Module Name: CSS3

CSS3

OBJECTIVES

1. To equip students with the basic understanding of CSS3 and its features
2. Students will learn about different advanced selectors & color models introduced with CSS3
3. To make students able to implement different transformations and animations on different elements

CSS3

OUTCOMES

- At the end of this module, students are expected to learn

✓ Features added to CSS3 & its applications

✓ Create a page with CSS3 & add advanced selectors on different HTML elements

✓ Understand & apply transformation, perspective & animations

CSS3

- Document/Video Links

Topics	URL
Learning CSS3	https://www.w3.org/Style/CSS/learning.en.html
Learn to style HTML using CSS	https://developer.mozilla.org/en-US/docs/Learn/CSS
CSS Tutorial – W3Schools	http://www.fadelk.com/files/Resources/153/005CSS.pdf
CSS3 Tutorials	https://www.youtube.com/watch?v=CUxH_rWSI1k https://www.youtube.com/watch?v=I_5or1sh8O0

Content

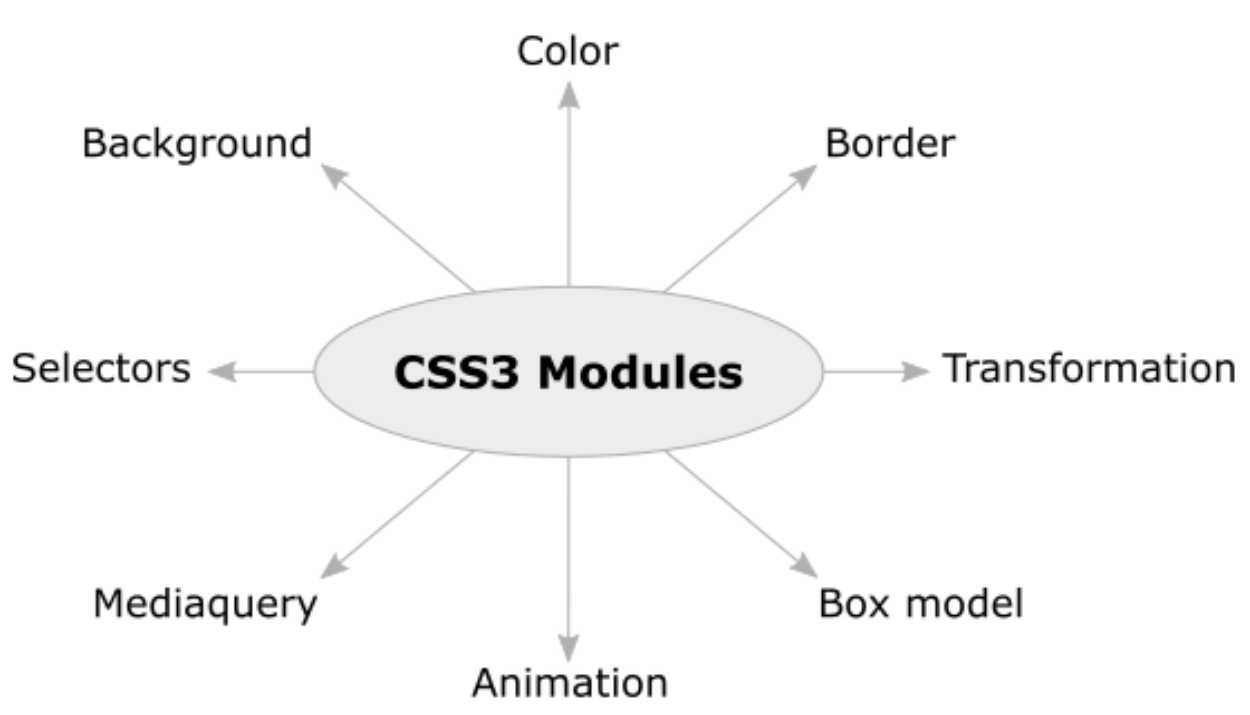
- Introduction
- Advanced Selectors
- Dynamic Selector
- Box Model
- Color Model
- Transformation
- Perspective
- Animation
- Flexbox

Introduction

- CSS3 is the newer version of CSS used for styling the web pages more effectively
- CSS3 is divided into modules which makes styling of web pages more simpler and easier
- It helps in creating dynamic web pages without using JavaScript
- It also helps in creating responsive web pages

Introduction

- Different CSS3 Properties



Introduction

- History

	1996	1998	1999
	CSS 1	CSS 2	CSS 3
Features:	id / class attribute Alignment, Positioning Margin, Padding, Border Foreground / Background color Type property - Typeface / Emphasis Letter-spacing, Word-spacing, Line-spacing	z-index Media types Text shadow Bidirectional text Aural style sheets Absolute, relative, fixed positioning	Selector Animation Namespace Mediaquery Color models Transformation

