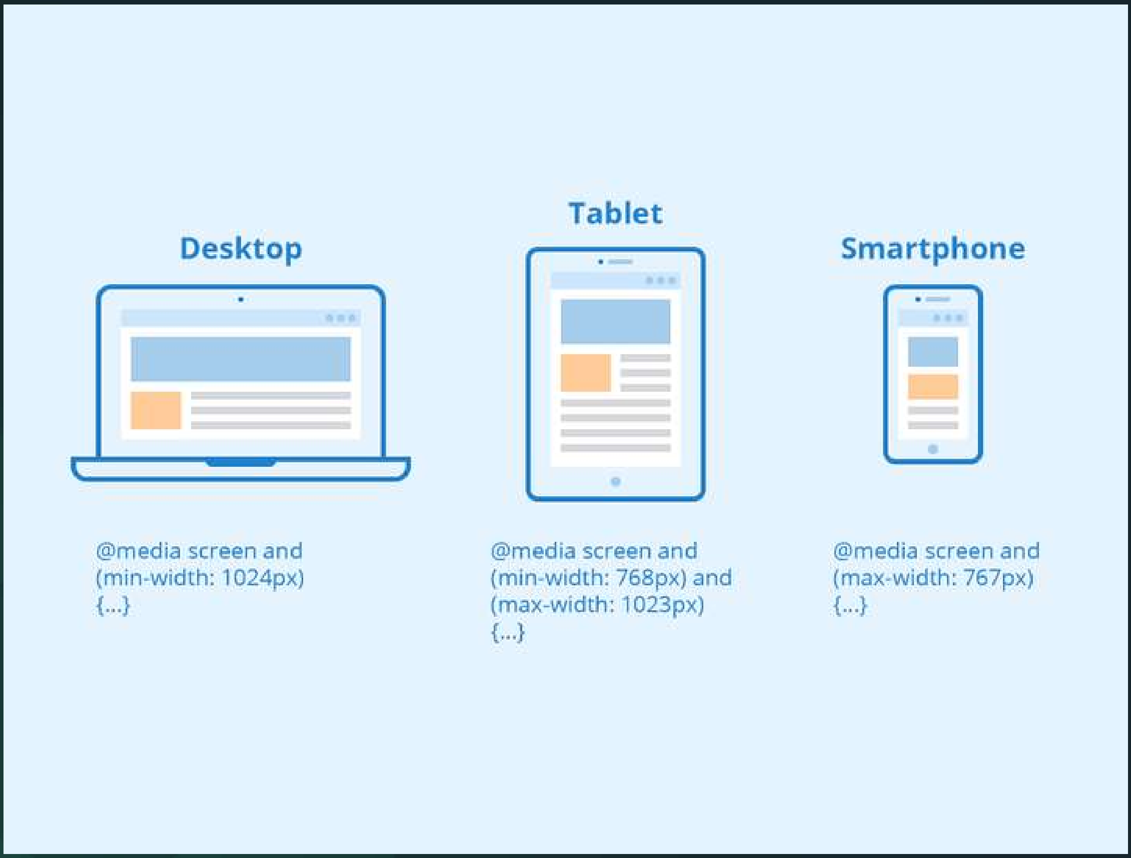
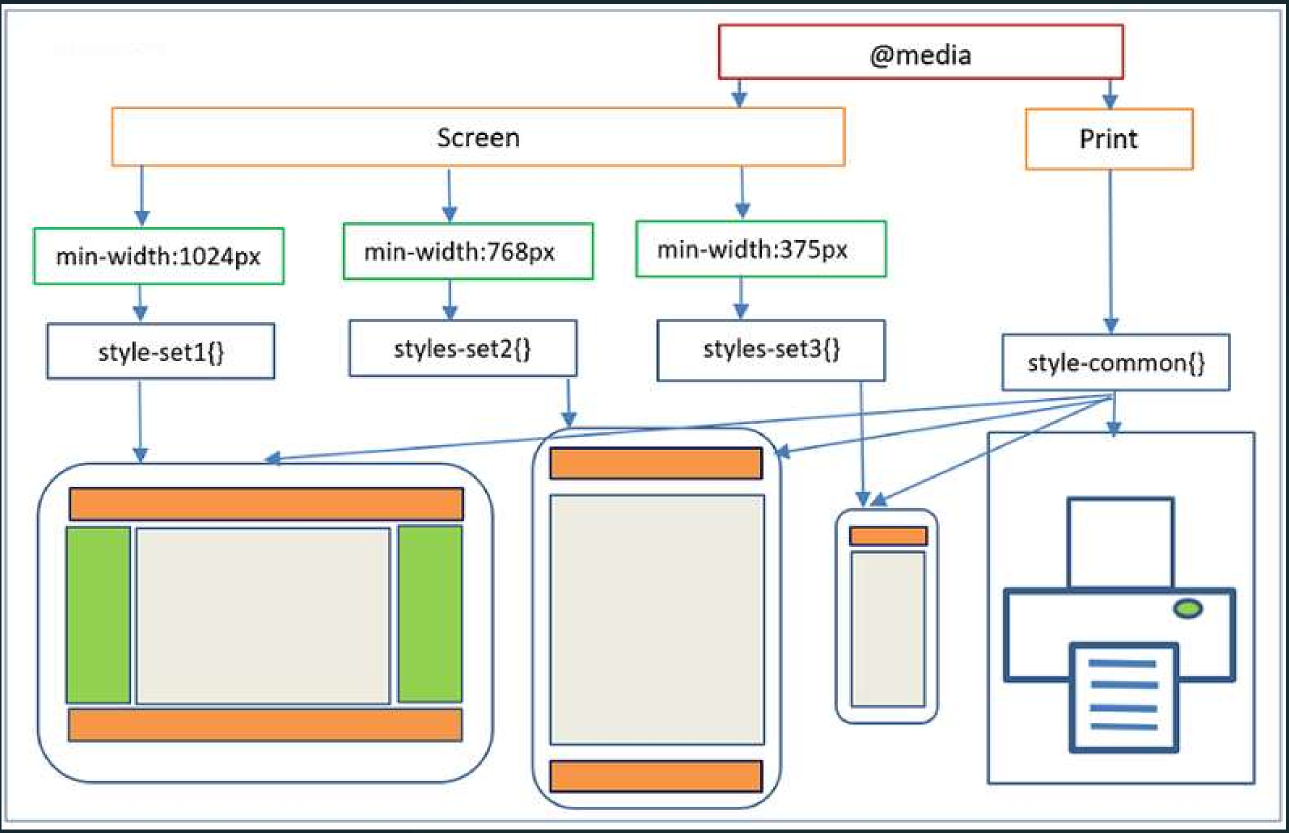
Media Quariries and Animation

