CH.2
CSS- Cascading Style Sheet

WEB PROGRAMMING AND PYTHON (CS104)

CSS

CSS

Three Ways to Insert CSS

- External CSS
- Internal CSS
- Inline CSS

CSS

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

CSS

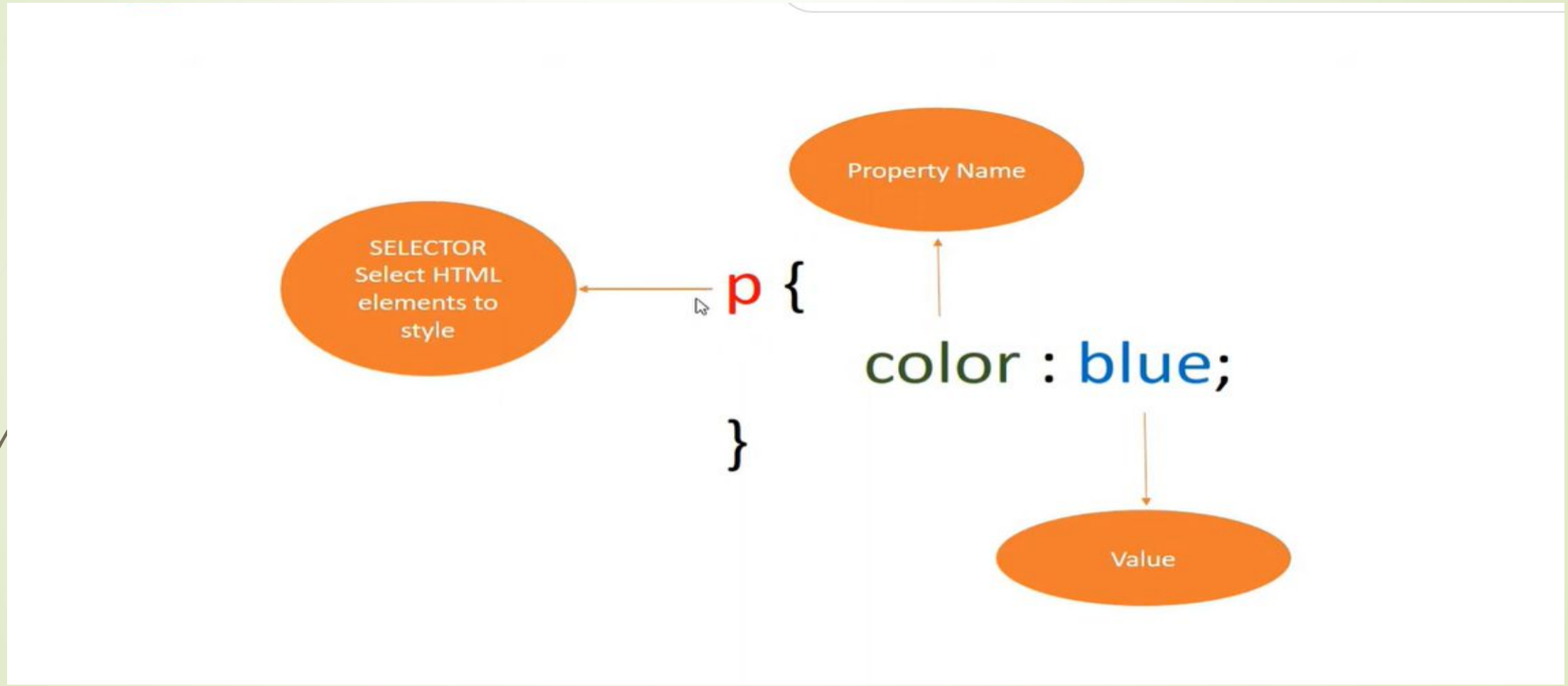
HTML: What to Display

CSS: How to Display

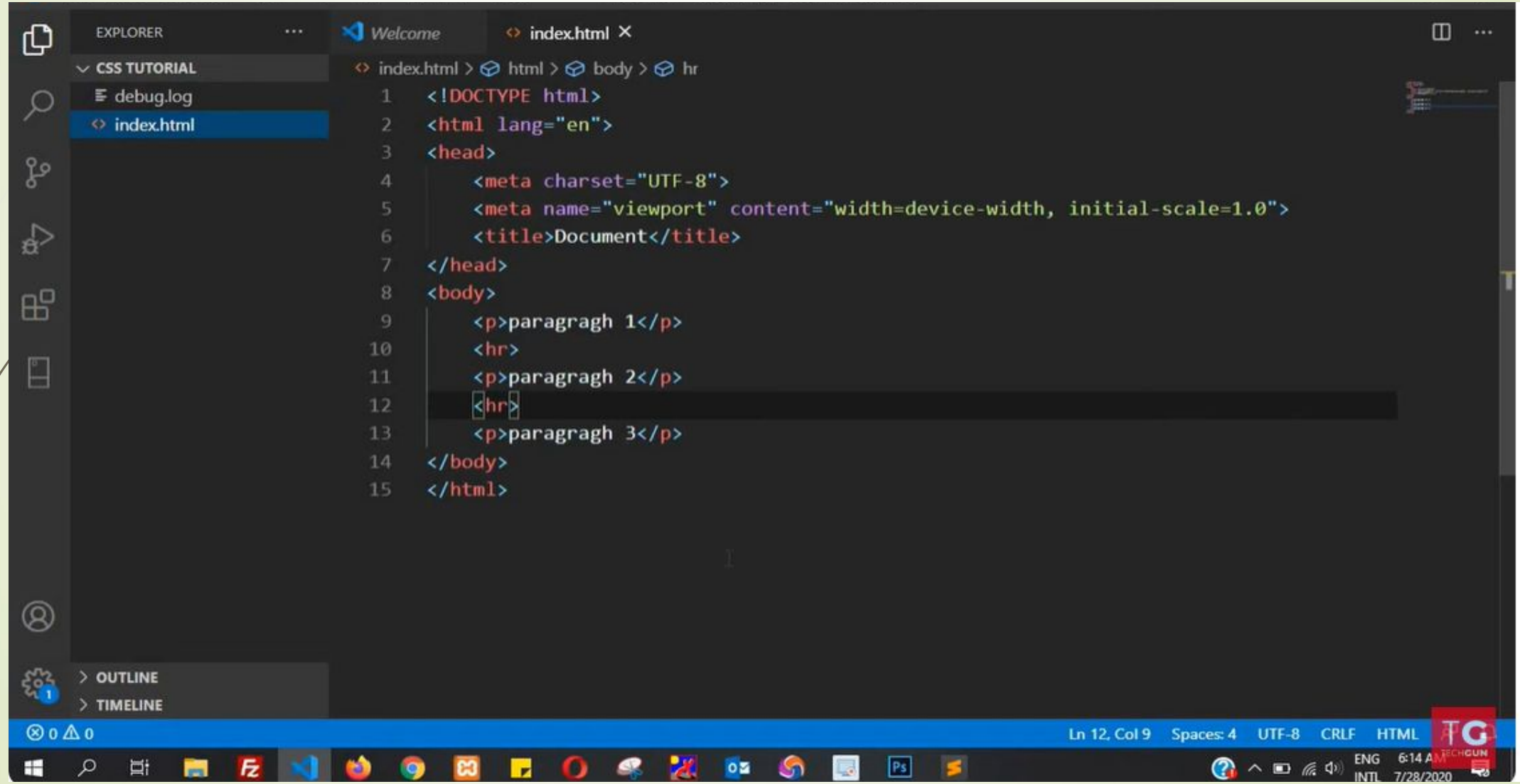
CSS

```
p {
        color : blue;
}
```