History of CSS

Advanced Selectors

- Selector is used to target and style an HTML element
- In CSS3 new selectors have been introduced for efficient selection of HTML element
- These selectors has been categorized into the following
    - Relational Selectors
    - Attribute Selectors

Advanced Selectors

- Universal Selector
  - The universal selector, written as * (asterisk) matches every single element on the page
  - It may be omitted if other conditions exist on the target element
  - It is often used to remove the default margins and paddings from the elements for quick testing purpose
  - E.g.

```
{
    margin: 0px;
}
```

Advanced Selectors

- Child Selectors
  - Used to select those elements that are the direct children of some element
  - A child selector is made up of two or more selectors separated by the greater than symbol (i.e. >)
  - These selectors can be used to select the first level of list elements inside a nested list that has more than one level

```
ul > li {
    list-style: square;
}
ul > li ol {
    list-style: none;
}
```

- Home
- About
- Services
  - Design
  - Development
- Contact

Advanced Selectors

- Adjacent Sibling Selectors
  - The adjacent sibling selectors can be used to select sibling elements
  - This selector has the syntax like: `E1 + E2`, where E2 is the target of the selector
  - E.g.
    - Let us consider two sibling selectors <h1> & <p>, where bother <h1> & <p> share the same parent in the document structure
    - Along with that <h1> immediately precedes the <p> element such that only those paragraphs that comes immediately after each <h1> heading will have the associated style rules

Advanced Selectors

- Adjacent Sibling Selectors
  - E.g. contd…

```
h1 + p {
    color: blue;
    font-size: 18px;
}
ul.task + p {
    color: #f0f;
    text-indent: 30px;
}
```

This is a paragraph.

This is another paragraph.

- Task 1
- Task 2
- Task 3

This is one more paragraph.

This is also a paragraph.

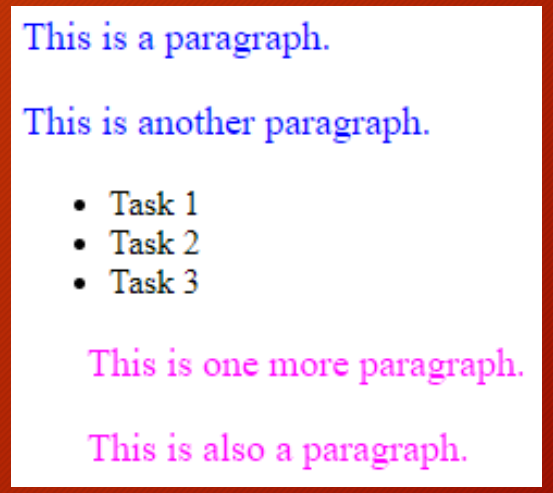
Advanced Selectors

- General Sibling Selectors
  - Similar to the adjacent sibling selector (E1 + E2), but it's less strict
  - A general sibling selector is made up of two simple selectors separated by the tilde (~) character
  - It can be written like: E1 ~ E2, where E2 is the target of the selector
  - E.g.
    - Consider the selector h1 ~ p
    - It will select all the <p> elements that are preceded by the <h1> element, provided that both the elements share the same parent in the document structure

Advanced Selectors

- General Sibling Selectors
  - E.g. contd…

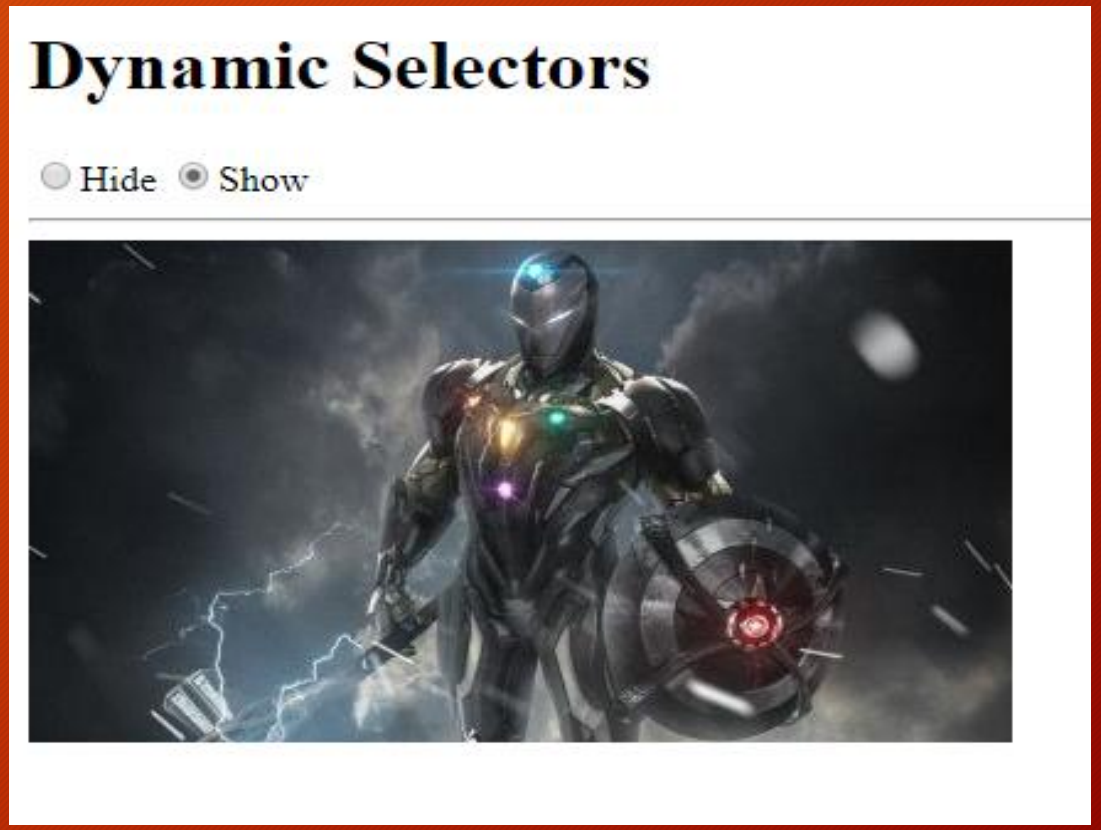
```
h1 ~ p {
    color: blue;
    font-size: 18px;
}
ul.task ~ p {
    color: #f0f;
    text-indent: 30px;
}
```



