Session-06

Advanced CSS

Thanos is on a mission to make his website standout from his rest of universe

Agenda : Advanced CSS

HTML

HTML + CSS

HTML + CSS
+ JAVASCRIPT

01 CSS Layout

02 Grid, Flex Layouts

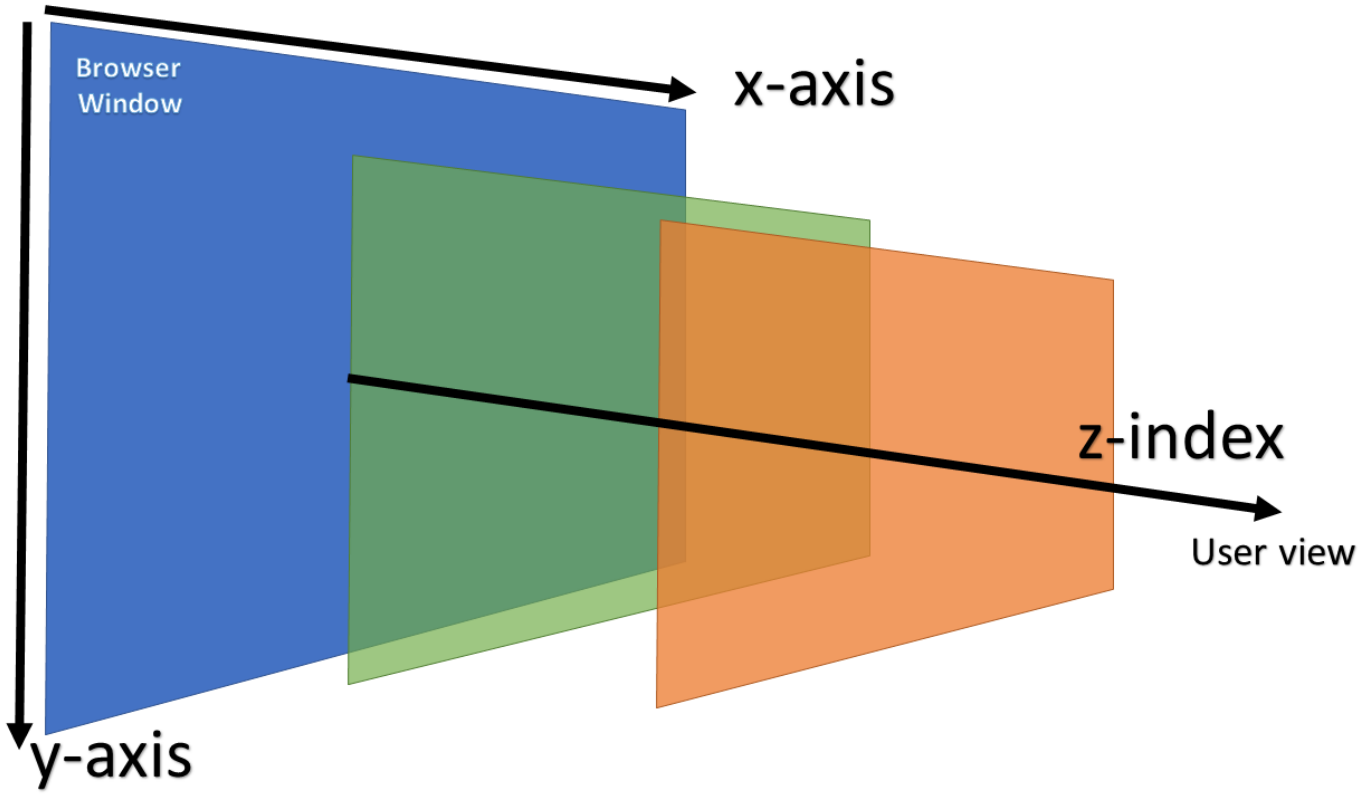
03 2D , 3D Transforms

04 Transitions, Animations

05 Special Effects

06 Preprocessors , Hands-On

Z-Index



<|>esto

Z-Index

Highest Value

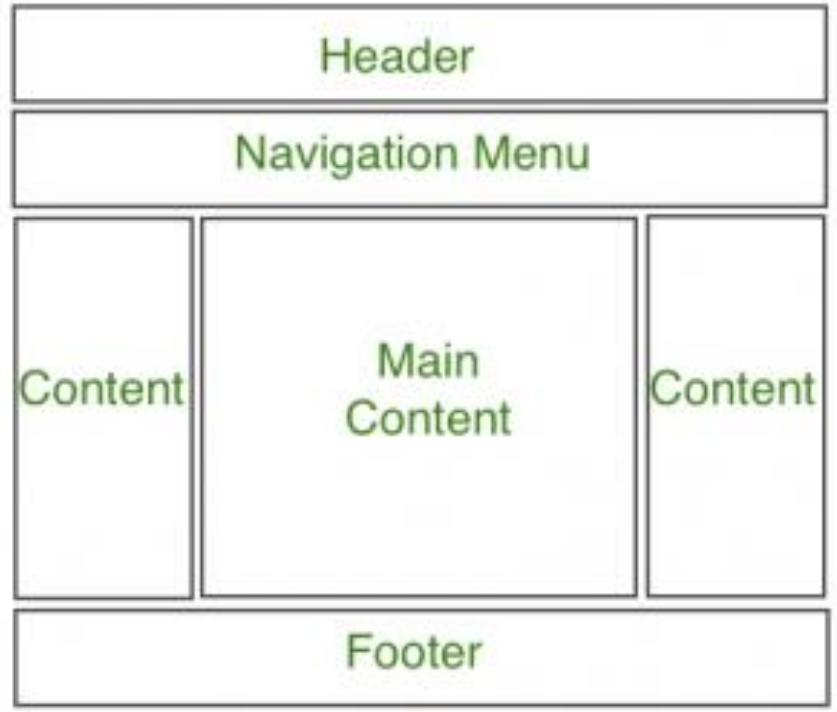
Lowest Value

Highest value always takes-over priority in viewport

What is a layout?

