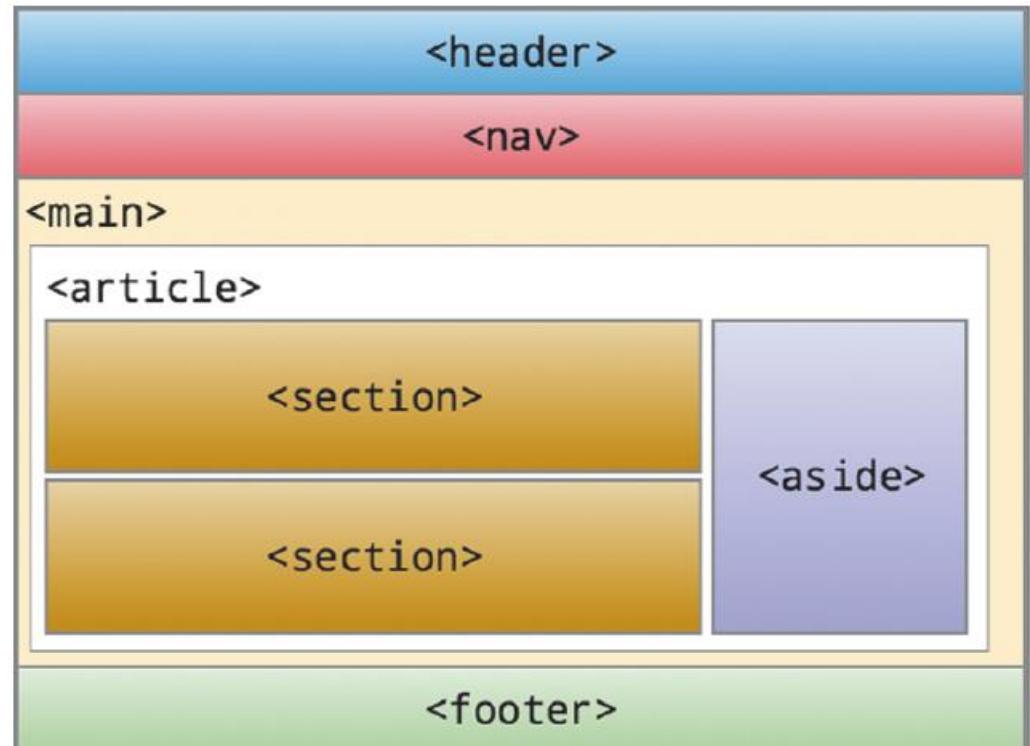


Seitenlayout



Standard-Seitenlayout mit semantischen Elementen

```
<header>
  
  <h1>Site Title</h1>
</header>
<nav>
  <ul>
    <li>Item 1</li>
  </ul>
</nav>
<main>
  <article>
    <section>
      <h2>Article Title</h2>
      <p>Article paragraph</p>
    </section>
    <aside>
      <p>Sidebar paragraph</p>
    </aside>
  </article>
</main>
<footer>
  <p>Footer paragraph</p>
</footer>
```



CSS-Attribut *display*

Mögliche Werte z.B.

- block (erzeugt einen Block)
 - Beschreibung:** Ein Block-Element erstreckt sich über die gesamte Breite seines übergeordneten Containers, wodurch es automatisch Zeilenumbrüche vor und nach sich erzeugt.
 - Typische Beispiele:** <div>, <p>, <h1>, <section>
 - inline (erzeugt ein Inline-Element)
 - Beschreibung:** Ein Inline-Element nimmt nur so viel Platz wie nötig ein und beeinflusst nicht den Fluss umgebender Elemente. Es verursacht keine Zeilenumbrüche.
 - Typische Beispiele:** , <a>, , .
 - inline-block (erzeugt einen Block im Textfluss)
 - **Beschreibung:** Ein Element, das sowohl Block- als auch Inline-Eigenschaften besitzt. Es verhält sich wie ein Inline-Element (kein Zeilenumbruch), erlaubt aber das Setzen von Breite und Höhe.
-

