Introducing CSS3

- Selectors and Pseudo Classes
- Fonts and Text Effects
- Colours, Gradients, Background Images, and Masks
- Borders and Box Effects, Transitions, Transforms, and Animations
- Layout: Columns and Flexible Box, Embedding Media.
- Responsive Web Design: Viewport, Grid View, Images, Flexible Box, Media Queries.

Introducing CSS3

What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

Introducing CSS3

https://slaveryfootprint.org/#where_do_you_live

https://www.art4web.co/

https://www.warchild.org.uk/

https://forms.gle/Ekp1xWDhaXrRB2wT8

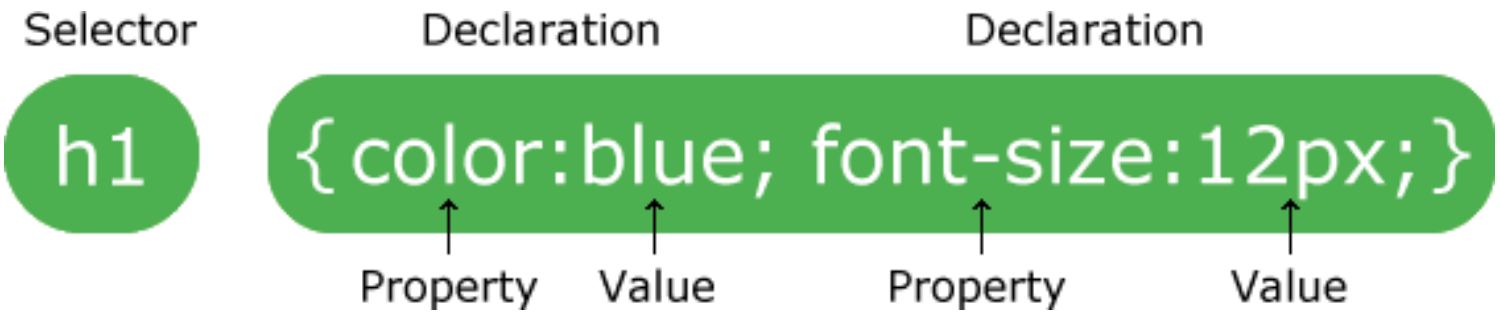
Introducing CSS3

What can we do with CSS that we can't do with HTML?

- Control of backgrounds.
- Set font size to the exact height you want.
- Highlight words, entire paragraphs, headings or even individual letters with background colors.
- Overlap words and make logo-type headers without making images.
- Precise positioning.
- Linked style sheets to control the look of a whole website from one single location.

Introducing CSS3

- CSS Syntax
- A CSS rule-set consists of a **selector and a declaration block:**



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.
- A CSS comment starts with /* and ends with */. Comments can also span multiple lines.

CSS Selectors

- CSS selectors are used to "find" (or select) the HTML elements you want to style.
- We can divide CSS selectors into five categories:
  - Simple selectors (select elements based on name, id, class)
  - Combinatory selectors (select elements based on a specific relationship between them)
  - Pseudo-class selectors (select elements based on a certain state)
  - Pseudo-elements selectors (select and style a part of an element)
  - Attribute selectors (select elements based on an attribute or attribute value)

CSS element Selectors

Selects HTML elements based on the element name.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
        <body>
        <p>Every paragraph will be affected by the style.</p>
        <p id="para1">Me too!</p>
        <p>And me!</p>
        </body>
 </html>
```

CSS ID Selectors

- The id selector **uses the id attribute of an HTML** element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
<!DOCTYPE html>
<html>
<head>
<style>
    #para1 {
      text-align: center;
      color: red;
     }
</style>
</head>
<body>
    <p id="para1">Hello World!</p>
    <p>This paragraph is not affected by the style.</p>
</body>
</html>
```

CSS Class Selectors

- The class selector selects HTML elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the class name.

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
        text-align: center;
        color: red;
    }
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p>Red and center-aligned paragraph.</p>

</body>
</html>
```

Three Ways to Insert CSS

- When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.
- There are three ways of inserting a style sheet:
  - External CSS
  - Internal CSS
  - Inline CSS

- External CSS
- With an external style sheet, you can change the look of an entire website by changing just one file!

- Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

- An external style sheet can be written in any text editor, and must be saved with a .css extension.

- The external .css file should not contain any HTML tags.

Three Ways to Insert CSS

- External CSS

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Three Ways to Insert CSS

- Internal CSS

  - An internal style sheet may be used if one single HTML page has a unique style.

  - The internal style is defined inside the <style> element, inside the head section.

Three Ways to Insert CSS

- Internal CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
    }
      h1 {
        color: maroon;
        margin-left: 40px;
       }
</style>
</head>
<body>
      <h1>This is a heading</h1>
      <p>This is a paragraph.</p>

</body>
</html>
```

Three Ways to Insert CSS

- Inline CSS
  - An inline style may be used to apply a unique style for a single element.
  - To use inline styles, add the style attribute to the relevant element.
  - The style attribute can contain any CSS property.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

CSS Fonts

These are some important font attributes:

CSS Font color: This property is used to change the color of the text. (standalone attribute)

CSS Font family: This property is used to change the face of the font.

CSS Font size: This property is used to increase or decrease the size of the font.

CSS Font style: This property is used to make the font bold, italic or oblique.

CSS Font variant: This property creates a small-caps effect.

CSS Font weight: This property is used to increase or decrease the boldness and lightness of the font.

CSS Text

You can set following text properties of an element −

The **color** property is used to set the color of a text.

The **direction** property is used to set the text direction.