Layout refers to the arrangement and positioning of elements on a webpage

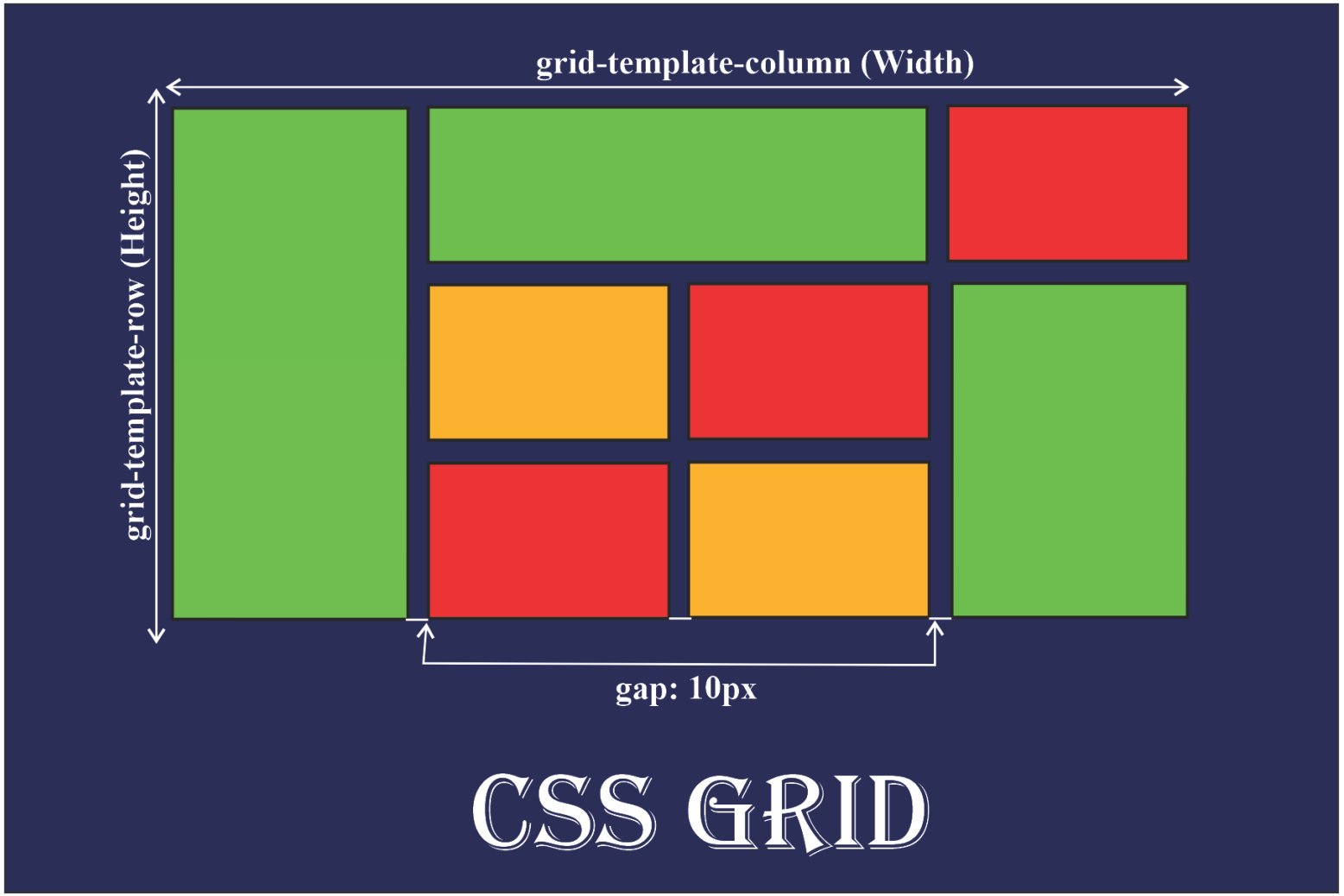
Typical layout



Different Layout Techniques

- Grid Layout
- Fluid Layout
- Flex Layout

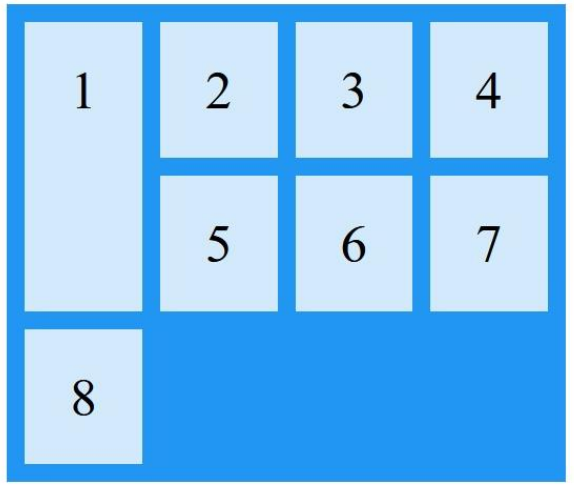
Grid Layout



Container based, 2D technique layout

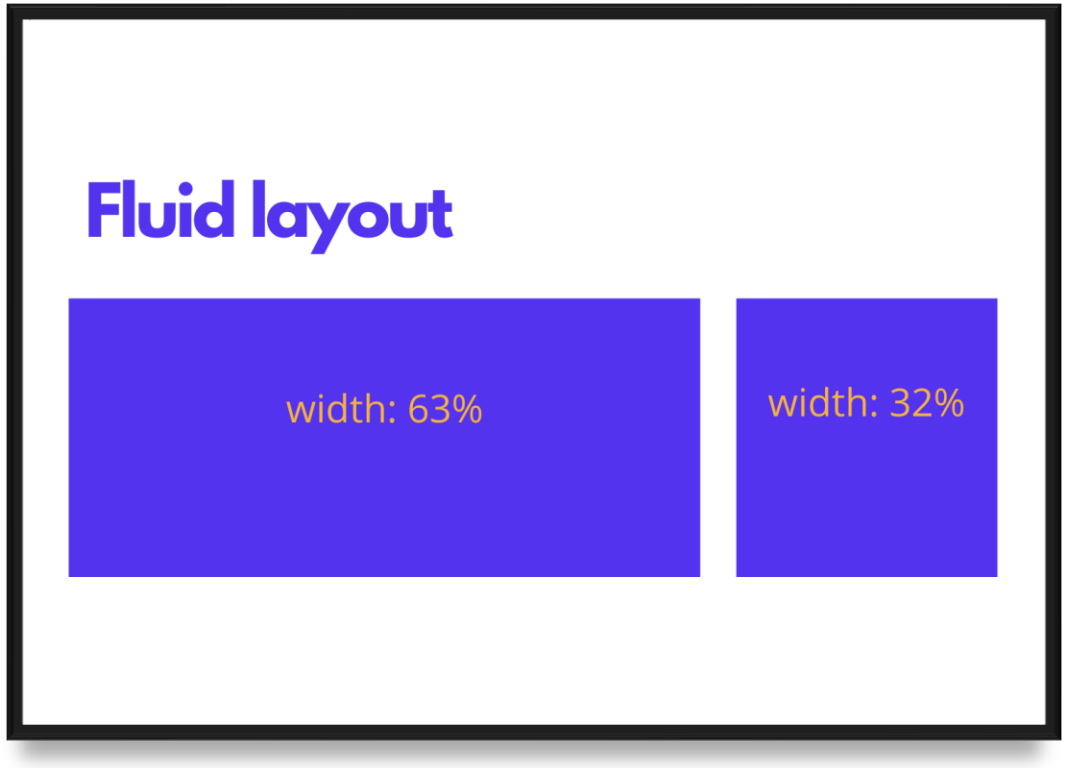
Grid Layout: Example

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}

.item1 {
  grid-row-start: 1;
  grid-row-end: 3;
}
```



You can refer to line numbers when placing grid items.

Fluid Layout



