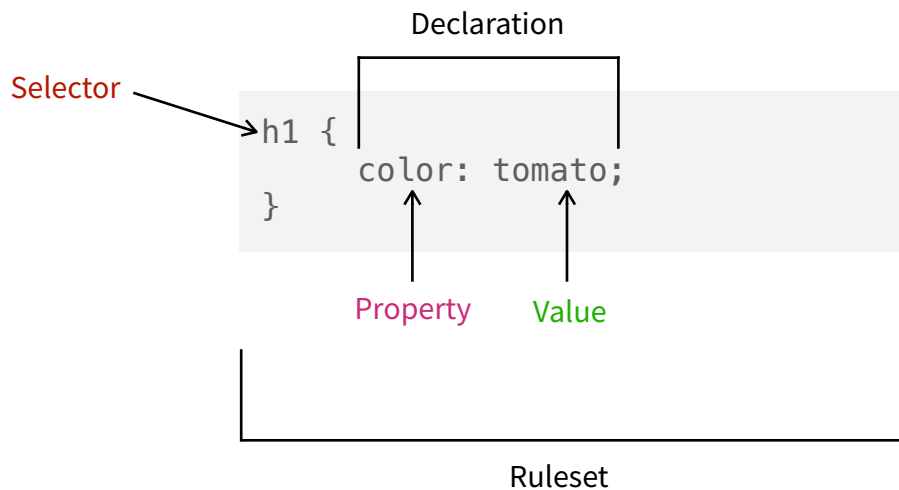




## Ruleset



## Pseudo Classes

Pseudo classes are applied to a selector to modify “specific states” of an element. The code on the right, demonstrates the usage of pseudo classes that are specifically applied to form elements, like input.

### SYNTAX

```
selector:pseudo-class {  
    property: value;  
}
```

```
div:click {  
    color: khaki;  
}  
div:hover {  
    color: tan;  
}  
div:focus {  
    color: deepskyblue;  
}  
div:visited {  
    color: dodgerblue;  
}
```

CSS

```
input:valid {  
    background: green;  
}  
input:invalid {  
    background: tomato;  
}  
input:required {  
    border-color: red;  
}  
input:required:invalid {  
    background: orange;  
}
```

CSS

More mostly used pseudo classes are -

`:active`, `:checked`, `:default`, `:disabled`, `:enabled`, `:first`, `:first-child`, `:last-child`, `:left`, `:right`, `:root`, `:lang()`, `:nth-child()`, etc.

## Pseudo Elements

Pseudo elements are applied to a selector to modify “specific parts” of an element.

### SYNTAX

```
selector::pseudo-element {  
    property: value;  
}
```

```
button::after {  
    content: “:-)”;  
}  
a::before {  
    content: “>>”;  
    color: red;  
}  
p::first-letter {  
    font-weight: 800;  
    font-size: 20pt;  
}  
p::selection {  
    color: white;  
    background: black;  
}
```

CSS

Button :-)

>> Link

**L**orem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

PREVIEW

More mostly used pseudo elements are -  
::cue, ::first-line, ::placeholder etc.

## Fonts

### Web Fonts

Fonts from the web can be used from various websites or foundries like Google Fonts, Adobe Typekit, etc.

```
<link rel=“stylesheet”  
href=“https://  
fonts.googleapis.com/css?  
family=Roboto”>
```

HTML

```
body {  
    font-family: ‘Roboto’;  
}
```

CSS

## System Fonts

As of June 2017, the recent syntax to include system fonts is as follows -

```
body {  
    font-family: "system-ui";  
}
```

CSS

The old syntax to include system fonts is as follows -

```
body {  
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,  
                Helvetica, Arial, sans-serif, "Apple Color Emoji",  
                "Segoe UI Emoji", "Segoe UI Symbol";  
}
```

CSS

Various font names are mentioned in the above example, to support font fallback, i.e. if one font is not found by the browser, then it will try to find successively mentioned font and so on. This is done to ensure that typography doesn't break on different browsers.