The **letter-spacing** property is used to add or subtract space between the letters that make up a word.

The **word-spacing** property is used to add or subtract space between the words of a sentence.

The **text-indent** property is used to indent the text of a paragraph.

The **text-align** property is used to align the text of a document.

CSS Text

You can set following text properties of an element −

The **text-decoration** property is used to underline, overline, and strikethrough text.

The **text-transform** property is used to capitalize text or convert text to uppercase or lowercase letters.

The **white-space** property is used to control the flow and formatting of text.

The **text-shadow** property is used to set the text shadow around a text.

CSS Colors

- RGBA Colors
- RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

- An RGBA color value is specified with: rgba(red, green, blue, alpha).
- The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

  - rgba(255, 0, 0, 0.2);
  - rgba(255, 0, 0, 0.4);
  - rgba(255, 0, 0, 0.6);
  - rgba(255, 0, 0, 0.8);

CSS Colors

Example:

```
<style>
#p1  {background-color:rgba(255,0,0,0.3);}
#p2  {background-color:rgba(0,255,0,0.3);}
#p3 {background-color:rgba(0,0,255,0.3);}
#p4 {background-color:rgba(192,192,192,0.3);}
#p5 {background-color:rgba(255,255,0,0.3);}
#p6 {background-color:rgba(255,0,255,0.3);}
</style>
```

CSS Colors

- HSL Colors
- HSL stands for Hue, Saturation and Lightness.
- An HSL color value is specified with: hsl(hue, saturation, lightness).
- Hue is a degree on the color wheel (from 0 to 360):
    - 0 (or 360) is red
    - 120 is green
    - 240 is blue
- Saturation is a percentage value: 100% is the full color.
- Lightness is also a percentage; 0% is dark (black) and 100% is white.
    - hsl(0, 100%, 30%);
    - hsl(0, 100%, 50%);
    - hsl(0, 100%, 70%);
    - hsl(0, 100%, 90%);

CSS Colors

Example 1:

```
<style>
#p1 {background-color:hsl(120,100%,50%);}
#p2 {background-color:hsl(120,100%,75%);}
#p3 {background-color:hsl(120,100%,25%);}
#p4 {background-color:hsl(120,60%,70%);}
#p5 {background-color:hsl(290,100%,50%);}
#p6 {background-color:hsl(290,60%,70%);}
</style>
```

CSS Opacity

- The opacity property specifies the opacity/transparency of an element.
- The opacity property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
<style>
    img {
      opacity: 0.5;
    }
    </style>
```

- ```
  img {
    opacity: 0.5;
    filter: alpha(opacity=50);          /* For IE8 and earlier */
  }

  img:hover {
    opacity: 1.0;
    filter: alpha(opacity=100);         /* For IE8 and earlier */
  }
  ```
  ```

CSS Gradients

- CSS gradients let you display smooth transitions between two or more specified colors.
- CSS defines two types of gradients:

1. Linear Gradients (goes down/up/left/right/diagonally)

- Syntax:

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
```

Example:

```
<style>
#grad1 {
  height: 200px;
  background-color: red; /* For browsers that do not support gradients */
  background-image: linear-gradient(red, yellow); /* Standard syntax (must be last) */
}
</style>
<div id="grad1"></div>
```

CSS Gradients

2. Radial Gradients (defined by their center)

- Syntax:

  background-image: radial-gradient(shape size at position, start-color, ..., last-color);

Example:

```
<style>
#grad1 {
  height: 150px;
  width: 200px;
  background-color: red; /* For browsers that do not support gradients */
  background-image: radial-gradient(red, yellow, green); /* Standard syntax (must be last) */
}
</style>
```

CSS Background

- CSS Multiple Backgrounds

- CSS allows you to add multiple background images for an element, through the background-image property.

- The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.

- Multiple background images can be specified using either the individual background properties (as above) or the background shorthand property.

CSS Background

You can set the following background properties of an element −

The **background-color** property is used to set the background color of an element.

The **background-image** property is used to set the background image of an element.

The **background-repeat** property is used to control the repetition of an image in the background.

The **background-position** property is used to control the position of an image in the background.

The **background-attachment** property is used to control the scrolling of an image in the background.

The **background** property is used as a shorthand to specify a number of other background properties.

CSS Background

- CSS Background Size

- The CSS background-size property allows you to specify the size of background images.

- The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.

CSS Background

- <style>
- #example1 {
  border: 1px solid black;
  background:url(img_flwr.gif);
  background-size: 100px 80px;
  background-repeat: no-repeat;
  padding:15px;
  }

- #example2 {
  border: 1px solid black;
  background:url(img_flwr.gif);
  background-repeat: no-repeat;
  padding:15px;
  }
  </style>

CSS Borders

- The CSS border properties allow you to specify the style, width, and color of an element's border.

- The border-style property specifies what kind of border to display.

- The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

CSS Borders

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

CSS Borders

- CSS Border Width
- The border-width property specifies the width of the four borders.
- The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

```
p.one {
    border-style: solid;
    border-width: 5px;
     }

  p.two {
    border-style: solid;
    border-width: medium;
  }

  p.three {
    border-style: solid;
    border-width: 2px 10px 4px 20px;
  }
```

CSS Borders

- CSS Border Color
- The border-color property is used to set the color of the four borders.
- The color can be set by:
  - name - specify a color name, like "red"
  - Hex - specify a hex value, like "#ff0000"
  - RGB - specify a RGB value, like "rgb(255,0,0)"
  - transparent

- p.one {
  border-style: solid;
  border-color: red;
  }

  p.three {
  border-style: solid;
  border-color: red green blue yellow;
  }

Pseudo Class

- A pseudo-class is used to define a special state of an element.

- For example, it can be used to:
  - Style an element when a user mouse over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus

- The syntax of pseudo-classes:

```
selector:pseudo-class
{
    property:value;
}
```

Pseudo Class

Sr.No.	Value & Description
1	**:link** Use this class to add special style to an unvisited link.
2	**:visited** Use this class to add special style to a visited link.
3	**:hover** Use this class to add special style to an element when you mouse over it.
4	**:active** Use this class to add special style to an active element.
5	**:focus** Use this class to add special style to an element while the element has focus.

Pseudo Class

- a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.

- a:active MUST come after a:hover in the CSS definition in order to be effective.

- Pseudo-class names are not case-sensitive.

- Pseudo-class are different from CSS classes but they can be combined.

Pseudo Class

1. Set the background color for visited and unvisited links to "lightblue", and the background color for the hover and active link states to "yellow".

2. Change the background color, when a user hovers over p elements, with the class "highlight", to "lightblue".

3. Set the background color of <input> elements that are in focus (clicked or active), to "lightblue".

Pseudo Class

```
1. <!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
  background-color: lightblue;
}
/* visited link */
a:visited {
  background-color: lightblue;
}
/* mouse over link */
a:hover {
  background-color: yellow;
}
```

Pseudo Class

```
/* selected link */
a:active {
  background-color: yellow;
}
</style>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
<p><a href="https://www.w3schools.com">W3Schools.com</a></p>

</body>
</html>
```

Pseudo Class

2. <!DOCTYPE html>

```
<html>
<head>
    <style>
    p.highlight:hover {
      background-color: lightblue;
    }
    </style>
</head>
<body>
<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
<p class="highlight">This is another paragraph.</p>
</body>
</html>
```

Pseudo Class

```
3. <!DOCTYPE html>
<html>
<head>
<style>
input:focus {
  background-color: lightblue;
}
</style>
</head>
<body>
        <form action="/action_page.php" method="get">
          First name: <input type="text" name="fname"><br>
          Last name: <input type="text" name="lname"><br>
          <input type="submit" value="Submit">
        </form>
</body>
</html>
```

Pseudo Class

Simple Tooltip Hover:

Hover over a element to show a <p> element (like a tooltip):

```
<!DOCTYPE html>
<html>
<head>
<style>
        p {
          display: none;
          background-color: yellow;
          padding: 20px;
        }
div:hover p {
  display: block;
}
</style>
</head>
```

Pseudo Class

```
<body>

<div>Hover over me to show the p element
  <p>Hey! Here I am!</p>
</div>

</body>
</html>
```

What are Pseudo-Elements?

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

-Syntax:

```
selector::pseudo-element
 {
        property:value;
   }
```

What are Pseudo-Elements?

All CSS Pseudo Elements

Selector	Example	Example description
::after	p::after	Insert content after every <p> element
::before	p::before	Insert content before every <p> element
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

What are Pseudo-Elements?

The ::first-line Pseudo-element

The ::first-line pseudo-element is used to add a special style to the first line of a text.

The ::first-line pseudo-element can only be applied to block-level elements.

The following properties apply to the ::first-line pseudo-element:

- font properties
- color properties
- background properties
- vertical-align

What are Pseudo-Elements?

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
</style>
</head>
<body>

<p>You can use the ::first-line pseudo-element to add a special effect to the first line of a text. Some more text. And even more, and more, and more, and more, and more, and more, and more, and more, and more, and more, and more, and more.</p>

</body>
</html>
```

What are Pseudo-Elements?

CSS ::after Selector

- The ::after selector inserts something after the content of each selected element(s).
- Use the content property to specify the content to insert.
- Use the ::before selector to insert something before the content.

```
::after {
  css declarations;
}
```

Example:

```
<style>
p::after {
  content: " - Remember this";
  background-color: yellow;
  color: red;
  font-weight: bold;
}
</style>
```

What are Pseudo-Elements?

CSS ::before Selector

- The ::before selector inserts something after the content of each selected element(s).
- Use the content property to specify the content to insert.

```
::before {
  css declarations;
}
```

Example

```
<style>
p::before {
  content: "Read this -";
  background-color: yellow;
  color: red;
  font-weight: bold;
}
</style> :
```

What are Pseudo-Elements?

CSS ::selection Selector

- The ::selection selector matches the portion of an element that is selected by a user.

- Only a few CSS properties can be applied to the ::selection selector: color, background, cursor, and outline.

```
::selection {
  css declarations;
}
```

What are Pseudo-Elements?

Example:

```
<style>
::-ie-selection { /* Code for Firefox */
  color: red;
  background: yellow;
}

::selection {
  color: red;
  background: yellow;
}
</style>
```

