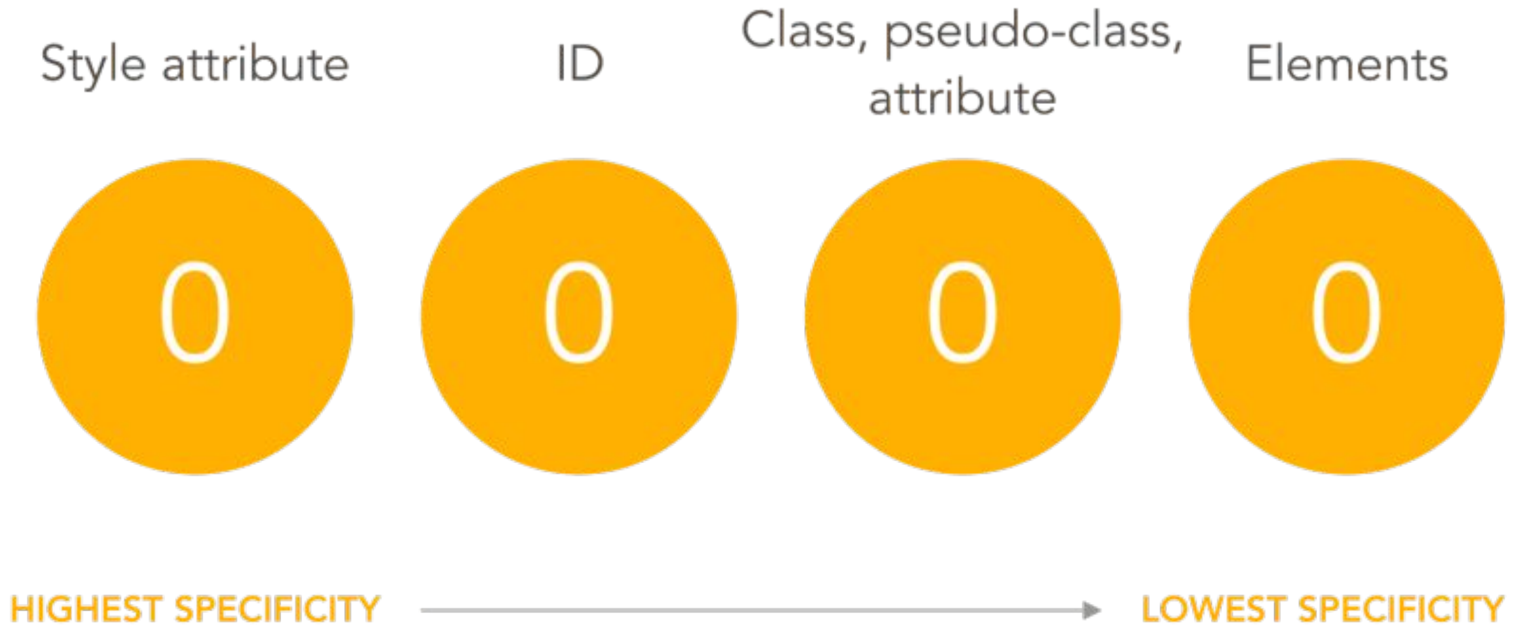
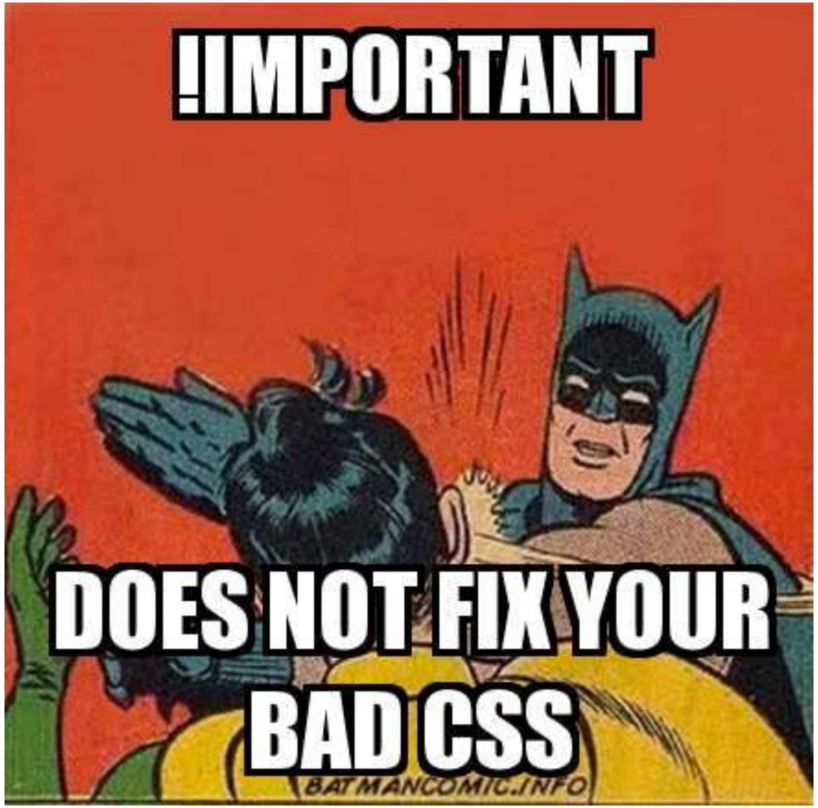
Web development basics