## Custom Fonts

To load custom font files in a webpage, export your font to one of the CSS compatible font format.

```
@font-face {  
    font-family: "MyCustomFont";  
    src: url("mycustomfont.woff2").format("woff2");  
}
```

CSS

Most widely used font file formats in CSS are WOFF/WOFF2, SVG/SVGZ, EOT, OTF/TTF.

## Styling

Following are the most used properties while styling fonts.

```
body {  
    font-family: 'Lato';  
    font-size: 16px;  
    font-style: italic;  
    font-stretch: normal;  
    font-variant: normal;  
    font-weight: 400;  
    line-height: 1em;  
}
```

CSS

## CSS Units

All the different CSS units can be divided into absolute and relative units.

Absolute	Relative
Pixels (px)	Percentages (%)
Inches (in)	Font-sizes (em & rem)
Centimeters (cm)	Character-sizes (ex & ch)
Millimeters (mm)	Viewport Width (vw)
Points (pt)	Viewport Height (vh)
Picas (pc)	Viewport Max (vmax)
	Viewport Min (vmin)

Relative units are based on -

- 1) The parents dimension (%)
- 2) The currently declared units for fonts (em, rem, ex, ch)
- 3) The viewport dimensions (vw, vh, vmin, vmax)

### Using Percentage

```
<div class="parent">  
  Hey  
  <div class="child">  
    There  
  </div>  
</div>
```

HTML

```
.parent {  
  background: tomato;  
  width: 200px;  
}  
.child {  
  background: khaki;  
  width: 50%;  
}
```

CSS



Hey  
There

PREVIEW

## Applying Units to Fonts - “em”

It's value is relative to the parent element.

```
<div class="parent">
  Hey
  <div class="child">
    There
  </div>
</div>
```

HTML

```
.parent {
  background: tomato;
  font-size: 20px;
  width: 200px;
}
.child {
  background: lime;
  font-size: 2em;
}
```

CSS



PREVIEW

## Applying Units to Fonts - “rem”

“rem” stands for Root EM, i.e. it's value is relative to the root instead of the parent element.

```
<div class="parent">
  Hey
  <div class="child">
    There
  </div>
</div>
```

HTML

```
:root {
  font-size: 10px;
}
.parent {
  background: tomato;
  font-size: 20px;
  width: 200px;
}
.child {
  background: lime;
  font-size: 1em;
}
```

CSS



PREVIEW

## Viewport Units

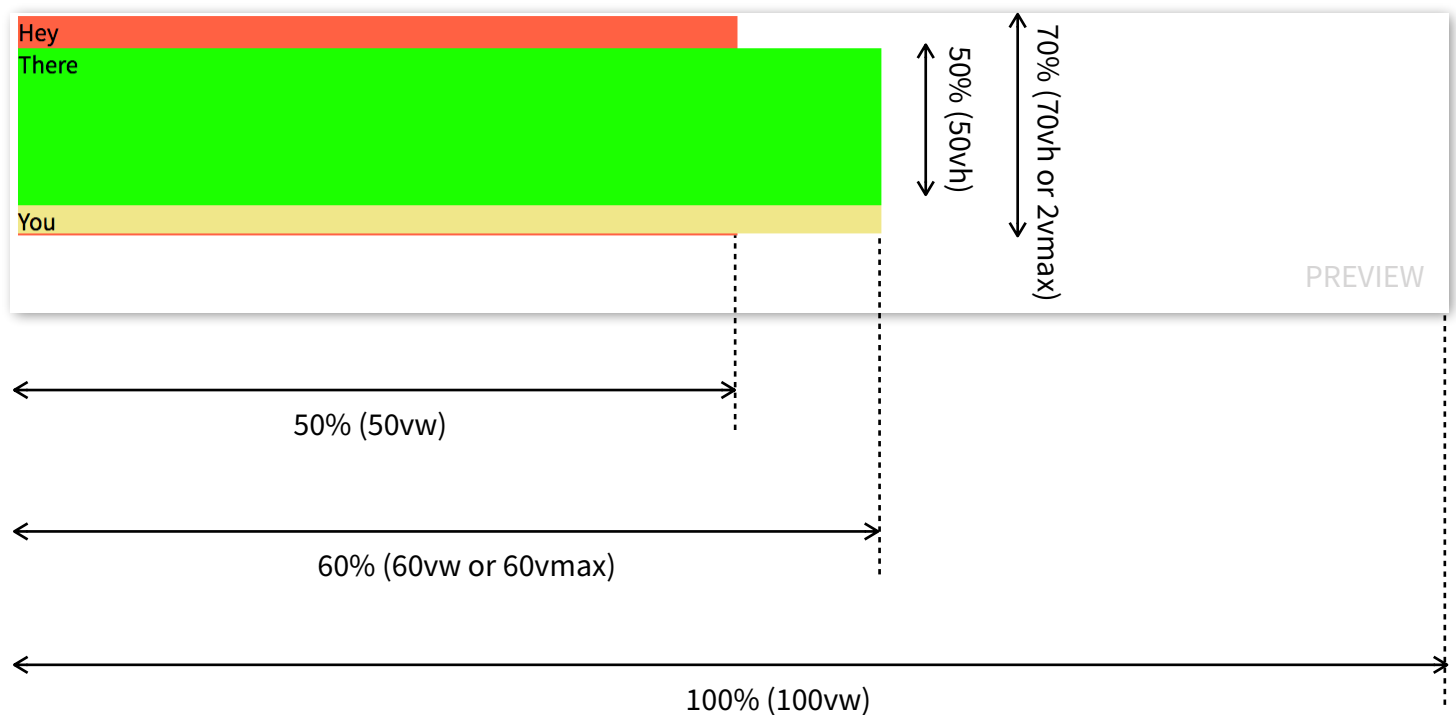
Viewport units divide the viewport into a grid of 100 units. Here, “vmax” is the maximum viewport length (width or height) which is larger in that viewport and similarly “vmin” is smaller between width and height.

```
<div class="parent">
  Hey
  <div class="child">
    There
  </div>
</div>
```