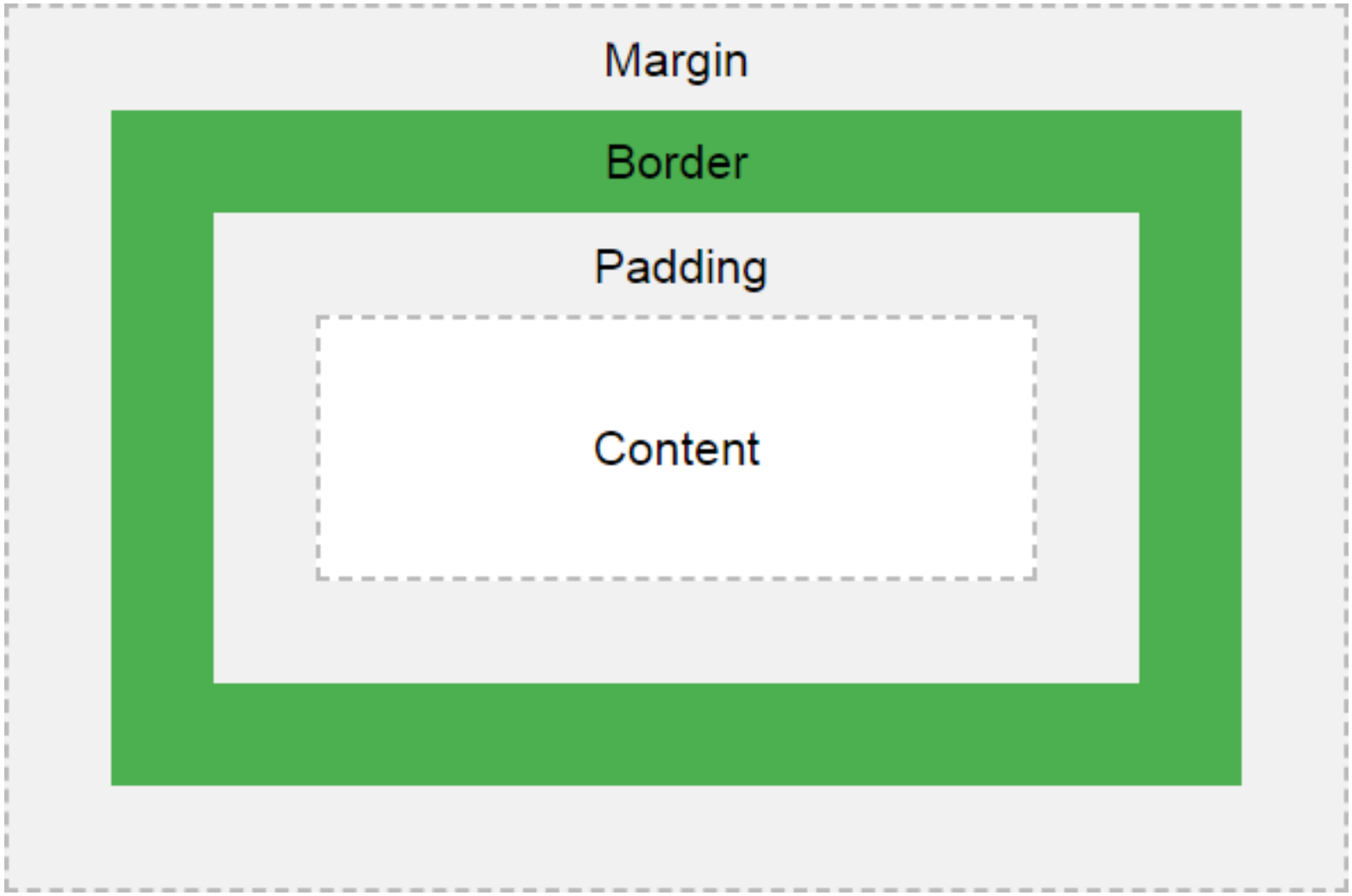
The CSS Box Model

- All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element.
- It consists of: margins, borders, padding, and the actual content.

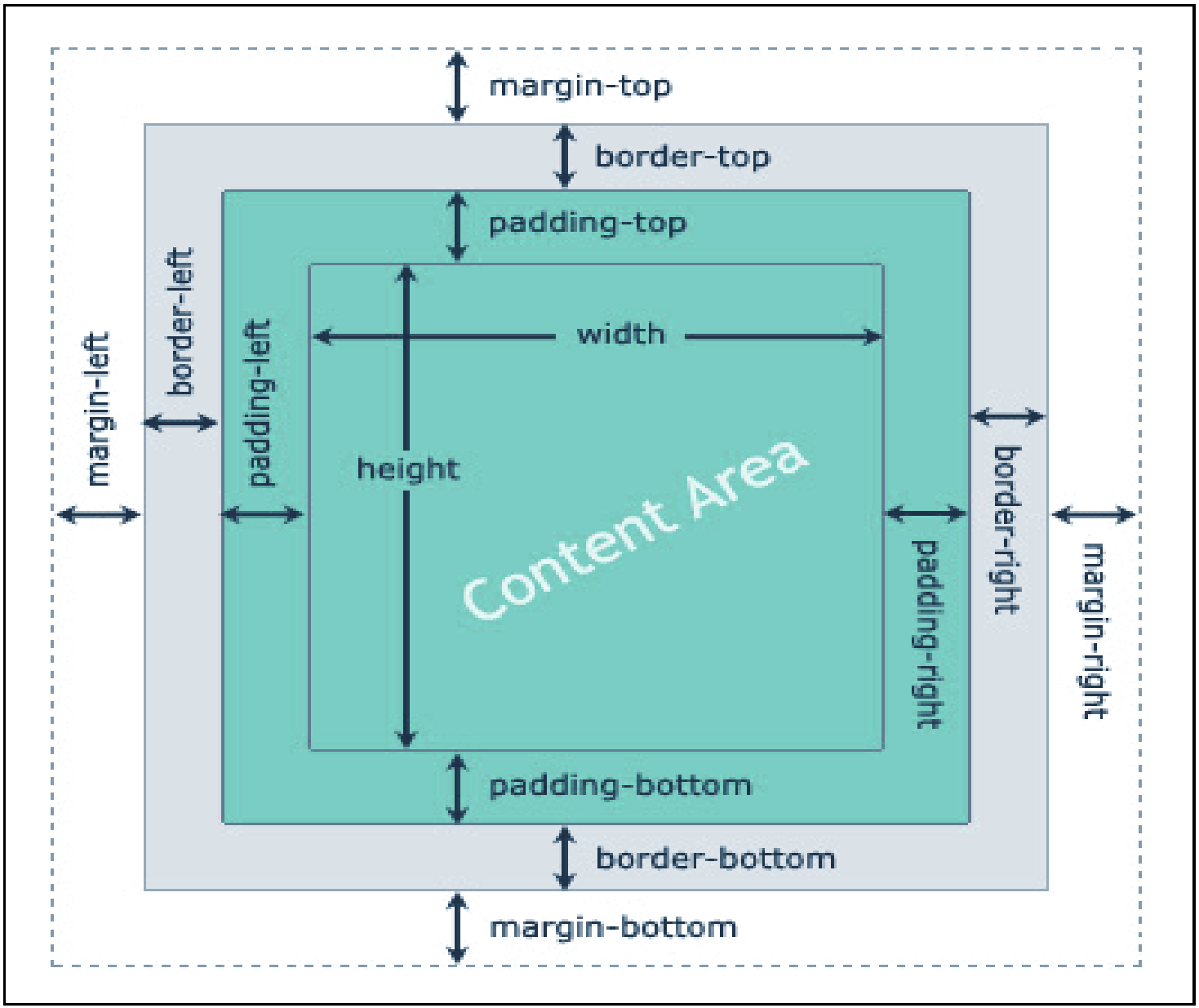
The different parts:

- Content - The content of the box, where text and images appear
- Padding - Clears an area around the content. The padding is transparent
- Border - A border that goes around the padding and content
- Margin - Clears an area outside the border. The margin is transparent

The CSS Box Model



The CSS Box Model



The CSS Box Model

•The total width of an element should be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

•The total height of an element should be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Example:

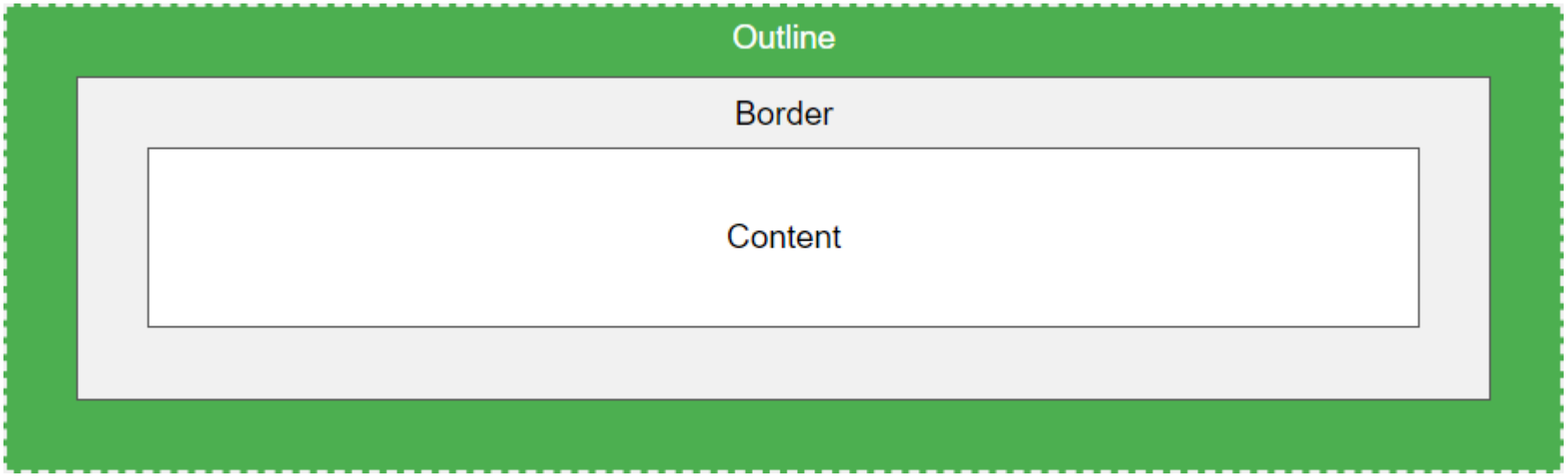
```
div {
    width: 300px;
    border: 15px solid green;
    padding: 50px;
    margin: 20px;
  }
```

The CSS Outline

•An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".

• CSS has the following outline properties:

- outline-style
- outline-color
- outline-width



The CSS Outline

Example:

```
p.ex1
{
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: thin;
}
```

•CSS has the following outline properties:

- outline-style

-The outline-style property specifies the style of the outline, and can have one of the following values:

- dotted - Defines a dotted outline
- dashed - Defines a dashed outline
- solid - Defines a solid outline
- double - Defines a double outline