CSS


Property Name
SELECTOR
Select HTML
elements to
style
p {
color : blue;
}
Value

HTML _ CSS


EXPLORER
CSS TUTORIAL
debug.log
index.html
Welcome
index.html
index.html > html > body > hr
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Document</title>
7 </head>
8 <body>
9 paragragh 1
10 <hr>
11 paragragh 2
12 <hr>
13 paragragh 3
14 </body>
15 </html>
OUTLINE
TIMELINE
Ln 12, Col 9
Spaces: 4
UTF-8
CRLF
HTML

HTML _ CSS

- The **lang** attribute specifies the language of the element's content.
- Common examples are "en" for English, "es" for Spanish, "fr" for French and so on.
- The lang attribute is a Global Attribute, and can be used on any HTML element.
- Some French text in a paragraph:

```
<p lang="fr">Ceci est un paragraphe.</p>
```

- The **<meta>** tag defines metadata about an HTML document. Metadata is data (information) about data.
- <meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.
- There is a method to let web designers take control over the viewport (the user's visible area of a web page), through the <meta> tag (See "Setting The Viewport" example below).

HTML _ CSS

Describe metadata within an HTML document:

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <meta name="author" content="John Doe">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

- Define keywords for search engines:

```
<meta name="keywords" content="HTML, CSS, JavaScript">
```

- Define a description of your web page:

```
<meta name="description" content="Free Web tutorials for HTML and CSS">
```

- Define the author of a page:

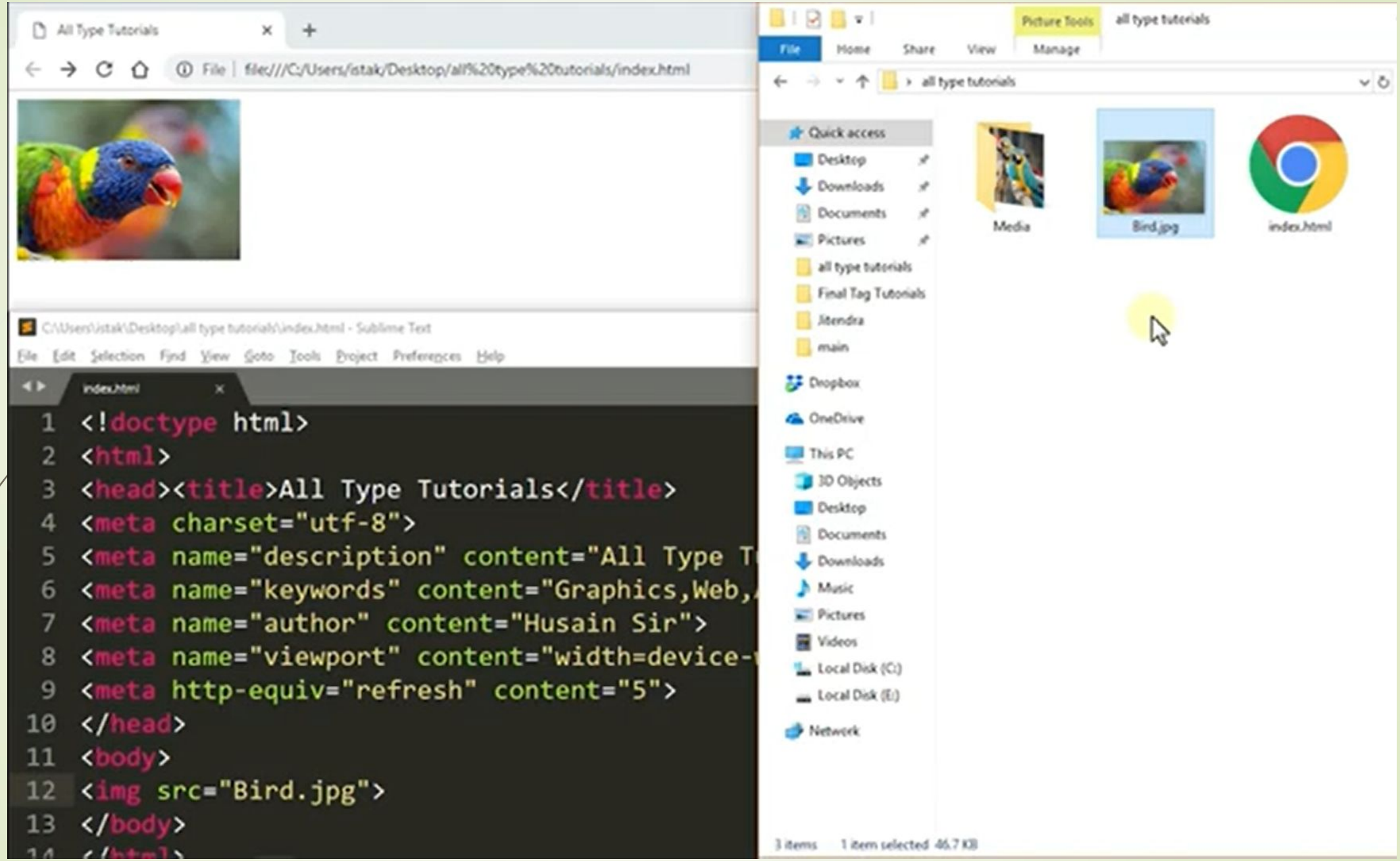
```
<meta name="author" content="John Doe">
```

- Refresh document every 30 seconds:

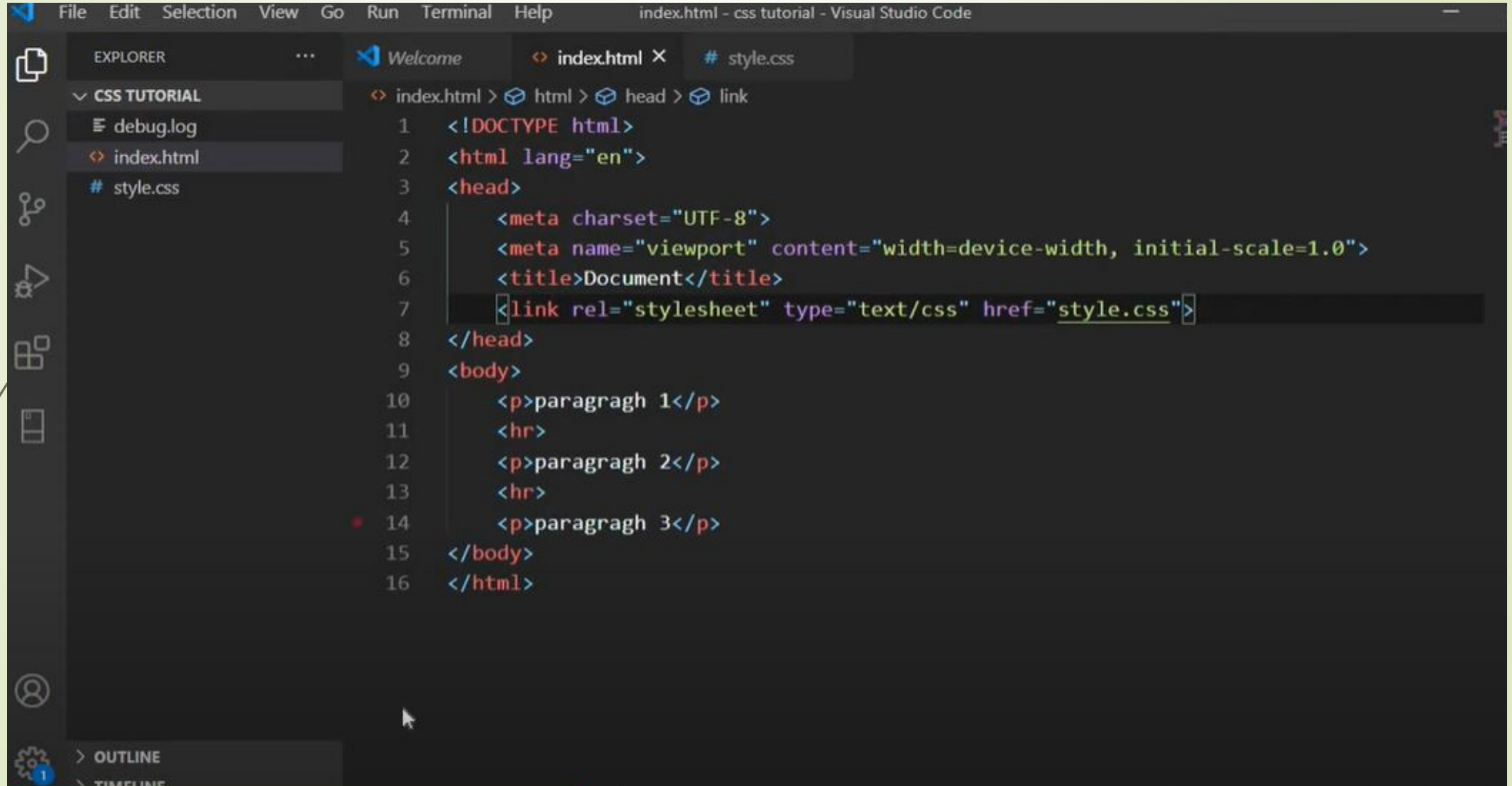
```
<meta http-equiv="refresh" content="30">
```

- Setting the viewport to make your website look good on all devices:

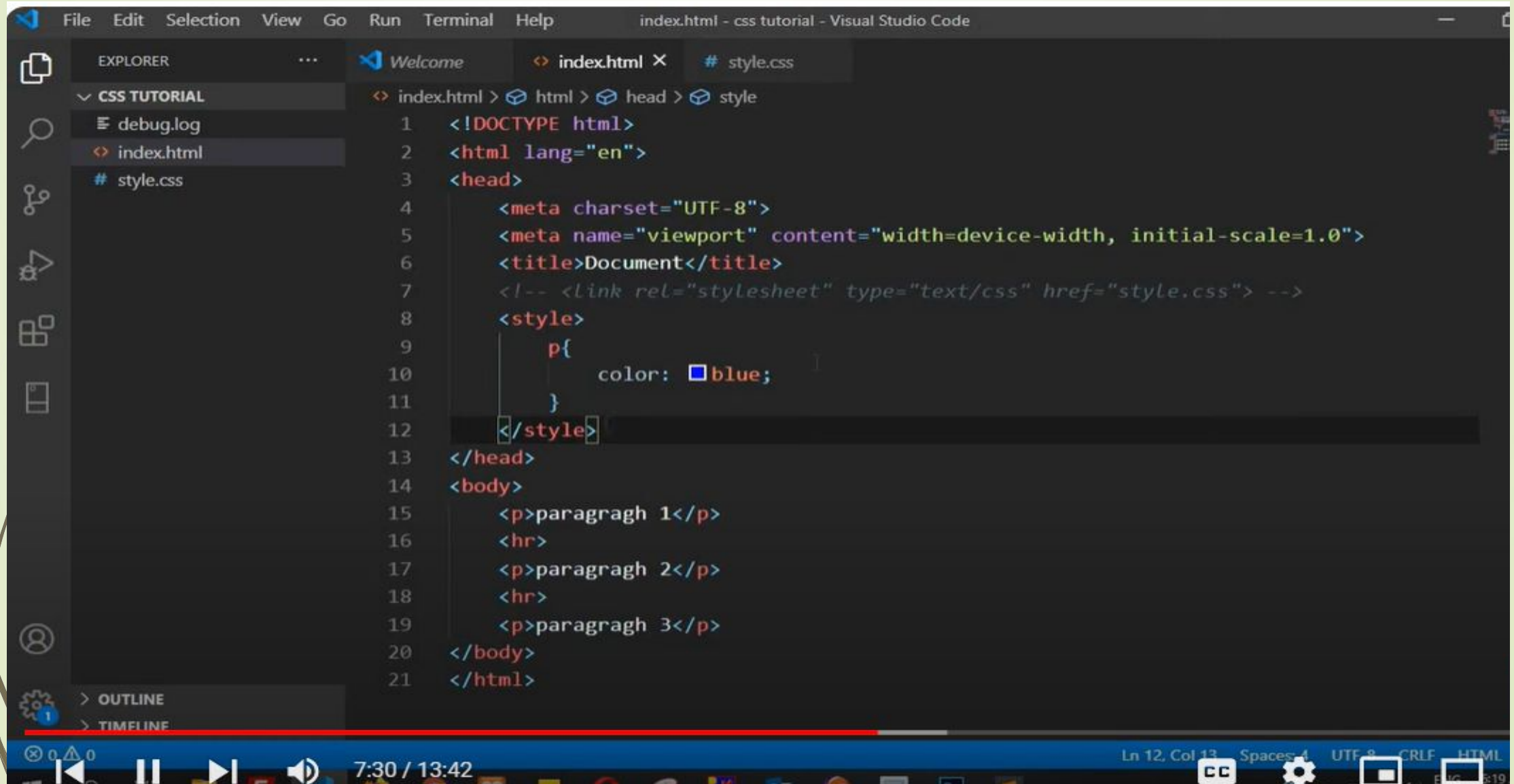
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```


All Type Tutorials
File | file:///C:/Users/istak/Desktop/all%20type%20tutorials/index.html
C:\Users\istak\Desktop\all type tutorials\index.html - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
index.html
1 <!doctype html>
2 <html>
3 <head><title>All Type Tutorials</title>
4 <meta charset="utf-8">
5 <meta name="description" content="All Type T
6 <meta name="keywords" content="Graphics,Web,
7 <meta name="author" content="Husain Sir">
8 <meta name="viewport" content="width=device-
9 <meta http-equiv="refresh" content="5">
10 </head>
11 <body>
12 <img src="Bird.jpg">
13 </body>
Picture Tools
all type tutorials
File Home Share View Manage
all type tutorials
Quick access
Desktop
Downloads
Documents
Pictures
all type tutorials
Final Tag Tutorials
Jitendra
main
Dropbox
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (E:)
Network
Media
Bird.jpg
index.html
3 items 1 item selected 46.7 KB