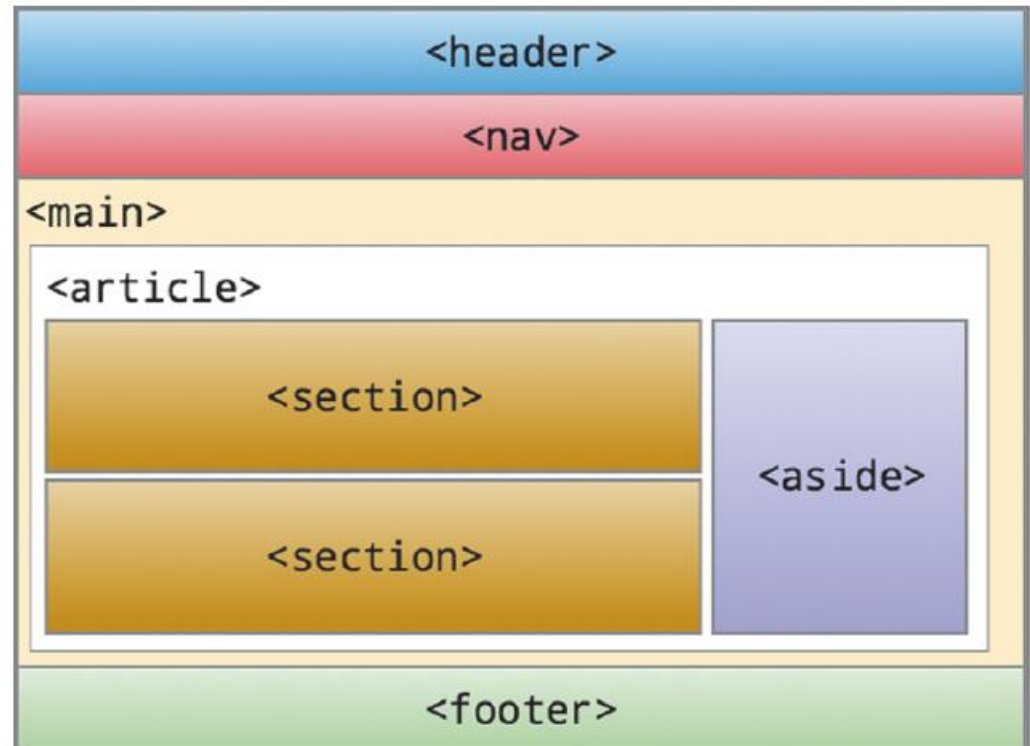
CSS-Attribut *display*

- none (Element unsichtbar, kein Platzhalter)
 - **Beschreibung:** Das Element wird vollständig aus dem Layout entfernt, ohne dass ein Platzhalter hinterlassen wird. Es ist weder sichtbar noch nimmt es Platz ein.
 - **Typische Verwendung:** Oft verwendet, um Elemente bedingt auszublenden.
 - flex (erzeugt Flex-Container als Block)
 - **Beschreibung:** Erzeugt einen Flex-Container, der eine flexible Anordnung der Kinder-Elemente ermöglicht. Flexbox ist ideal für Layouts, die sowohl horizontal als auch vertikal ausgerichtet werden müssen
 - inline-flex (erzeugt Flex-Container als Inline-Element)
 - **Beschreibung:** Ein Flex-Container, der sich wie ein Inline-Element verhält, aber die Flexbox-Eigenschaften beibehält.
-

Herausforderungen beim Standard-Layout

Inhalte nebeneinander:

- im header
- im nav-Bereich
- im article



Wie bekomme ich die Inhalte nebeneinander?