A fluid layout relies on dynamic values like a **percentage of the viewport width**.

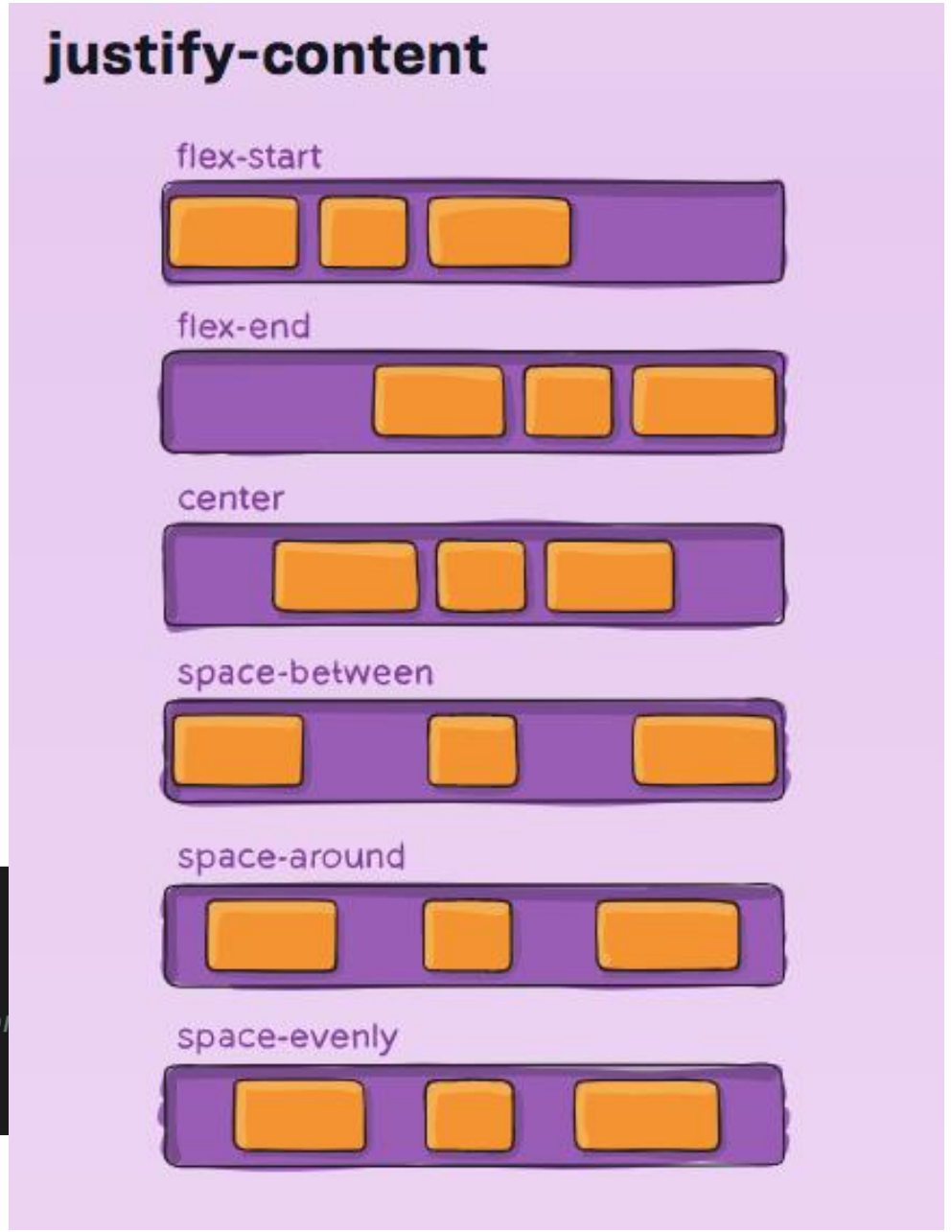
Fluid Layout: Example

```
.sidebar {
  width: 20%;
  /* Sidebar width is set to 20% of the container width */
  float: left;
  /* Sidebar is floated to the left */
  background-color: #f2f2f2;
  /* Example background color */
}

.content {
  width: 80%;
  /* Content width is set to 80% of the container width */
  float: left;
  /* Content is floated to the left */
  padding: 20px;
  /* Example padding for content */
}
```

<|>esto

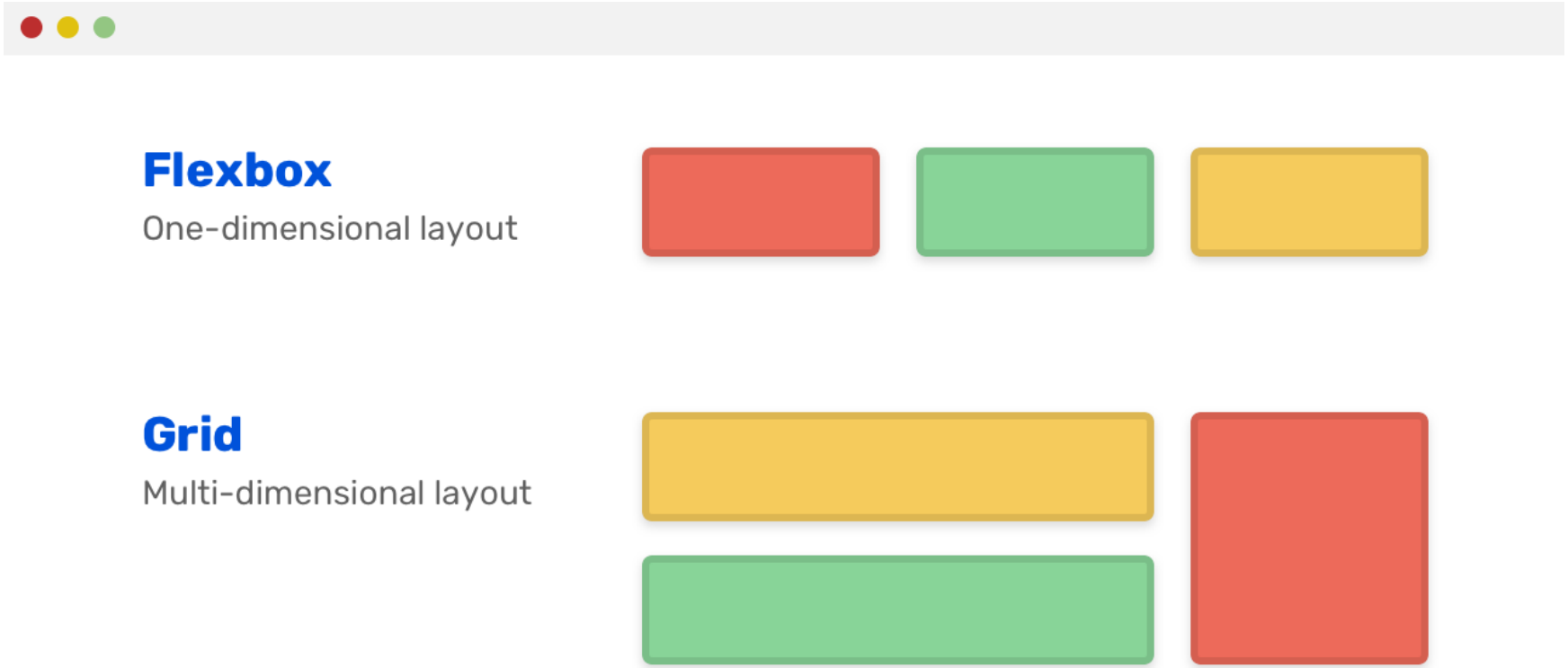
Flex Layout