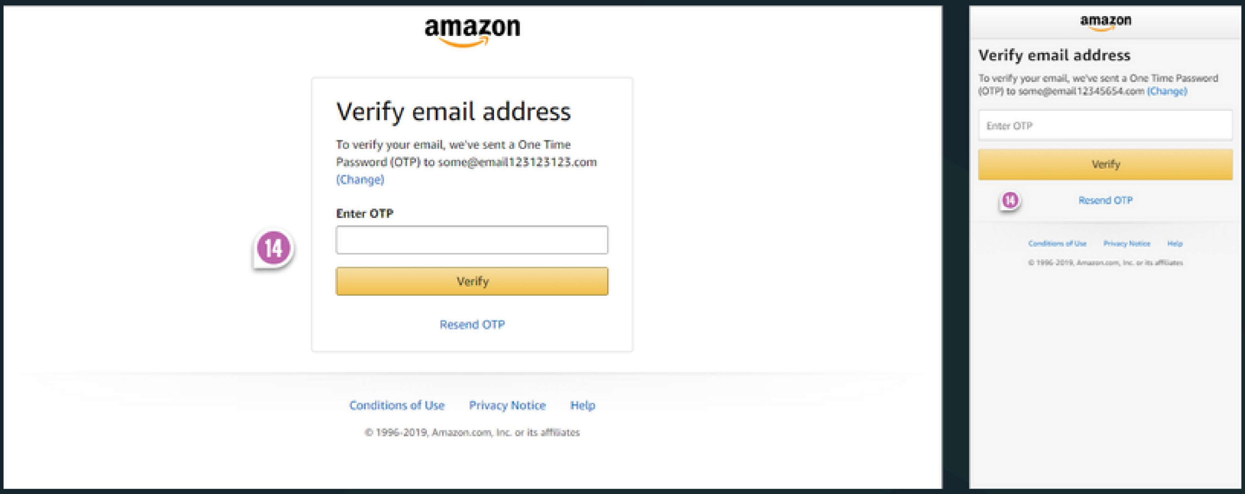

Desktop
Tablet
Smartphone
@media screen and (min-width: 1024px) {...}
@media screen and (min-width: 768px) and (max-width: 1023px) {...}
@media screen and (max-width: 767px) {...}

Media Queries

Media queries are a CSS feature that allows developers to apply specific styles based on certain device characteristics, such as screen size, resolution, orientation, and more. It enables the creation of responsive web designs that adapt seamlessly to different devices, ensuring a consistent and user-friendly experience across various platforms.



