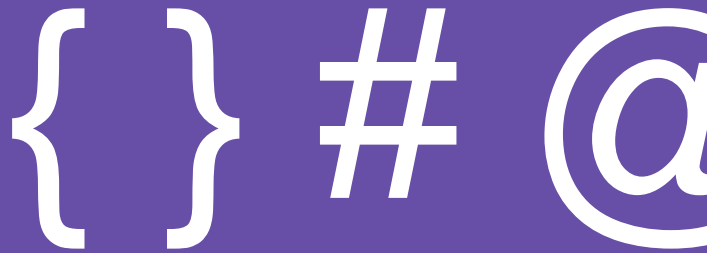
CSS (continued)
Responsive Design



CSS and Fonts

Choosing a font with the font-family property:

```
p{
     font-family: Arial, sans-serif;
}


.entry{

     font-family: “Times New Roman”;
}


p.handwritten{

     font-family: cursive;
}
```

A *font stack* defines the first font to check the user’s system for and then alternatives if not found.

Setting a web font with @font-face:

```
@font-face {
    font-family: myFirstFont;
    src: url(sansation_light.woff);
}

div {
    font-family: myFirstFont;
}
```

@font-face allows you to define a font and link to its file so it can be used with your styles. Fonts may be .ttf, .otf, and . woff

@font-face should be defined at the beginning of your css styles so it is available.

Other services like Google Font API allow you to include their web fonts via url

font-size

```
p{
     font-size: 14px;
}

p.entry{
     font-size: 16pt;
}

h1 {
    font-size: 1.5em;
}

h2 {
    font-size: 90%;
}
```

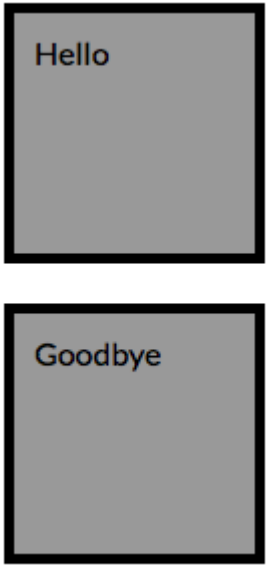
Fixed font sizes are in pixels or point sizes.

Relative font sizes such as em or % use the user's default font size as the baseline. Most browsers have the default set to 16px. So 1em would be 16px and 2em 36px.

CSS Box Model

CSS Box Model

```
p{

  padding: 10px;
  margin: 20px;
  background-color: #999;
  border: 5px solid #000;
  width: 100px;
  height: 100px;

}
```



Each html element has a default display property that defines how it is layed out on the page. Most are block or inline.

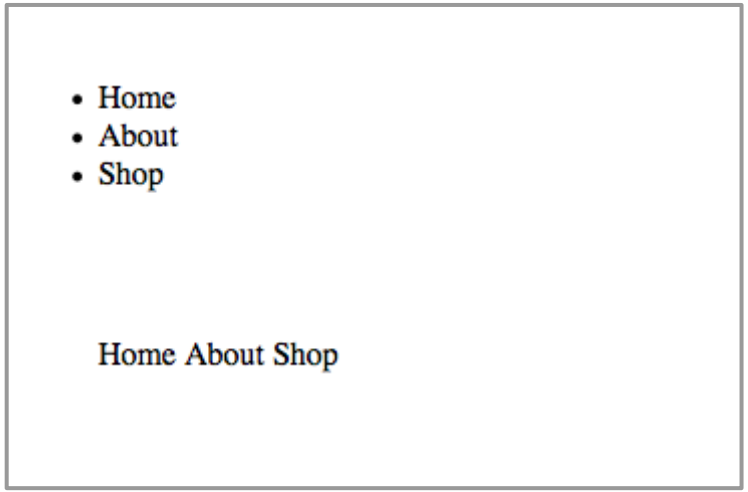
A *block* element always starts on a new line and stretch out to fill as much width as they can. ex: <p> <h1>

An *inline* element does not start a new line and only occupies the space needed. ex: <a> <img>

A display value of *none* can be used to render an element invisible.