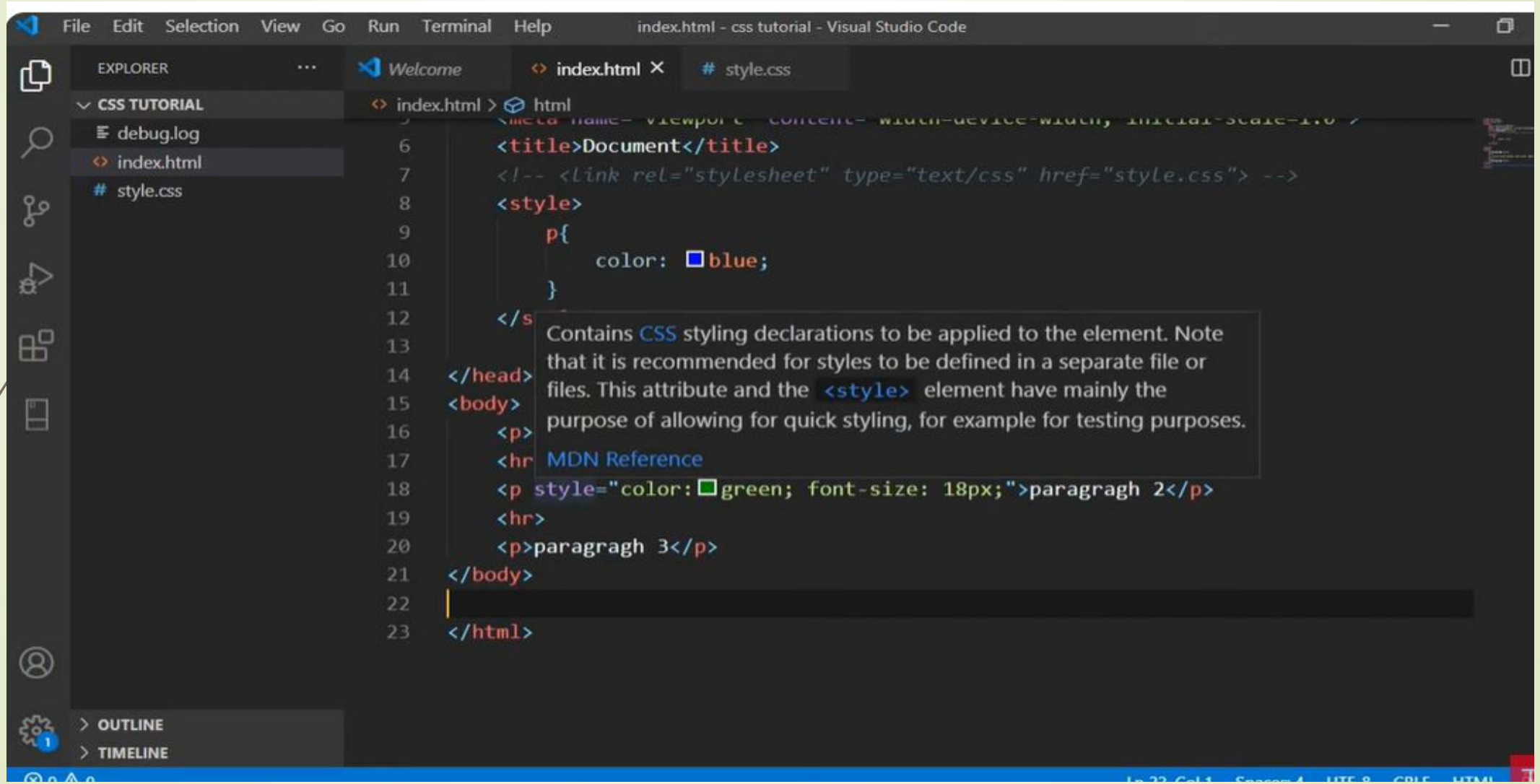
External CSS



Internal CSS



Inline CSS


File Edit Selection View Go Run Terminal Help
index.html - css tutorial - Visual Studio Code
EXPLORER
CSS TUTORIAL
debug.log
index.html
style.css
Welcome
index.html
style.css
index.html > html
<title>Document</title>
<!-- <link rel="stylesheet" type="text/css" href="style.css"> -->
<style>
p{
color: blue;
}
Contains CSS styling declarations to be applied to the element. Note that it is recommended for styles to be defined in a separate file or files. This attribute and the <style> element have mainly the purpose of allowing for quick styling, for example for testing purposes.
MDN Reference
</head>
<body>
paragragh 2
<hr>
paragragh 3
</body>
</html>
OUTLINE
TIMELINE

CSS

- Internal CSS, also known as embedded CSS, involves adding CSS rules directly within the <style> element in the <head> section of an HTML document. It allows styling specific to that document.

  - **Internal CSS Syntax:**
  - **<style>**
  - **// CSS Properties**
  - **</style>**

- External CSS is used to place CSS code in a separate file and link to the HTML document. To use external CSS, create a separate file with the .css file extension that contains your CSS rules. You can link this file to your HTML document using the "link" tag in the head section of your HTML document.

External CSS Syntax:

```
<head>
    <link rel="stylesheet"
       type="text/css" href="style.css">
</head>
```

CSS

- Inline CSS is a way of defining the styling of an HTML element by adding CSS rules directly to the element's tag using the "style" attribute. It is used for quick and simple styling changes to specific elements, without creating a separate CSS file.

Inline CSS Syntax:

```
<p style="css_styles">
    // Content
</p>
```

CSS- Selector

CSS selectors are used *to select the content you want to style*. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

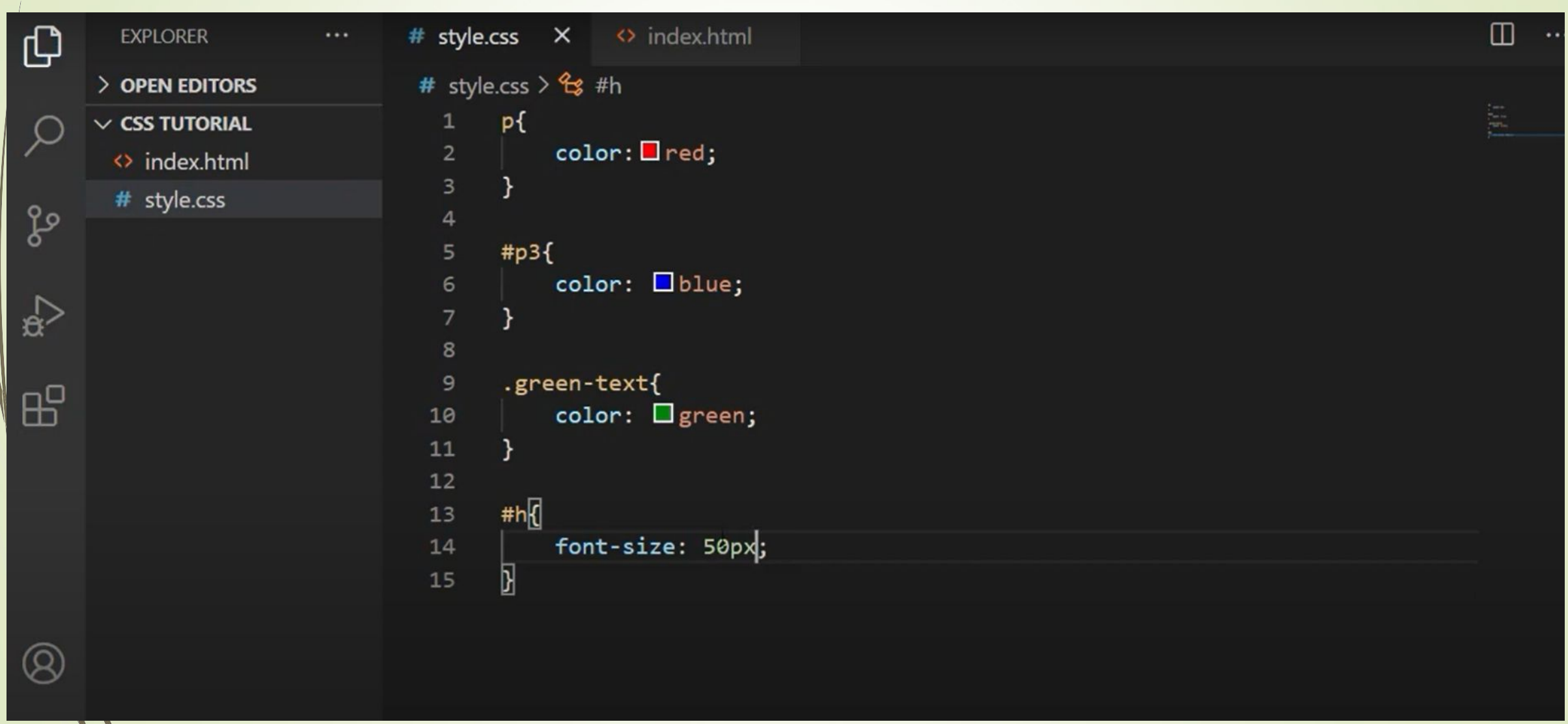
- **Element Selector:** The element selector selects the HTML element by name.
- **:id Selector :** The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