CSS specificity, responsiveness, transitions, animations

CSS specificity



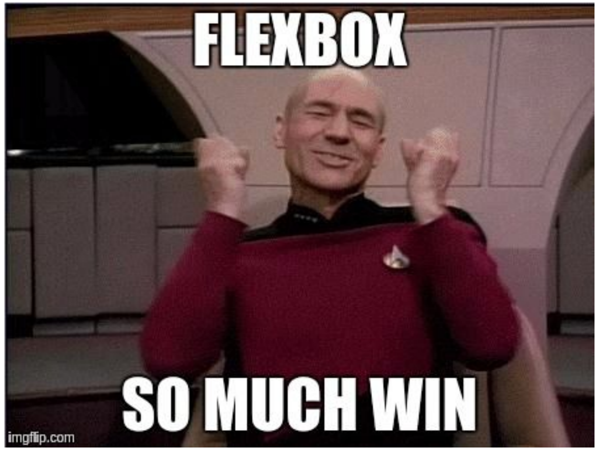
CSS specificity !important

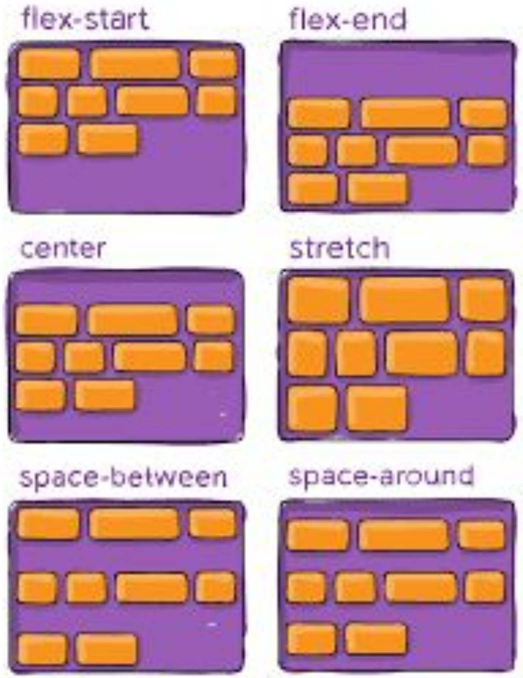


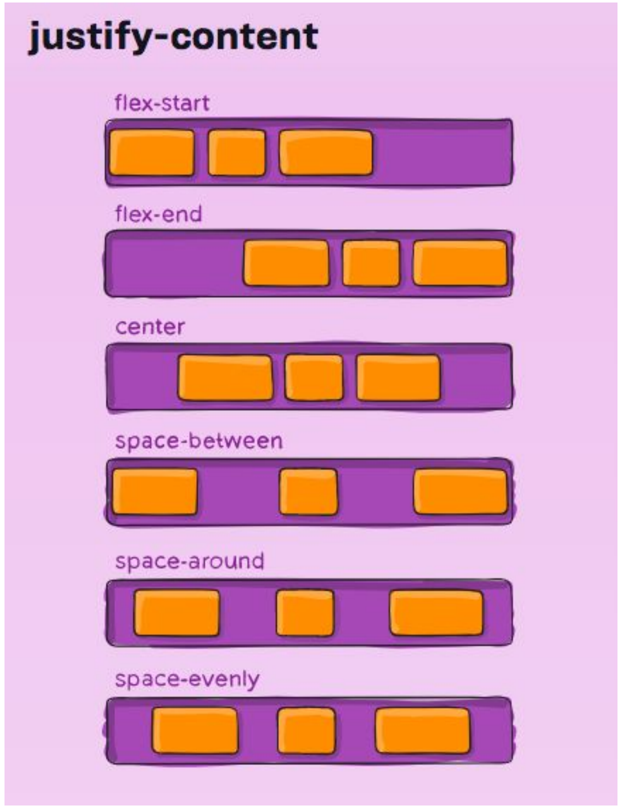
CSS flexbox

CSS Flexbox is a **layout module** in CSS that makes it easy to create flexible, responsive layout structures without using floats or positioning. It allows you to align elements horizontally and vertically, distribute space between elements, and reorder elements within a container. Flexbox is particularly useful when creating **responsive design**, as it allows you to control how elements are displayed on different screen sizes and orientations. To use flexbox, you first create a container element and set its **display** property to "**flex**". Then, you can use various flexbox properties to control the layout of the child elements within that container.