The CSS Outline

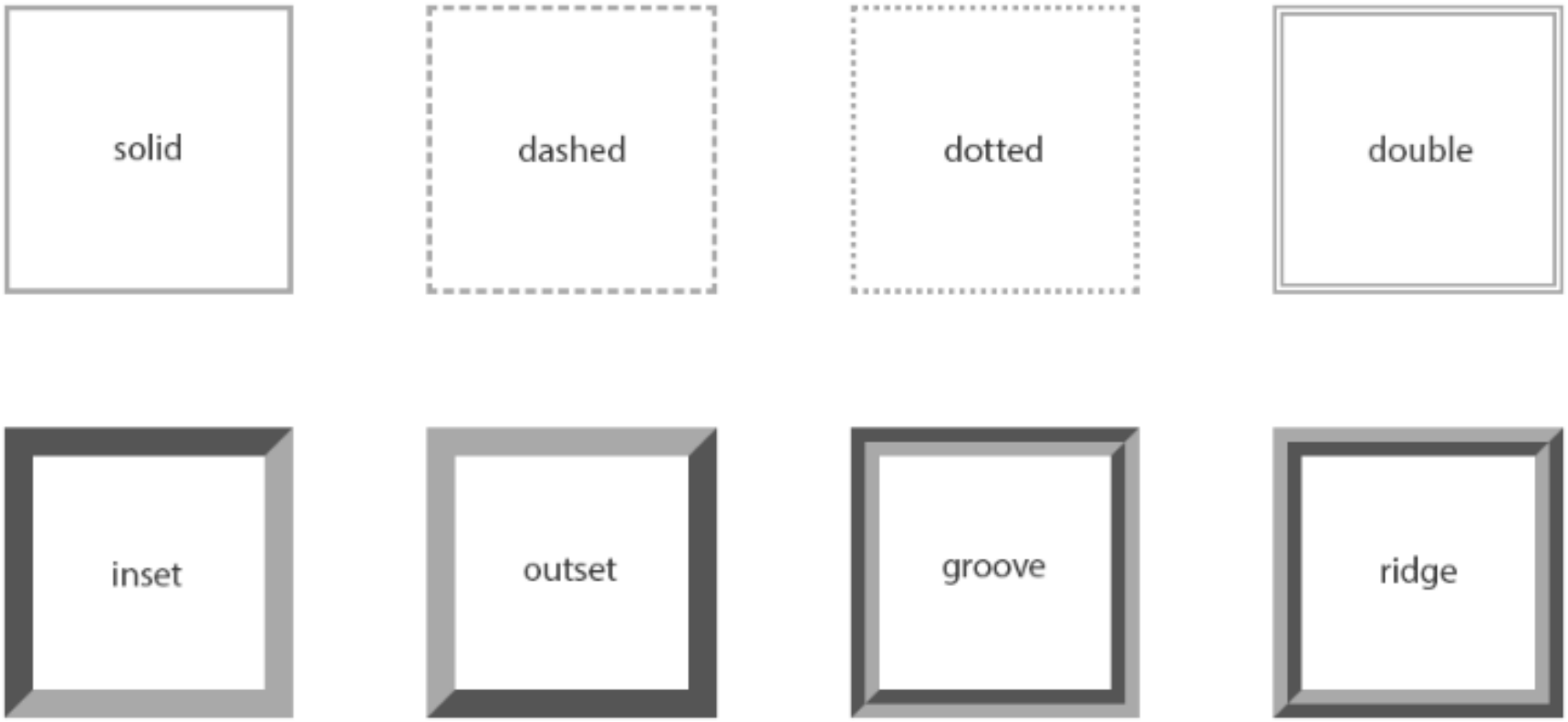
- CSS has the following outline properties:
  - outline-color

  - The outline-color property is used to set the color of the outline.
  The color can be set by:
  - name - specify a color name, like "red"
  - RGB - specify a RGB value, like "rgb(255,0,0)"
  - Hex - specify a hex value, like "#ff0000"

  - outline-width

  -The outline-width property specifies the width of the outline, and can have one of the following values:
  - thin (typically 1px)
  - medium (typically 3px)
  - thick (typically 5px)
  - A specific size (in px, pt, cm etc)

The CSS Outline



The CSS Transition

• CSS transitions allows you to change property values smoothly, over a given duration.

•To create a transition effect, you must specify two things: the CSS property you want to add an effect to and the duration of the effect.

•**Note:** If the duration part is not specified, the transition will have no effect, because the default value is 0.

Properties:

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function

The CSS Transition

The following table lists all the CSS transition properties:

Property	Description
transition	A shorthand property for setting the four transition properties into a single property
transition-delay	Specifies a delay (in seconds) for the transition effect
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
transition-property	Specifies the name of the CSS property the transition effect is for
transition-timing-function	Specifies the speed curve of the transition effect

The CSS Transition

```
transition-delay: time | initial | inherit
```

Value	Description
time	Defines the number of seconds (s) or milliseconds (ms) to wait before the transition will start. The default value is 0s.
initial	Sets this property to its default value.
inherit	If specified, the associated element takes the computed value of its parent element transition-delay property.

The CSS Transition

```
transition-timing-function: linear | ease | ease-in | ease-out | ease-in-out |
                            cubic-bezier(n,n,n,n) | initial | inherit
```

Value	Description
linear	Specifies that the transition goes from its initial state to its final state, with a constant speed.
ease	Similar to ease-in-out, though it accelerates more sharply at the beginning and the acceleration already starts to slow down near the middle of it.
ease-in	Specifies that the transition begins slowly, then progressively accelerates until the final state is reached and the transition stops abruptly.
ease-out	Specifies that the transition starts quickly then slow progressively down when approaching to its final state.
ease-in-out	Specifies that the transition starts slowly, accelerates then slows down when approaching its final state.
cubic-bezier(*n*,*n*,*n*,*n*)	Defines a cubic Bezier curve. It is also known as speed curve. Possible values are numeric values from 0 to 1.

The CSS Animation

- To use CSS animation, you must first specify some keyframes for the animation.

- Keyframes hold what styles the element will have at certain times.

The @keyframes Rule

- When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

- To get an animation to work, you must bind the animation to an element.

