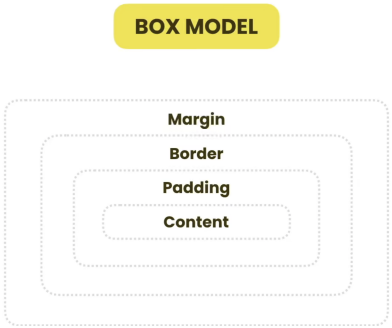
Box Model



padding and border will enlarge the box, while margin not.

we can use box-sizing to fix the size of a box.

box-sizing:content-box;
box-sizing:border-box;

Universal selector

```
*, *::before, *::after{
    box-sizing:border-box;
}
```

to give a inline element height and width

display:inline-block;

overflow:

overflow:visible
overflow:scroll
overflow:auto
overflow:hidden auto

Measurement Units

Measurement Units

1. Absolute: px
2. Relative:
   - % -- relative to the size of the container
   - vw vh -- relative to the viewport
   - relative to the font size

```
.box{
    width:15rem;   //fontsieze of the root element
```

```
    width:10em;
}
```

62.5% of 16px = 10px

```
html{
    font-size:62.5%;/* then 15px equeals 150px*/
}
```

Position

```
.box{
    position:relative;
    position:fixed;
    position:absolute;
    left:4rem;
    bottom:4rem;
    z-index:3
}

.clear{
    clear:left;
    clear:both;
}
```

Whenever we use float, we should clear after, otherweise we'll have layout issues.

3 ways to solve this problem.

1 Add an extra div

```
<body>
    <article tweet>
    <div class="avatar"></div>
    <p>Lorem ipsum dolor sit amet.</p>
    <div class="clear"></div>
    </article>
</body>
```

2 Use pseudo elements, this is a better way.

```
<body>
    <article tweet clearfix>
    <div class="avatar"></div>
    <p>Lorem ipsum dolor sit amet.</p>
    </article>
</body>
```

```
.clearfix::after{
    content: '';
    display:block;
    clear:both;
}
```

3 or a dirty solution(so do not use it, althogth it works sometimes):

```
.tweet{
    overflow:hidden;
}
```

Flexbox:

```
.container{
    boarder:3px solid grey;
    display:flex;
    flex-direction: row;
    justify-content:space-around;
    align-items:flex-end;
    flex-wrap:nowrap;
    align-content:center;
}
```

Axes

- Main(primary)
- Cross(secondary)

Aligning Items

- justify-conten(along the main axis)
- align-items(along the cross axis)
-

```
<div class="container">
    <div class="box box-one">A</div>
    <div>B</div>
    <div>C</div>
</div>
```

```
.box-one{
    /* this can overwrite the align-item property*/
    align-self:flex-start;
}
```

Sizing Items(item properties)

- flex-basis(the initial size of a flex item)
- flex-grow(the growth factor)
- flex-shrink(the shrink factor)
- flex

```
.box{
/* if flex dirction is row, this value will be translated to width.*/
/* if flex dirction is column, this value will be translated to height.*/
    flex-basis: 10rem;
    flex-grow: 1;
    flex-shrink: 1;
    flex:1 1 10rem; /*flex-grow flex-shrink flex-basis*/
}
```

Grid

```
.container{
    display: grid;
    grid-template-rows:100px 100px 100px;
    grid-template-colums:100px 100px;
}
```

or

```
.container{
    display: grid;
    grid-template-rows: repeat(3, 100px);
    grid-template-colums: repeat(2, 100px);
    grid-template: repeat(3, 100px)/repeat(2,100px)
    justify-items:center;
    align-items:center;
    justify-content:center;/* align the whole grid to the center*/
    align-content:end;

}
```

Aligning Items

- justify-items(along the horizontal axis)
- align-items(along the vertical axis)

Grid-template

If we write 100px 30% 70% ,the content will be bigger then the container.

Because the size of 30% and 70% is base on the container.

So we should use fraction.

```
.container{
    display: grid;
    /*grid-template-rows:repeat(3, 100px)/ 30% 70%;*/
    grid-template-rows:repeat(3, 100px)/ 30fr 70fr;
}
```

Gap

- row-gap
- column-gap
- gap

```
.container{
    display: grid;
    row-gap:10px;
    column-gap:10px;
    gap:10px;
}
```

Placing Items(Item Properties)

- grid-row
- grid-column
- grid-area

```
.box-one{
    grid-column: 1/span;
    grid-column: 1/3;
    grid-column: 1/-1; /* this is the same as the last one*/
    gird-area: 1/1/1/3;
}
```

Placing Items in Named Areas

- grid-template-areas
- grid-area

Name the areas.

```
.container{
    display: grid;
    grid-template-area:
        "header header"
        "siderbar main"
        "footer footer";
}
```

Put box-one to the named area header.

```
.box-one{
    grid-column: 1/span 2;
    grid-area: header;
}
```

Hiding Elements

Display none hides the element as if it's never there.

```
.first{
    display: none;
}
```

The element is just invisible.

```
.first{
    visibility: hidden;
}
```

Media Queries

To provide different styles for different devices depending on their features.

```
@media screen and (min-width:600px) and (max-width: 900px)){
    .container{
        flex-direction:row;
    }
}
```