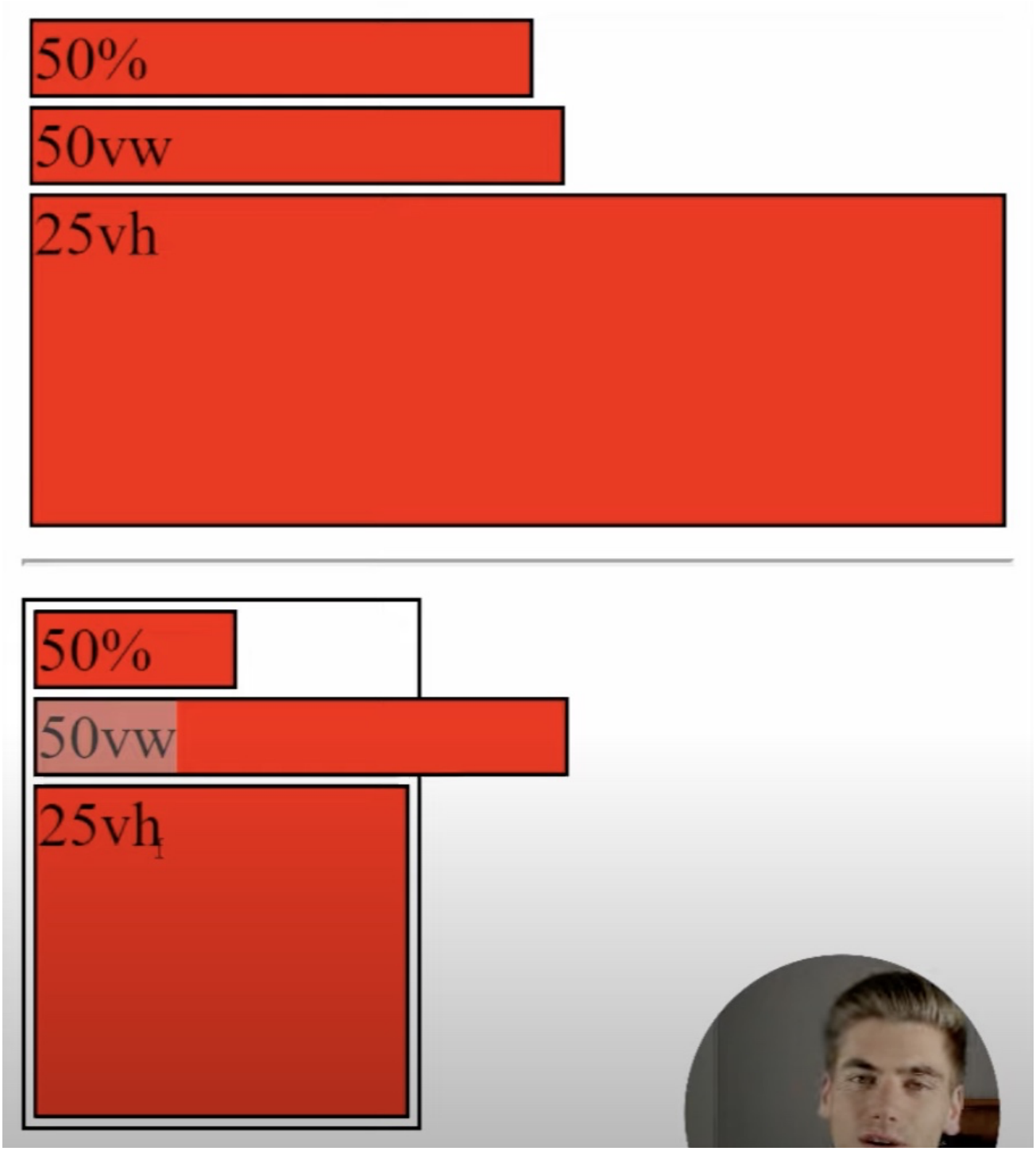
css

[Here Is the playlist I followed for the speedrun]