HTML

```
.parent {
  background: tomato;
  height: 70vh;
  width: 50vw;
}
.child {
  background: lime;
  height: 50vh;
  width: 60vw;
}
.second-child {
  background: khaki;
  height: 2vmax;
  width: 60vmax;
}
```

CSS



### Using Absolute Units

The preview on the RHS, displays all the currently supported font sizes and how they vary from each other. Points and Picas are generally used in print media.

20px

0.5in

1.5cm

10mm

5pt

2pc

PREVIEW

### Shorthands

Following are the most widely used shorthands. The code on RHS is the shorthand of the code on LHS.

```
h3 {  
  font-style: oblique;  
  font-weight: bold;  
  font-size: 2rem;  
  line-height: 1;  
  font-family: 'Roboto';  
}
```

CSS

```
h3 {  
  font: oblique bold 2rem/1  
        Roboto;  
}
```

CSS

```
div {  
  border-width: 1px;  
  border-style: dashed;  
  border-color: #afafaf;  
}
```

CSS

```
div {  
  border: 1px dashed #afafaf;  
}
```

CSS



The following margin property shorthands also apply to padding property.

```
div {  
    margin-top: 10px;  
    margin-right: 10px;  
    margin-bottom: 10px;  
    margin-left: 10px;  
}
```

CSS

```
div {  
    margin: 10px;  
}
```

CSS

```
div {  
    margin-top: 10px;  
    margin-right: 10px;  
    margin-bottom: 20px;  
    margin-left: 10px;  
}
```

CSS

```
div {  
    margin: 10px 20px;  
}
```

CSS

```
div {  
    margin-top: 10px;  
    margin-right: 20px;  
    margin-bottom: 30px;  
    margin-left: 20px;  
}
```

CSS

```
div {  
    margin: 10px 20px 30px;  
}
```

CSS

The following shorthand is used for the background property.

```
div {  
    background-color: #ddd;  
    background-image: url(image.png);  
    background-repeat: cover;  
    background-position: right bottom;  
}
```

