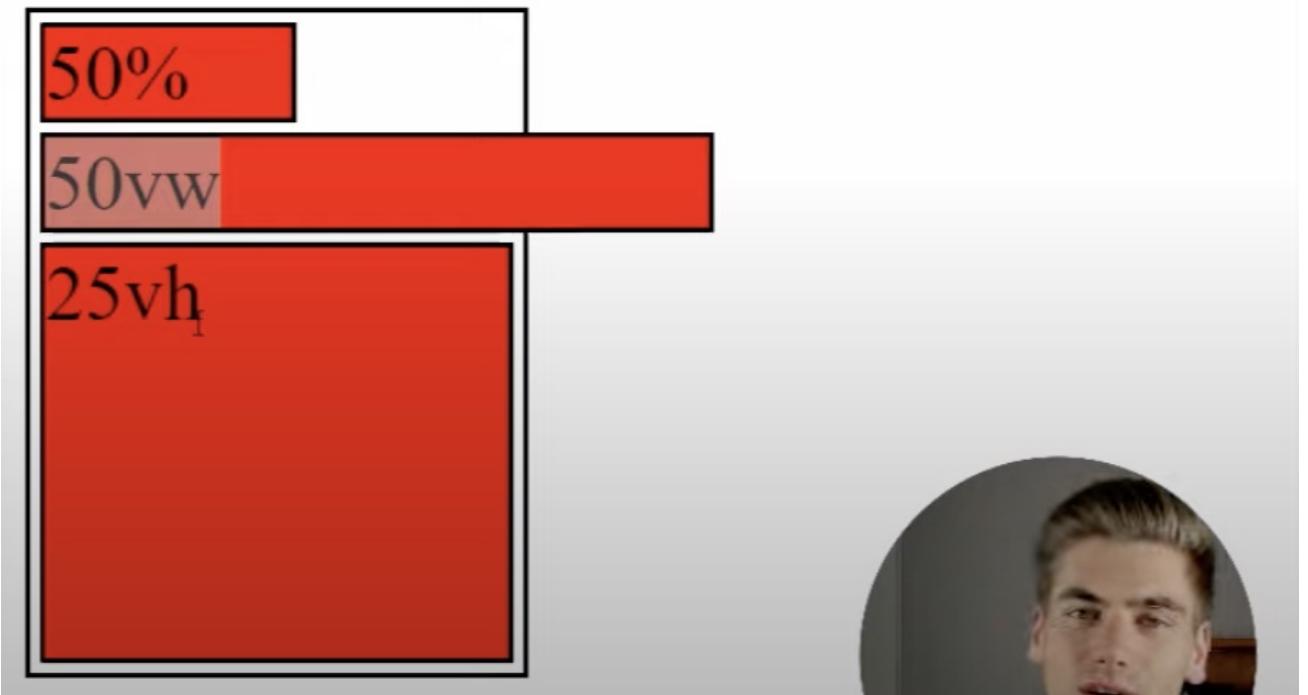
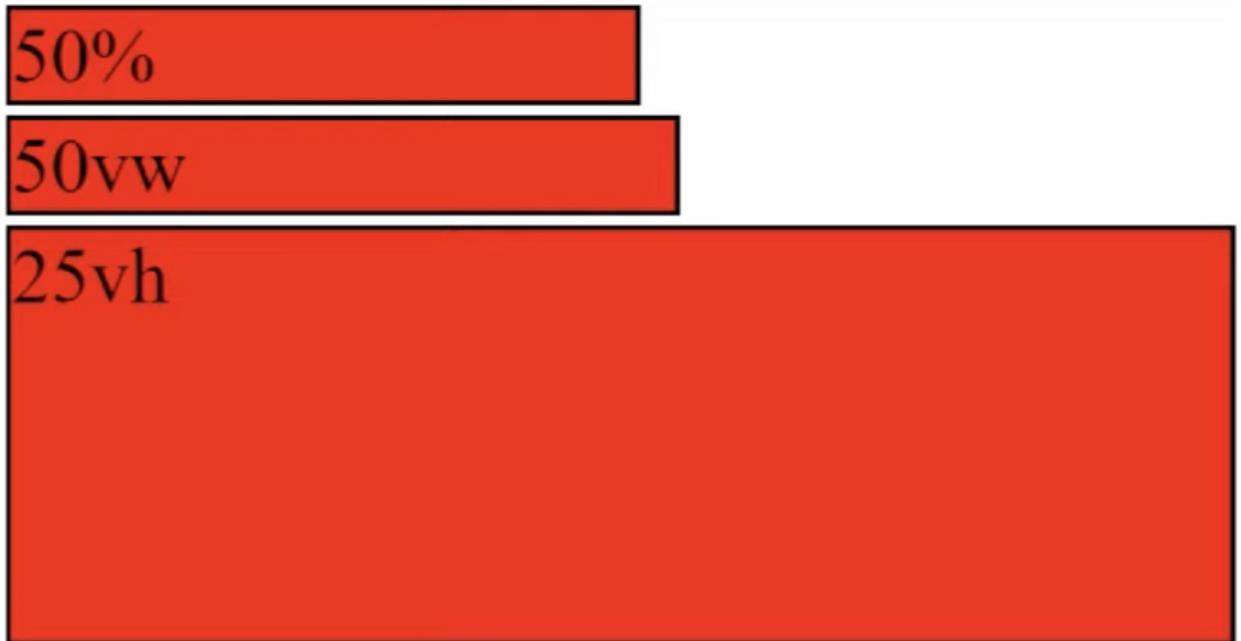