Dynamic Selector

- Show/Hide based on radio button click

  - Using advanced selectors design a web page that contains two radio buttons namely *show* & *hide* such that as we click on either of them, it should show/hide the image



Dynamic Selector

- Show/Hide based on radio button click
  - Code

Using sibling selector

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        input[value="hide"]:checked ~ .img {
            display: none;
        }
    </style>
</head>
<body>
    <h1>Dynamic Selectors</h1>
    <input type="radio" name="toggle" value="hide">Hide
    <input type="radio" name="toggle" value="show">Show
    <hr>
    <img src="https://wallpapersite.com/images/pages/ico_n/16470.jpg" alt="" class="img">
</body>
</html>
```

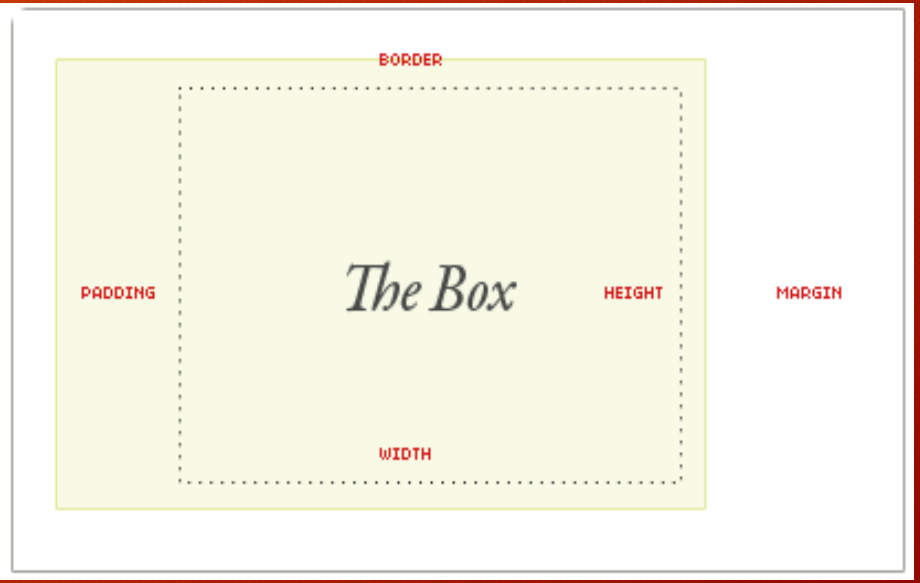
Box Model

- CSS box model is a container which contains multiple properties including borders, margin, padding and the content itself

- It is used to create the design and layout of web pages

- CSS Box model has multiple properties such as

  - Borders
  - Margins
  - Padding
  - Content



Box Model

- Since the beginning of CSS, the box model has worked like this by default
  - Width + padding + border = actual visible width of an element's box
  - Height + padding + border = actual visible height of an element's box
- *With Box Sizing*
  - Width/height + padding + border = actual width or height (not affected by padding or border)

Box Model

Box Sizing

- As the responsive design has become more prominent among the developers, there is need for an update to the *old style* box model