CSS

```
div {  
    background: #ddd url(image.png) cover right bottom;  
}
```

CSS

## Positioning & Layouts

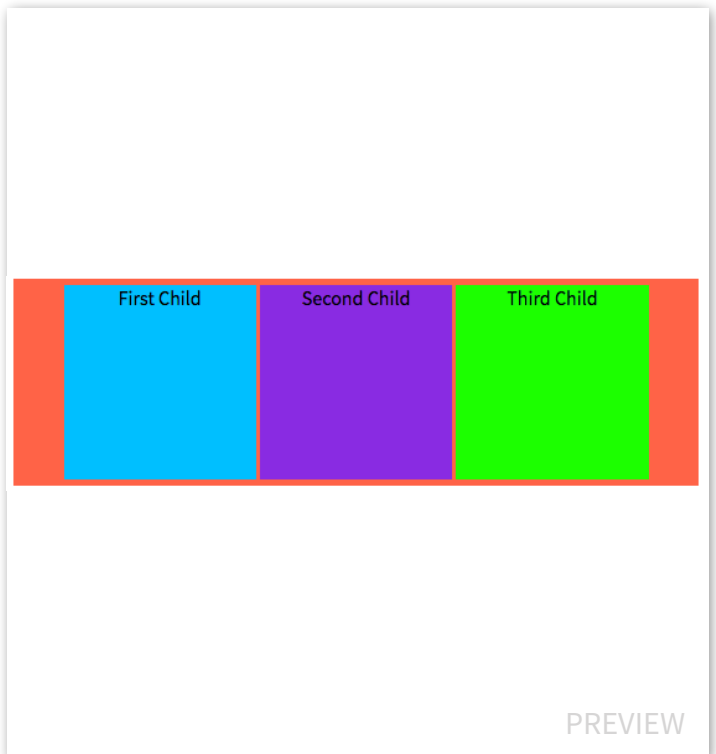
Following are the most widely used properties for positioning elements and making layouts.

- 1) display
- 2) position
- 3) top
- 4) bottom
- 5) left
- 6) right
- 7) z-index

Few examples of different usages are as follows:

```
.parent {  
    background: tomato;  
    padding: 5px;  
    text-align: center;  
    width: 70vw;  
}  
.child {  
    background: deepskyblue;  
    display: inline-block;  
    height: 20vw;  
    width: 20vw;  
}  
.second-child {  
    background: blueviolet;  
    display: inline-block;  
    height: 20vw;  
    width: 20vw;  
}  
.third-child {  
    background: lime;  
    display: inline-block;  
    height: 20vw;  
    width: 20vw;  
}
```

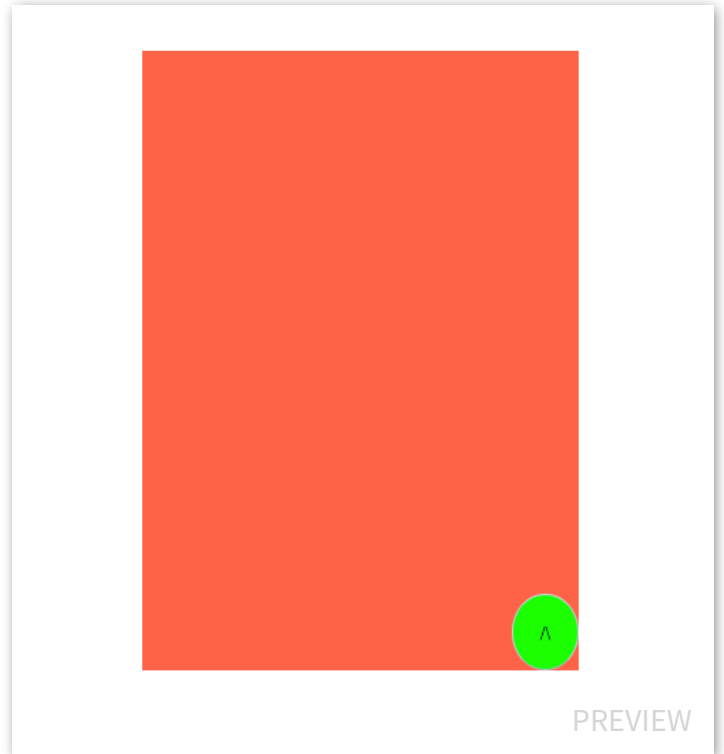
CSS



Making a simple floating action button, using positioning properties.

```
.parent {  
  background: tomato;  
  padding: 5px;  
  height: 70vh;  
  position: relative;  
  width: 40vw;  
}  
.fab {  
  background: lime;  
  bottom: 0;  
  border-radius: 50%;  
  position: absolute;  
  padding: 20px;  
  right: 0;  
}
```

CSS



## Styling

The following section covers basics of styling elements.