```
.parent {
  display: flex;
  flex-flow: row wrap; /* OK elements, go as far as you can on o
}
```



<|>esto

Flex Vs Grid



Flex Vs Grid

Content-Based vs Container-Based

A Flexbox layout is calculated **after its content is loaded**. It's content-based.

CSS Grid, however, requires you to **define your layout first**. It's container-based.

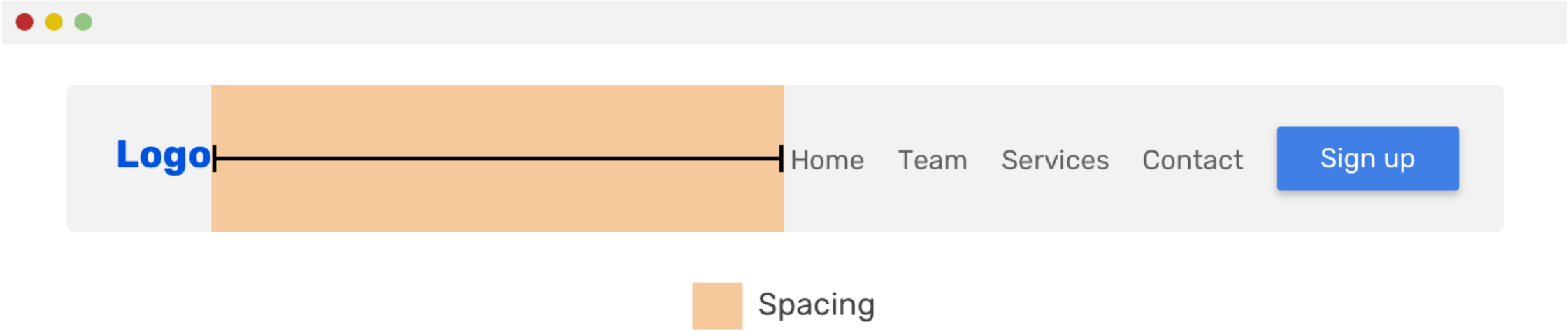
Confused Bro??

When to use which one?

Which one to use

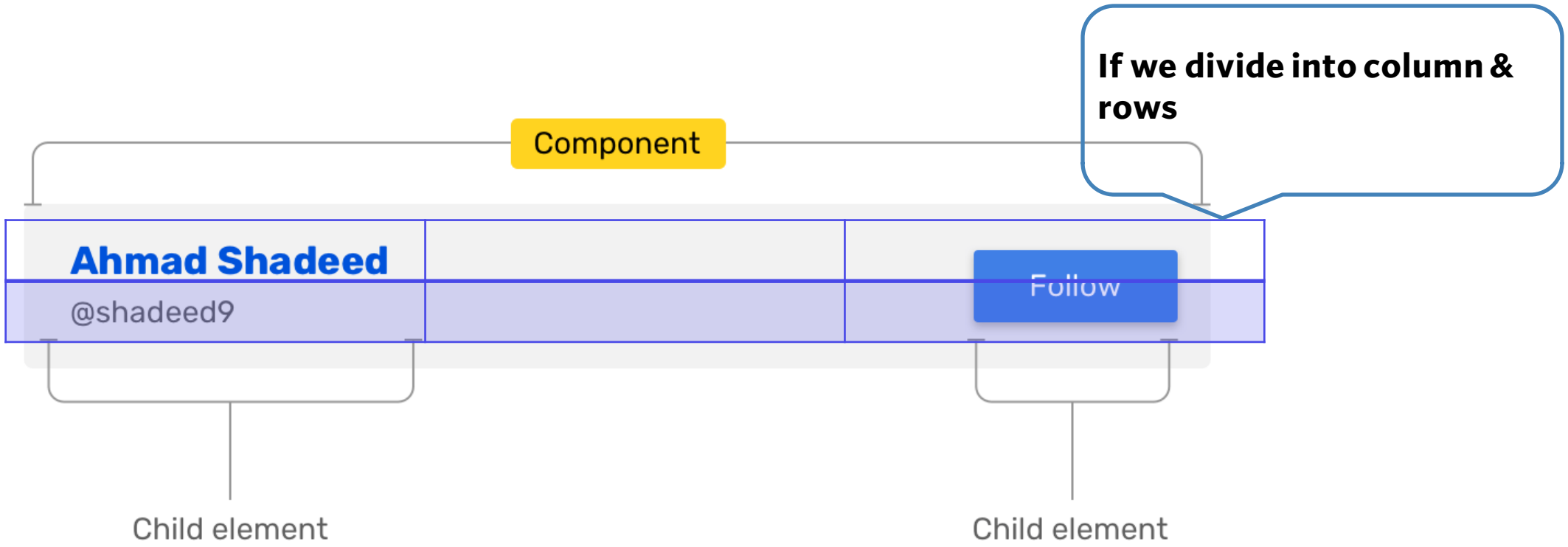
- Web layout that **only consists of rows or columns(1D)**, then Flexbox is the best
- If you have a complex, **multi-row and multi-column layout(2D)**, then you'll want to use CSS Grid.

Example Decision Practice-1

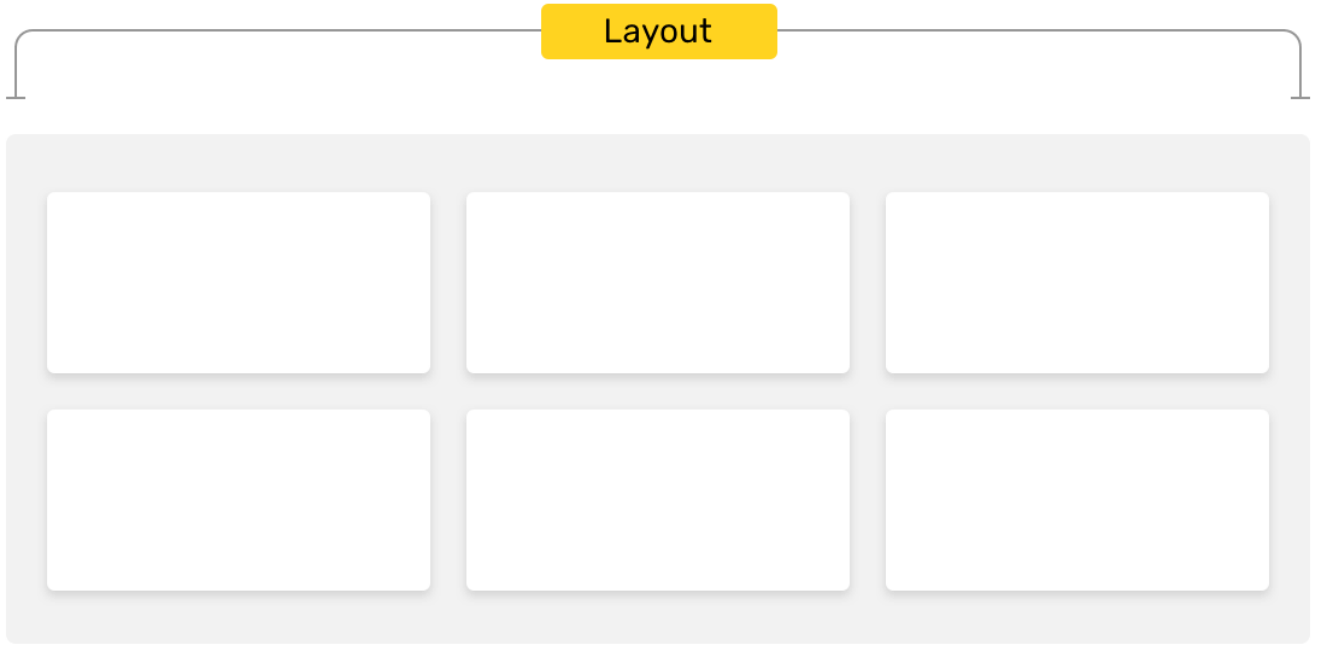


Which one will you use?

Example Decision Practice-2



Example Decision Practice-2



It’s layout goes like this... Which one will you use?

You might be like..

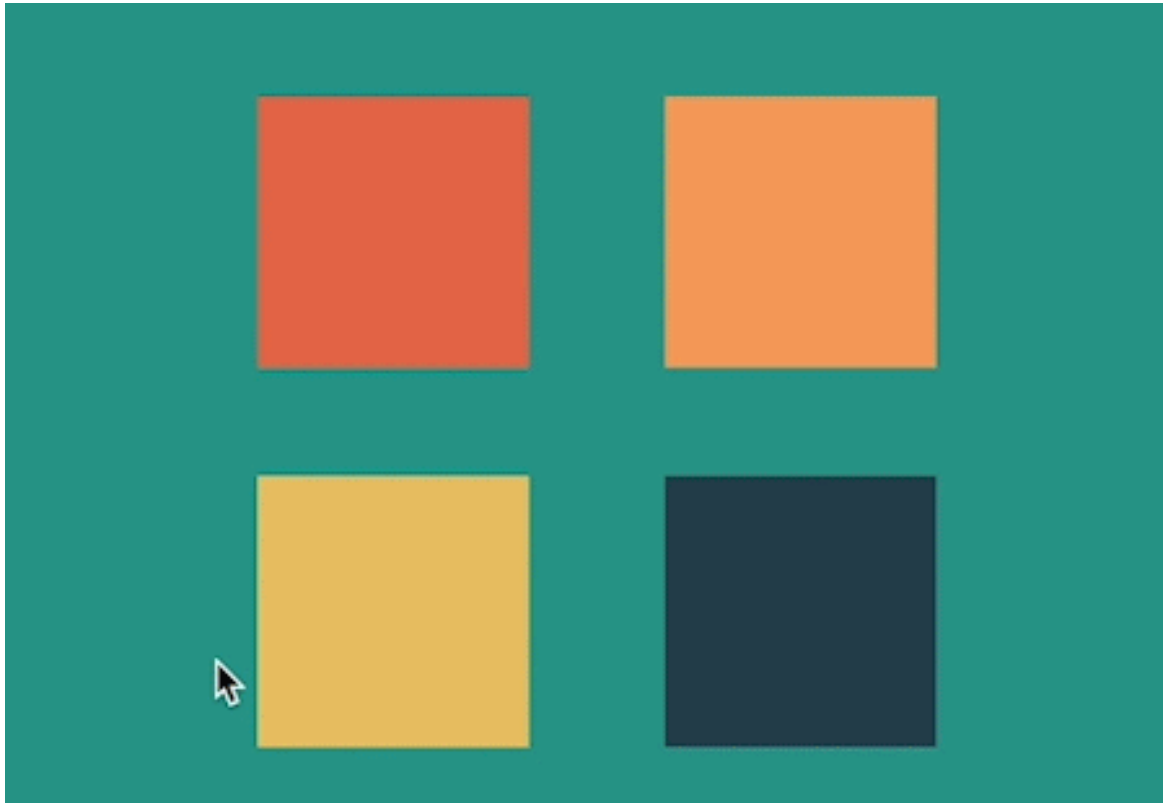
All this is fine.. But How to add animations in html?

CSS Animations- Key Concepts

1. 2D Transforms
2. 3D Transforms
3. Transitions
4. Animations

CSS 2D Transforms

2D transforms allow you to move, rotate, scale, and skew elements.



Methods: =>

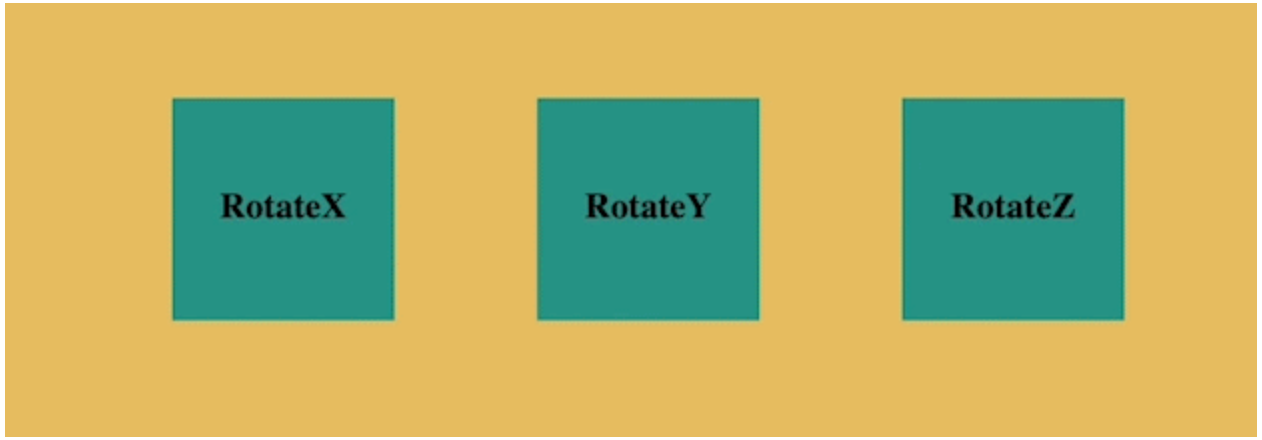
```
translate()
rotate()
scaleX()
scaleY()
scale()
skewX()
skewY()
skew()
```

Syntax: =>