- In CSS box model, the width & height of an element is applied to the element’s content box only
- If the element has any border or padding, this is then added to the width and height to arrive at the size of the box that's rendered on the screen
- The *box-sizing* property can make building CSS layouts easier and a lot more intuitive where an element's specified width and height aren't affected by padding or borders

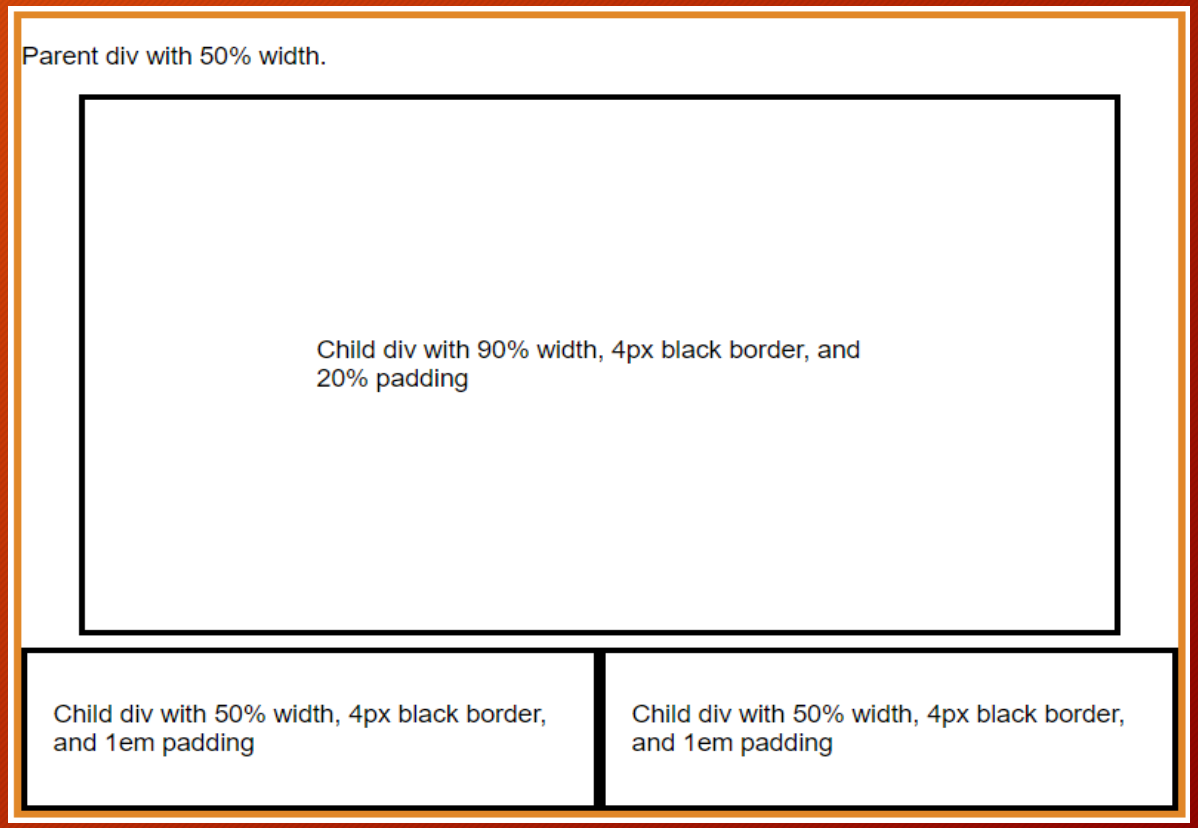
Box Model

Box Sizing

- The *box-sizing* property has the following possible values
- ***Border-box***: sets any border and padding in the values you specify for an element's width and height
  - If you set an element's width to 100 pixels, that 100 pixels will include any border or padding you added, and the content box will shrink to absorb that extra width
  - *Syntax*

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

```
<style>
    * {
    box-sizing: border-box;
    }
    .parent {
        width: 50%;
        border: 5px solid #E18728;
        float: left;
    }

    .child {
        width: 90%;
        padding: 20%;
        border: 4px solid black;
        margin: .5em auto;
    }

    .twins {
        width: 50%;
        padding: 1em;
        border: 4px solid black;
        float: left;
    }

    body {
        font-family: sans-serif;
        font-size: 1.1em;
    }
</style>
```



Color Models

- Color models have been introduced using which we can display color with opacity
- Opacity property is used to change the opacity or transparency level of your HTML elements
- Using opacity property the opacity of the entire element changes
- Syntax