CSS flexbox 2







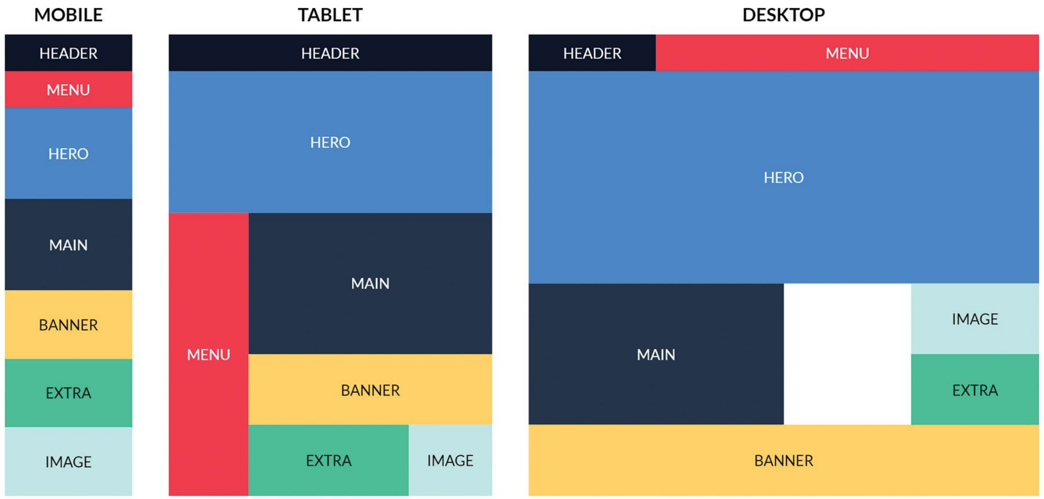
CSS grid

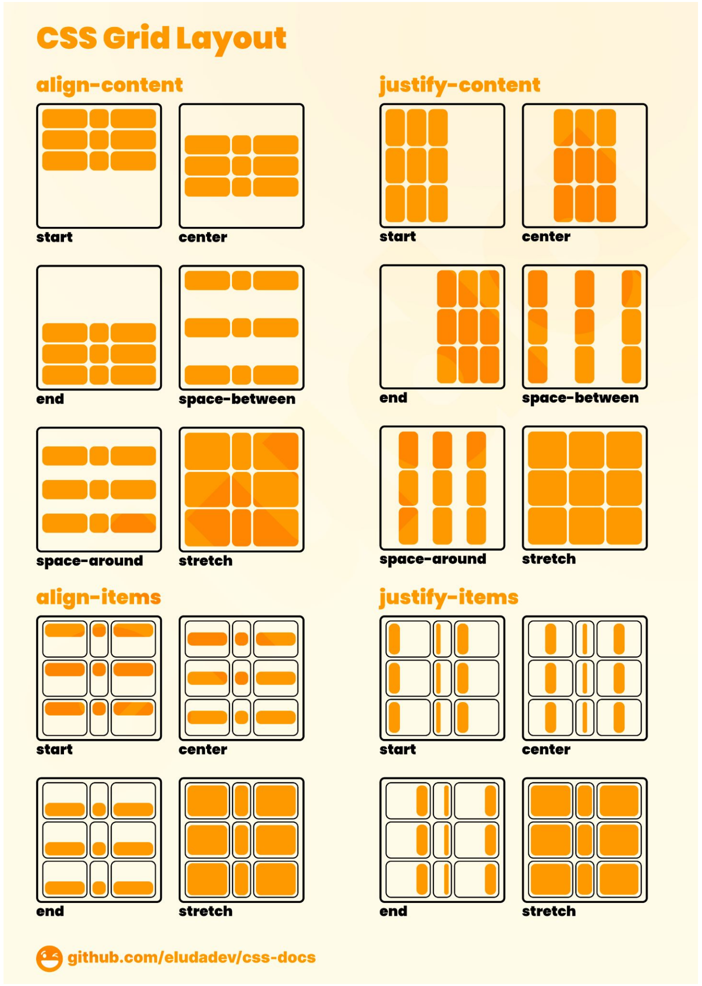
CSS Grid is a two-dimensional layout system for the web that allows you to create **rows** and **columns** to organize elements on a webpage. It is designed to make it easy to create complex, responsive layouts with minimal code. With CSS Grid, you can define a **grid container** and then specify how many **rows** and **columns** the grid should have, and how the elements within the grid should be placed. You can also control the **size** of rows and columns, and specify how elements should span multiple rows or columns. CSS Grid also allows you to control the alignment of elements within the grid and how the grid should respond when the **viewport size** changes.

One of the main advantages of CSS Grid is that it allows you to create layouts that are not possible with traditional CSS techniques such as **floats** and **positioning**. Grid also make it easy to create **responsive design** as it allows you to control how elements are displayed on **different screen sizes and orientations**.

To use CSS Grid, you first create a container element and set its **display** property to "**grid**". Then, you can use various grid properties to define the structure of the grid and the placement of the child elements within it.

CSS grid 2

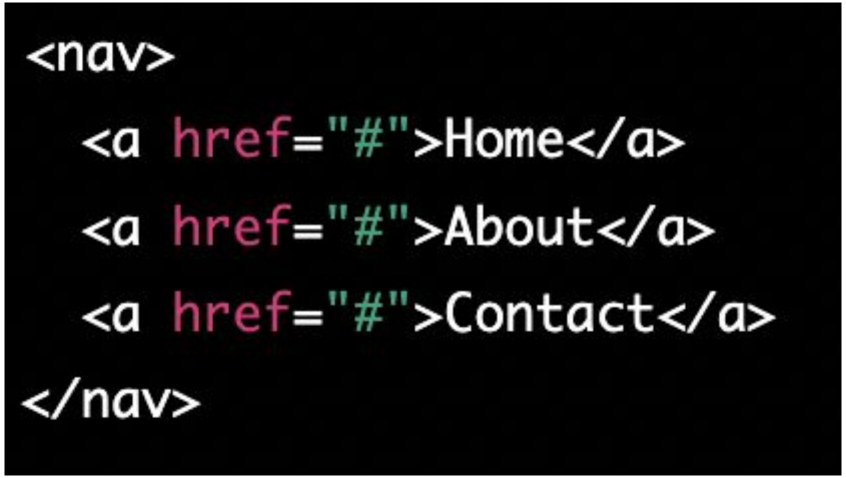

MOBILE
HEADER
MENU
HERO
MAIN
BANNER
EXTRA
IMAGE
TABLET
HEADER
HERO
MENU
MAIN
BANNER
EXTRA
IMAGE
DESKTOP
HEADER
MENU
HERO
MAIN
IMAGE
EXTRA
BANNER