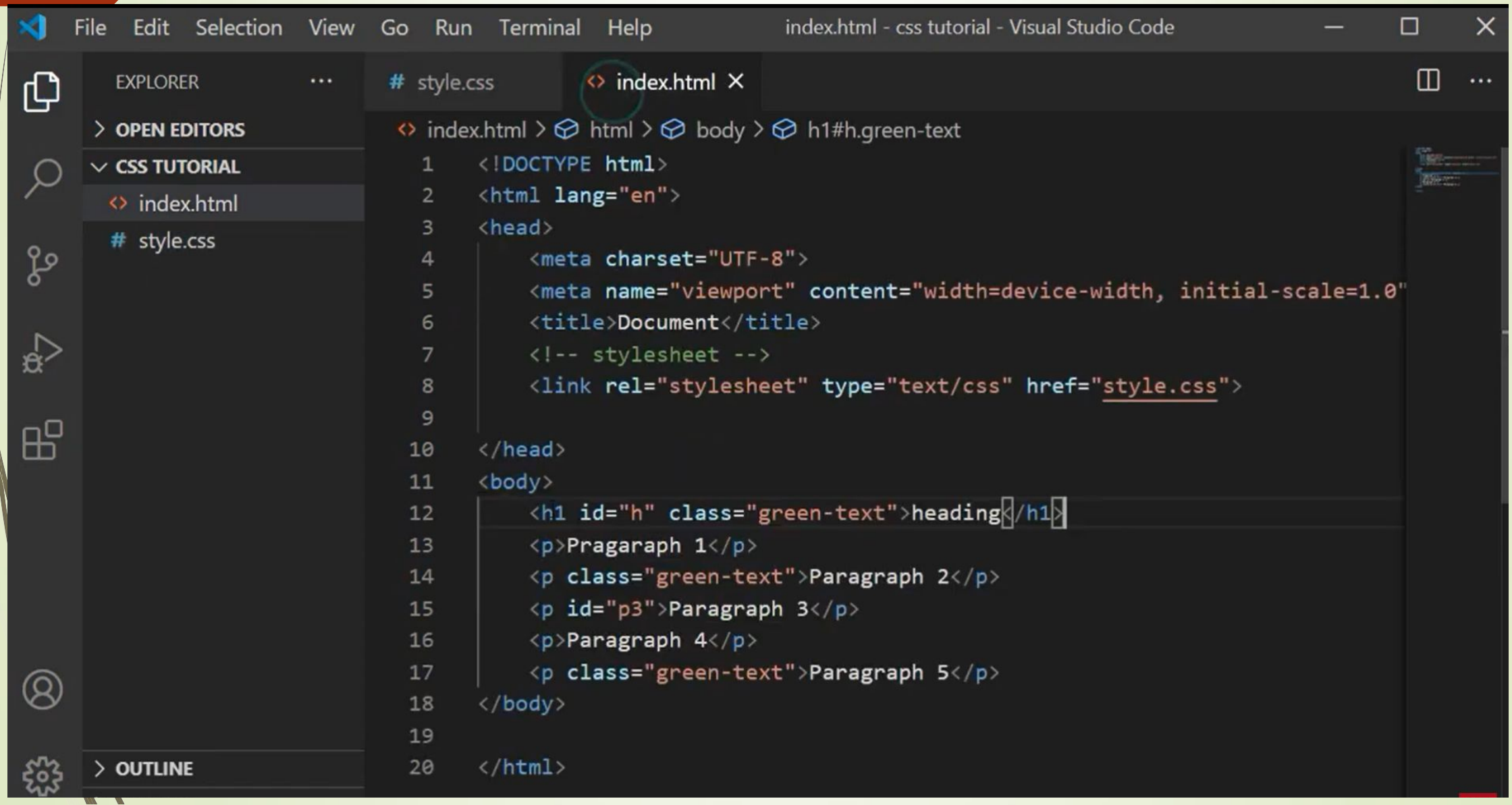
- **Class Selector:** The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name.

Note: A class name should not be started with a number.