Default display values can be overidden.

```
li {
     display: inline;
}
```

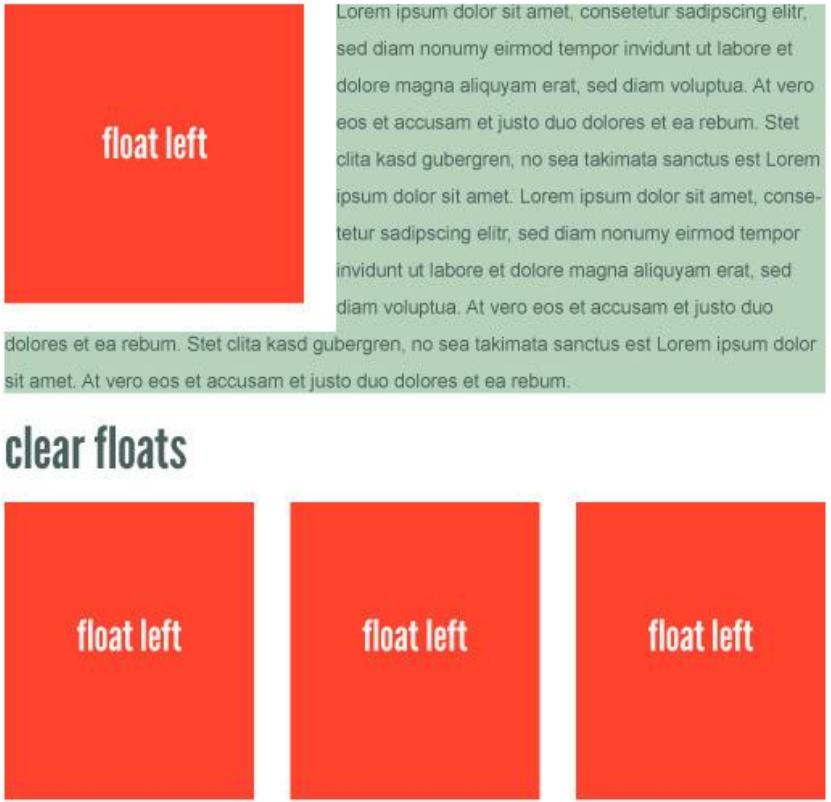


Float and clear

Float can determine how text flows around an image or other block element. It can also be used to make elements float pu adjacent to each other to create adjustable layouts.

Float values can be left, right, or center (or none).

Clear determines whether elements can float on each side. Values can be left, right, both, none. If clear is set to right, no elements can float to the right.



Columns

Column layouts can be designed in a number of ways. Often they are achieved by using a container and percent width div's that float.

CSS3 also introduced a column property.

Finally, CSS3's flexbox can be used to create more complex layouts that were previously difficult such as equal height columns or vertical alignment. Note that only new browsers will support flexbox.

```
.container{
     width: 100%;
}

.col{
     float: left;
     width: 33.33%;
}
```

```
<div class="container">
     <div class="col">
          Column 1
     </div>
     <div class="col">
          Column 2
     </div>
     <div class="col">
          Column 3
     </div>
</div>
```