```
div {
  transform: translate(50px, 100px);
}
```

CSS 3D Transforms

3D transforms happen in 3 dimensional axis



Syntax: =>

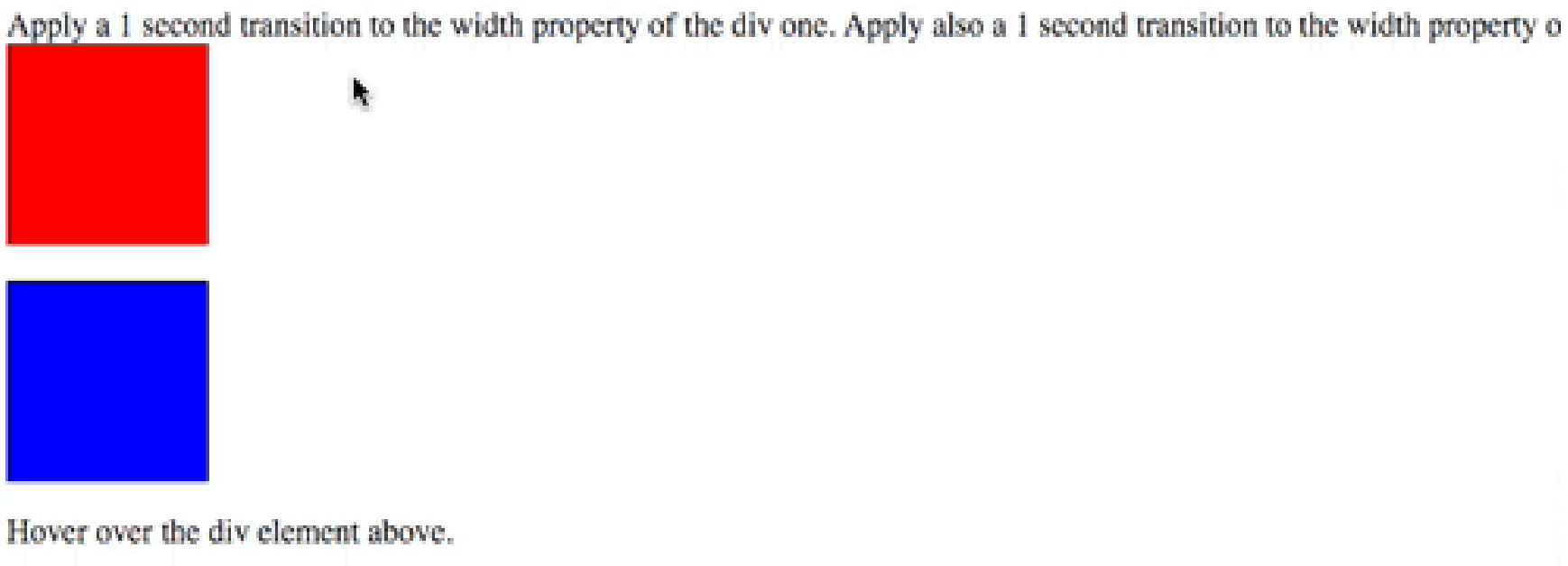
```
#myDiv {
  transform: rotateY(150deg);
}
```

Methods: =>