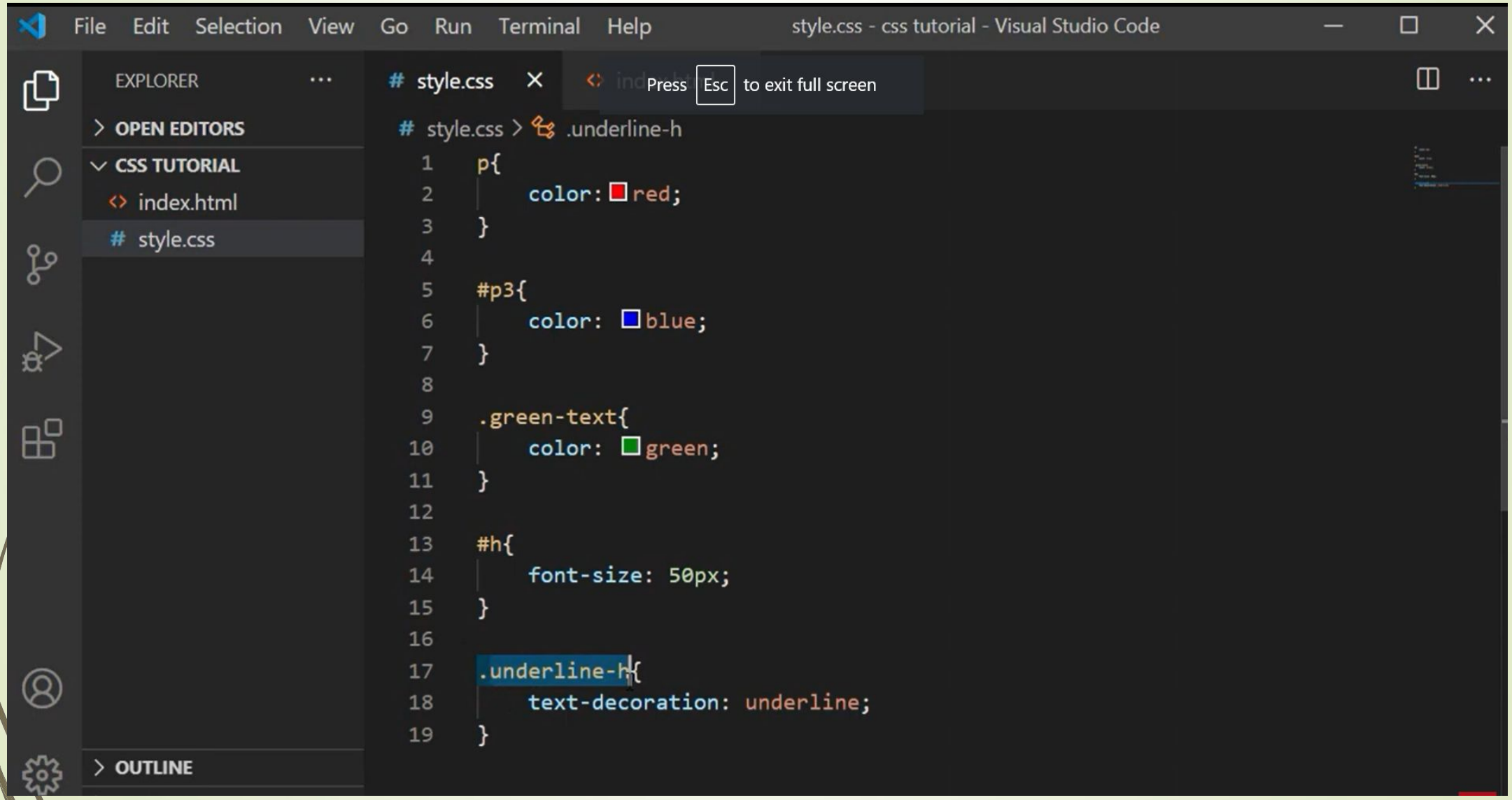
CSS- Selector



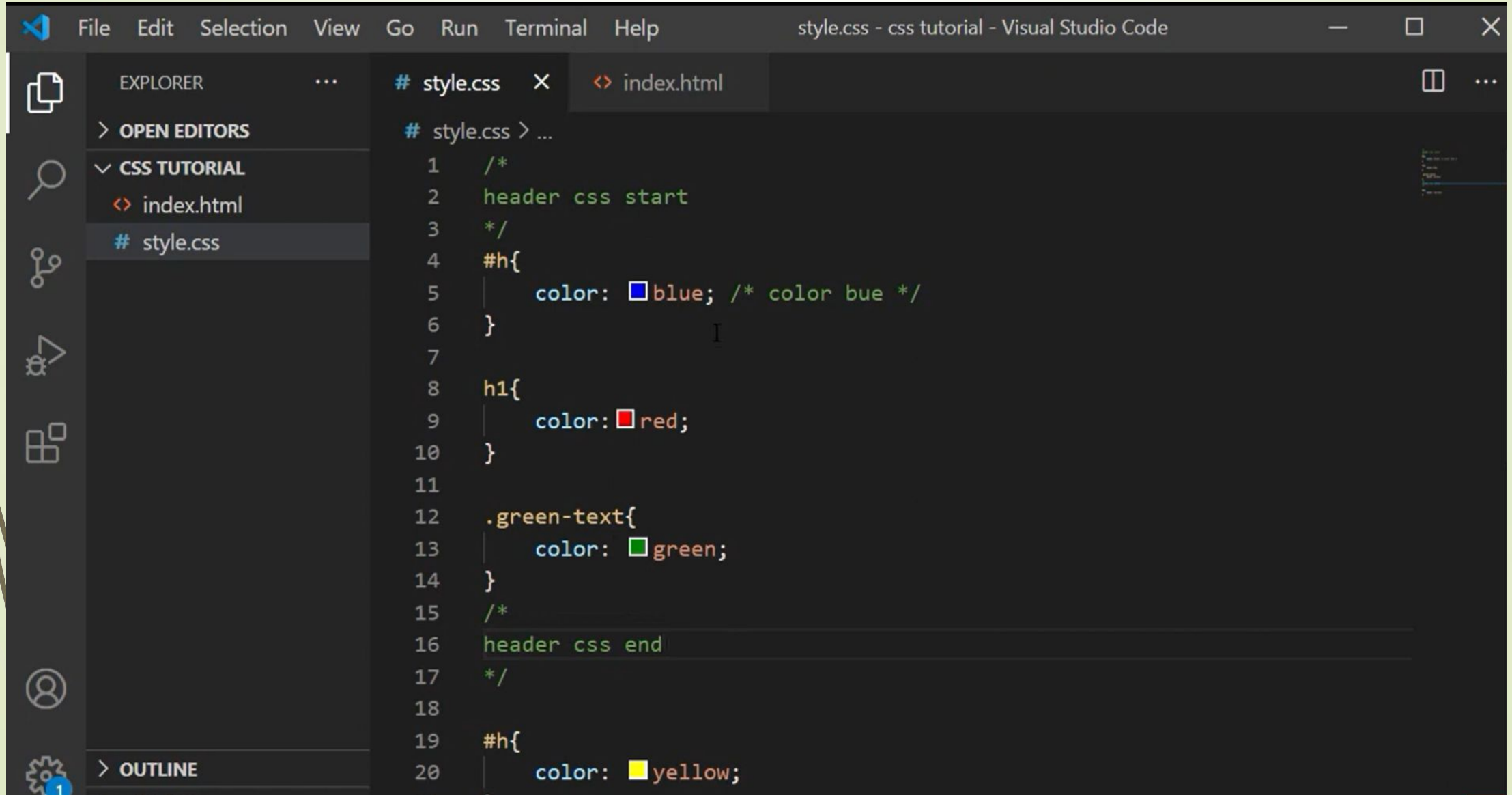
CSS- Selector



CSS- Selector



CSS- Comments in CSS



CSS- Colours in CSS

Predefined color names

- **Modern browsers support 140 named colors**
- https://htmlcolorcodes.com/color-names/

In CSS Colors can be specified using

- Predefined color names
- RGB
- RGBA
- HEX
- HSL
- HSLA

CSS- Colours in CSS

RGB

- rgb(*red, green, blue*)
- Color value between 0 and 255
- Black : **rgb(0, 0, 0)**
- White : **rgb(255, 255, 255)**
- Red : **rgb(255, 0, 0)**
- Green : **rgb(0, 255, 0)**
- Blue : **rgb(0, 0, 255)**

RGBA

- rgb(*red, green, blue*, ***alpha***)
- alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all)
- **rgba(255, 99, 71, 0.5)**