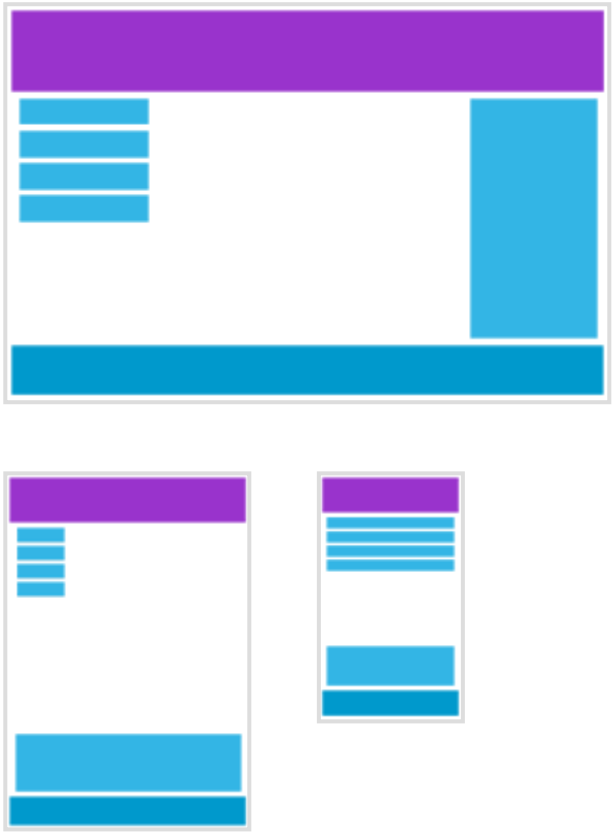
3 Column Layout

Responsive Design

Responsive design for the web refers to making web layouts that can adjust to different devices' screen resolution.

It is accomplished using html and css (no programming or javascript is needed).

Involves adjusting layouts using percentages for widths of elements as well as changing the layout depending on the viewport size.



In order to make responsive designs behave correctly on various devices, the viewport must be set using the meta tag in the head.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This indicates that the layout of the page corresponds to the width of the current device’s viewport so that the device does not attempt to scale or zoom the content.

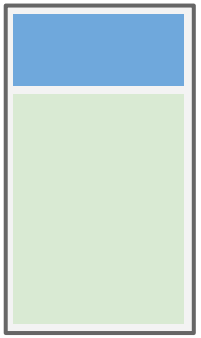
Media queries are rules that allow css to adjust the layout based on certain breakpoints and positions.

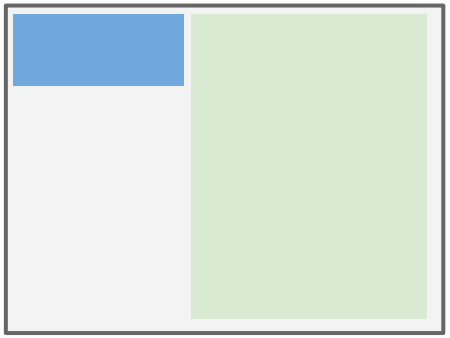
Many designers begin designing for the smallest layout (usually mobile) and use media queries to adjust for increasingly larger screen widths.

This often uses breakpoints with min-width rules. If a screen's width is at least 600px wide the layout expands for example.

```
div.main-content{
    width: 100%;
}

div.navigation{
    width: 100%;
}

@media screen and (min-width: 600px){
    div.navigation{
        width: 30%;
    }

    div.main-content{
        width: 70%;
    }
}
```

```
div.main-content{
     width: 100%;
}

div.navigation{
     width: 100%;
}

@media screen and (min-width: 600px){
     div.navigation{
          width: 30%;
     }

     div.main-content{
          width: 70%;
     }
}
```