CSS

[Here Is the playlist I followed for the speedrun](#)

units

- percentage units - parent size
- vh vw units will overflow parent



- em relative to parent

- rem relative to root
 - based on font size
-

1rem
1em
2rem
I
2em

■ 1em icon
■ 1rem icon
■ 1em icon
■ 1rem icon

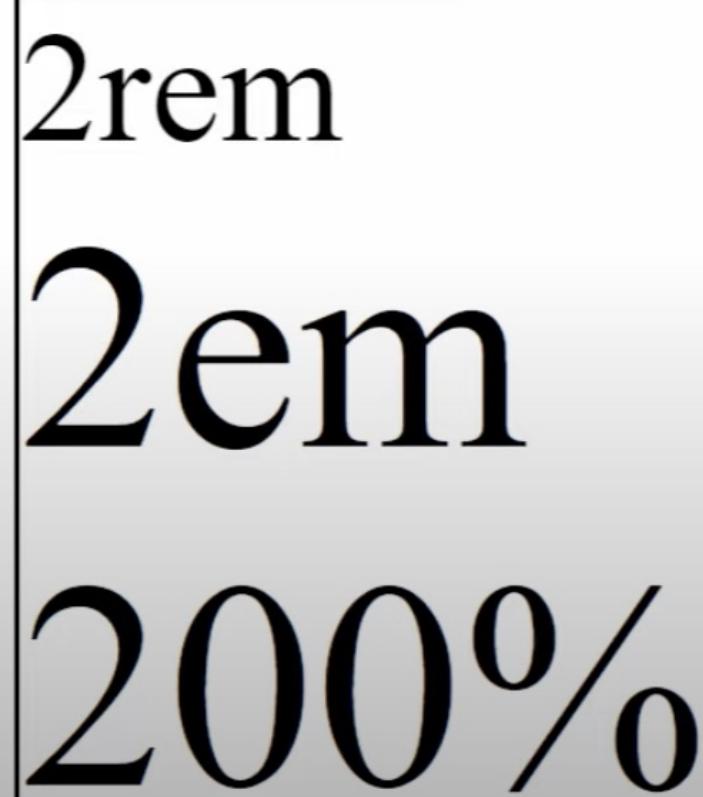


- em get nested and grow quickly

- percentages exactly same as em



2rem
2em
200%



2rem
2em
200%



viewports

- vh,vw percentage of screen
- vmax,vmin percentage of what is larger/smaller
- vi/vb (inline/block) to support different languages and their writing mode
- small, dynamic, large - for mobile
 - small - all obstructions
 - dynamic - obstruction transition
 - large - no obstructions
 - obstructions(top bottom bars,navbar)