units

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

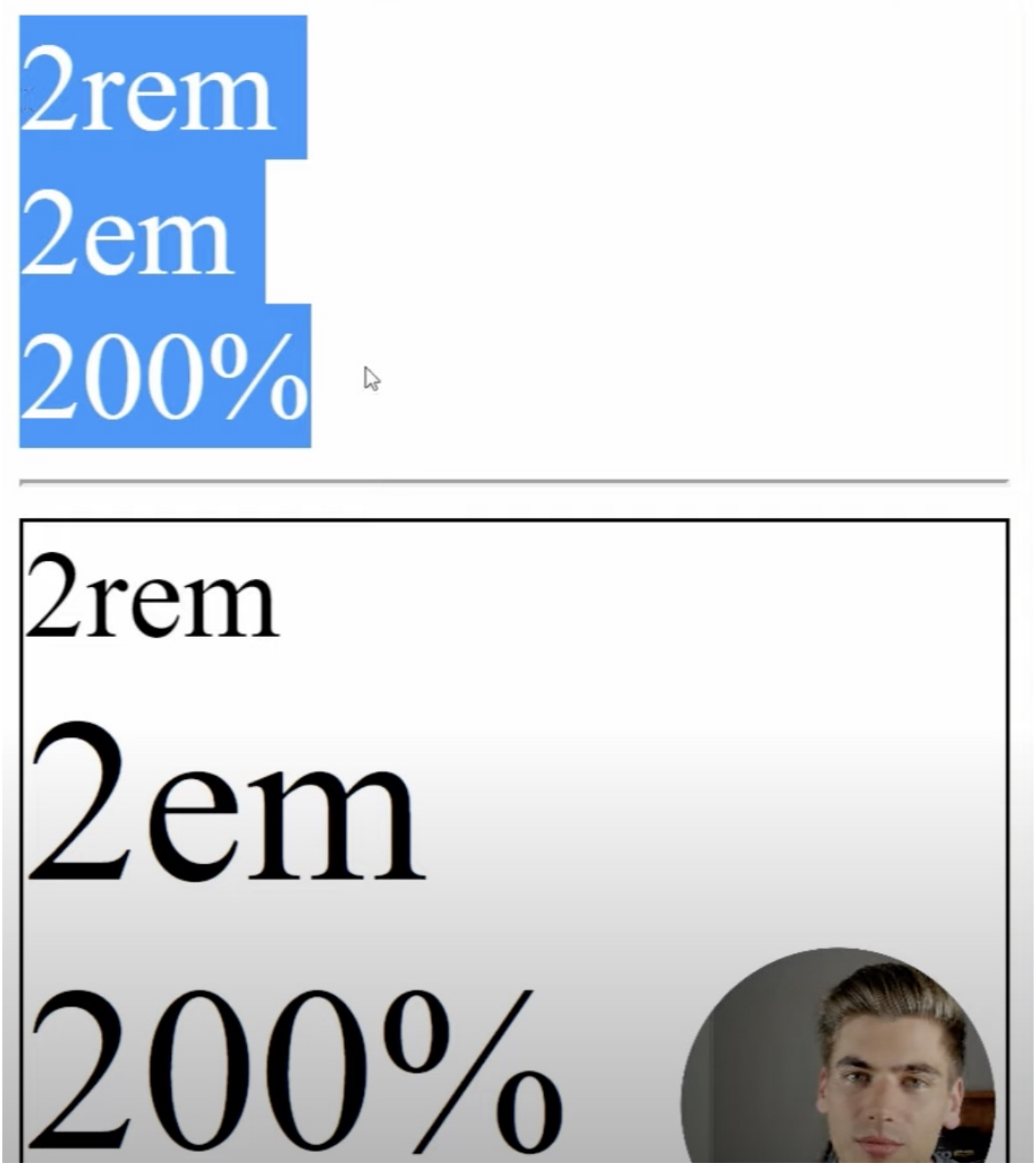


- em relative to parent

- rem relative to root
- based on font size

- em get nested and grow quickly

- percentages exactly same as em



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 obstuctions
  dynamic - obstruction transition
  large - no obstructions
  obstructions(top bottom bars,navbar)