Ältere Ansätze:

- Blöcke mit *float* formatieren
 - **Beschreibung:** Mit float können Elemente nach links oder rechts "schweben". Der Text und andere Inline-Elemente fließen um das gefloatete Element herum.
 - **Nachteile:**

Umständlich für komplexe Layouts, da es oft zu Problemen mit "überlappenden" Elementen kommt.

Erfordert den Einsatz von clear-Techniken, um Layout-Probleme zu vermeiden.

- Blöcke zu Inline-Blöcken machen (*display:inline-block*)
 - **Beschreibung:** Diese Methode ermöglicht es Block-Elementen, wie Inline-Elemente zu agieren und nebeneinander angezeigt zu werden. Anders als float bleibt das Layout sauberer und einfacher zu kontrollieren.
 - **Nachteile:** Der Abstand zwischen Inline-Block-Elementen kann durch den Zeilenumbruch im HTML-Code beeinflusst werden.
-

Wie bekomme ich die Inhalte nebeneinander?

Neuerer Ansatz:

- Blöcke zu Flexboxen machen (*display:flex*)
 - **Beschreibung:** Flexbox ist ein leistungsfähiges Layout-Modell, das die flexible Anordnung von Elementen sowohl in horizontaler als auch in vertikaler Richtung ermöglicht. Flexbox löst viele der Probleme, die mit älteren Layout-Techniken wie float und inline-block verbunden sind.
 - **Vorteile:**
 - Flexibel und dynamisch: Elemente können wachsen, schrumpfen und neu angeordnet werden, ohne das HTML ändern zu müssen.
 - Einfachere Vertikale Zentrierung.
 - Ideal für responsive Designs
-

Wie bekomme ich die Inhalte nebeneinander?

Zukünftiger Ansatz:


Grid Layout (*display:grid*)

Beschreibung: CSS Grid ist ein zweidimensionales Layout-System, das sowohl die Anordnung in Zeilen als auch in Spalten ermöglicht. Es eignet sich besonders für komplexe Layouts, die mehr als nur eine einfache Zeilen- oder Spaltenstruktur erfordern.

Vorteile:

- Sehr mächtig und flexibel, besonders bei der Erstellung von komplexen Layouts.
 - Bietet explizite Kontrolle über das Layout.
 - Einfaches Umstrukturieren von Layouts für verschiedene Bildschirmgrößen (responsive Design).
-

Aufgabe: Kapitel 11 (zu den älteren Ansätzen)

 WEB DESIGN PLAYGROUND

Run Code ▶ MENU ≡

11.1: Creating the Holy Grail Layout with Floats

◀ Previous Page Next Page ▶

Lesson 11.1

Creating the Holy Grail Layout with Floats

Introduction

The holy grail includes three instances where we need content side-by-side:

- In the header, you'll usually want a site title beside the site logo.
- In the navigation bar, you'll usually want the navigation items to appear in a row.
- The sidebar has to appear to the right of the main content.

All of these instances require the use of the `float` property to get the elements out of the default page flow and rendered beside each other.

HTML

CSS

Hide Editors New Sandbox

Geschachtelte div-Container: Standard-Verhalten

≡

HTML

```
1 <div class="container">
2   <div class="item itemA">A</div>
3   <div class="item itemB">B</div>
4   <div class="item itemC">C</div>
5   <div class="item itemD">D</div>
6   <div class="item itemE">E</div>
7   <div class="item itemF">F</div>
8 </div>
```

≡

CSS

```
1 * {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5   font-size: 100%;
6 }
7
8 /*.container {
9   display: flex;
10  flex-direction: row;
11 }*/
12
13 .item {
14   padding: .1em;
15   color: white;
16   font-family: "Verdana";
```

A

B

C

D

E

F

Eltern-Container als Flexbox

flex-direction:

- **Beschreibung:** Diese Eigenschaft definiert die Hauptachse des Flex-Containers, entlang derer die Kinder-Elemente ausgerichtet werden.
 - **Mögliche Werte:**
 - row: Die Kinder-Elemente werden in einer Zeile von links nach rechts (Standard) angeordnet.
 - row-reverse: Die Kinder-Elemente werden in einer Zeile von rechts nach links angeordnet.
 - column: Die Kinder-Elemente werden in einer Spalte von oben nach unten angeordnet.
 - column-reverse: Die Kinder-Elemente werden in einer Spalte von unten nach oben angeordnet.
 - Beispiel: https://www.w3schools.com/cssref/tryit.php?filename=trycss3_flex-direction
-

Eltern-Container als Flexbox

flex-direction: row | row-reverse | column | column-reverse; (Bestimmung der Hauptachse)



HTML

```
1 <div class="container">
2   <div class="item itemA">A</div>
3   <div class="item itemB">B</div>
4   <div class="item itemC">C</div>
5   <div class="item itemD">D</div>
6   <div class="item itemE">E</div>
7   <div class="item itemF">F</div>
8 </div>
```



CSS

```
1 * {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5   font-size: 100%;
6 }
7
8 .container {
9   display: flex;
10  flex-direction: row;
11 }
12
13 .item {
14   padding: .1em;
15   color: white;
16   font-family: "Verdana";
```

A B C D E F

Flexbox-Attribut zur Verteilung der "Kinder" (*Items*)

justify-content:

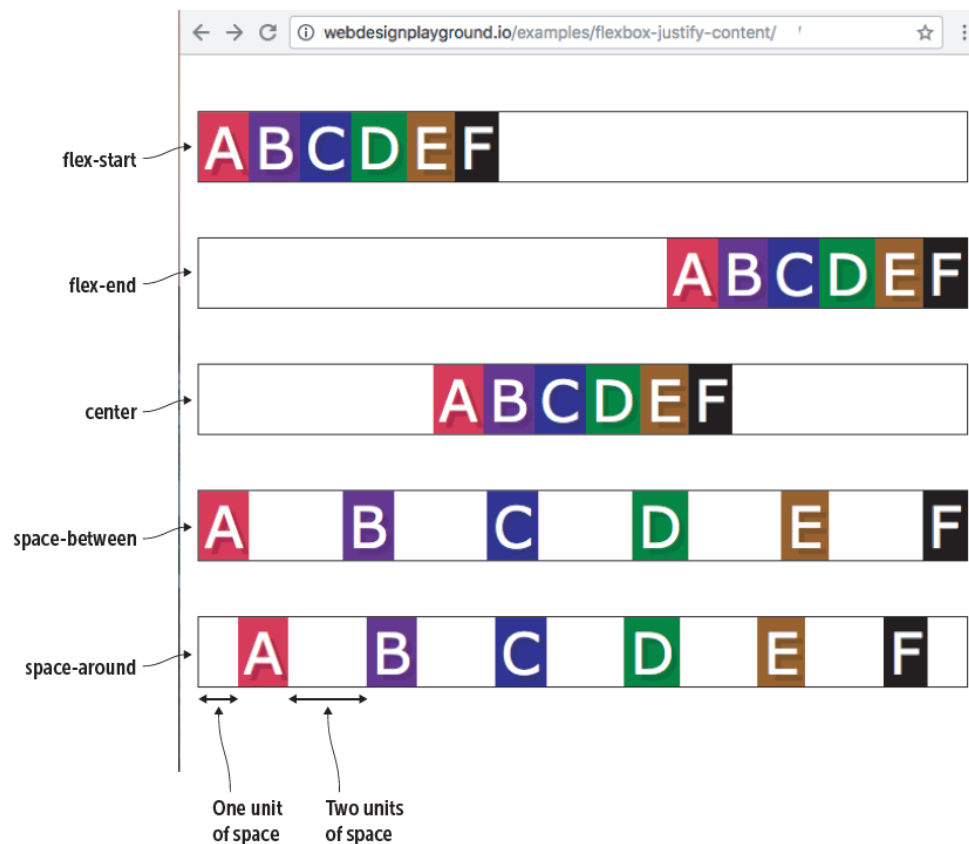
• **Beschreibung:** Diese Eigenschaft steuert die Verteilung der Kinder-Elemente entlang der Hauptachse des Flex-Containers.