- dvh scales between svh and lvh

display

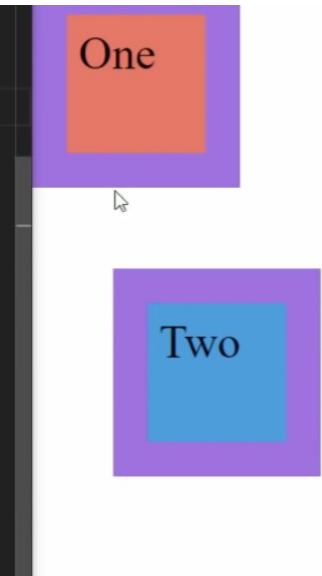
- block have space above and below them take up the full width
- inline - take up the least possible space - img, span, a
- inline - can't set width and height
- inline-block - takes up the min space in the same line but can set width and height
- none - acts as if element deleted

box model

- height and width are of the innermost content
- margin collapses with the element next to it
- the larger of the defined margin takes over

```

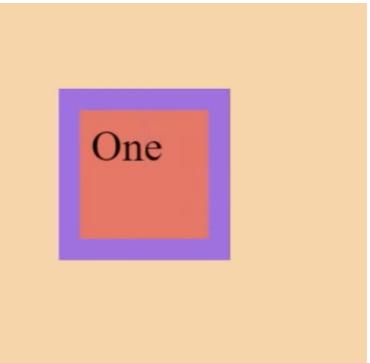
11  width: 100px;
12  border: 30px solid #9B51E0;
13  margin: 0px;
14 }
15
16 .box-two {
17  background-color: #2D9CDB;
18  border: 0 solid #27AE60;
19  padding: 10px;
20  height: 100px;
21  width: 100px;
22  border: 30px solid #9B51E0;
23  margin: 70px;
24 }
```



- total height and width = content + padding + border
- margin doesn't contribute to total width height
- **box-sizing:border-box** - sets the height and width css rules to total height and width i.e with padding and border

```

8  background-color: #EB5757;
9  padding: 10px;
10 height: 100px;
11 width: 100px;
12 border: 20px solid #9B51E0;
13 margin: 100px;
14 /* box-sizing: border-box; */
15 }
16
17 .box-two {
```



```
background-color: #EB5757;
padding: 10px;
height: 100px;
width: 100px;
border: 20px solid #9B51E0;
margin: 100px;
box-sizing: border-box;
```



- content size depends upon padding and border

position

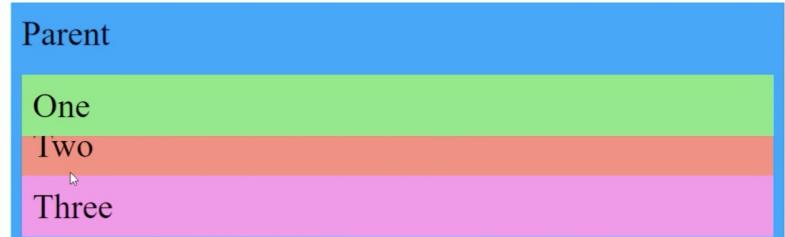
- static - follow the document flow
- static cannot have anything positioned absolute to it
- relative - acts like static but allows defining top left bottom right
- relative elements can overflow the parent
- relative - allows to change position relative to its static position (gets removed from the document flow)
- the space(position) where the element would normally be will be taken up

```
.parent {
}

.child-one {
    position: relative;
    top: 10px;
}

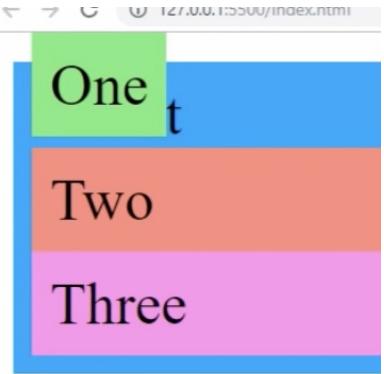
.child-two {
}

.child-three {
}
```



- absolute deletes the element from document flow itself all the other elements follow normally
- the space(position) the element would take normally is not taken up
- for other elements this one does not exist
- top left right bottom are with respect the page
- in a relative position parent the absolute element children would have top left bottom right with respect to the parent

```
1 .parent {  
2   |  
3 }  
4  
5 .child-one {  
6   position: absolute;  
7   top: 0;  
8 }  
9
```