CSS Grid Layout
align-content
start
center
end
space-between
space-around
stretch
justify-content
start
center
end
space-between
space-around
stretch
align-items
start
center
end
stretch
justify-items
start
center
end
stretch
github.com/eludadev/css-docs

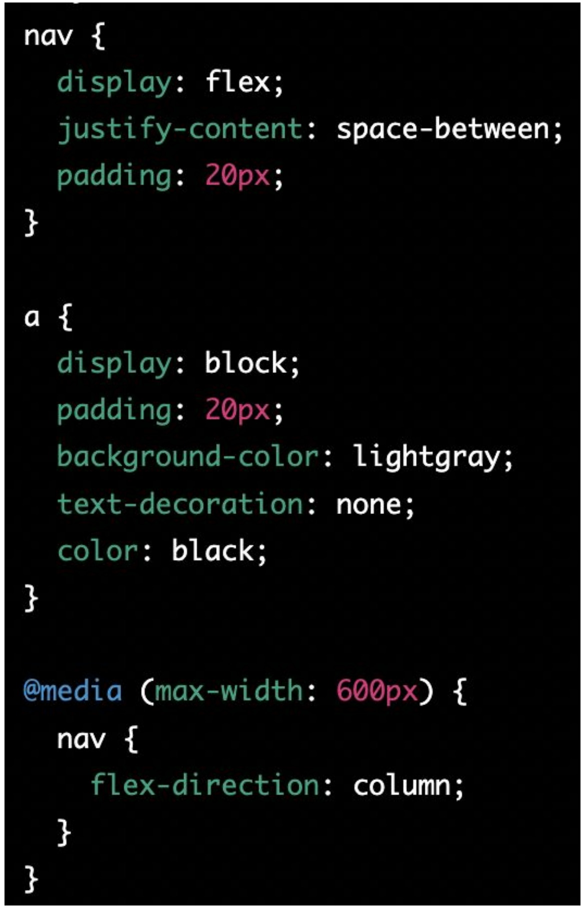
CSS flex classwork

1) Create an HTML file with a nav element to serve as the navigation bar container.
2) Inside the nav element, create a list of a elements to represent each navigation item.
3) Set the nav element's display property to flex and its justify-content property to space-between to evenly distribute the navigation items across the container
4) Add styles to the a elements to make them look like navigation items (e.g. display as block elements and give them padding and a background color).
5) Use media queries to adjust the flex-direction property of the nav element to change the orientation of the navigation items on different screen sizes **(optional)**
6) Preview your changes in the browser and adjust the styles as needed.

CSS flex classwork - solution



```html
<nav>
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Contact</a>
</nav>
```

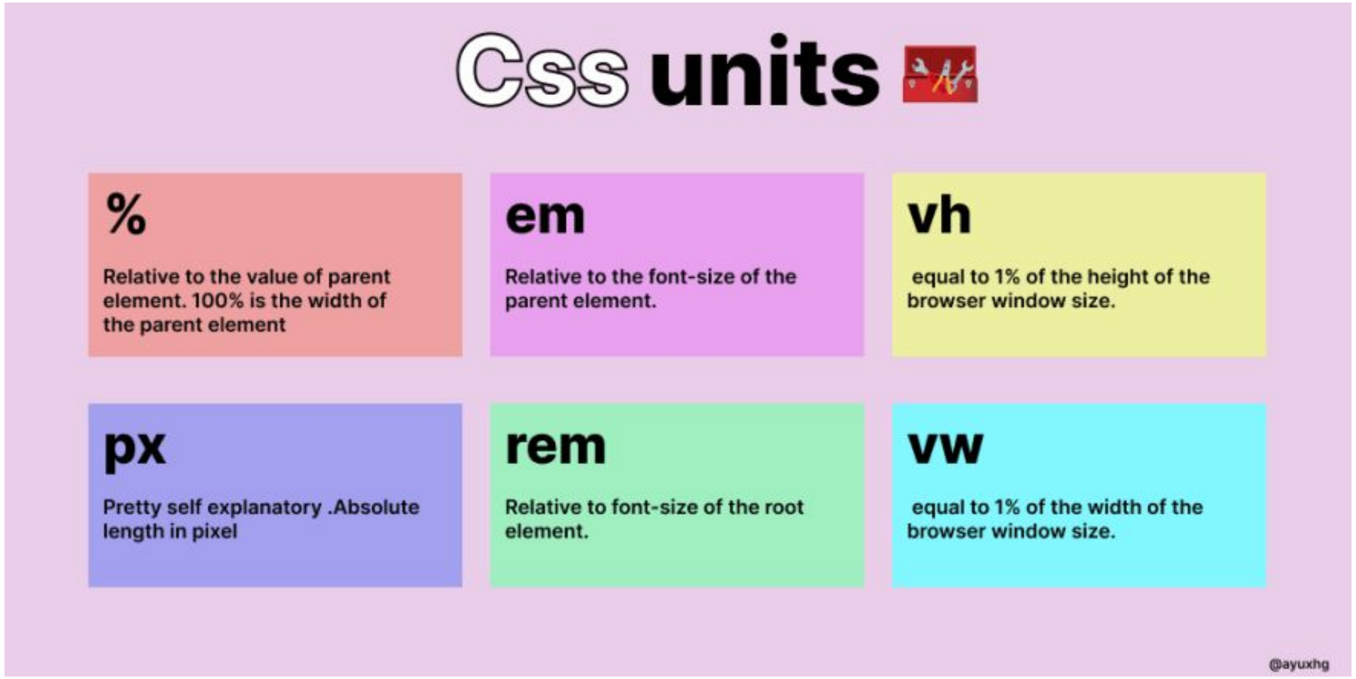


```css
nav {
  display: flex;
  justify-content: space-between;
  padding: 20px;
}

a {
  display: block;
  padding: 20px;
  background-color: lightgray;
  text-decoration: none;
  color: black;
}

@media (max-width: 600px) {
  nav {
    flex-direction: column;
  }
}
```

CSS Measurements



Css units

%

Relative to the value of parent element. 100% is the width of the parent element

em

Relative to the font-size of the parent element.

vh

equal to 1% of the height of the browser window size.

px

Pretty self explanatory .Absolute length in pixel

rem

Relative to font-size of the root element.

vw

equal to 1% of the width of the browser window size.