• Mögliche Werte:

- flex-start: Die Elemente werden am Anfang der Hauptachse ausgerichtet.
 - flex-end: Die Elemente werden am Ende der Hauptachse ausgerichtet.
 - center: Die Elemente werden in der Mitte der Hauptachse zentriert.
 - space-between: Die Elemente werden gleichmäßig verteilt, das erste und letzte Element befinden sich am Rand.
 - space-around: Die Elemente werden gleichmäßig verteilt, mit gleichen Abständen um jedes Element.
-

Flexbox-Attribut zur Verteilung der "Kinder" (*Items*)

`justify-content: flex-start | flex-end | center | space-between | space-around;`



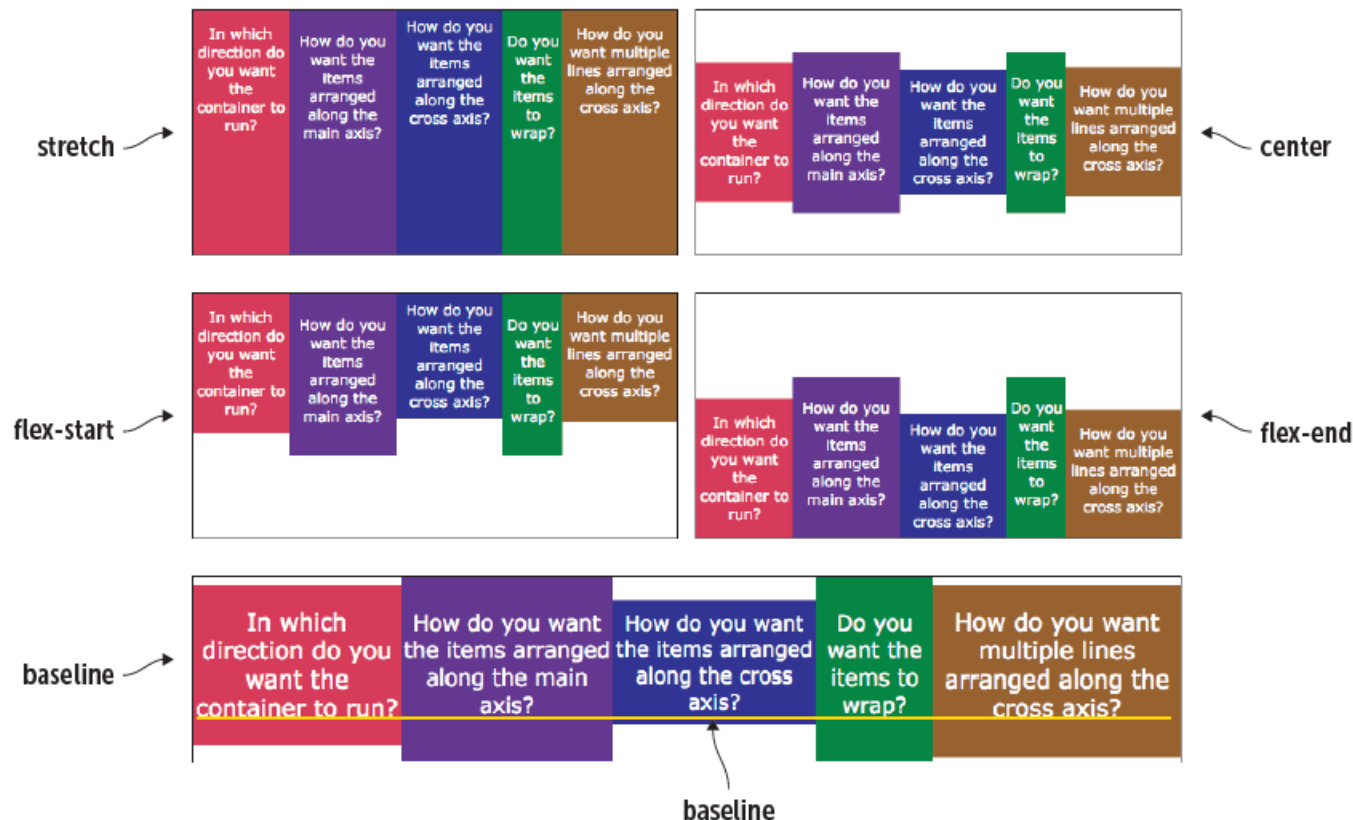
Flexbox-Attribut zur Ausrichtung entlang der orthogonalen Achse