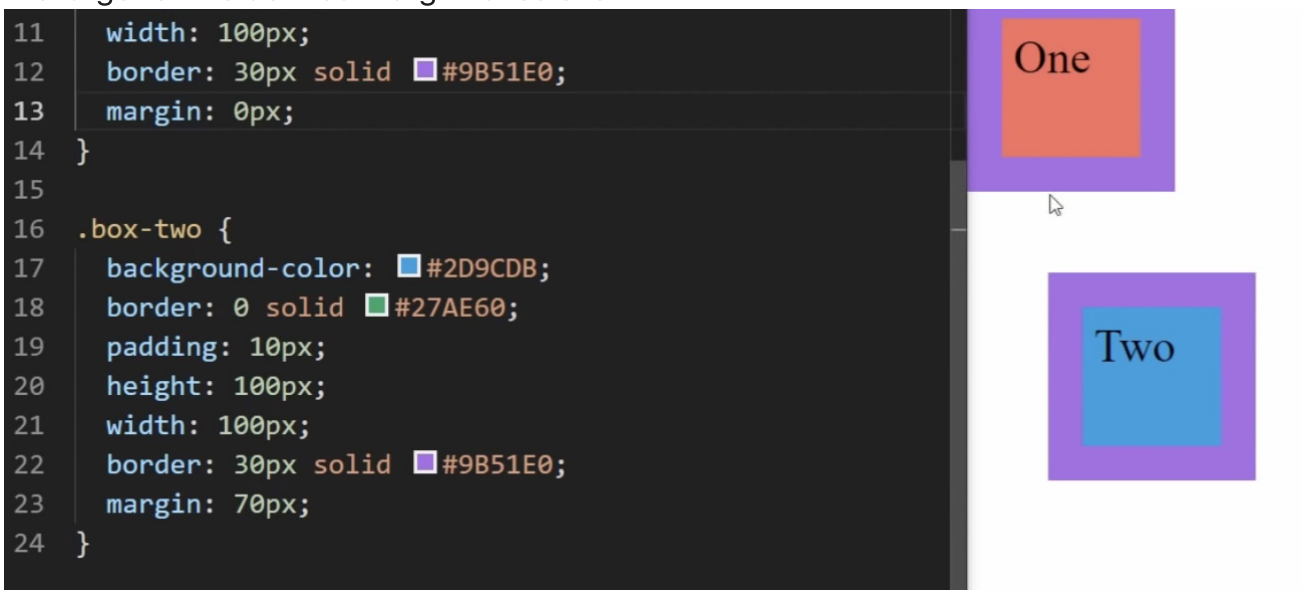
- dvh scales between svh and lvh

display

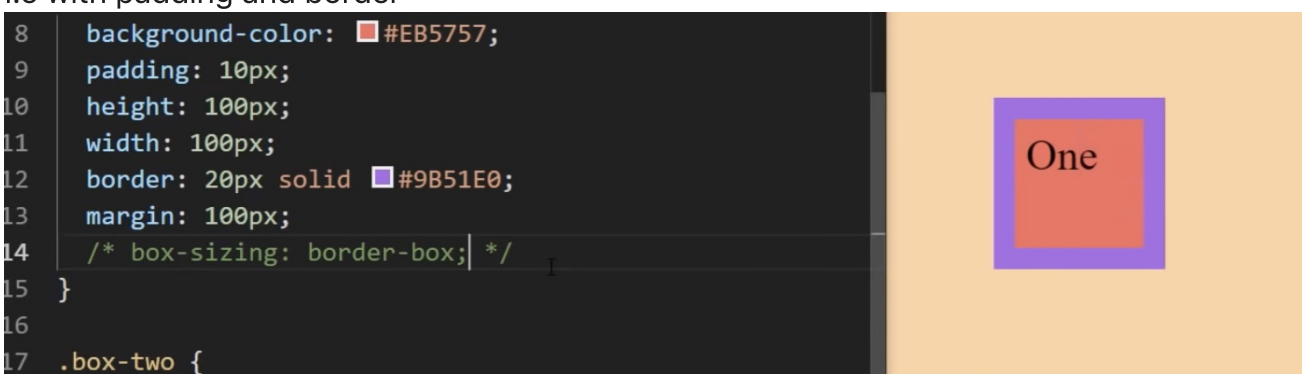
- block have space aboove and below them take up the full width
- inline - take up the least possible space - img, span, a
- inline - cant set width and height
- inline-block - takes up the min space in the same line but can set width and height
- none - acts as if element deleted

box model

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



- total height and width = content + padding + border
- margin doesnt 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



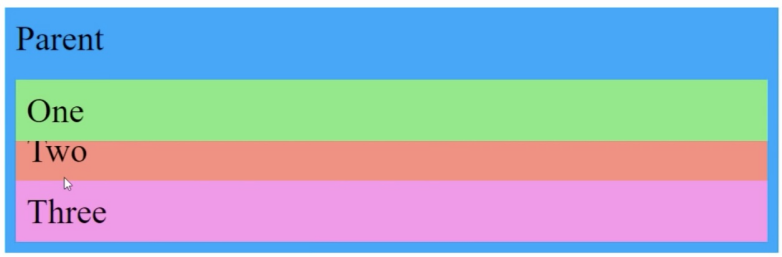
```
background-color: #EB5757;
padding: 10px;
height: 100px;
width: 100px;
border: 20px solid #9B51E0;
margin: 100px;
box-sizing: border-box;
```