```
rotateX()
rotateY()
rotateZ()
```

CSS Transitions

Transitions allows you to change values smoothly, over a given duration



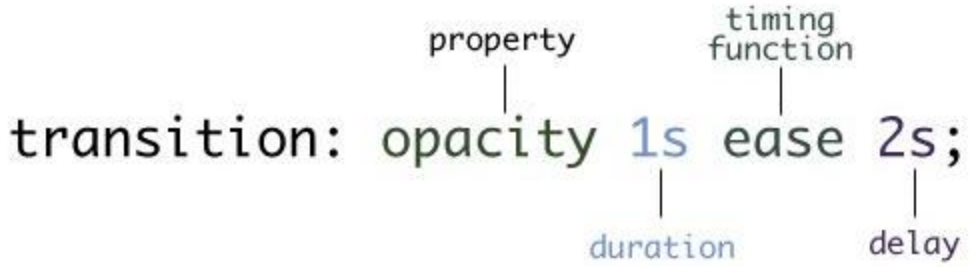
<|>esto

CSS Transitions : Syntax

Transitions allows you to change values smoothly, over a given duration

Properties

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function



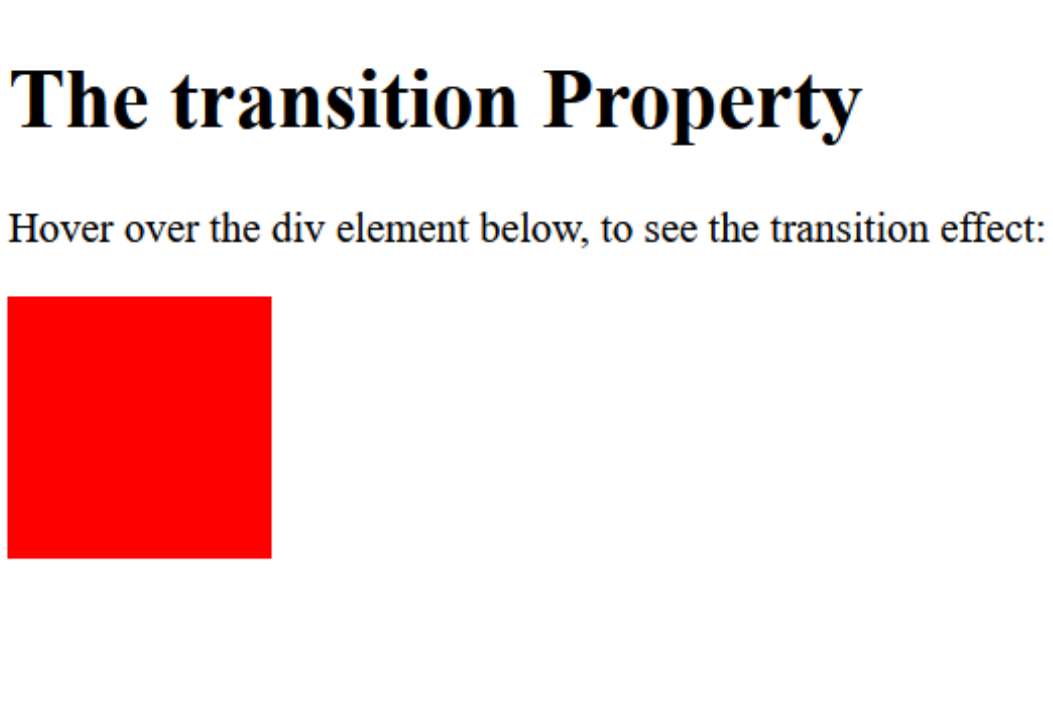

```
div {
  transition: width 2s linear 1s;
}
```

CSS Transitions: Example

Property =>

Trigger =>

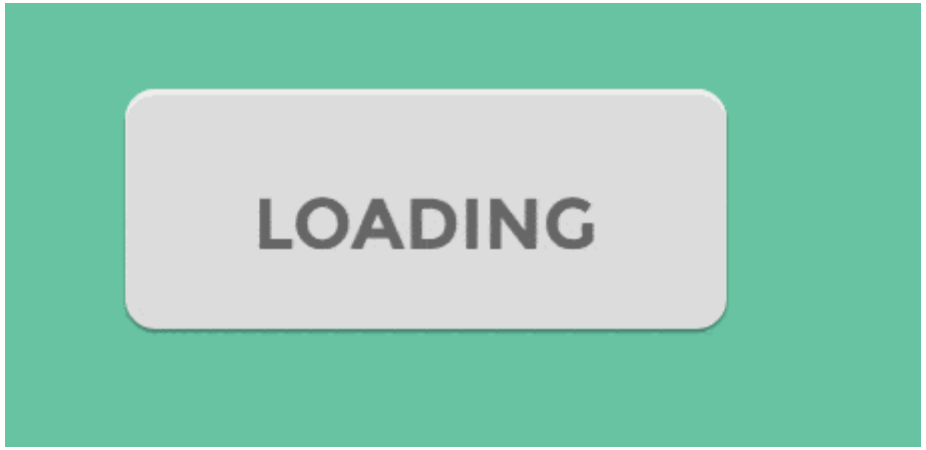
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}

div:hover {
  width: 300px;
}
</style>
```



CSS Animations

Animations allow you to add dynamic and interactive effects to pages

- No flash
- No Javascript



CSS Animations

- Element gradually change from **one style to another**
- Change as **many CSS properties** you want and **as many times**
- Uses **@keyframes for changing property**

Template/Properties

```css
.element {
    /* Animation name */
    animation-name: myAnimation;
    /* Duration of the animation */
    animation-duration: 2s;
    /* Timing function of the animation */
    animation-timing-function: ease-in-out;
    /* Delay before the animation starts */
    animation-delay: 1s;
    /* Number of times the animation should repeat */
    animation-iteration-count: infinite;
    /* Direction of the animation */
    animation-direction: alternate;

    /* Styles at different percentages of the animation */
    @keyframes myAnimation {

        /* Styles at 0% (start) of the animation */
        0% {
            /* CSS properties and values */
        }

        /* Styles at 50% (middle) of the animation */
        50% {
            /* CSS properties and values */
        }

        /* Styles at 100% (end) of the animation */
        100% {
            /* CSS properties and values */
        }
    }
```

Web Fonts : @font-face

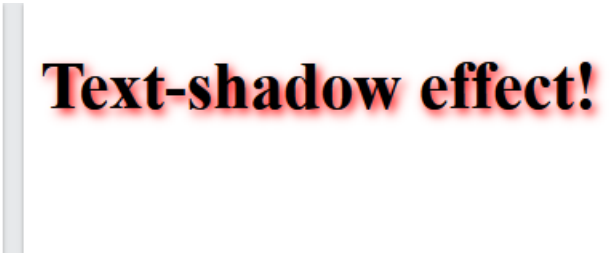
- Use fonts that are not installed on the user's computer
- include the font file url and will automatically be downloaded

```
/* Define the @font-face rule */
@font-face {
    font-family: 'MyCustomFont';
    /* Font family name */
    src: url('https://example.com/fonts/myfont.woff2') format('woff2'),
        /* URL to the font file */
        url('https://example.com/fonts/myfont.woff') format('woff');
    /* URL to the font file */
    /* Additional font properties (optional) */
    font-weight: normal;
    font-style: normal;
}

/* Use the custom font in CSS */
h1 {
    font-family: 'MyCustomFont', sans-serif;
    /* Font family name and fallback font */
}
```

Special Effects

Text Shadow Effect

```
<!DOCTYPE html>

h1 {
  text-shadow: 2px 2px 5px red;
}
```