Media queries consist of a @media rule and a set of CSS styles enclosed within curly braces. Within the media query, you can use various sub-properties to target specific device characteristics.


amazon

Verify email address

To verify your email, we've sent a One Time Password (OTP) to some@email123123123.com (Change)

Enter OTP

14

Verify

Resend OTP

Conditions of Use Privacy Notice Help

© 1996-2019, Amazon.com, Inc. or its affiliates

amazon

Verify email address

To verify your email, we've sent a One Time Password (OTP) to some@email12345654.com (Change)

Enter OTP

Verify

14

Resend OTP

Conditions of Use Privacy Notice Help

© 1996-2019, Amazon.com, Inc. or its affiliates

Let's say you have an e-commerce website. You can use media queries to adjust the layout and font size for mobile devices to ensure easy navigation and readability, while on larger screens like laptops or desktops, you can display more product images and details to utilize the available space effectively.

width and height of the viewport:

These sub-properties allow you to target screens with specific width and height dimensions. For example:

```css
@media screen and (max-width: 768px) {
  /* Styles for screens with a width of 768px or less */
}
```

width and height of the device:

You can target devices based on their physical width and height using the device-width and device-height sub-properties. For instance:

```css
@media screen and (min-device-width: 320px) {
  /* Styles for devices with a minimum width of 320px */
}
```

orientation (portrait or landscape mode):

The orientation sub-property allows you to apply different styles based on the device's orientation. For example:

```css
@media screen and (orientation: landscape) {
  /* Styles for devices in landscape mode */
}
```

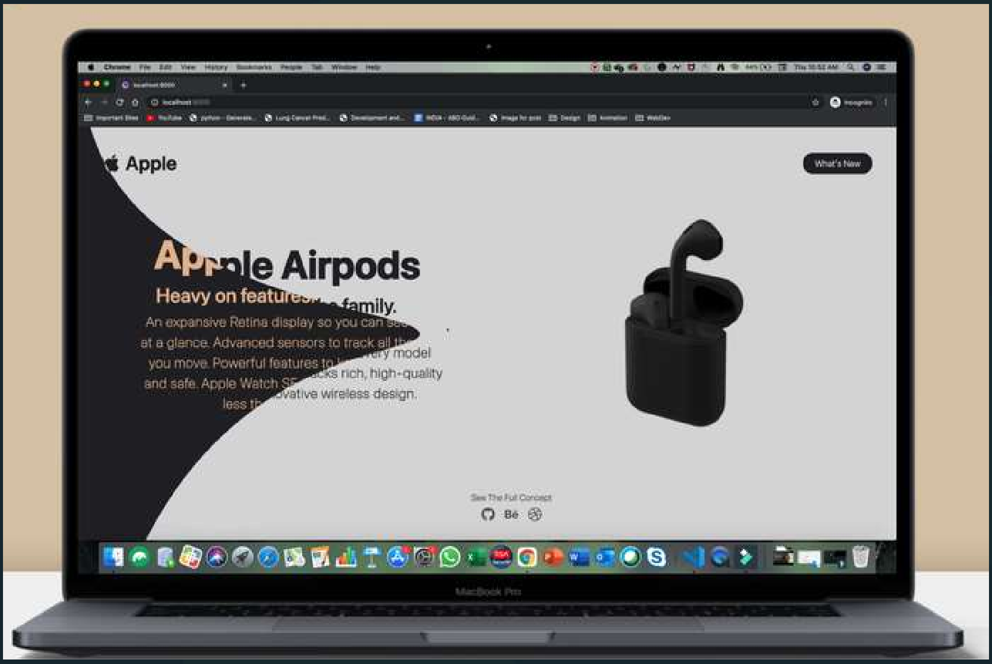
resolution:

The resolution sub-property allows you to target devices based on their display resolution, such as DPI (dots per inch). For example:

```css
@media (min-resolution: 300dpi) {
  /* Styles for devices with a resolution greater than 300dpi */
}
```

Animations:

Animations in web development refer to adding dynamic movement to web elements, creating visually appealing effects that capture users' attention and enhance the overall interaction with the website. CSS animations are a powerful tool for bringing life to static web pages and can be used for various purposes, such as illustrating processes, providing feedback, or guiding users through a user interface.