```
div{
        background-color: red;
        opacity: 0.5;
}
```

The default value of opacity is 1

Opacity value ranges between 0 & 1. 0 being fully transparent and 1 being fully opaque

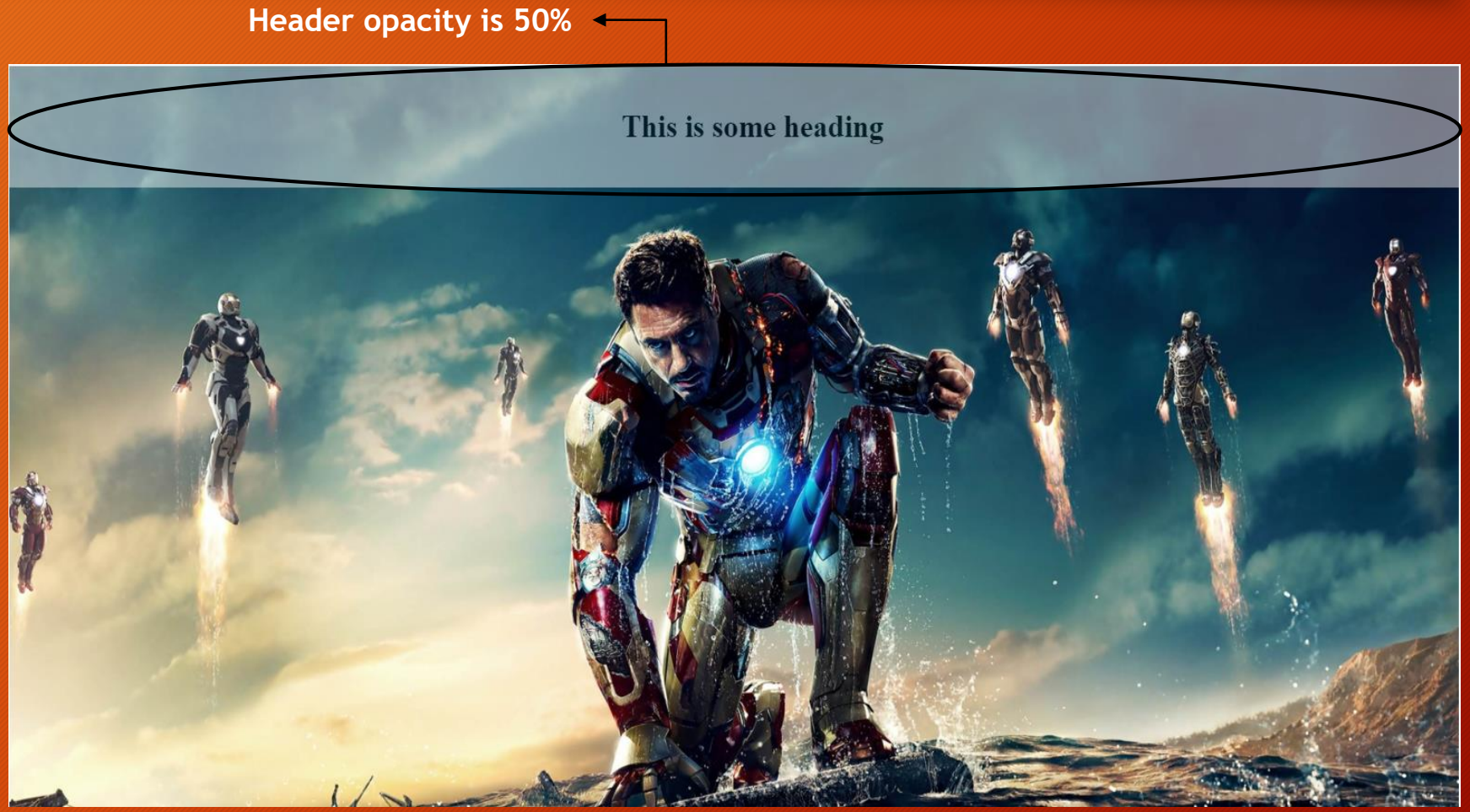
Color Models

- E.g.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        header{

            position: absolute;
            width: 100%;
            text-align: center;
            padding: 20px;
            left: 0px;
            background-color: #eeeeee;
            opacity: 0.5;
        }
        img{
            width: 100%;
            /* height: 700px; */
        }
    </style>
</head>
<body>
    <header>
        <h1>This is some heading</h1>
    </header>
    <div>
        <img src="http://www.avatarys.com/downloadfullsize/send/14715" alt="Image not available">
    </div>
</body>
</html>
```

Color Models

- Output



Color Models

- RGBA Color Values
  - Colors can be defined in the RGBA model (red-green-blue-alpha) using the rgba() functional notation
  - RGBA color model are an extension of RGB color model with an alpha channel — which specifies the opacity of a color
  - The alpha parameter accepts a value from 0.0 (fully transparent) to 1.0 (fully opaque)

Color Models

- E.g.