### Colors

The color property accepts following different color value formats:

- 1) HEX (eg: #ff0000)
- 2) RGB/RGBA (eg: rgb(255, 0, 0))
- 3) HSL/HSLA (eg: hsl(0, 100%, 50%))
- 4) Color Names (eg: red)

```
.hey {  
  background: rgb(255, 0, 0);  
}  
.there {  
  background: rgba(255, 0, 0,  
0.7);  
}
```

CSS



Here the difference between “rgb” and “rgba” values is that, in “rgba” one more value is defined which is for the alpha channel, setting the opacity.

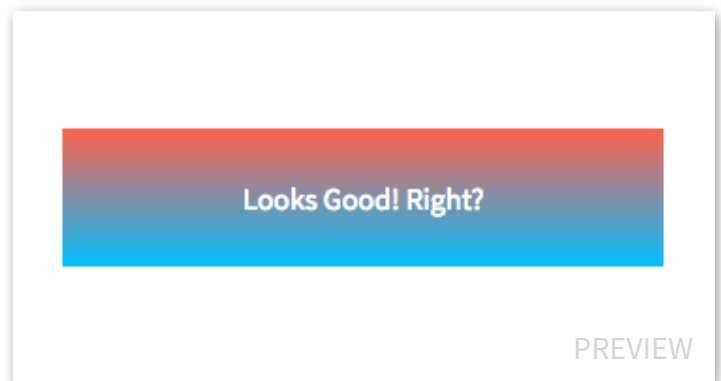
## Backgrounds

Backgrounds can be styled using different types of gradients from simple(see below) ones to more complex, using the following properties:

- 1) linear-gradient
  - 1) Directions: to right, to left, to top, to bottom
  - 2) Angles: <value>deg
  - 3) Positioning color stops
  - 4) Stacking gradients
- 2) radial-gradient
  - 1) Sizes: closest-side, farthest-corner
  - 2) Stacking gradients

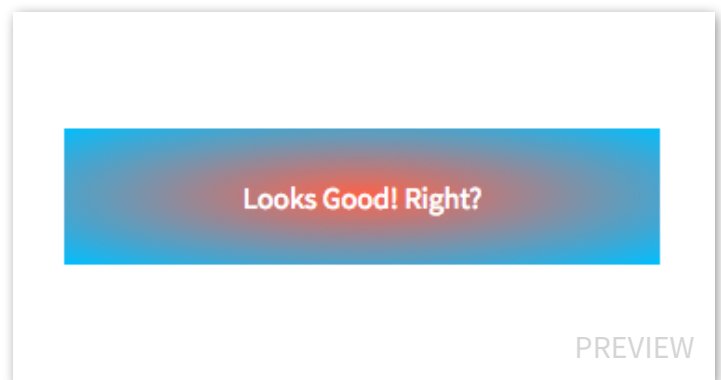
```
.parent {  
  background: linear-  
gradient(tomato, deepskyblue);  
  padding: 5px;  
  height: 10vh;  
  position: relative;  
  width: 40vw;  
}  
.text {  
  color: #fff;  
  text-align: center;  
  line-height: 4;  
}
```

CSS



```
.parent {  
  background: radial-  
gradient(tomato, deepskyblue);  
  padding: 5px;  
  height: 10vh;  
  position: relative;  
  width: 40vw;  
}  
.text {  
  color: #fff;  
  text-align: center;  
  line-height: 4;  
}
```

CSS



## Borders

Borders can be styled using `border-style` property of the `border` selector. The following styles are available:

- 1) none
- 2) hidden
- 3) dotted
- 4) dashed
- 5) solid
- 6) double
- 7) groove
- 8) ridge
- 9) inset
- 10) outset

```
.parent {  
    background: beige;  
    border: 10px ridge orange;  
    border-top-color: lime;  
    border-bottom-color: deeppink;  
    border-top-left-radius: 10%;  
    border-bottom-right-radius: 10%;  
    border-bottom-right-radius: 30%;  
    border-left-style: dotted;  
    padding: 5px;  
    height: 10vh;  
    position: relative;  
    width: 40vw;  
}
```

CSS



PREVIEW

## Cascading

```
<div class="parent">
  <div class="child">
    First Child
  </div>
  <div class="child" id="second-child">
    Second Child
  </div>
</div>
```

HTML

```
.parent {
  background: deepskyblue;
  padding: 5px;
  height: 10vh;
  width: 40vw;
}
.child {
  background: khaki;
}
.parent .child {
  background: tomato;
  border: 5px dotted white !important;
}
#second-child {
  background: lime;
  border: 5px solid black;
}
```

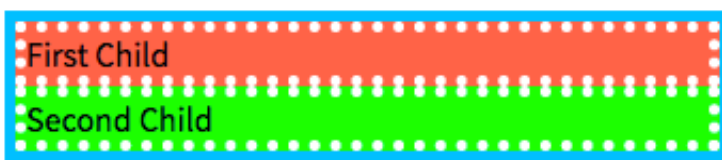
CSS

The ruleset which comes later in the source order wins(see background property).

`.parent .child` has higher specificity than `.child`, that is why former wins. Only way to over-ride ID selectors is to use `!important`.

**(Not Recommended)**

ID selectors over-ride class selectors.



PREVIEW

## Inheritance

By making use of inheritance, same rulesets can be applied to different elements without repetitively applying rulesets to each child elements of the parent(see below). But, not all rulesets should be inherited from the parent elements, for example `margin`.

Values supported for inheritance are:

- 1) `inherit`
- 2) `unset`
- 3) `initial`

```
<div class="parent">
  <div class="child">
    First Child
  </div>
  <div class="second-child">
    Second Child
  </div>
  <div class="third-child">
    Third Child
  </div>
</div>
<div class="outside">
  Outside
</div>
```

HTML

```
* {
  font-size: 10px;
}
.parent {
  background: deepskyblue;
  font-size: 20px;
  padding: 5px;
  height: 10vh;
  width: 40vw;
}
.second-child {
  font-size: unset;
}
.third-child {
  font-size: initial;
}
.outside {
  font-size: inherit;
}
```

CSS

First Child

**Second Child**

Third Child

Outside

PREVIEW

## At-Rules

Each at-rule has its own different syntax and following are the most used at-rules:

- 1) @media
- 2) @import
- 3) @font-face
- 4) @charset
- 5) @supports
- 6) @page
- 7) @keyframes
- 8) @counter-style
- 9) @font-feature-values

```
@charset "utf-8";
@supports (display: flex) {
    .parent {
        display: flex;
    }
}
@import "someCssfile.css";
@page {
    margin: 1cm;
}
@font-face {
    font-family: "Roboto";
    font-weight: 300;
    src: url("/roboto.woff2") format("woff2");
}
.child {
    background: deeppink;
}
.parent {
    background: deepskyblue;
    font-size: 20px;
    padding: 5px;
    height: 10vh;
    width: 40vw;
}
@media screen and(max-width: 600px) {
    .child {
        background: orangered;
    }
}
```

CSS



## Transitions

CSS transitions enable to control the properties of the element to animate, by defining delay, duration and timing function. Transitions only control the visualisations from start to end, but in order to loop the visualisations, CSS animation property is used. Along with transitions, `transform` property can also be used to scale, skew, rotate and translate the element.