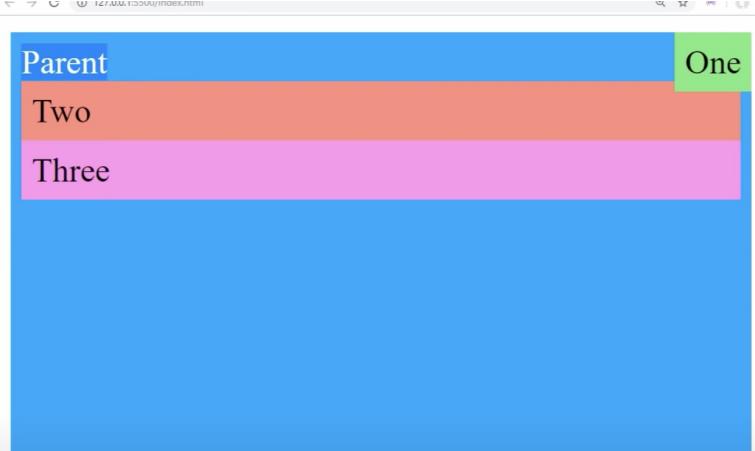


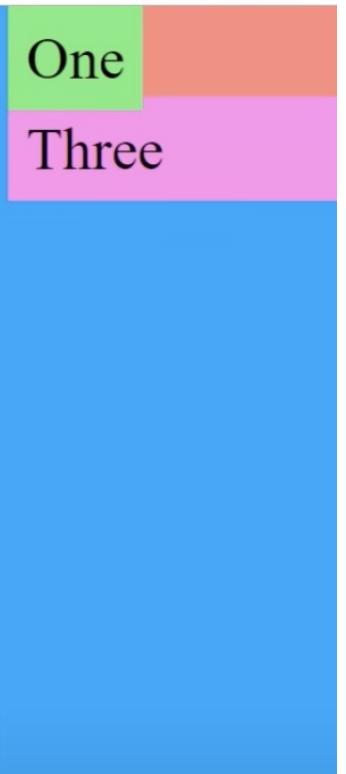
```
1 .parent {  
2   position: relative;  
3 }  
4  
5 .child-one {  
6   position: absolute;  
7   top: 0;  
8 }  
9
```



- fixed top left right bottom based on the entire html element does not copy with relative parent
- fixed elements do not scroll with page

```
1 .parent {  
2   position: relative;  
3   height: 200vh;  
4 }  
5  
6 .child-one {  
7   position: absolute;  
8   top: 0;  
9   right: 0;  
10 }  
11  
12 .child-two {  
13 }  
14  
15 .child-three {  
16 }  
17  
18 }
```





```
1 .parent {
2   position: relative;
3   height: 200vh;
4 }
5
6 .child-one {
7   position: fixed;
8   top: 0;
9 }
10
11 .child-two {
12
13 }
14
15 .child-three {
16
17 }
```

- sticky - relative by default but as soon its outside of the viewport it becomes fixed position

media query

- syntax

```
@media [screen,print,screenreader,all] [operator] (selector) {rules}
```

- default is all, and
- max-width - css in {} applies if width is equal to greater than specified width
- media queries are like other selectors if the specificity is same the css in bottom of the page applies ignoring media query selected styles

```
.subtitle {
  font-size: 4rem;
}

@media (max-width: 500px) {
  body {
    color: blue;
  }
}

body {
  color: red;
}
```

Title

Sub Title

- media queries generally placed at the bottom of the document, style tage
- landscape/portrait

```
@media (orientation: landscape/portrait)
```

```
@media (max-width: 500px) {  
  body {  
    color: blue;  
  }  
}  
  
@media (orientation: landscape) {  
  .title {  
    color: green;  
  }  
}  
  
@media (orientation: portrait) {  
  .subtitle {  
    color: cyan;  
  }  
}
```

Title

Sub Title

```
@media (max-width: 500px) {  
  body {  
    color: blue;  
  }  
}  
  
@media (orientation: landscape) {  
  .title {  
    color: green;  
  }  
}  
  
@media (orientation: portrait) {  
  .subtitle {  
    color: cyan;  
  }  
}
```

Title

Sub Title

- can combine query selectors

```
@media (max-width: 500px) {  
  body {  
    color: blue;  
  }  
}  
  
@media (orientation: landscape) and (max-width: 500px) {  
  .title {  
    color: green;  
  }  
}  
  
@media (orientation: portrait) {  
  .subtitle {  
    color: cyan;  
  }  
}
```