align-items:

- **Beschreibung:** Diese Eigenschaft bestimmt die Ausrichtung der Kinder-Elemente entlang der Querachse des Flex-Containers.
 - **Mögliche Werte:**
 - stretch: Standardwert, bei dem die Elemente gedehnt werden, um den gesamten Container entlang der Querachse auszufüllen.
 - flex-start: Die Elemente werden am Anfang der Querachse ausgerichtet.
 - flex-end: Die Elemente werden am Ende der Querachse ausgerichtet.
 - center: Die Elemente werden entlang der Querachse zentriert.
 - baseline: Die Elemente werden an ihrer Textbasislinie ausgerichtet.
-

Flexbox-Attribut zur Ausrichtung entlang der orthogonalen Achse

align-items: stretch | flex-start | flex-end | center | baseline;



Attribute für Flexbox-Items (also die "Kinder")

flex-grow:

- **Beschreibung:** Bestimmt, wie viel ein Element im Verhältnis zu den anderen wachsen kann, wenn zusätzlicher Platz verfügbar ist.

flex-shrink:

- **Beschreibung:** Bestimmt, wie viel ein Element im Verhältnis zu den anderen schrumpfen kann, wenn weniger Platz verfügbar ist.

flex-basis:

- **Beschreibung:** Bestimmt die Basisgröße des Elements, bevor die flex-grow und flex-shrink-Eigenschaften angewendet werden.

flex:

- **Beschreibung:** Eine Kurzform für flex-grow, flex-shrink und flex-basis.

order:

- **Beschreibung:** Bestimmt die Reihenfolge der Elemente unabhängig von der Reihenfolge im HTML-Dokument.

align-self:

- **Beschreibung:** Überschreibt die align-items-Eigenschaft für ein einzelnes Kind und ermöglicht so eine individuelle Ausrichtung.
-

Attribute für Flexbox-Items (also die "Kinder")

flex-grow: *value*

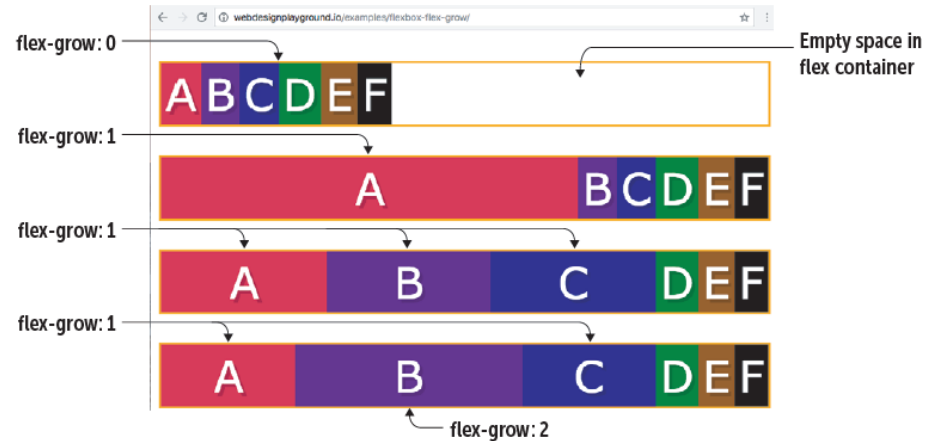
flex-shrink: *value*

flex-basis: *value* | auto | content


flex: 0 1 auto (*zusammengefasst*)