Special Effects

Gradient effect

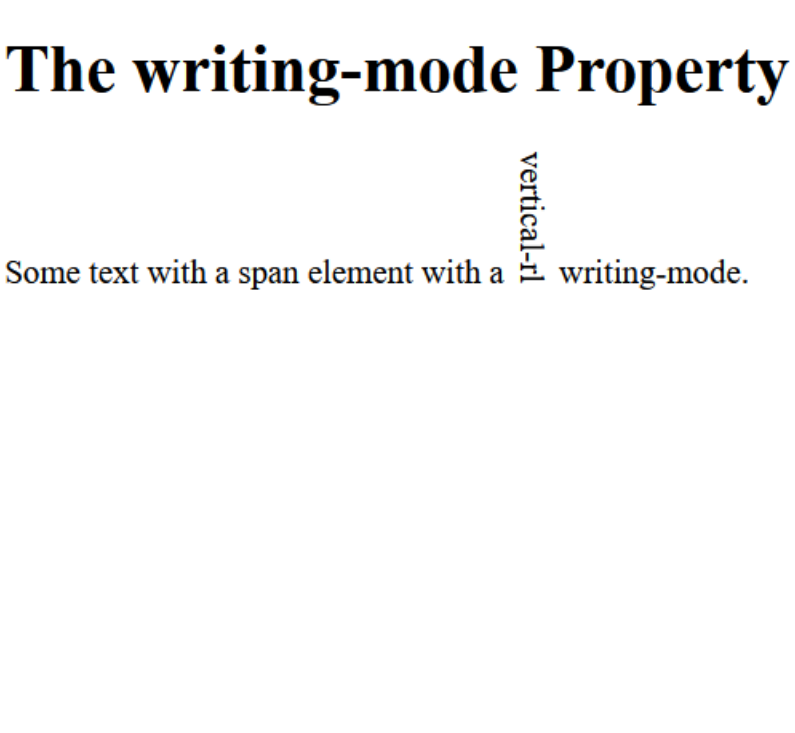
```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
  height: 200px;
  background-color: red; /* For browsers that do not support
gradients */
  background-image: linear-gradient(to right, red , yellow);
}
</style>
</head>
<body>
<p>This linear gradient starts red at the left, transitioning
to yellow (to the right):</p>
```

This linear gradient starts red at the left, transitioning to yellow (to the right):

Special Effects

Writing Mode

```
<!DOCTYPE html>
<html>
<head>
<style>

span.test2 {
  writing-mode: vertical-rl;
}

</style>
</head>
<body>
<h1>The writing-mode Property</h1>


<p>Some text with a span element
with a <span
class="test2">vertical-rl</span>
writing-mode.</p>
```



I feel like We can introduce reusability in CSS

And Modularity

```
nav {
  background-color: #007bff;
}

button {
  background-color: #007bff;
  color: white;
}
```

CSS Preprocessors – Programming(Dynamic Style Sheets)

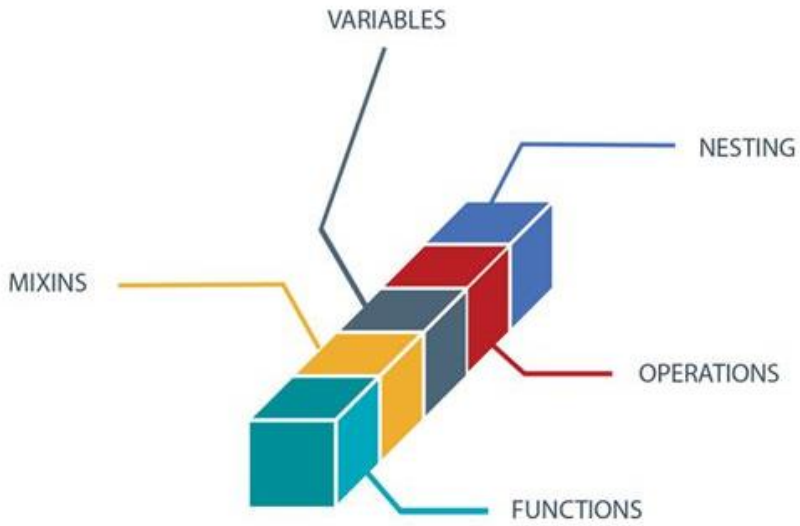
```
$primary-color: #007bff;

nav {
  background-color: $primary-color;
}

button {
  background-color: $primary-color;
  color: white;
}
```

CSS Preprocessors are capable of

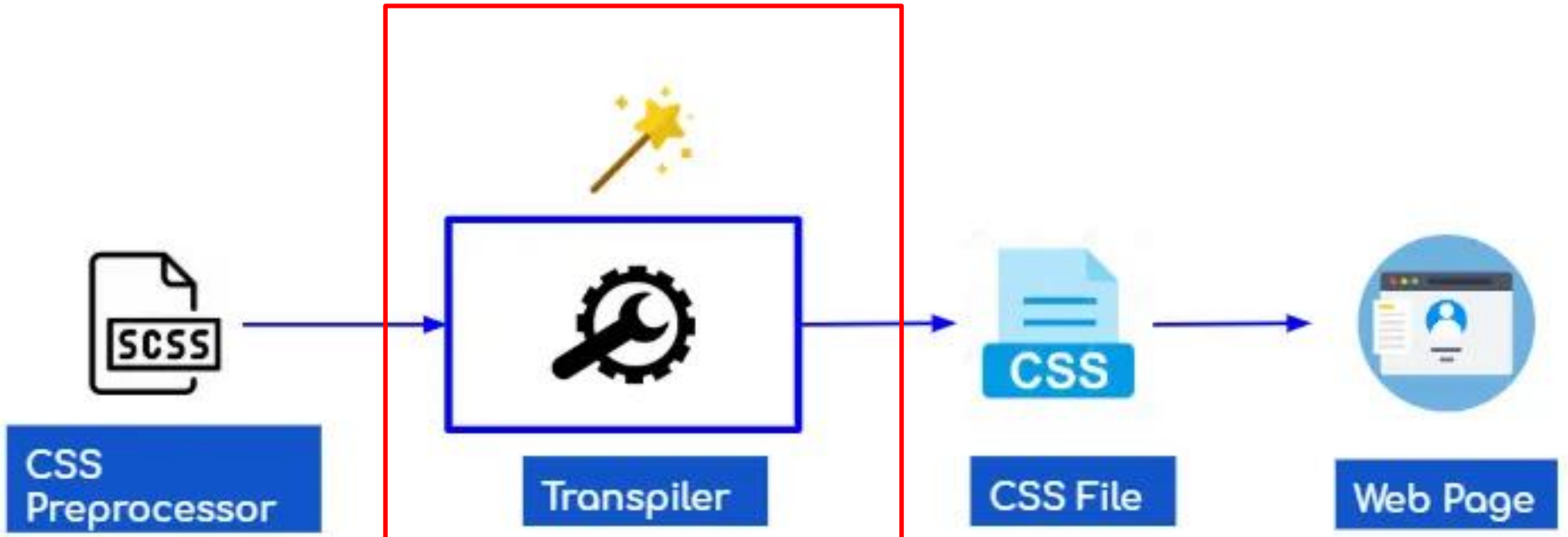
- Variables
- Functions
- Nesting
- Operations etc

CSS Preprocessors – High level Overview (Dynamic StyleSheets)