- content size depends upon padding and border

position

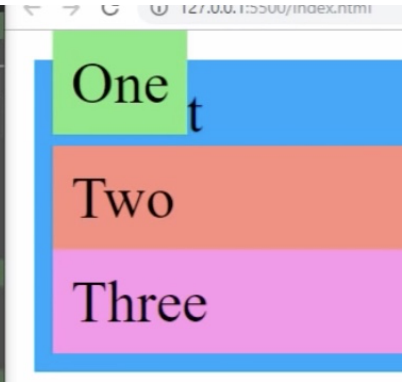
- static - follow the document flow
- static cannot have anything positoned absolute to it
- relative - acts like static but allows defining top left bottom right
- relative elements can overflow the parent
- relative - allows to change positioon relative to its static position (gets removed from the 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 {

}
```

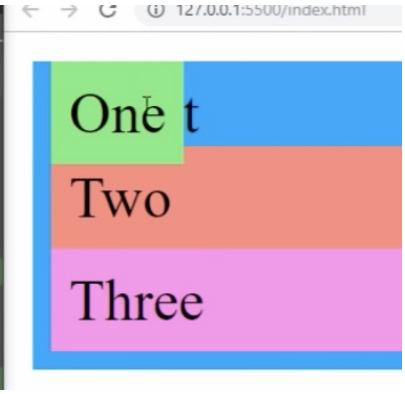


- absolute deletes the element from document flow itself all the other elemets follow normally
- the space(position) the element would take normally is not taken up
- for other elements this one doest exist
- top left right bottom are with respect the page
- in a relative position parent the absolute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

```
.parent {

}

.child-one {
  position: absolute;
  top: 0;
}
```

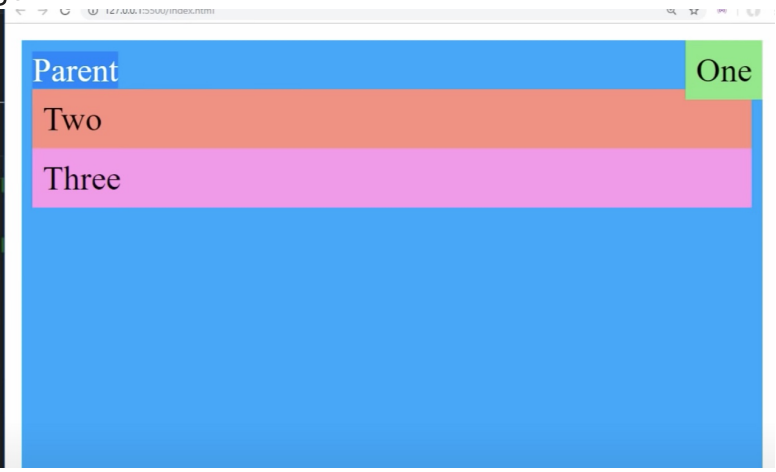


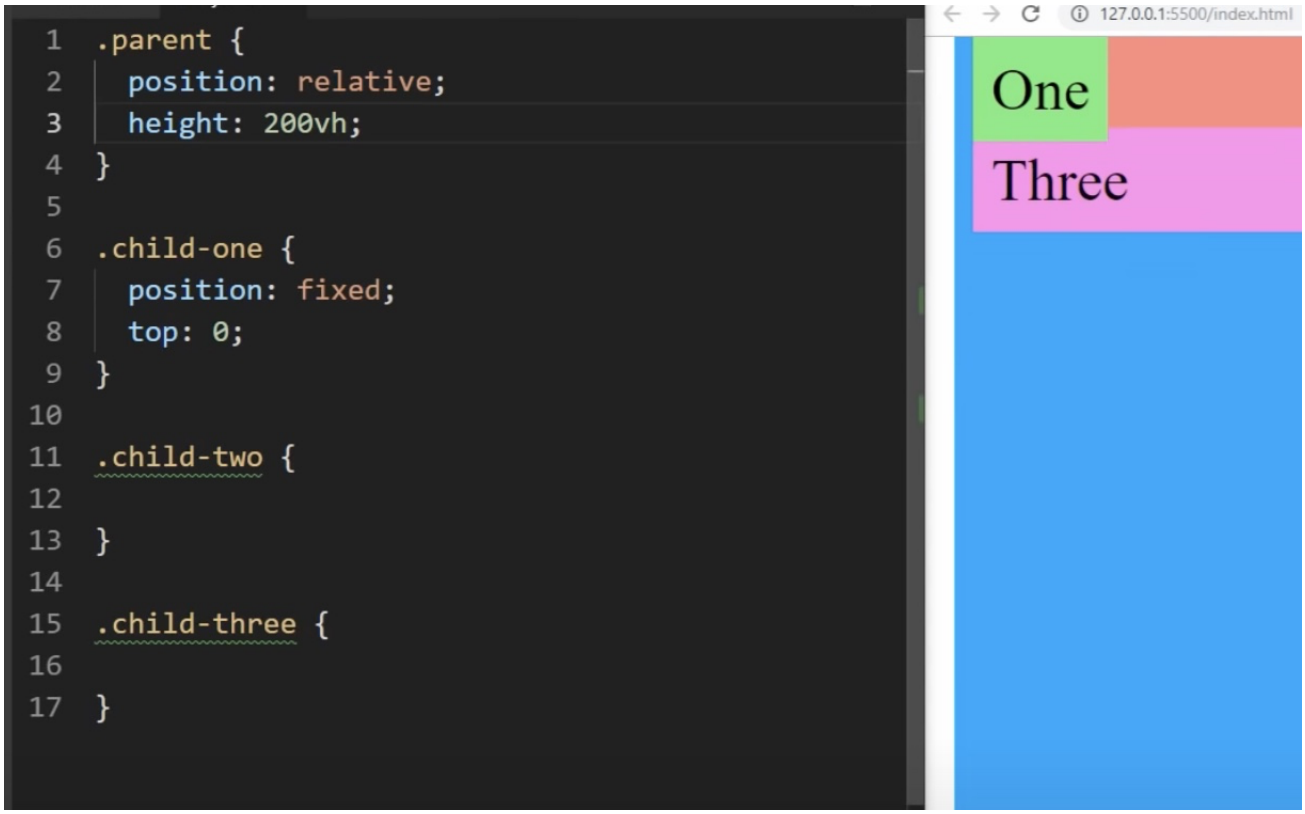