**SYNTAX**

```
selector {  
  transition: <property> <duration> <timing-function> <delay>  
}
```

```
.parent {  
  background: deepskyblue;  
  font-size: 20px;  
  padding: 5px;  
  height: 10vh;  
  line-height: 3;  
  transition: padding 0.3s linear 0.1s;  
  width: 40vw;  
}  
.parent:hover {  
  padding: 20px;  
  transform: rotate(0.5turn);  
}
```

CSS

The above code is same as the following:

```
.parent {  
  background: deepskyblue;  
  font-size: 20px;  
  padding: 5px;  
  height: 10vh;  
  line-height: 3;  
  transition-property: padding;  
  transition-duration: 0.3s;  
  transition-timing-function: linear;  
  transition-delay: 0.1s;  
  width: 40vw;  
}  
.parent:hover {  
  padding: 20px;  
  transform: rotate(0.5turn);  
}
```

CSS

## Animations

CSS animation property is the short-hand for the following properties:

- 1) animation-name
- 2) animation-duration
- 3) animation-timing-function
- 4) animation-delay
- 5) animation-iteration-count
- 6) animation-direction
- 7) animation-fill-mode
- 8) animation-play-state

SYNTAX

```
selector {  
  animation: <name> <duration> <timing-function> <delay>  
            <count> <direction> <fill-mode> <play-state>  
}
```


```
<div class="parent">  
  <div class="child">  
    First Child  
  </div>  
  <div class="second-child">  
    Second Child  
  </div>  
  <div class="third-child">  
    Third Child  
  </div>  
</div>  
<div class="outside">  
  Outside  
</div>
```

HTML

```
.bar {  
  background: #ddd;  
  padding: 5px;  
  background: deepskyblue;  
  width: 10%;  
  animation: 3s linear 1s  
            infinite  
            alternate  
            extends;  
  @keyframes extends {  
    from {  
      width: 10%;  
    } to {  
      width: 50%;  
    }  
  }  
}
```

CSS

Processing...



PREVIEW

## Common CSS Mistakes

- 1) Reset the browser's CSS, as each browser defines its own default styles for the HTML elements. Recommended to use [normalize.css](#), in case of writing custom CSS from scratch.
- 2) Before using latest CSS3 properties, verify that the end-users' browsers will support them. Recommended to use [caniuse](#) to verify support for different browsers.
- 3) Writing long selector chains. In the following example, to set the font style of the description inside child div, a separate class should be created. So, the developer needs to maintain balance between the number of classes and the length of the selector chains and accordingly make the decision.

```
.parent .child .head p {  
    font-size: 16px;  
}
```

✗  
CSS

```
.child-desc {  
    font-size: 16px;  
}
```

✓  
CSS

- 4) Not using shorthand properties. The aim is to decrease the size of the CSS files as much as possible and the amount of characters/lines directly affects the size of file.
- 5) Using color names like black, instead of hexadecimal color codes, like #000.
- 6) In medium to large code bases, the amount of classes are also large and after several iterations, some part of the CSS code is left unused and this adds up to the overall junk code the project; which affects the response time and file loading time on the browser.
- 7) Fallback fonts should be provided, in order to ensure that the user-interface doesn't break even if some specific fonts are not loaded for the user.
- 8) Never use one CSS file for all the HTML files, in a project. Rather club each HTML file with its own CSS file. This ensure quick development, easy bug-fixes and enhancements.
- 9) Ignoring [accessibility web standards](#) while making components like modals. This directly impacts the user experience and can degrade the overall performance on different devices.
- 10) Hard-coding units for font-sizes or container paddings/margins, which renders an unresponsive user-interface. Rather relative units should be used, wherever applicable.
- 11) Making heavy use of !important, in case of using some third-party framework like Bootstrap, Foundation, MDL, etc. This adds up to a chunks of un-used or over-ridden code.
- 12) Writing inline styles in HTML files, is not recommended. Instead separate CSS files should be used and linked to their specific HTML files.
- 13) Using depreciated CSS properties should be avoided as they might be supported on few browsers, but since they are depreciated so they will be removed by browser vendors in their next few updates.
- 14) Using long CSS class names is not desirable as they add up the overall size of the CSS and HTML files. It is recommended to use some naming convention throughout the project.