order: *value*


align-self: stretch | flex-start | flex-end | center | baseline



Beispiel: Menüzeile

WEB PAGE	
CSS	<pre>nav { padding: .5em; border: 1px solid black; } nav ul { display: flex; justify-content: flex-start; list-style-type: none; padding-left: .5em; } nav li { padding-right: 1.5em; }</pre> <p>Display the ul element as a flex container.</p>
HTML	<pre><nav> Home Item Item Item </nav></pre>

Aufgabe: Kapitel 12

 WEB DESIGN PLAYGROUND

Run Code ▶ MENU ≡

12.1: Working with Flexbox Containers

◀ Previous PageNext Page ▶

Lesson 12.1

Working with Flexbox Containers

In which direction do you want the container to run?

Flexbox containers always have two axes: the *main axis* runs in the same direction as the container's items, and the *secondary axis* runs perpendicular to the main axis. You determine the main axis direction when you choose a `flex-direction` value for the container:

☒ `row` (main axis is horizontal, left-to-right)

☐ `row-reverse` (main axis is horizontal, right-to-left)

☐ `column` (main axis is vertical, top-to-bottom)

☐ `column-reverse` (main axis is vertical, bottom-to-top)

HTML

```
1 <div class="container">
2   <div class="item itemA">A</div>
3   <div class="item itemB">B</div>
4   <div class="item itemC">C</div>
5   <div class="item itemD">D</div>
6   <div class="item itemE">E</div>
7   <div class="item itemF">F</div>
8 </div>
```

CSS

```
1 * {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5   font-size: 100%;
6 }
7
8 .container {
9   display: flex;
10  flex-direction: row;
11 }
12
13 .item {
14   padding: .1em;
15   color: white;
16   font-family: "Vardana";
```

ABCDEF

Hide EditorsNew Sandbox



Hausaufgabe

Gestalten Sie das Hauptmenü Ihrer Website mithilfe von Flexboxen.

Responsives Design



CSS Media Queries

Syntax für media queries:

```
@media (expression) {  
    selector {  
        declarations  
    }  
    etc.  
}
```

Bedeutung: Wenn die Bedingung *expression* gilt, werden die Formate angewandt.

Was ist Responsive Design?

Inhalt:

- **Definition:** Responsive Design ist ein Ansatz im Webdesign, der darauf abzielt, Websites so zu gestalten, dass sie auf verschiedenen Geräten und Bildschirmgrößen optimal dargestellt werden.
- **Ziel:** Sicherstellen, dass eine Website auf Desktops, Tablets und Smartphones gleichermaßen gut funktioniert und benutzerfreundlich ist.

Visualisierung:

Eine Illustration oder ein Bild, das verschiedene Geräte (Desktop, Tablet, Smartphone) zeigt, auf denen eine Website angezeigt wird.

Warum ist Responsive Design wichtig?

Inhalt:

- **Vielzahl von Geräten:** Benutzer greifen auf Websites über eine Vielzahl von Geräten zu – von großen Monitoren bis hin zu kleinen Smartphone-Bildschirmen.
 - **SEO-Vorteile:** Google priorisiert mobilfreundliche Websites in den Suchergebnissen.
 - **Bessere Benutzererfahrung:** Eine responsive Website sorgt für eine konsistente und angenehme Benutzererfahrung, unabhängig vom Gerät.
-

Was sind Media Queries?