```
.parent {
  position: relative;
}

.child-one {
  position: absolute;
  top: 0;
}
```



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

```
.parent {
  position: relative;
  height: 200vh;
}

.child-one {
  position: absolute;
  top: 0;
  right: 0;
}

.child-two {

}

.child-three {

}
```





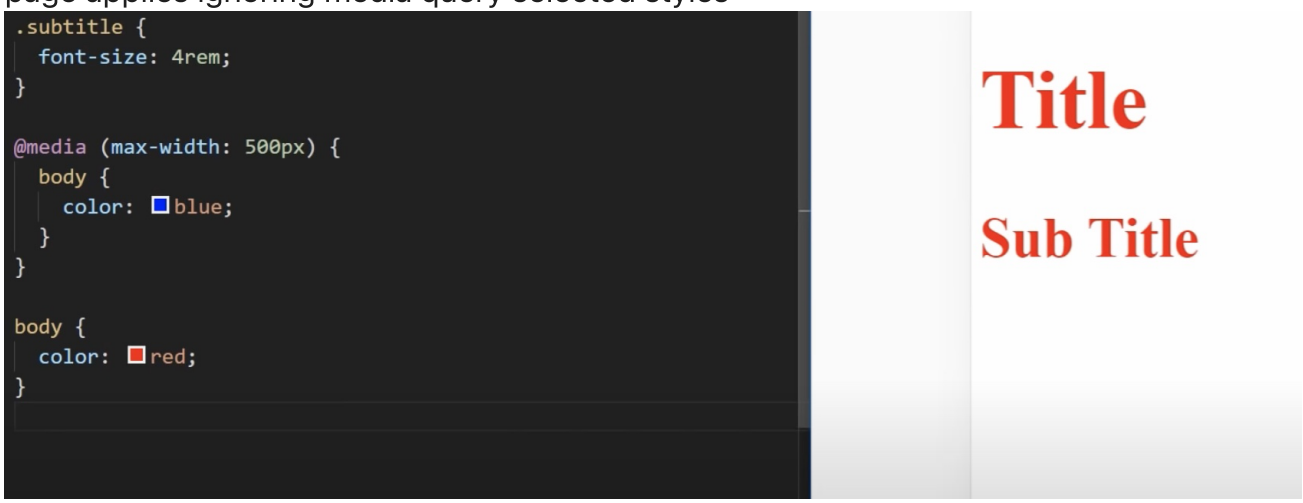
- sticky - relative by default but as soon its outside of the viewport it becomes fixed position

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

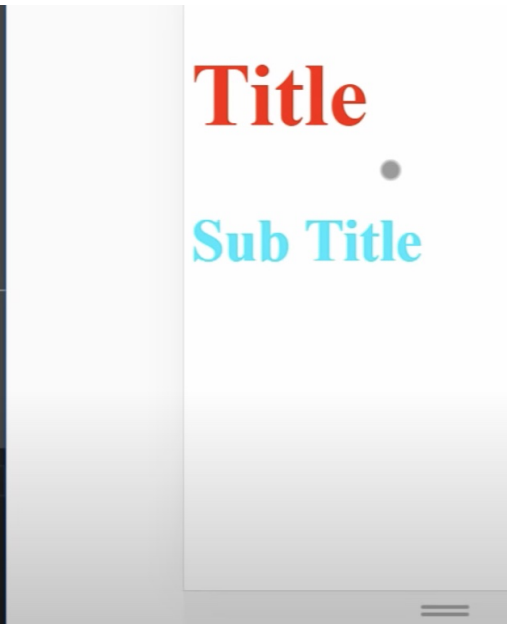


- media queries generally placed at the bottom of the document, style tage
- landscape/portrait

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

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

@media (orientation: landscape) {
  .title {
    color: green;
  }
}

@media (orientation: portrait) {
  .subtitle {
    color: cyan;
  }
}
```