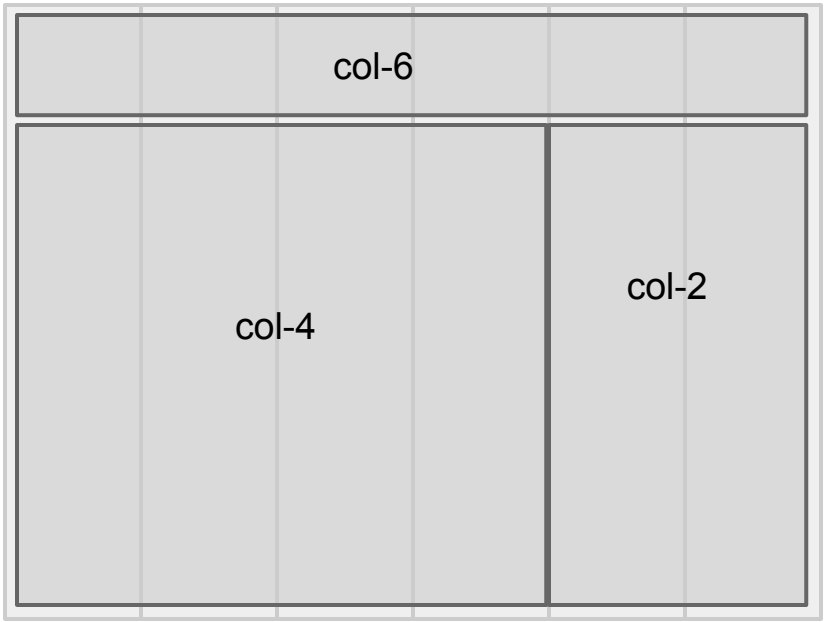




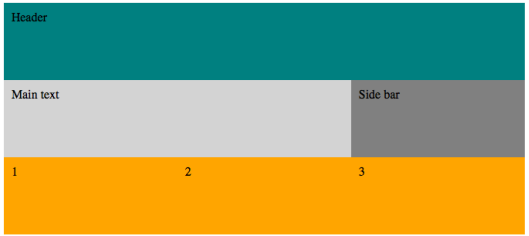
Grids

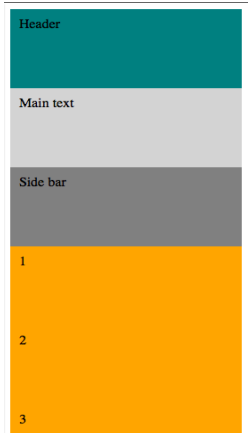
Most responsive layouts rely on a grid usually depending on defining column widths. Media queries can then be used to adjust width sizes based on the device viewport.

This can be a useful method for creating layouts, but sometimes means adding extra container tags (that don’t relate to content structure) and non-semantic classes.



```
*{
  box-sizing: border-box;
}

col-1{width: 100%;}
col-2{width:100%;}
col-3{width:100%;}

[class*='col-']{
    float:left;
}

@media screen and (min-width:600px){
    col-1{ width: 33.33%;}
    col-2{ width: 66.66%;}
    col-3{ width: 100%;}
}
```





Frameworks

There are many responsive design frameworks of varying complexity that can be used in developing sites. Some designers prefer to use them for ease of use, while others believe they are slower and include unnecessary extras for many sites.

Two of the most commonly used frameworks are: Bootstrap and Foundation,

Bootstrap

Foundation

Start here, build everywhere.

CSS Animation

CSS 2D Transforms

Transforms can be used to translate, scale or rotate elements.

```
div    {
       -ms-transform: rotate(7deg); /* IE 9 */
        -webkit-transform: rotate(7deg); /* Chrome, Safari, Opera */
        transform: rotate(7deg);
}
```

CSS Transitions

Transitions allow css element properties to change smoothly over a defined duration of time.

Using only CSS, transitions are triggered when a property for an element changes such as on a mouse hover, active or focus.

For more complex interactions, javascript can be used to add or remove classes or change properties thus triggering the transition.

```
div {
    width: 100px;
    -webkit-transition: width 2s; /* Safar
    transition: width 2s;
}

div:hover {
    width: 300px;
}
```

CSS Transitions

Transitions can have easing:

linear
ease-in
ease-out
ease-in-out
cubic-bezier

```
div {
    width: 100px;
    -webkit-transition: width 2s;
    transition: width 2s;

    -webkit-transition-timing-function: ease-in-out;
    transition-timing-function: ease-in-out;
}

div:hover {
    width: 300px;
}
```

CSS Animation

Animations also animate between style properties like transitions, but do not require a property change to be triggered. Animations can also be looped or repeated and can have keyframes to control or change the animation.