animation-name: Specifies the name of the keyframe animation that defines the sequence of states. For example:

```css
@keyframes fadeInOut {
  0% { opacity: 0; }
  50% { opacity: 1; }
  100% { opacity: 0; }
}
```

animation-duration:
Sets the duration of the animation. It defines how long the animation takes to complete one cycle. For example:

```css
.element {
  animation-name: fadeInOut;
  animation-duration: 3s;
}
```

animation-timing-function: Determines the pace of the animation. It controls the acceleration and deceleration of the animation over time. For example:

```css
.element {
  animation-timing-function: ease-in-out;
}
```

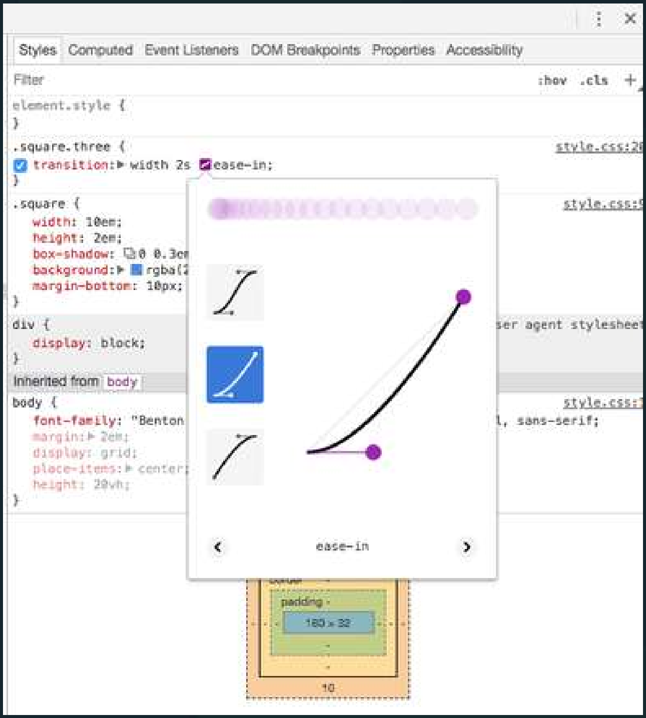
animation-iteration-count:
Specifies how many times the animation should play. You can use values like infinite for endless loops. For example:

```css
.element {
  animation-iteration-count: infinite;
}
```

Transition Property:

The transition property in CSS enables developers to apply smooth transitions to CSS properties when changes occur, like when hovering over an element or when it receives focus. Transitions allow for gradual changes in styles rather than instant switches, improving the visual experience and reducing abruptness.

With the transition property, you can specify which CSS properties should transition, the duration of the transition, and the timing function that determines how the transition progresses over time.



```
element.style {
}
.square.three {                                        style.css:2
  transition: ▸ width 2s ease-in;
}
.square {                                              style.css:9
  width: 10em;
  height: 2em;
  box-shadow: 0 0.3em
  background: ▸ rgba(2
  margin-bottom: 10px;
}
div {                                          ser agent stylesheet
  display: block;
}
Inherited from body
body {                                                 style.css:1
  font-family: "Benton                          l, sans-serif;
  margin: ▸ 2em;
  display: grid;
  place-items: ▸ center;
  height: 20vh;
}
```

transition-property:

Defines the CSS property or properties that should transition. For example:e:

```
.button {
  transition-property: background-color, color;
}
```

transition-duration:

Specifies the time it takes for the transition to complete. For example:

```
.button {
  transition-duration: 0.3s;
}
```

transition-timing-function:

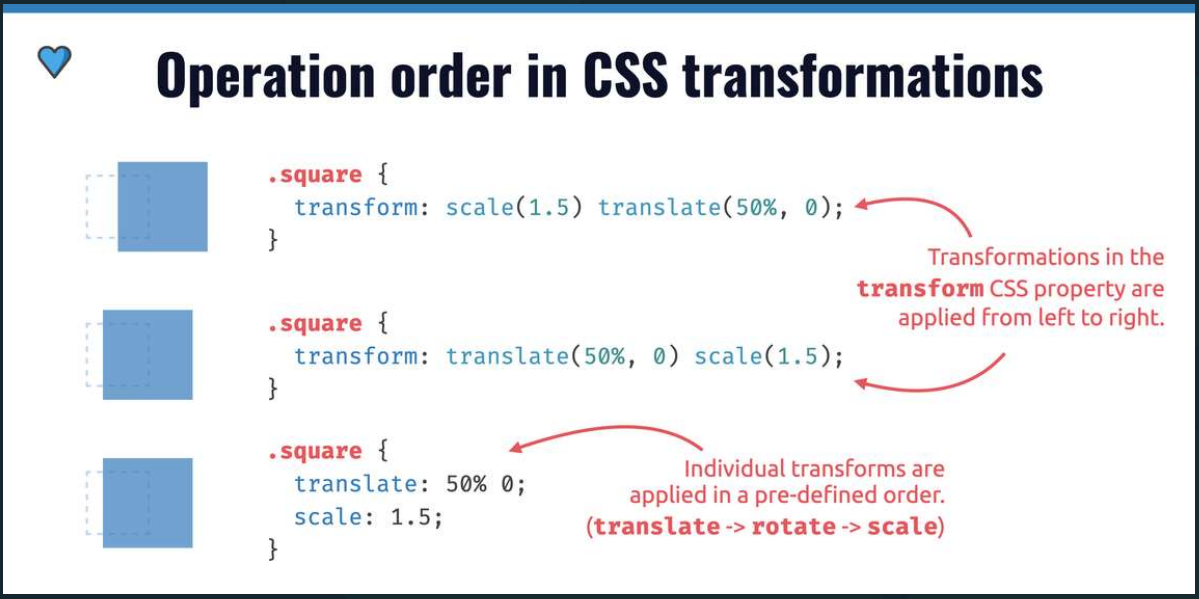
Similar to the animation timing function, this controls the pace of the transition. For example:

```
.button {
  transition-timing-function: ease-in-out;
}
```

Transform Property:

The transform property in CSS allows developers to apply various 2D and 3D transformations to an element. This includes scaling, rotating, skewing, and translating (moving) elements, allowing for creative and visually engaging effects without the need for complex animations.

Using transform, you can change the appearance and position of elements dynamically. These transformations are applied relative to the element's original position, making them efficient and flexible for achieving stunning visual effects.



Operation order in CSS transformations

```
.square {
  transform: scale(1.5) translate(50%, 0);
}

.square {
  transform: translate(50%, 0) scale(1.5);
}

.square {
  translate: 50% 0;
  scale: 1.5;
}
```

Transformations in the **transform** CSS property are applied from left to right.

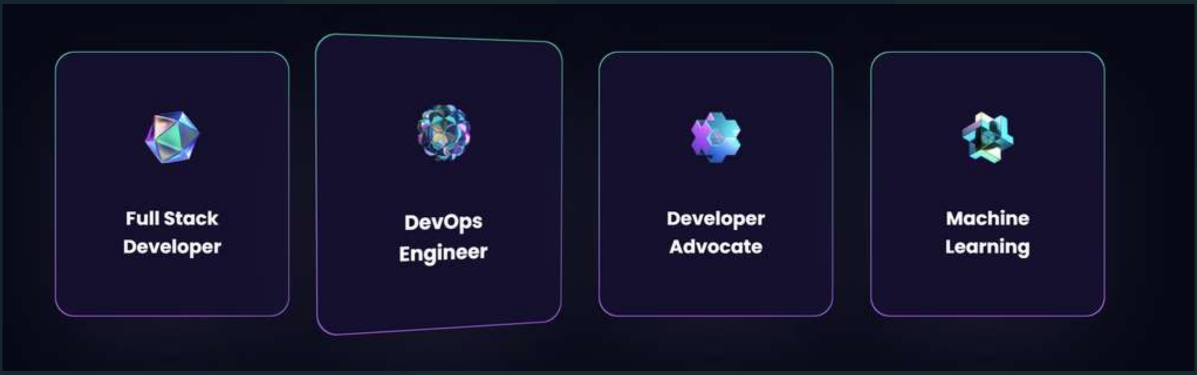
Individual transforms are applied in a pre-defined order. (**translate** -> **rotate** -> **scale**)

transform:

The main property to apply transformations. You can combine multiple transformations using this property. For example:

```css
.box {
  transform: scale(1.2) rotate(45deg);
}
```

Consider a portfolio website with a grid of project images. When users hover over an image, you can use CSS transforms to enlarge the image slightly, adding a subtle zoom-in effect that enhances the user's focus on the selected project. Additionally, you can add a rotation to the image to give it a dynamic feel as the user hovers over it.


Full Stack Developer
DevOps Engineer
Developer Advocate
Machine Learning