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

@media (orientation: landscape) {
  .title {
    color: green;
  }
}

@media (orientation: portrait) {
  .subtitle {
    color: cyan;
  }
}
```



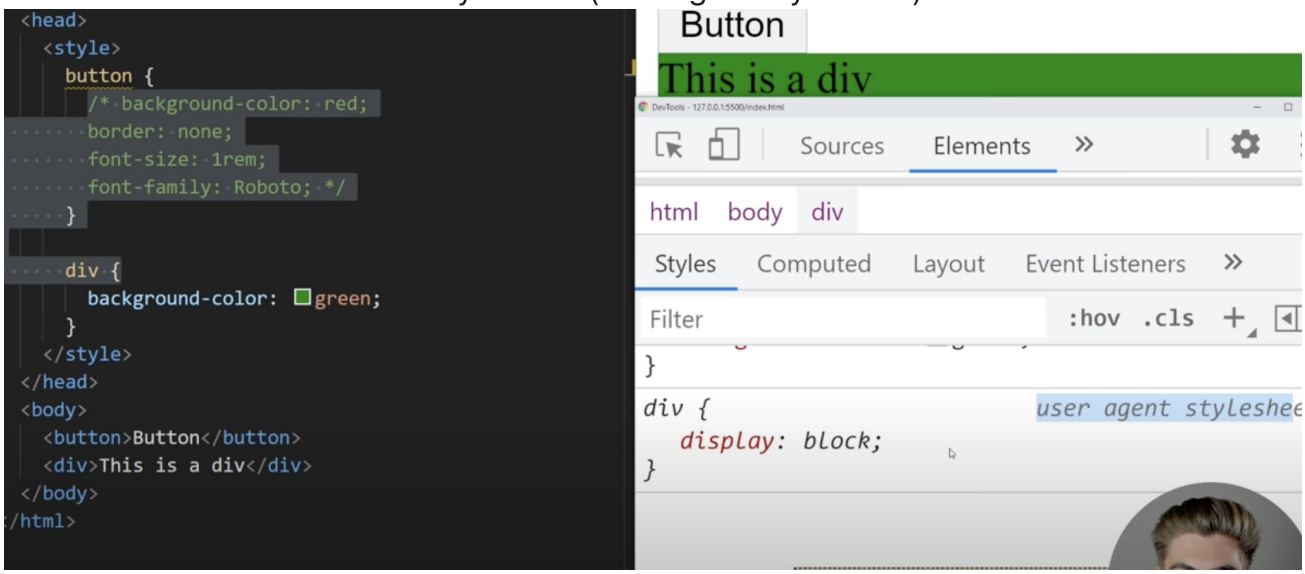
- can combine query selectors

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

@media (orientation: landscape) and (max-width: 500px) {
  .title {
    color: green;
  }
}

@media (orientation: portrait) {
  .subtitle {
    color: cyan;
  }
}
```