@ayuxhg

CSS Position

Static: This is the **default** value of the position property, and it means that the element is positioned according to the **normal flow** of the document. Elements with position: static **are not affected** by the **top**, **right**, **bottom**, and **left** properties.

Relative: This value positions the element **relative to its normal position** in the document flow. Setting top, right, bottom, or left properties will move the element in the **respective direction from its original position**.

Absolute: This value positions the element relative to its nearest positioned ancestor (ancestor elements with a position value **other than static**). If there is no positioned ancestor, then the element is positioned relative to the **initial containing block** (usually the <html> element).

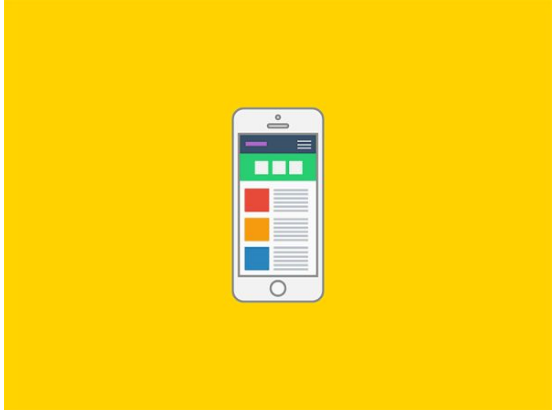
Fixed: This value positions the element **relative to the viewport** (the visible area of the browser window). Fixed elements **are always visible**, even when the user **scrolls the page**.

Sticky: This value is a **hybrid of relative and fixed positioning**. A sticky positioned element is positioned **relative to its nearest positioned ancestor or the viewport**. However, it behaves like a fixed element when it **reaches a certain threshold** (specified with top, right, bottom, or left), and then it sticks to that position.

CSS Pseudo-Classes

https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

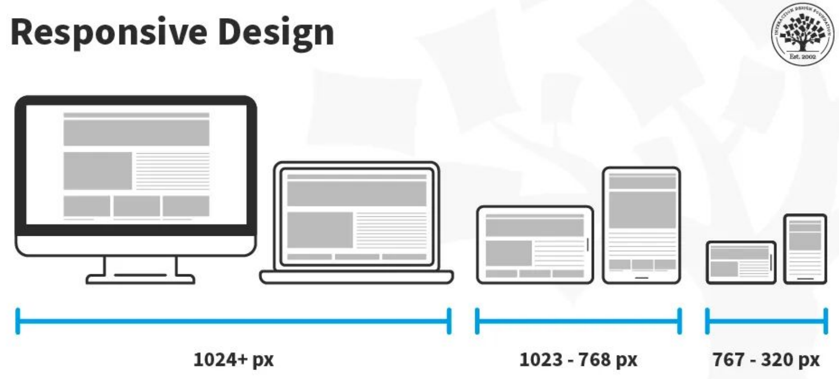
Responsive Design and Animation



Responsive design ensures that a website or application looks and functions correctly on a variety of devices, such as laptops, tablets, and smartphones. This is important because more and more people are accessing the internet on mobile devices, and a website that is not optimized for mobile may be difficult to navigate or use.

Animation can help to guide the user through the interface and make the experience more engaging. For example, animation can be used to highlight important information, provide feedback on user interactions, and create a sense of hierarchy on the page. Additionally, animation can make a website or application feel more dynamic and alive, which can lead to a more positive user experience.

What is Responsive Design?