Inhalt:

- **Definition:**

Media Queries sind eine CSS-Technik, die es ermöglicht, verschiedene Stile für verschiedene Bildschirmgrößen und -eigenschaften anzuwenden.

- **Ziel:**

Das Ziel von Media Queries ist es, das Layout und Design einer Webseite dynamisch an die Eigenschaften des Geräts anzupassen, auf dem sie betrachtet wird.



Wie funktionieren Media Queries?

Inhalt:

- **Grundstruktur:**

Media Queries nutzen Schlüsselwörter wie `min-width` und `max-width`, um bestimmte Stile nur dann anzuwenden, wenn die Bildschirmgröße diese Bedingungen erfüllt.

```
@media (max-width: 600px) {  
  
  body {  
  
    background-color: lightblue;  
  
  }  
  
}
```

Erklärung:

Dieser Code ändert die Hintergrundfarbe des `body` auf hellblau, wenn die Breite des Bildschirms 600px oder weniger beträgt.

Anwendung auf verschiedene Bildschirmgrößen:

Media Queries können verwendet werden, um Stile für verschiedene Gerätegrößen wie Smartphones, Tablets und Desktops zu definieren.

Wichtige Werte in Media Queries

Inhalt

- Extra kleine Geräte:
 - Breite: bis zu 576px
 - Anwendung: Für Smartphones im Hochformat
 - Kleine geräte:
 - Breite: 576px bis 768px
 - Anwendung: Für größere Smartphones und Tablets im Hochformat.
 - Mittlere Geräte:
 - Breite: 768px bis 992px
 - Anwendung: Für Tablets im Querformat und kleine Laptops.
 - Große Geräte:
 - Breite: 992px und größer
 - Anwendung: Für Desktops und große Laptops.
-

Beispiel

► **Example** ➡ Online: wdpg.io/13-4-1

This code uses a media query to remove the floats from the article and aside elements, as well as perform a few other tasks as noted.

CSS

```
h1 {
  float: left;
  font-size: 32px;
}
main {
  background-color: #b4a7d6;
}
article {
  float: left;
  width: 66.67%;
}
aside {
  float: left;
  width: 33.33%;
}
@media (max-width: 450px) {
  article {
    float: none;
    width: 100%;
  }
  aside {
    float: none;
    width: 100%;
  }
  h1 {
    font-size: 24px;
  }
  main {
    background-color: white;
  }
}
```

The media query applies to screen widths up to 450px.


Floats are removed from the article and aside elements.

The page title is reduced to 24px.

The main element background color is changed to white.



Aufgabe: Kapitel 13

 WEB DESIGN PLAYGROUND

Run Code ▶ MENU ≡

13.1: Why Fixed-Width Layouts Are the ...

◀ Previous Page

Next Page ▶

Lesson 13.1

Why Fixed-Width Layouts Are the Enemy

Introduction

Why don't web pages just fit whatever screen they're being displayed on? In most cases, the culprit is the use of large, fixed-width elements. These elements stay the same size no matter how wide a screen they're shown on, so if their width is greater than that of the screen, the dreaded horizontal scrollbar appears.

HTML

1


CSS

1

Hide Editors

New Sandbox

Aufgabe: Kapitel 14

 WEB DESIGN PLAYGROUND

Run Code ▶ MENU ≡

14.1: Creating Fluid Images

◀ Previous Page

Next Page ▶

Lesson 14.1

Creating Fluid Images

Introduction

An image comes with a predetermined width and height, so at first blush it seems impossible to overcome these fixed dimensions. Fortunately, an `` tag is just another page element. That means that, yes, by default the image is displayed at its full width and height, just like a `div` or any other block element. But in the same way that you can make a block element fluid by using percentages, you can also make an image fluid.

HTML

1

CSS

1

Hide Editors

New Sandbox



Hausaufgabe

Schaffen Sie ein responsives Seitenlayout für Ihre Website.