- and both true for or we just use comma

ineheritance

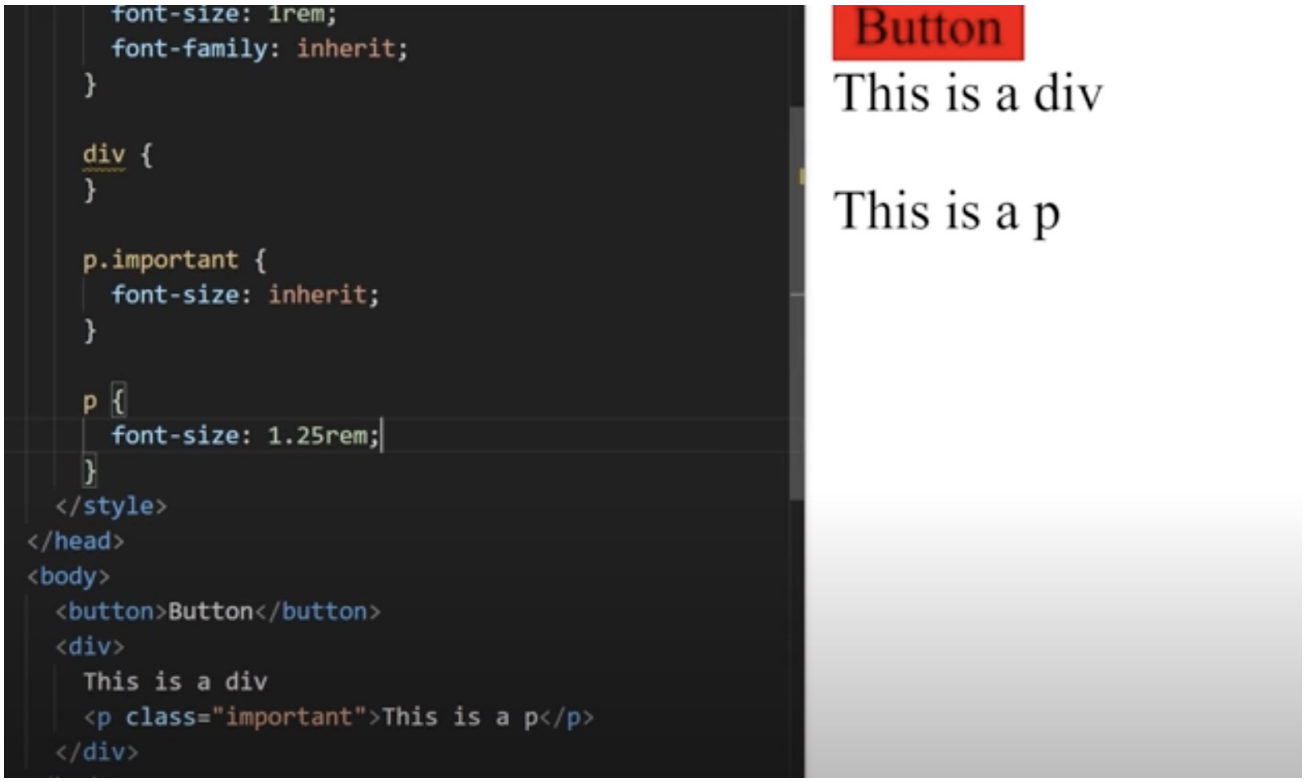
- intial - sets it to the default value according to formal definition in css (mdn docs)
- it also resets the browser set stylesheet (user agent stylesheet)