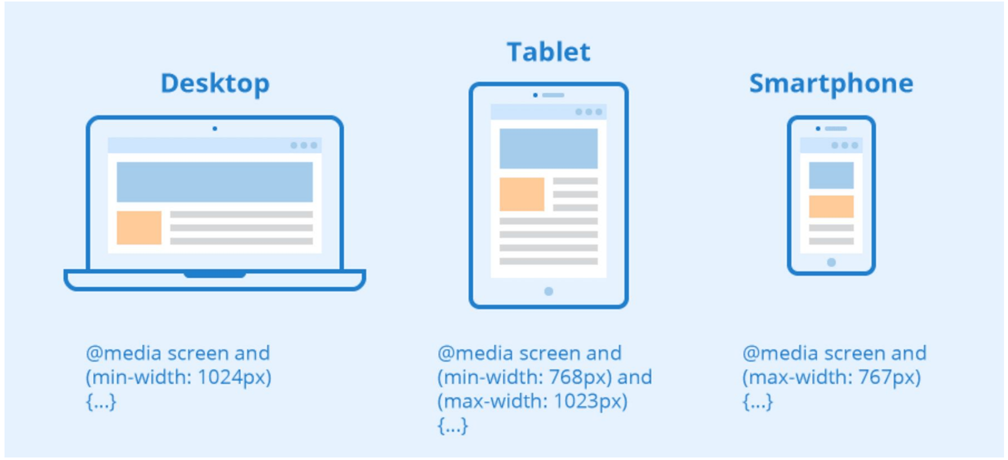


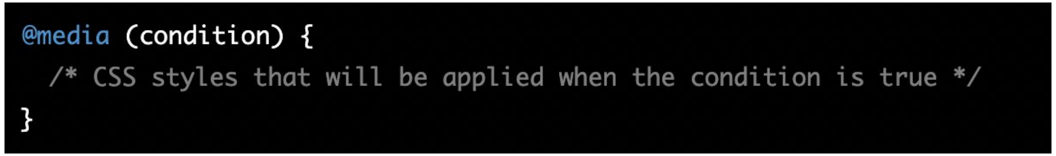
Responsive design is a method of designing and developing websites and applications that adapt to the size of the user's screen. This allows the website or application to look and function correctly on a variety of devices, such as laptops, tablets, and smartphones.

One key aspect of responsive design is the use of media queries, which are a way to apply CSS styles based on the characteristics of the device. Media queries allow designers to create different layouts for different screen sizes, so that the content can be rearranged and resized to fit the screen.

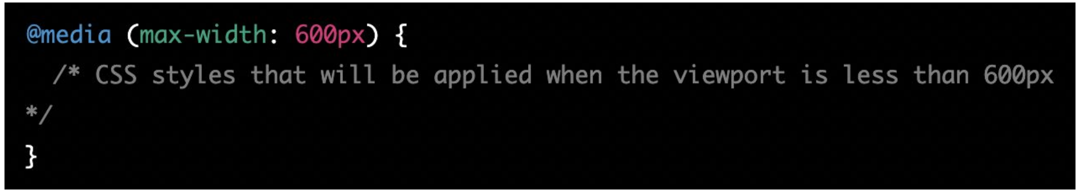
CSS media queries

By using media queries, designers can create websites and applications that adapt to the size of the user's screen, ensuring that the content is always easy to read and navigate, regardless of the device being used.


Desktop
Tablet
Smartphone
@media screen and (min-width: 1024px) {...}
@media screen and (min-width: 768px) and (max-width: 1023px) {...}
@media screen and (max-width: 767px) {...}

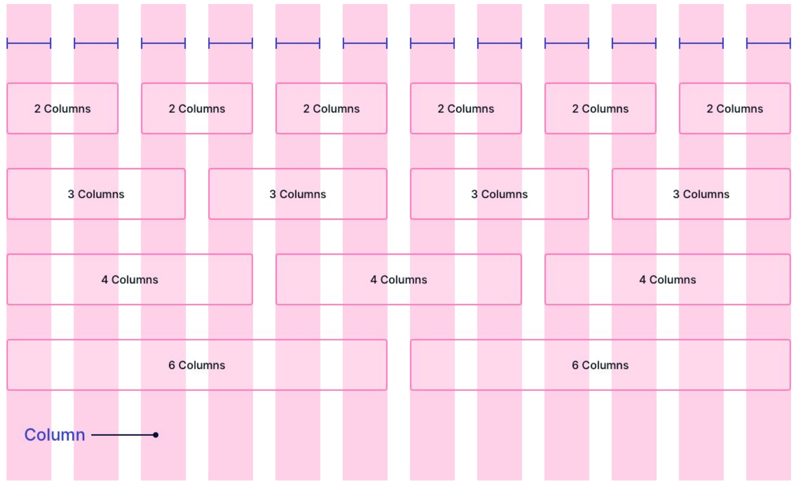


```
@media (condition) {
  /* CSS styles that will be applied when the condition is true */
}
```



```
@media (max-width: 600px) {
  /* CSS styles that will be applied when the viewport is less than 600px
*/
}
```

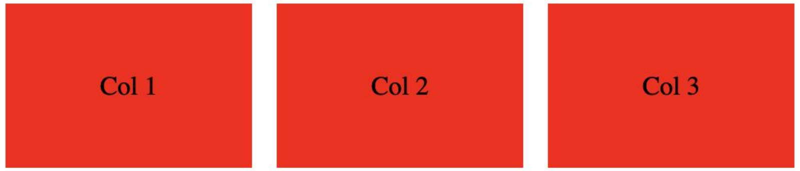
Fluid grids

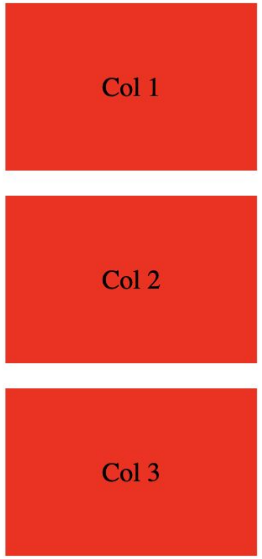


Another important aspect of responsive design is the use of fluid grids, which are a way to create layouts that automatically adjust to the size of the screen. This is done by defining the layout in terms of relative units, such as percentages, rather than absolute units, such as pixels.

This allows the layout to automatically adjust to the size of the screen, so that the content can be rearranged and resized to fit the screen. Together, media queries and fluid grids allow designers to create websites and applications that adapt to the size of the user's screen, ensuring that the content is always easy to read and navigate, regardless of the device being used.