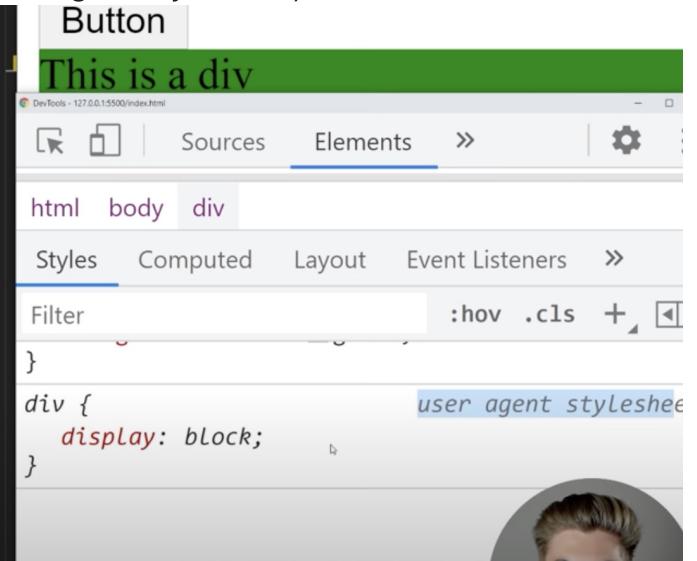
Title

Sub Title

- and both true for or we just use comma

inheritance

- initial - sets it to the default value according to formal definition in css (mdn docs)
- it also resets the browser set stylesheet (user agent stylesheet)



The screenshot shows a browser's developer tools with the "Elements" tab selected. A green bar at the top contains the text "This is a div". Below the bar, the "user agent stylesheet" is visible in the "Styles" section, showing a rule for "div" with "display: block;". The "Computed" tab shows the final rendered style for the green bar.

```

<head>
  <style>
    button {
      /* background-color: red; */
      border: none;
      font-size: 1rem;
      font-family: Roboto; */
    }

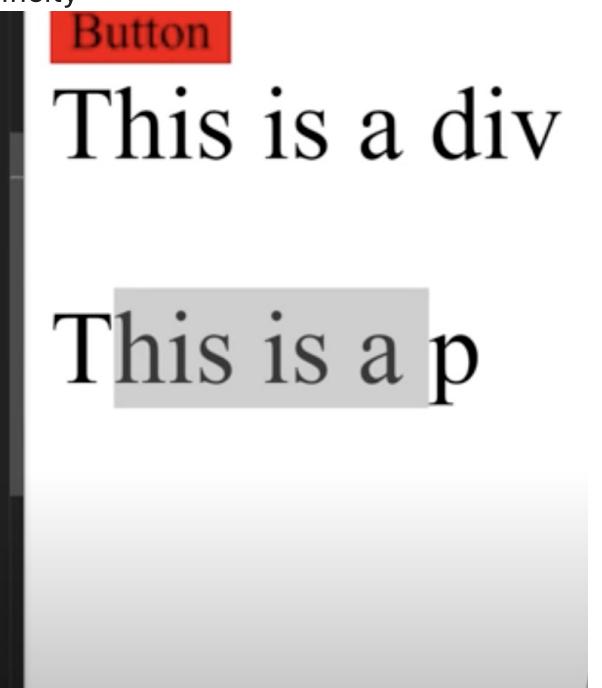
    div {
      background-color: green;
    }
  </style>
</head>
<body>
  <button>Button</button>
  <div>This is a div</div>
</body>
</html>

```

- hierarchy

user css >> browser css >> formal definition default values

- inherit - gets the values applied to the parent
- some properties are inherited by default like font size
- by setting the value of a rule to inherit we get the same value set in the parent
- inherits based on parent not on the selector specificity



The screenshot shows a browser's developer tools with the "Elements" tab selected. It displays three elements: a red button, a green "div" containing the text "This is a div", and a grey "p" containing the text "This is a p". The "user agent stylesheet" shows a "div" rule with "font-size: 2rem;". The "p" rule has "font-size: inherit;" set. The "Computed" tab shows the final rendered font sizes: the button is 1.25rem, the green "div" is 2rem, and the grey "p" is 2rem.

```

}
div {
  font-size: 2rem;
}

p.important {
  font-size: inherit;
}

p {
  font-size: 1.25rem;
}
</style>
</head>
<body>
  <button>Button</button>
  <div>
    This is a div
    <p class="important">This is a p</p>
  </div>
</body>

```

```
    font-size: 1rem;
    font-family: inherit;
}

div {

p.important {
    font-size: inherit;
}

p {
    font-size: 1.25rem;
}

```

</style>

</head>

<body>

<button>Button</button>

<div>

This is a div

<p class="important">This is a p</p>

</div>

Button

This is a div

This is a p

```
body {
    font-size: 2rem;
}

p.important {
    font-size: inherit;
}

p {
    font-size: 1.25rem;
}