The CSS Animation

Property	Description
@keyframes	It is used to specify the animation.
animation	This is a shorthand property, used for setting all the properties, except the animation-play-state and the animation-fill- mode property.
animation-delay	It specifies when the animation will start.
animation-direction	It specifies if or not the animation should play in reserve on alternate cycle.
animation-duration	It specifies the time duration taken by the animation to complete one cycle.
animation-fill-mode	it specifies the static style of the element. (when the animation is not playing)
animation-iteration-count	It specifies the number of times the animation should be played.

The CSS Animation

```
@keyframes example
{
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */

div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

The CSS Animation

```
div {
        width: 100px;
        height: 100px;
        background-color: red;
        position: relative;
        animation-name: example;
        animation-duration: 4s;
        animation-iteration-count: 2;
        animation-direction: alternate; //to make the animation run forwards first, then  backwards
      animation-direction: alternate-reverse;
      }
```

The CSS Transforms

- CSS transforms allow you to move, rotate, scale, and skew elements.

- With the CSS transform property you can use the following 2D transformation methods:
  - translate()
  - rotate()
  - scale()
  - scaleX()
  - scaleY()

The CSS Transforms

- translate()
- The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

```
div {
    transform: translate(50px, 100px);
    //the <div> element 50 pixels to the right, and 100 pixels down from its current  position
  }
```

Rotate( )

- The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

```
div {
    transform: rotate(20deg);      //the <div> element clockwise with 20 degrees:
  }
```

•Using negative values will rotate the element counter-clockwise.

```
div {
    transform: rotate(-20deg);
  }
```

The CSS Transforms

- Scale ( )
- The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

```
div {
    transform: scale(2, 3);

      // the <div> element to be two times of its original width, and three times of its  original
          height
  }
```

scaleX( )

•The scaleX() method increases or decreases the width of an element.

```
div {
    transform: scaleX(2);
  }
```

scaleY( )

The scaleY() method increases or decreases the height of an element.

Function	Description
matrix(*n,n,n,n,n,n*)	Defines a 2D transformation, using a matrix of six values
translate(*x,y*)	Defines a 2D translation, moving the element along the X- and the Y-axis
translateX(*n*)	Defines a 2D translation, moving the element along the X-axis
translateY(*n*)	Defines a 2D translation, moving the element along the Y-axis
scale(*x,y*)	Defines a 2D scale transformation, changing the elements width and height
scaleX(*n*)	Defines a 2D scale transformation, changing the element's width
scaleY(*n*)	Defines a 2D scale transformation, changing the element's height
rotate(*angle*)	Defines a 2D rotation, the angle is specified in the parameter
skew(*x-angle,y-angle*)	Defines a 2D skew transformation along the X- and the Y-axis
skewX(*angle*)	Defines a 2D skew transformation along the X-axis
skewY(*angle*)	Defines a 2D skew transformation along the Y-axis

CSS Flexbox

- The CSS3 flexbox is used to make the elements behave predictably when they are used with different screen sizes and different display devices.

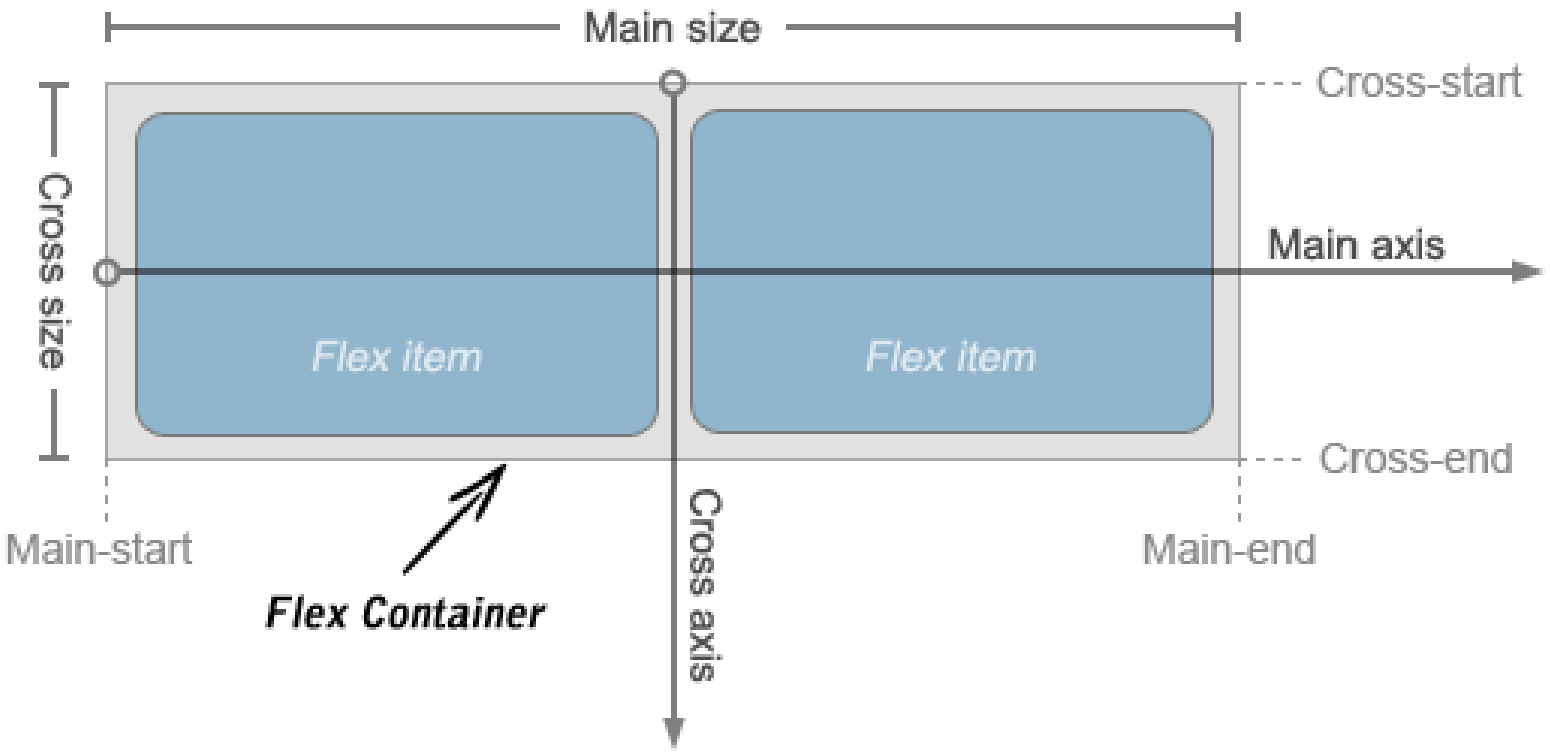
- It provides a more efficient way to layout, align and distribute space among items in the container.

The CSS3 flexbox contains flex containers and flex items.

- **Flex container:** The flex container specifies the properties of the parent. It is declared by setting the display property of an element to either flex or inline-flex.

- **Flex items:** The flex items specify properties of the children. There may be one or more flex items inside a flex container.

CSS Flexbox


Main size
Cross-start
Cross size
Main axis
Flex item
Flex item
Cross-end
Main-start
Main-end
Flex Container
Cross axis

CSS Flexbox

Property	Description
display	it is used to specify the type of box used for an html element.
flex-direction	it is used to specify the direction of the flexible items inside a flex container.
justify-content	it is used to align the flex items horizontally when the items do not use all available space on the main-axis.
align-items	it is used to align the flex items vertically when the items do not use all available space on the cross-axis.
flex-wrap	it specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line.
align-content	it is used to modify the behavior of the flex-wrap property. it is similar to align-items, but instead of aligning flex items, it aligns flex lines.

• Media Queries and Responsive Web Design

- Media queries allow you to customize the presentation of your web pages for a specific range of devices like mobile phones, tablets, desktops, etc. without any change in markups.

- A media query consists of a media type and zero or more expressions that match the type and conditions of a particular media features such as device width or screen resolution.

- Since media query is a logical expression it can be resolve to either true or false.

- The result of the query will be true if the media type specified in the media query matches the type of device the document is being displayed on, as well as all expressions in the media query are satisfied.

• Media Queries and Responsive Web Design

Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

- Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

• Media Queries and Responsive Web Design

Media Query Syntax

A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