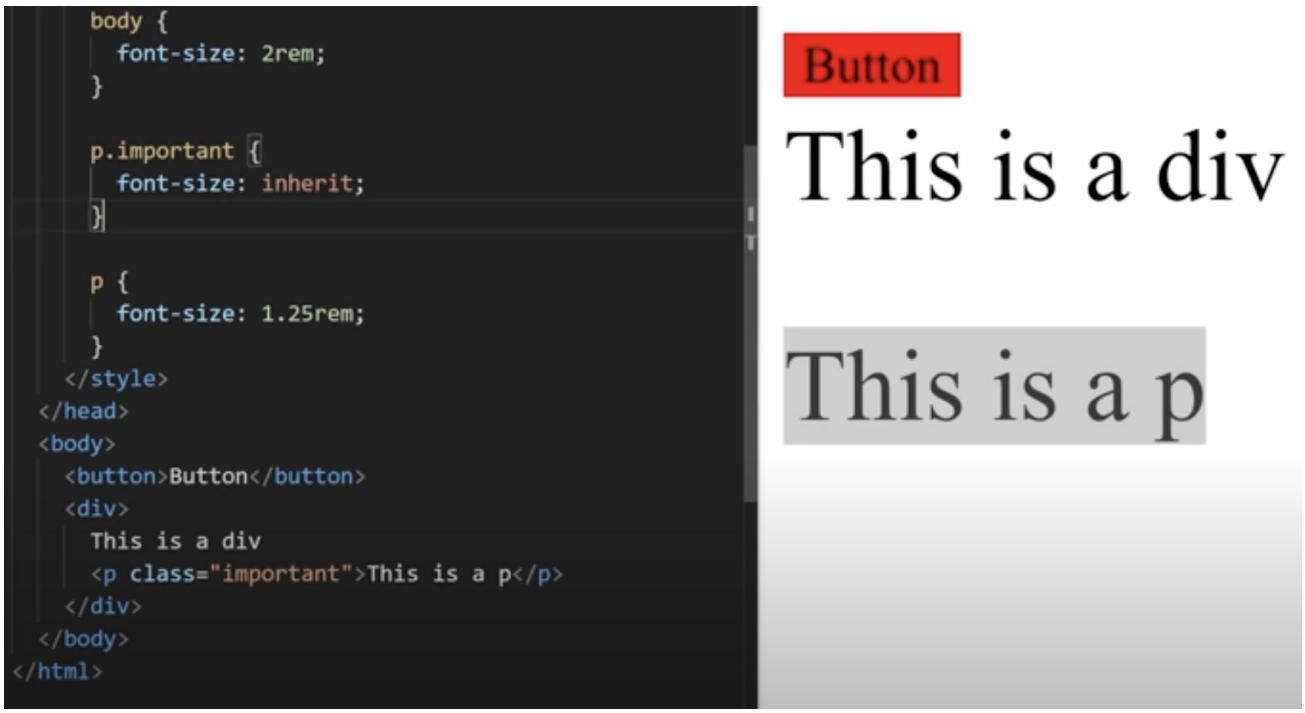


- hierarchy

```
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





- unset - if the property inherits by default it is inherited from the parent if doesnt it is set to initial
- all: unsets - sets all the properties to their inital value if they dont inherit and parent values if it inherits

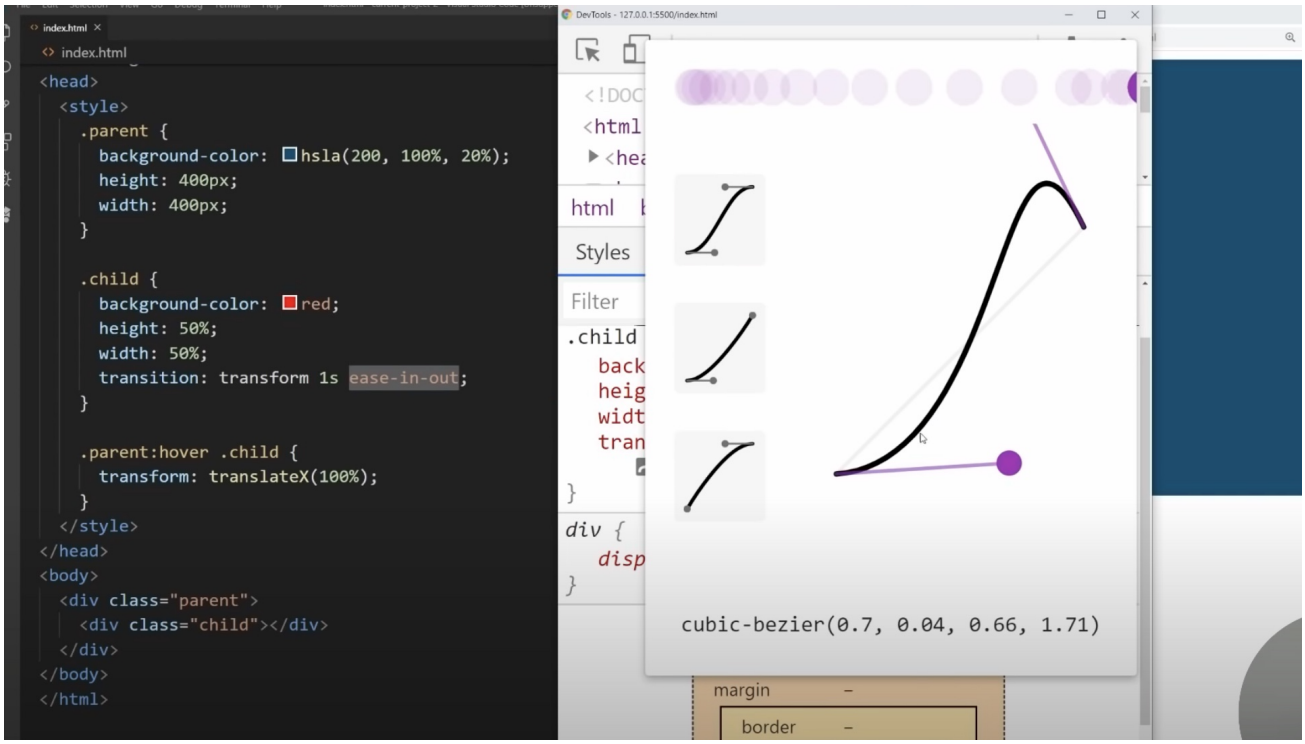
```
<style>
  button {
    all: unset;
    background-color: red;
    border: none;
    font-size: 1rem;
    font-family: inherit;
  }
</style>
```

index.html

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

animations

- trnasform - changes properties
- transisition - defines properties of transform
- trnsition can be all, transform, color etc.
- if we set transition to only tranform only the tranform 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] [dalay] [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% {tranform:[values]}
  2{transform:[values]}
  .
  .
  100%{transform:[values]}
}
```

variables