: 768px)`
- **Kurzform:** `@media (max-width: 768px)` → meist ausreichend für Screens

### d) Anwendungsbeispiele

1. **Zweispalten-Layout auf einspaltig umstellen:**

```
@media (max-width: 768px) {
  .left, .right {
    float: none;
    width: 100%;
  }
}
```

2. **Bildgröße anpassen:**

```
@media (max-width: 600px) {
  img {
    width: 80%;
  }
}
```

3. **Schriftgröße skalieren:**

```
@media (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

---

### e) Gängige Breakpoints

- `1200px`: Desktops
- `992px`: Tablets quer
- `768px`: Tablets hoch
- `576px`: Smartphones

> Passe Layout bei typischen Gerätegrößen an.

---

### f) Mobile First Ansatz

Start mit mobiler Ansicht → danach Erweiterung für größere Bildschirme mit `min-width`

```
/* Basis für Smartphones */
.container {
  width: 100%;
}

/* Erweiterung für Tablets/Desktop */
@media (min-width: 768px) {
  .container {
    width: 50%;
  }
}
```

> Vorteil: Optimierung für die meistgenutzten Geräte (mobil)

---

### g) Weitere gebräuchliche Media Queries

Neben der Bildschirmbreite (`width`) gibt es viele weitere sinnvolle Eigenschaften, auf die du mit Media Queries reagieren kannst:

**Gerätespezifische Eigenschaften**

- `(max-width: 768px)` – typische Tablets
- `(min-width: 1024px)` – größere Desktops
- `(orientation: portrait)` – Hochformat
- `(orientation: landscape)` – Querformat

**Benutzerpräferenzen**

- `(prefers-color-scheme: dark)` – erkennt Dark Mode-Einstellung des Systems

```
@media (prefers-color-scheme: dark) {
  body {
    background-color: #121212;
    color: #ffffff;
  }
}
```

- `(prefers-reduced-motion: reduce)` – erkennt, ob Nutzer Animationen vermeiden möchten

```
@media (prefers-reduced-motion: reduce) {
  * {
    transition: none;
    animation: none;
  }
}
```

**Weitere nützliche Media Features**

- `(resolution: 300dpi)` – für hochauflösende Bildschirme (z. B. Retina)
- `(hover: none)` – Touchgeräte, auf denen kein Hover möglich ist
- `(pointer: coarse)` – erkennt grobe Eingabegeräte wie Finger

> Teste Media Queries in der Dev-Console deines Browsers mit verschiedenen Geräteeinstellungen und Simulationen.

---

## 3. Float + Media Query kombiniert

```
<div class="container">
  <div class="left">Links</div>
  <div class="right">Rechts</div>
</div>
```

```
.left {
  float: left;
  width: 70%;
}
.right {
  float: right;
  width: 30%;
}

@media (max-width: 768px) {
  .left, .right {
    float: none;
    width: 100%;
  }
}
```

## 4. Übungsaufgaben – Float & Media Queries

### Aufgabe 1: Float-Layout bauen

Erstelle zwei Boxen nebeneinander mit `float: left/right` und `width: 50%`. Teste mit Inhalt.

### Aufgabe 2: Clearfix anwenden

Erstelle drei gefloatete Boxen innerhalb eines Containers. Stelle sicher, dass der Container korrekt die Höhe hat. Verwende `::after`-Clearfix oder `overflow: auto`.

### Aufgabe 3: Bild und Text kombinieren

Baue ein Beispiel mit einem Bild `float: left` und daneben Text. Teste mit Rand (`margin-right`).

### Aufgabe 4: Responsive Schriftgröße

Passe die Schriftgröße für Viewports unter `600px` an (z. B. `font-size: 14px`).

### Aufgabe 5: Layout-Umschaltung mit Media Query

Erstelle zwei Boxen (z. B. `.left` & `.right`) mit `float`, die sich unter `768px` auf `100%` Breite ändern.

### Aufgabe 6: Mobile First testen

Baue ein einfaches Layout, das auf kleinen Bildschirmen einspaltig ist und ab `768px` zweispaltig wird. Nutze `min-width`.