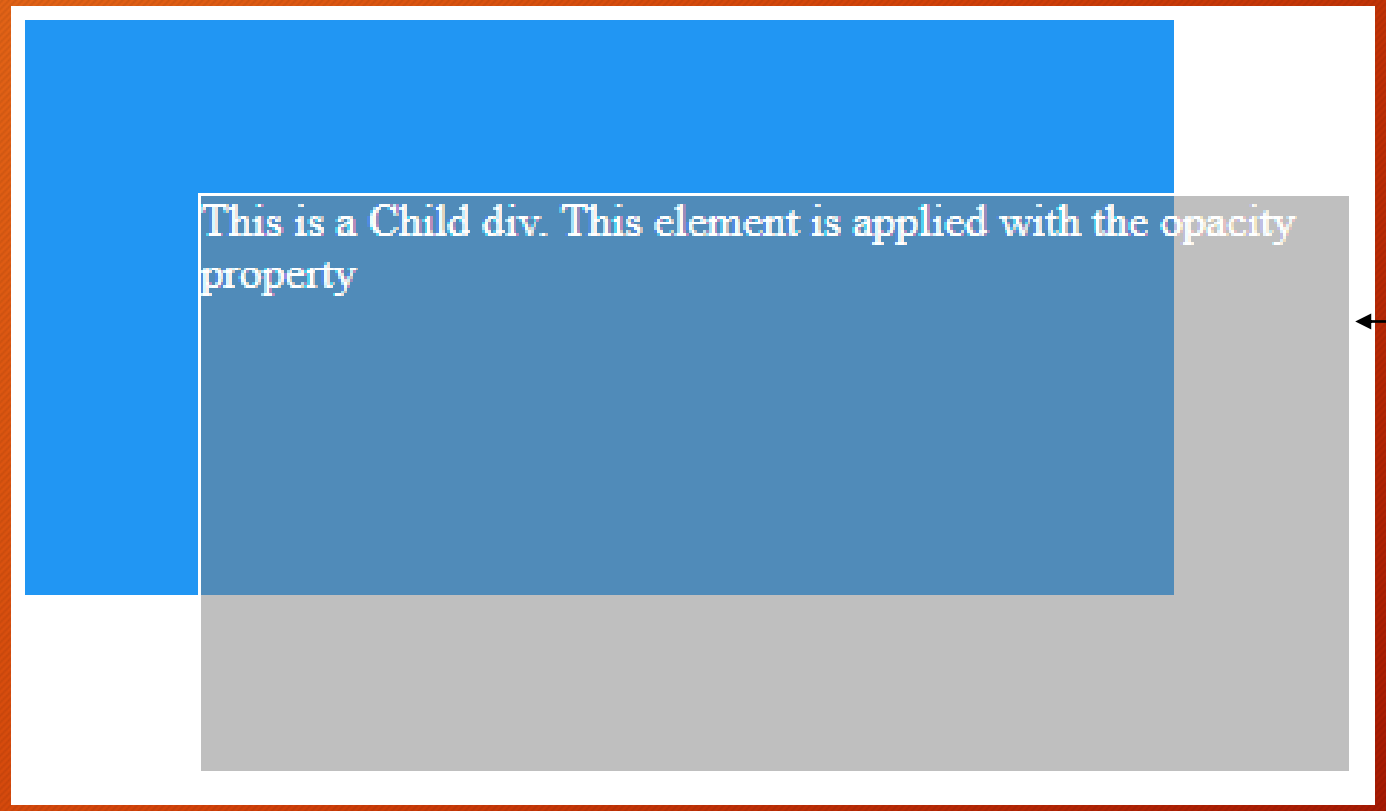
```html
<html>
<head>
        <title>Opacity Property</title>
        <link rel="stylesheet" type="text/css" href="opacity.css">
</head>
<body>
        <div id="parent">
                <div id="child"> This is a child div </div>
        </div>
</body>
</html>
```

```css
#parent
{
        background-color: #2196f3;
        border: 1px solid white;
        height:200px;
        width:200px;
}
#child
{
        background-color: RGBA(128,128,128,0.5);
        border: 1px solid white;
        color: white;
        height:200px;
        margin-left:50px;
        margin-top:50px;
        width:400px;
}
```

Opacity of 0.5 is applied to child element

Color Models

- Output



Here the opacity is applied to the background-color alone, not to the text color

Transformation

- The ***transform*** css property lets us rotate, scale, skew or translate an element
- It modifies the coordinate space of the css formatting model
- We manipulate an elements appearance using transform functions

Transformation

translate()

- The *translate(x,y)* function is similar to relative positioning, translating, or relocating an element by x from the left and y from the top

transform: translate(15px, -15px)

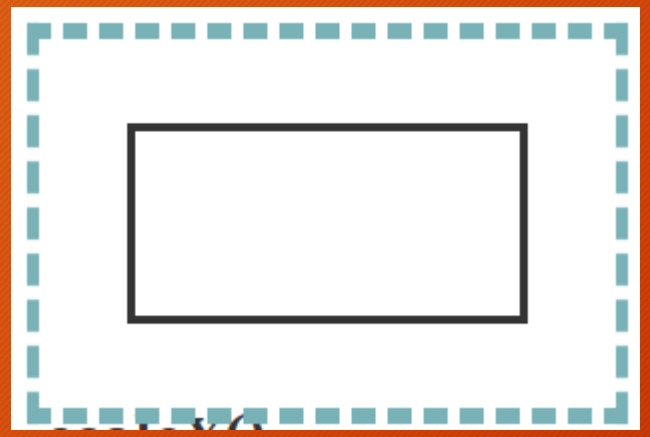
- *translatex() & translatey()* are the subsidiary functions for specifying only left/right values

```css
div {
    height: 300px;
    width: 300px;
    margin: 0 auto;
    border: 5px solid red;
    border-radius: 50%;
    overflow: hidden;
    box-shadow: 0px 0px 20px red;
    transition-duration: 1s;
}

img {
    height: inherit;
    width: 100%;
    transition-duration: 2s;
}

div:hover {
    /* transform: translateX(900px); */
    /* transform: translateY(100px); */
    transform: translate(900px, 100px);
}
```