Using Media Queries to create Responsive Design



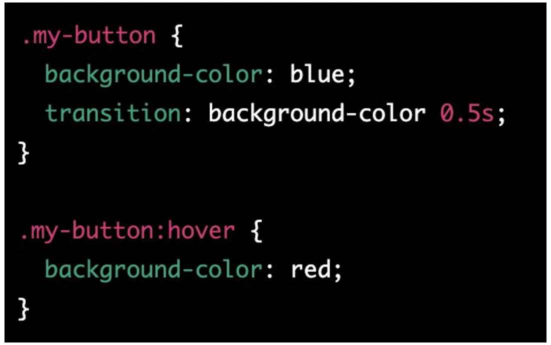


```
<div class="wrapper">
  <div>Col 1</div>
  <div>Col 2</div>
  <div>Col 3</div>
</div>
```

Creating Smooth Transitions with CSS

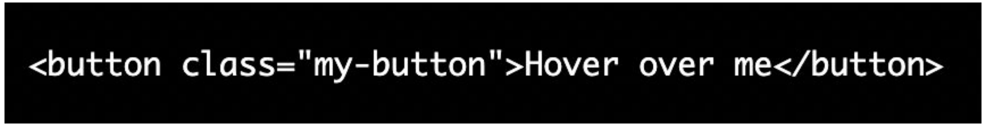
CSS transitions allow you to smoothly change the values of CSS properties over a given duration. They are used to add visual effects and make changes in the layout of a webpage look more natural.

https://www.w3schools.com/css/css3_transitions.asp



```css
.my-button {
  background-color: blue;
  transition: background-color 0.5s;
}

.my-button:hover {
  background-color: red;
}
```



```html
<button class="my-button">Hover over me</button>
```

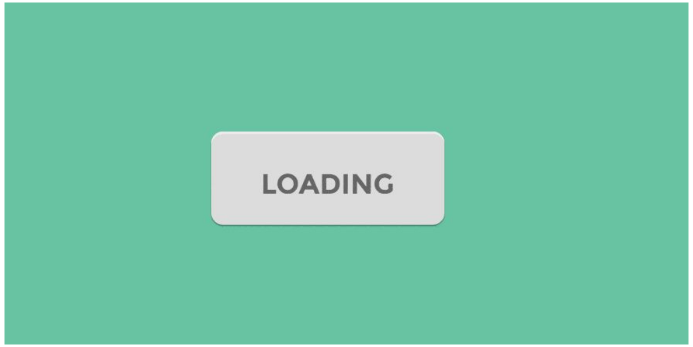
Introducing Animation to your website

Animation is the process of creating the illusion of motion by rapidly displaying a sequence of static images, called frames. When these frames are played back in quick succession, the human eye perceives them as a continuous motion. Animation can be used to create engaging and interactive user interfaces by providing visual feedback, guiding the user's attention, and creating a sense of motion and dynamism.

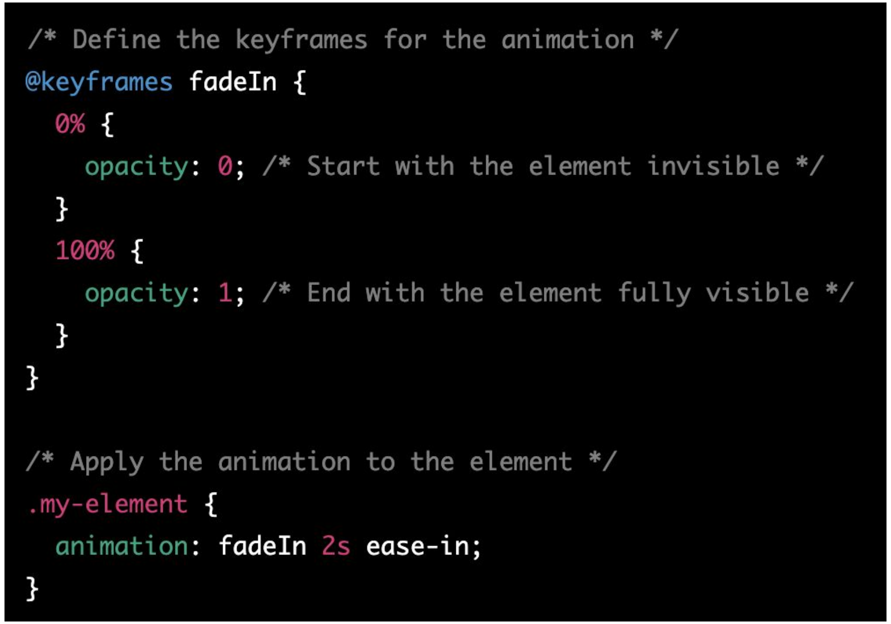
Animation use case examples

Animation can be used in a variety of ways to enhance the user experience. For example, an animation can be used to:

- Provide visual feedback when a button is clicked or a form is submitted
- Guide the user's attention by highlighting important elements on the page
- Create a sense of motion and dynamism, making the interface feel more alive and engaging
- ...