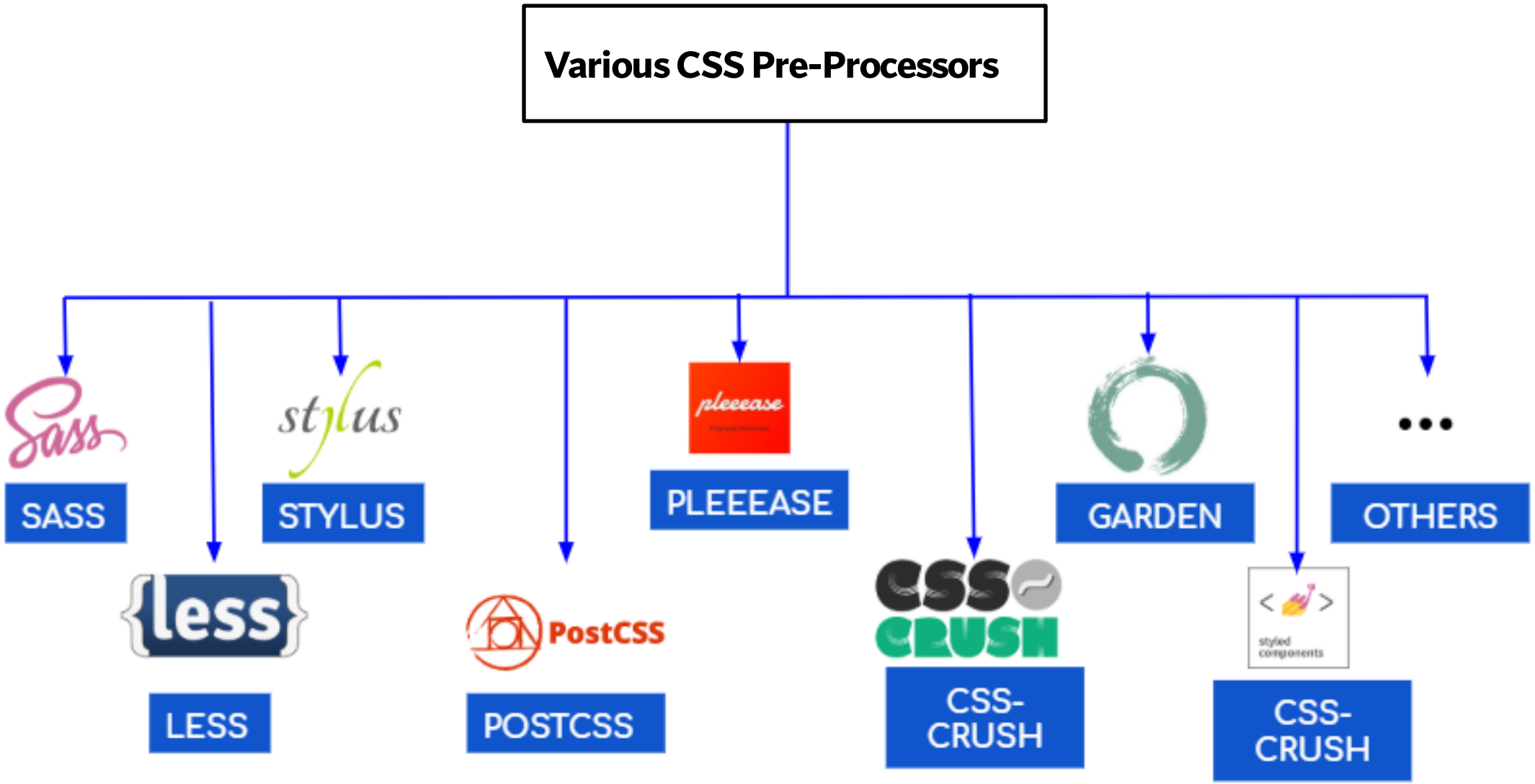
CSS preprocessors are also referred to as being 'dynamic style sheet languages'. They have been developed to add a programming functionality to the editing of Cascading Style Sheets. CSS Preprocessors convert code into a true CSS by taking the same written code from a simple preprocessed language (CSS with added extensions). Besides CSS preprocessors are used to add extensions which aren't used in CSS yet like: functions, mixins, nested rules, variables, operations and inheritance.



CSS Preprocessors – Has a Transpiler that converts into CSS



All this happens during build phase, not in production phase


Various CSS Pre-Processors
SASS
STYLUS
PLEEEASE
GARDEN
OTHERS
LESS
POSTCSS
CSS-CRUSH
CSS-CRUSH

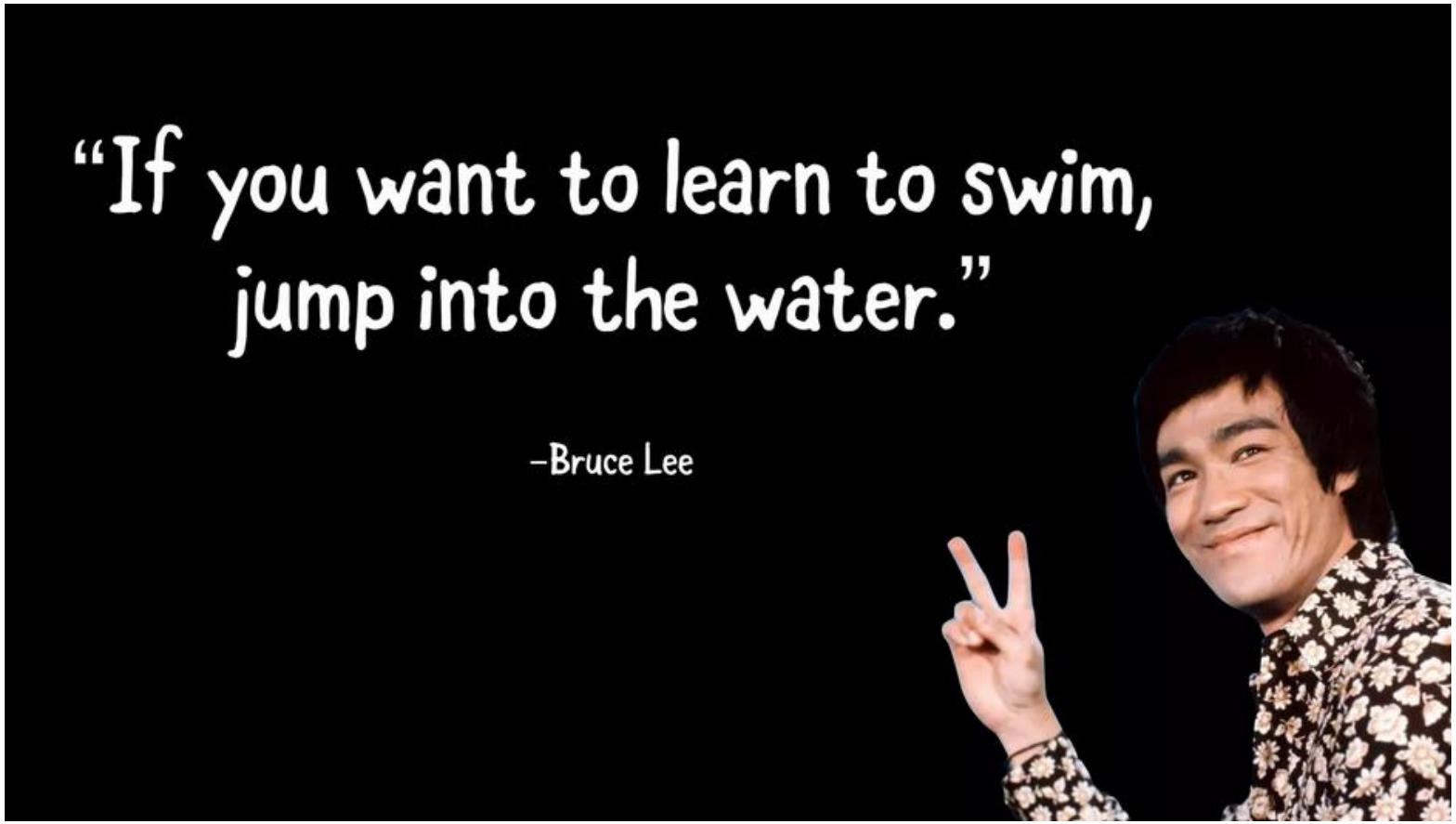
CSS Pre-Processors : Pros & Cons

Advantages

- CSS is made **more maintainable.**
- Easier to write nested selectors.
- **Variables for consistent theming**. Can share theme files across different projects. This is not necessarily useful with CSS custom properties (frequently called CSS variables).
- Sass and Less have features like **loops, lists, and maps can make configuration easier** and less verbose.
- **Splitting your code into multiple files** during development. CSS files can be split up too but doing so will require an HTTP request to download each CSS file and can increase performance

Disadvantages

- Requires tools **for preprocessing. Re-compilation time can be slow.**


“If you want to learn to swim,
jump into the water.”
–Bruce Lee

Q & A