# Custom Components

## Button Component

```
<div class="container">
  <div class="btn">
    Button
  </div>
</div>
```

HTML

```
* {
  font-family: 'Source Sans Pro';
}
.container {
  display: flex;
}
.btn {
  background: #FD0156;
  border: 10px solid #FF614F;
  border-radius: 2px;
  color: #fff;
  cursor: pointer;
  font-size: 18px;
  padding: 5px;
  text-transform: uppercase;
  transition:
    color 0.3s linear 0.1s,
    background 0.3s linear 0.1s,
    box-shadow 0.3s linear 0.1s,
    border 0.3s linear 0.1s;
  letter-spacing: 1px;
}
.btn:hover {
  background: #FF614F;
  border: 10px dashed #FD0156;
  box-shadow: 0px 0px 20px #FD0156;
  color: #ddd;
}
```

CSS



PREVIEW

## List Component

```
<div class="list">
  <ul>
    <li class="list-head">
      List Head
    </li>
    <li class="list-item">
      Item One
    </li>
    <li class="list-item">
      Item Two
    </li>
    <li class="list-item">
      Item Three
    </li>
    <li class="list-item">
      Item Four
    </li>
  </ul>
</div>
```

HTML

```
.list {
  box-shadow: 0px 0px 20px 0px
    #000000024;
  margin: 25%;
}
ul {
  padding: 0;
}
.list-head {
  background: tomato;
  border-top: 1px solid tomato;
  border-top-left-radius: 5px;
  border-bottom: 1px solid tomato;
  border-top-right-radius: 5px;
  color: #fff;
  font-family: 'Source Sans Pro';
  font-size: 15px;
  list-style: none;
  padding: 10px;
  text-transform: uppercase;
}
.list-item {
  border-bottom: 1px solid #ddd;
  border-left: 1px solid tomato;
  border-right: 1px solid tomato;
  color: salmon;
  font-family: 'Source Sans Pro';
  font-size: 20px;
  list-style: none;
  padding: 5px;
  text-transform: capitalize;
  transition:
    font-size 0.3s linear 0.1s;
}
.list-item:last-child {
  border-bottom: 1px solid tomato;
}
.list-item:hover {
  background-color: salmon;
  color: gold;
  cursor: pointer;
  font-size: 23px;
}
```

CSS

LIST HEAD

Item One

Item Two

Item Three

Item Four

PREVIEW

## Form Component

```
<form>
  <div class="form-element">
    <label for="firstName">first name</label>
    <input id="name" type="text" placeholder="Enter Your First Name" required>
  </div>
  <div class="form-element">
    <label for="lastName">last name</label>
    <input id="lastName" type="text" placeholder="Enter Your Last Name" required>
  </div>
  <div class="form-element">
    <label for="address">address</label>
    <textarea id="address" type="text" rows="5"></textarea>
  </div>
  <div class="form-element">
    <label for="email">email</label>
    <input id="email" type="email" placeholder="Enter Your Email Address" required>
  </div>
  <div class="form-element">
    <label for="password">password</label>
    <input id="password" type="password" placeholder="Enter Password" required>
  </div>
  <div class="form-element">
    <button class="btn" type="submit">Submit</button>
  </div>
</form>
```

HTML

```
* {
  font-family: 'Source Sans Pro', sans-serif;
  font-size: 15px;
}
form {
  box-shadow: 0px 0px 20px 0px #1c807c;
  padding: 10px;
}
.form-element {
  display: flex;
  flex-direction: column;
  padding-bottom: 3vh;
}
label {
  font-weight: 500;
  padding-bottom: 0.5vh;
  text-transform: uppercase;
}
input, textarea {
  border: none;
  border-bottom: 1px solid deepskyblue;
```

CSS

```
  color: dodgerblue;
  padding: 5px;
  transition: border-bottom 0.3s linear 0.1s;
}
input:focus, textarea:focus {
  outline: none;
  border-bottom: 3px solid mediumturquoise;
}
form button {
  background-color: mediumturquoise;
  color: #fff;
  font-size: 18px;
  margin-top: 5vh;
  padding: 10px;
  text-transform: uppercase;
  transition: border 0.3s linear 0.1s;
}
form button:hover {
  border: 3px solid deepskyblue;
  cursor: pointer;
}
```

CSS

FIRST NAME

Enter Your First Name

LAST NAME

Enter Your Last Name

ADDRESS

EMAIL

Enter Your Email Address

PASSWORD

Enter Password

SUBMIT

PREVIEW