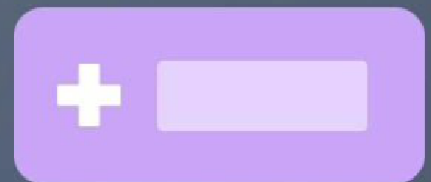
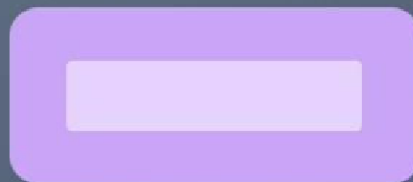




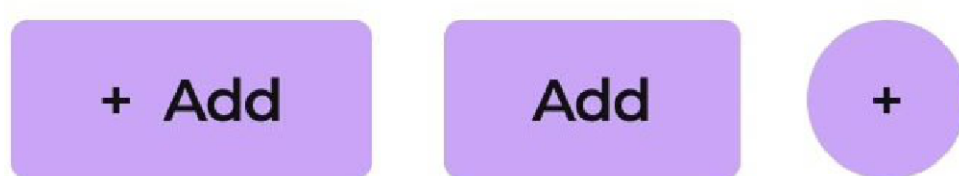
# CSS Features





# Container Queries

Similar to media queries, but allows you to define different styles for an element based on the dimensions of it's parent.



```
.card {  
  container-type: inline-size;  
}  
  
@container (max-width: 10rem) {  
  /* Hide button label if  
   less space is available */  
  .card .btn .label {  
    display: none;  
  }  
}
```



CONTAINER QUERIES

2

## :has()

Style parent element based on the state of children. This was not possible previously in CSS!

`<figure>`

without caption



`<figure>`


with caption



A good boi

////

```
.card {  
  padding: 0;  
  border: none;  
}
```

```
.card:has(figcaption) {  
  padding: 0.25rem;  
  border: solid 2px  #dddddd;  
}
```

////////////////////

:HAS()



# Sub-grids

Allow the grid template defined in an element to be used also in it's children (nested grids)

**.sub-grid**

**This is a very long title**

This is a piece of information present in the first card. This is only a placeholder to demonstrate sub-grids in CSS

**.sub-grid**

**This is short**

This is a piece of information present in the first card. This is only a placeholder to demonstrate sub-grids in CSS



```
.main-grid {  
  grid-template-rows: 1fr 1fr;  
  grid-template-columns: auto 1fr;  
}
```

```
.main-grid .sub-grid {  
  grid-template-columns: subgrid;  
}
```

////////////////////

**SUB GRIDS**




## **:is()**

This pseudo class can be used to select one or more of the provided arguments.

It is really useful when we have multiple repeated selectors.





```
/* Traditional syntax */  
h1 > b, h2 > b, h3 > b {  
  color:  hotpink;  
}
```

```
/* Same implemented using :is() */  
:is(h1, h2, h3) > b {  
  color:  hotpink;  
}
```

////////////////////

:IS()