Transformation

scale()

- The *scale(w,h)* property scales an element by ‘w’ width & ‘h’ height
- In case if only one value is declared, the scaling will be proportional



transform: scale(1.5, 2)

Transformation

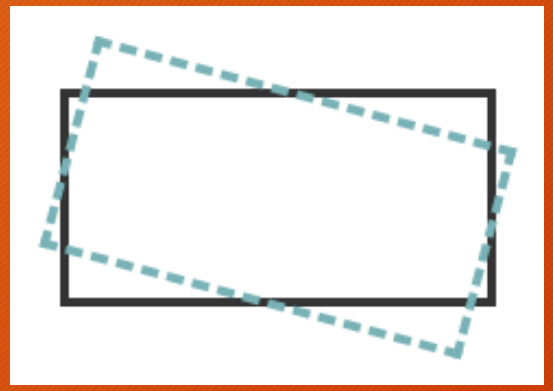
- E.g.

```
div {
    height: 300px;
    width: 300px;
    margin: 0 auto;
    border: 5px solid red;
    border-radius: 50%;
    overflow: hidden;
    box-shadow: 0px 0px 20px red;
    transition-duration: 1s;
}

img {
    height: inherit;
    width: 100%;
    transition-duration: 2s;
}

div:hover {
    transform: scale(1.5, 1.5);
}
```

Transformation

rotate()

- The *rotate(angle)* function will rotate an element about the point of origin but the angle value specified



transform: rotate(15deg)

Transformation

- E.g.

```
div {
    height: 300px;
    width: 300px;
    margin: 0 auto;
    border: 5px solid red;
    border-radius: 50%;
    overflow: hidden;
    box-shadow: 0px 0px 20px red;
    transition-duration: 1s;
}

img {
    height: inherit;
    width: 100%;
    transition-duration: 2s;
}

div:hover {
    transform: rotate(360deg);
}
```

Transformation

skew()

- The *skew(x,y)* function specifies a skew along the X and Y axes
- The x specifies the skew on the x-axis, & y specifies the skew on the y-axis
- If there is only one parameter, then its the same as *skew(x,0)* or *skewX(x)*

transform: skewx(15deg)

Transformation

- E.g.

```
div {
    height: 300px;
    width: 300px;
    margin: 0 auto;
    border: 5px solid red;
    border-radius: 50%;
    overflow: hidden;
    box-shadow: 0px 0px 20px red;
    transition-duration: 1s;
}

img {
    height: inherit;
    width: 100%;
    transition-duration: 2s;
}

div:hover {
    transform: skewX(180deg);
}
```

Perspective

- The perspective CSS property gives an element a 3D-space by affecting the distance between the Z plane and the user
- It defines how far the object is away from the user
- Hence, a lower value will result in a more intensive 3D effect than a higher value
- When defining the perspective property for an element, it is the CHILD elements that get the perspective view, NOT the element itself

Perspective

- E.g.

```
div{

    width: 300px;
    height: 400px;
    margin: 0 auto;
    border: 2px solid red;
    padding: 10px;
    perspective: 300px;
}
img{
    width: 100%;
    height: inherit;
    transition-duration: 1s;
    transform-origin: left;
}
div:hover img{
    transform: rotateY(-180deg);
}
```

Animation

- CSS3 introduces animations that makes it possible to animate transitions from one CSS style configuration to another
- Animations consist of two components, a style describing the CSS animation and a set of '*keyframes*' that indicate the start and end states of the animation's style, as well as possible intermediate waypoints
- To create a CSS animation sequence, you style the element you want to animate with the animation property or its sub-properties

Animation

- E.g.

```
img{
    position: absolute;
    width: 20%;
    bottom: 0px;
    left: 45%;
    animation-name: anim_1;
    animation-duration: 3s;
    animation-iteration-count: infinite;
    /* animation-direction: alternate; */
    animation-timing-function: linear;
}
@keyframes anim_1{
    0%{
        bottom: 0px;
        left: 45%;
    }
    25%{
        bottom:45%;
        left: 0%;
        transform: rotate(360deg);
    }
    50%{
        bottom: 80%;
        left: 45%;

    }
    75%{
        bottom:45%;
        left:90%;
    }
    100%{
        bottom:0px;
        left: 45%;
    }
}
```