```

</style>

</head>

<body>

<button>Button</button>

<div>

This is a div

<p class="important">This is a p</p>

</div>

</body>

</html>

Button

This is a div

This is a p

- unset - if the property inherits by default it is inherited from the parent if doesn't it is set to initial
- all: unsets - sets all the properties to their initial value if they don't inherit and parent values if it inherits

```
<style>
  button {
    all: unset;
    background-color: red;
    border: none;
    font-size: 1rem;
    font-family: inherit;
  }
</style>
```

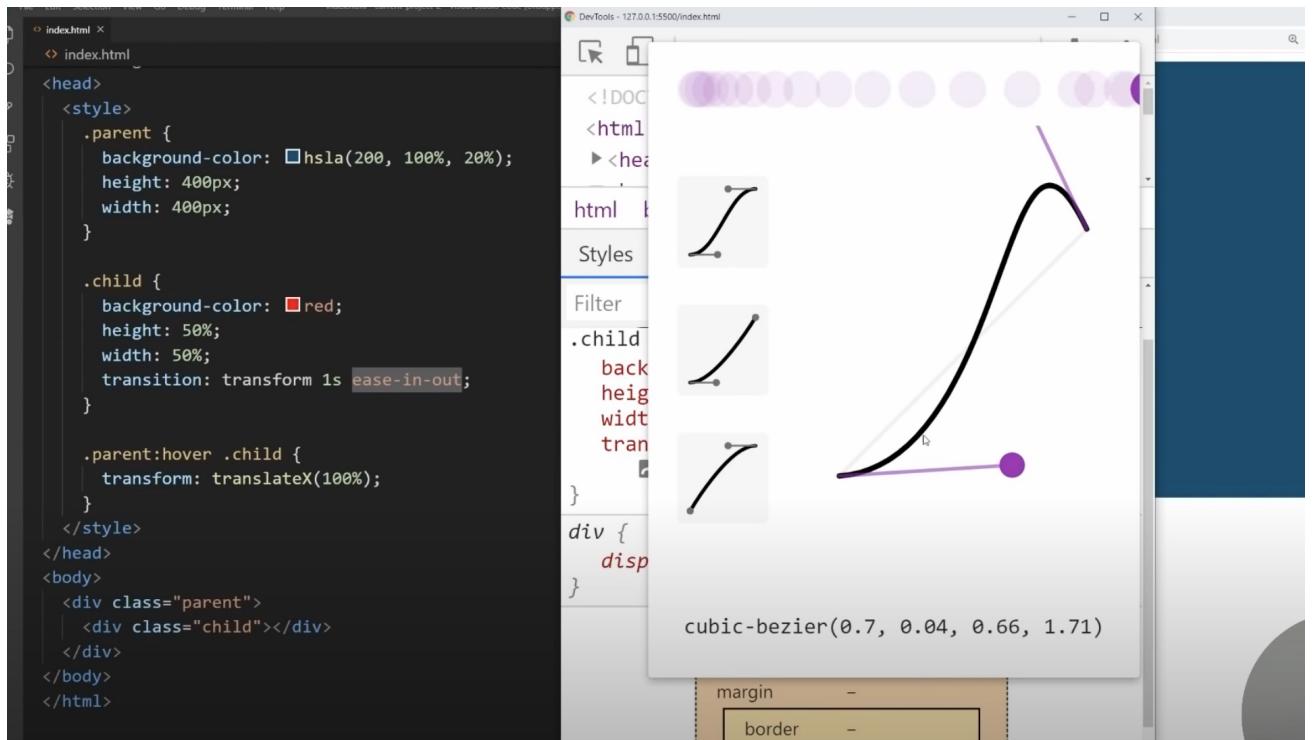
Button

```
<index.html>
<style>
  button {
    all: unset;
    background-color: red;
  }
</style>
</head>
<body>
  <button>Button</button>
</body>
</html>
```

Button

animations

- transform - changes properties
- transition - defines properties of transform
- transition can be all, transform, color etc.
- if we set transition to only transform only the transform will be animated with properties of transition like duration other changes like color change will be instant
- transform is defined as a property of parent or hovered object
- transition is defined as property of child
- to set own timing function



animation

animation: [name] [duration] [delay] [timing function] [animation-fill-mode]
– forwards – same properties as **100%** after animation
– backwards – before animation set values as **0%**
– both – combines both forwards and backwards]

```
@keyframes [name-of-animation] {  
    1% {transform: [values]}  
    2{transform: [values]}  
    .  
    .  
    100%{transform: [values]}  
}
```

variables