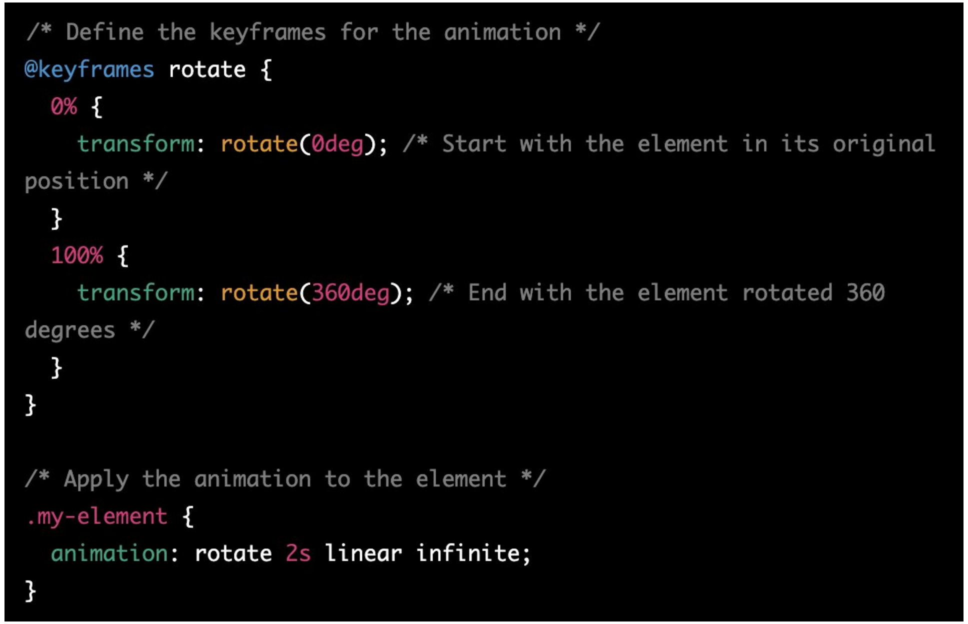


Creating Animations with CSS



```css
/* Define the keyframes for the animation */
@keyframes fadeIn {
  0% {
    opacity: 0; /* Start with the element invisible */
  }
  100% {
    opacity: 1; /* End with the element fully visible */
  }
}

/* Apply the animation to the element */
.my-element {
  animation: fadeIn 2s ease-in;
}
```



```css
/* Define the keyframes for the animation */
@keyframes rotate {
  0% {
    transform: rotate(0deg); /* Start with the element in its original
position */
  }
  100% {
    transform: rotate(360deg); /* End with the element rotated 360
degrees */
  }
}

/* Apply the animation to the element */
.my-element {
  animation: rotate 2s linear infinite;
}
```

Animations task

Task: Create a simple animation that makes a square move across the screen horizontally.
Steps:

1) Create the HTML structure for the square. In the body of your HTML file, add a div with the class "square"
2) In your CSS file, give the square a size and a background color
3) Define the keyframes for the animation. In your CSS file, create a new @keyframes rule called "move" that defines the starting and ending positions of the square:
4) Apply the animation to the square. Using the animation property, specify the animation name, duration, and timing function:
5) Test your animation by running the HTML file in a browser. The square should move horizontally across the screen from left to right over a duration of 2 seconds, using a linear timing function.

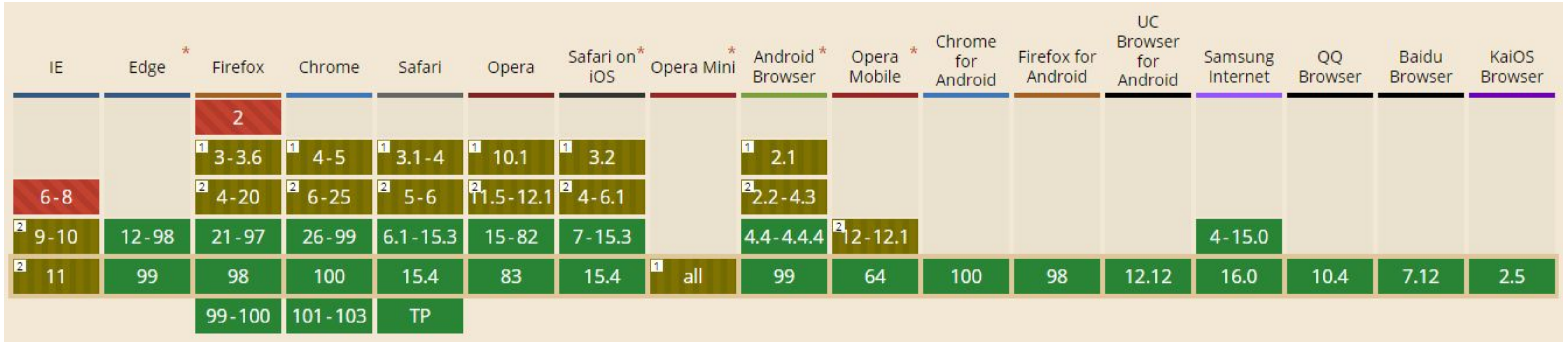
Animations task explanation

```html
<div class="square"></div>
```

```css
.square {
  width: 50px;
  height: 50px;
  background-color: blue;
  animation: move 2s linear;
}
```

```css
@keyframes move {
  0% {
    transform: translateX(0);
  }
  100% {
    transform: translateX(300px);
  }
}
```

Can I use it?



IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
		2														
		[1] 3-3.6	[1] 4-5	[1] 3.1-4	[1] 10.1	[1] 3.2		[1] 2.1								
6-8		[2] 4-20	[2] 6-25	[2] 5-6	[2] 11.5-12.1	[2] 4-6.1		[2] 2.2-4.3								
[2] 9-10	12-98	21-97	26-99	6.1-15.3	15-82	7-15.3		4.4-4.4.4	[2] 12-12.1				4-15.0			
[2] 11	99	98	100	15.4	83	15.4	[1] all	99	64	100	98	12.12	16.0	10.4	7.12	2.5
		99-100	101-103	TP												

Homework

1) Add responsiveness/transitions/animations to portfolio

Portfolio ideas -
https://bashooka.com/html/free-html-css-portfolio-web-design-templates/