Flexbox

- The Flexbox (Flexible Box) layout module aims at providing a more efficient way to lay out, align and distribute space among items in a container (even if their size is unknown or dynamic)

- It gives container the ability to alter its items' width/height (& order) to best fill the available space

- A flex container expands items to fill available free space, or shrinks them to prevent overflow

- Syntax:

```css
.container {
  display: flex; /* or inline-flex */
}
```

Flexbox

- Prior to *flexbox*

```
.left{
    float: left;
}
.right{
    float: right;
}
.row:after{
    content: "";
    clear: both;
    display: block;
}
```

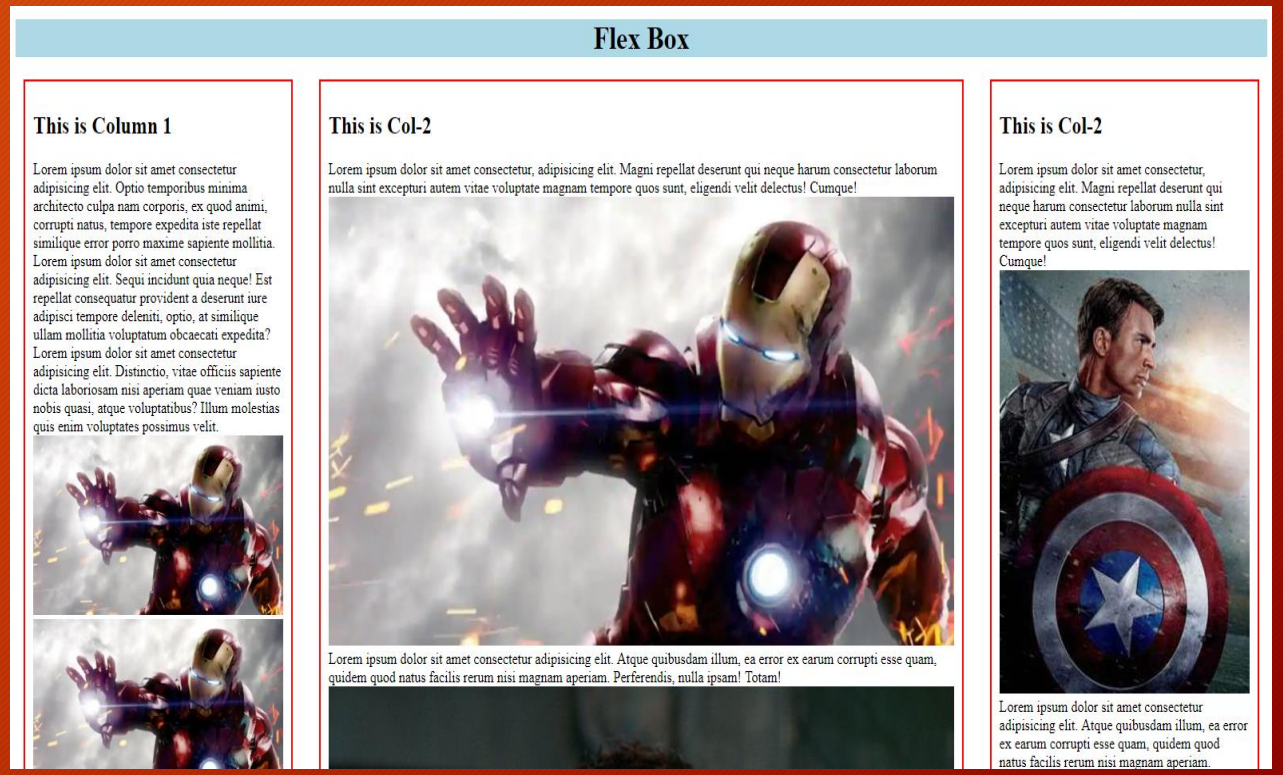
- After using *flexbox*

```
.row {
    display: flex;
    justify-content: space-between;
}
```

Flexbox

- E.g.

```
#parent{
    display: flex;
    justify-content: space-between;
    align-items: flex-start;
}
.col-1, .col-2, .col-3{
    border: 2px solid red;
    padding: 10px;
    margin: 0px 10px;
}
header{
    background-color: lightblue;
    width: 100%;
    text-align: center;
}
p{
    margin: 0px;
}
img{
    width: 100%;
}
.col-1{
    width:20%;
}
.col-2{
    width: 50%;
}
.col-3{
    width: 20%;
}
ul{
    margin: 0px;
    padding: 0px;
    list-style: none;
    display: flex;
    flex-wrap: wrap;
    justify-content: space-around;
}
li{
    border: 2px solid red;
    height: 200px;
    width: 30%;
    margin: 10px 0px;
}
```



Thank You