```
@media not|only mediatype and (expressions)
 {
  CSS-Code;
}
```

Unless you use the not or only operators, the media type is optional and the all type will be implied. You can also have different stylesheets for different media:

```
<link rel="stylesheet" media="mediatype and|not|only (expressions)" href="print.css">
```

- Media Queries and Responsive Web Design

Media Types

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

• Responsive Web Design

What is Responsive Web Design?

- Responsive web design makes your web page look good on all devices.
- Responsive web design uses only HTML and CSS.
- Responsive web design is not a program or a JavaScript.

• Responsive Web Design

What is The Viewport?

- The viewport is the user's visible area of a web page.
- The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.
- Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.
- Then, when we started surfing the internet using tablets and mobile phones, fixed size web pages were too large to fit the viewport. To fix this, browsers on those devices scaled down the entire web page to fit the screen.
- This was not perfect!! But a quick fix.

• Responsive Web Design: Viewport

Setting The Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

You should include the following <meta> viewport element in all your web pages:

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

This gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

• Responsive Web Design: Viewport

Rules to follow:

1. **Do NOT use large fixed width elements** - For example, if an image is displayed at a width wider than the viewport it can cause the viewport to scroll horizontally. Remember to adjust this content to fit within the width of the viewport.

2. **Do NOT let the content rely on a particular viewport width to render well** - Since screen dimensions and width in CSS pixels vary widely between devices, content should not rely on a particular viewport width to render well.

3. **Use CSS media queries to apply different styling for small and large screens** - Setting large absolute CSS widths for page elements will cause the element to be too wide for the viewport on a smaller device. Instead, consider using relative width values, such as width: 100%. Also, be careful of using large absolute positioning values. It may cause the element to fall outside the viewport on small devices.

• Responsive Web Design: Images

Using The width Property:

If the width property is set to a percentage and the height property is set to "auto", the image will be responsive and scale up and down:

Example:

```
img {
  width: 100%;
  height: auto;
}
```

Using The max-width Property

If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

Example

```
img {
  max-width: 100%;
  height: auto;
}
```

• Responsive Web Design: Images

Different Images for Different Devices

A large image can be perfect on a big computer screen, but useless on a small device.

To reduce the load, or for any other reasons, you can use media queries to display different images on different devices.

```
/* For width smaller than 400px: */
body {
  background-image: url('img_smallflower.jpg');
}

/* For width 400px and larger: */
@media only screen and (min-width: 400px) {
  body {
    background-image: url('img_flowers.jpg');
  }
}
```

• Responsive Web Design: Images

The HTML <picture> Element

The HTML <picture> element gives web developers more flexibility in specifying image resources.

The most common use of the <picture> element will be for images used in responsive designs. Instead of having one image that is scaled up or down based on the viewport width, multiple images can be designed to more nicely fill the browser viewport.

The <picture> element works similar to the <video> and <audio> elements. You set up different sources, and the first source that fits the preferences is the one being used:

Example

```
<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 400px)">
  <source srcset="img_flowers.jpg">
  <img src="img_flowers.jpg" alt="Flowers">
</picture>
```