HEX

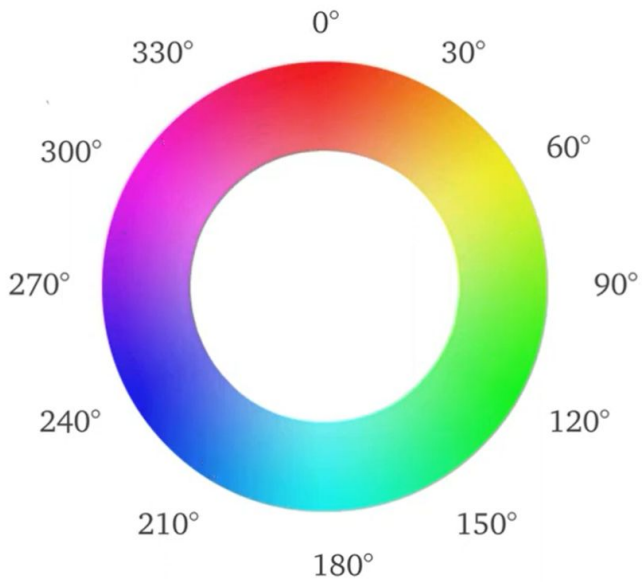
- ***#rrggbb***
- rr (red), gg (green) and bb (blue)
- hexadecimal values between 00 and ff (same as decimal 0-255)
- Red: **#ff0000**
- Black: **#000000**
- White: **#ffffff**

CSS- Colours in CSS

HSL

- **hsl(*hue, saturation, lightness*)**
- hue, saturation, and lightness (HSL)
- Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
- Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.
- Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white
- **RED: hsl(0, 100%, 50%)**

HSL



HSLA

- **hsla(*hue, saturation, lightness, alpha*)**
- alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all)
- **hsla(0, 100%, 50%, 0.5)**

CSS- Border & Background in CSS

- The background-clip property defines how far the background (color or image) should extend within an element. **Padding** is used to create space around an element's content, inside of any defined borders.

Specify how far the background should extend within an element:

```
div {
  border: 10px dotted black;
  padding: 15px;
  background: lightblue;
  background-clip: padding-box;
}
```

CSS- Border & Background in CSS

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

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

CSS- Border & Background in CSS

Example
Demonstration of the different border styles:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

CSS- Border & Background in CSS

Specific Side Widths

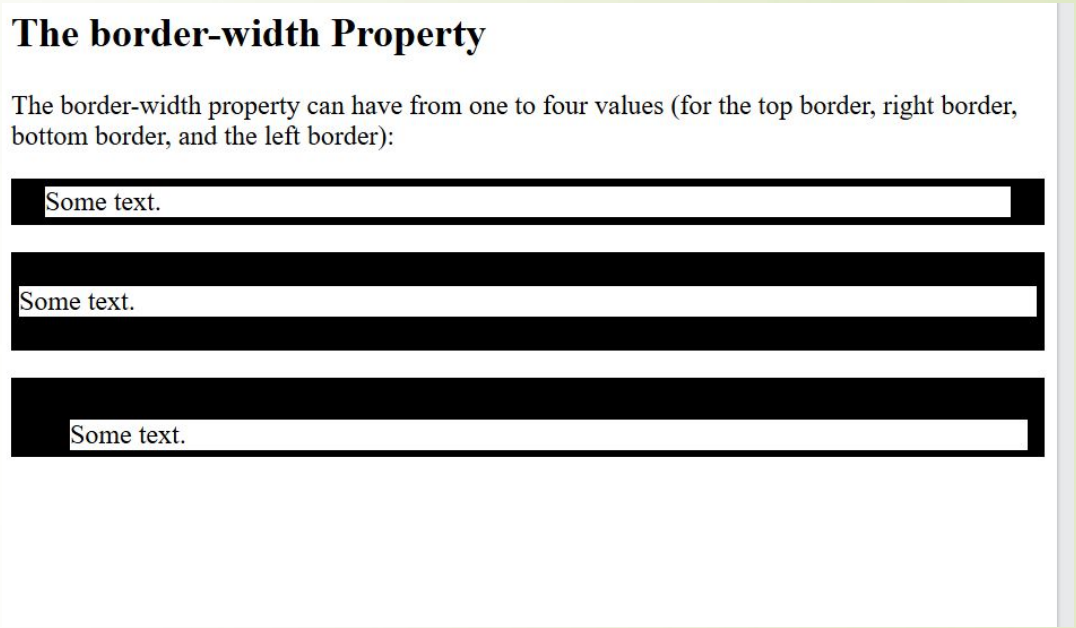
The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border):

Example

```
p.one {
  border-style: solid;
  border-width: 5px 20px; /* 5px top and bottom, 20px on the sides */
}

p.two {
  border-style: solid;
  border-width: 20px 5px; /* 20px top and bottom, 5px on the sides */
}

p.three {
  border-style: solid;
  border-width: 25px 10px 4px 35px; /* 25px top, 10px right, 4px bottom and 35px left */
}
```

CSS- Border & Background in CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  border-style: solid;
  border-width: 5px 20px; /* 5px top and bottom, 20px on the sides */
}
p.two {
  border-style: solid;
  border-width: 20px 5px; /* 20px top and bottom, 5px on the sides */
}
p.three {
  border-style: solid;
  border-width: 25px 10px 4px 35px; /* 25px top, 10px right, 4px bottom and 35px left */
}
</style>
</head>
<body>
<h2>The border-width Property</h2>
<p>The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border):</p>

<p class="one">Some text.</p>
<p class="two">Some text.</p>
<p class="three">Some text.</p>

</body>
</html>
```



CSS- CSS padding

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

Padding - Individual Sides
CSS has properties for specifying the padding for each side of an element:

padding-top
padding-right
padding-bottom
padding-left

CSS- CSS padding

All the padding properties can have the following values:

length - specifies a padding in px, pt, cm, etc.
% - specifies a padding in % of the width of the containing element
inherit - specifies that the padding should be inherited from the parent element
Note: Negative values are not allowed.

Example

Set different padding for all four sides of a element:

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

CSS-CSS Setting height and width

- The height and width properties are used to set the height and width of an element.

- The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

- CSS height and width Values
- The height and width properties may have the following values:

  - auto - This is default. The browser calculates the height and width
  - length - Defines the height/width in px, cm, etc.
  - % - Defines the height/width in percent of the containing block
  - initial - Sets the height/width to its default value
  - inherit - The height/width will be inherited from its parent value

CSS-CSS Setting height and width

CSS height and width Examples

This element has a height of 200 pixels and a width of 50%

Example

Set the height and width of a element:

```
div {
  height: 200px;
  width: 50%;
  background-color: powderblue;
}
```

CSS-The CSS Box Model

- 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: content, padding, borders and margins. The image below illustrates the box model:
- Explanation of the different parts:

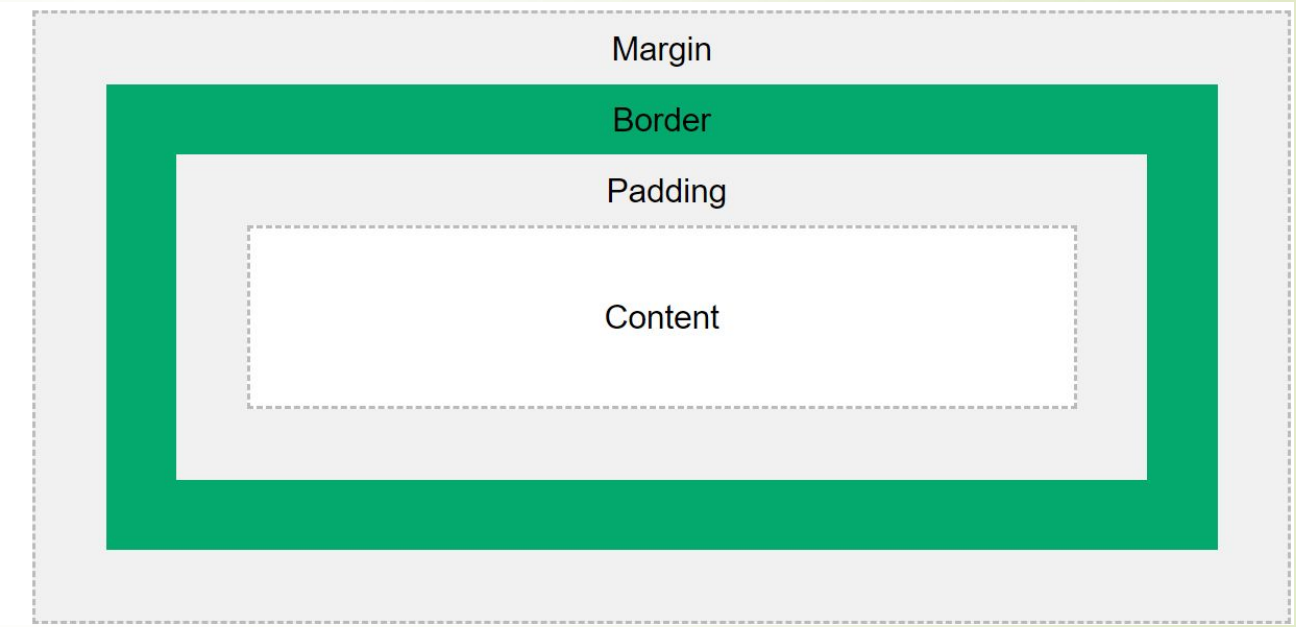
  - Content - The content of the box, where text and images appear
  - Padding - Clears an area around the content. The padding is transparent
  - Border - A border that goes around the padding and content
  - Margin - Clears an area outside the border. The margin is transparent
- The box model allows us to add a border around elements, and to define space between elements.

CSS-The CSS Box Model

Example

Demonstration of the box model:

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



CSS-CSS 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
groove - Defines a 3D grooved outline
ridge - Defines a 3D ridged outline
inset - Defines a 3D inset outline
outset - Defines a 3D outset outline
none - Defines no outline
hidden - Defines a hidden outline

CSS-CSS Lists

- Unordered Lists:
  - Coffee
  - Tea
  - Coca Cola
    - Coffee
    - Tea
    - Coca Cola
- Ordered Lists:
  1. Coffee
  2. Tea
  3. Coca Cola
     1. Coffee
     i. Tea
     ii. Coca Cola

CSS-CSS Lists

- HTML Lists and CSS List Properties
- In HTML, there are two main types of lists:

- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters
- The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

CSS-CSS Lists

Example

```
ul.a {
  list-style-type: circle;
}

ul.b {
  list-style-type: square;
}

ol.c {
  list-style-type: upper-roman;
}

ol.d {
  list-style-type: lower-alpha;
}
```

CSS/HTML – Input Type

Definition and Usage
The <input> tag specifies an input field where the user can enter data.

The <input> element is the most important form element.

The <input> element can be displayed in several ways, depending on the type attribute.

The different input types are as follows:

<input type="button">
<input type="checkbox">
<input type="color">
<input type="date">
<input type="email">
<input type="file">
<input type="hidden">
<input type="image">
<input type="month">
<input type="number">
<input type="password">
<input type="radio">
<input type="range">
<input type="reset">
<input type="search">
<input type="submit">
<input type="tel">
<input type="text"> (default value)
<input type="time">
<input type="url">
<input type="week">

CSS-CSS Tables

- The look of an HTML table can be greatly improved with CSS:
- Table Borders
- To specify table borders in CSS, use the border property.

- The example below specifies a solid border for <table>, <th>, and <td> elements:

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Example

```
table, th, td {
  border: 1px solid;
}
```

CSS-CSS Overflow

- The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

- The overflow property has the following values:

- visible - Default. The overflow is not clipped. The content renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible
- scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
- auto - Similar to scroll, but it adds scrollbars only when necessary
- Note: The overflow property only works for block elements with a specified height.

- Note: In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

CSS-CSS Overflow

overflow: visible

By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:

You can use the overflow property when
you want to have better control of the layout.

The overflow property specifies what happens if content overflows an element's box.

Example

```
div {
  width: 200px;
  height: 65px;
  background-color: coral;
  overflow: visible;
}
```

CSS-CSS Layout - display: inline-block

- Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.
- Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.
- Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

The following example shows the different behavior of display: inline, display: inline-block and display: block:

Example

```
span.a {
  display: inline; /* the default for span */
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}
```

Program

```
<!DOCTYPE html>
<html>
<head>
<style>
span.a {
  display: inline; /* the default for span */
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}

span.b {
  display: inline-block;
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}

span.c {
  display: block;
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}
</style>
</head>
<body>

<h1>The display Property</h1>

<h2>display: inline</h2>
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat.
<span class="a">Aliquam</span>
<span class="a">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.
</div>

<h2>display: inline-block</h2>
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat.
<span class="b">Aliquam</span>
<span class="b">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.
</div>

<h2>display: block</h2>
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat.
<span class="c">Aliquam</span>
<span class="c">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.
</div>
</body>
</html>
```

CSS-CSS Combinators

- A combinator is something that explains the relationship between the selectors.

- A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

1. descendant selector (space)
2. child selector (>)
3. adjacent sibling selector (+)
4. general sibling selector (~)

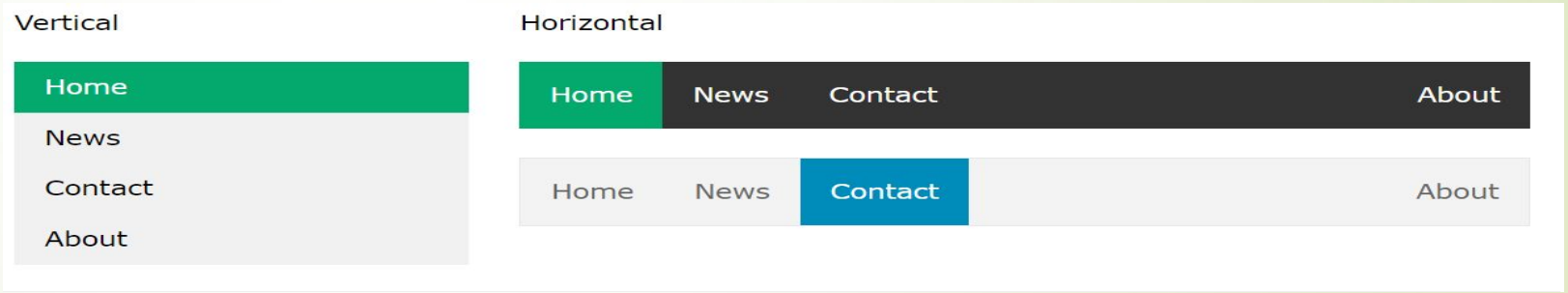
The following example selects all <p> elements that are children of a element:

Example

```
div > p {
  background-color: yellow;
}
```

CSS-Navigation Bar = List of Links

- Having easy-to-use navigation is important for any web site.
- With CSS you can transform boring HTML menus into good-looking navigation bars.
- A navigation bar needs standard HTML as a base.
- In our examples we will build the navigation bar from a standard HTML list.
- A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense:



Example

```
<ul>
  <li><a href="default.asp">Home</a></li>
  <li><a href="news.asp">News</a></li>
  <li><a href="contact.asp">Contact</a></li>
  <li><a href="about.asp">About</a></li>
</ul>
```

CSS-Basic Dropdown

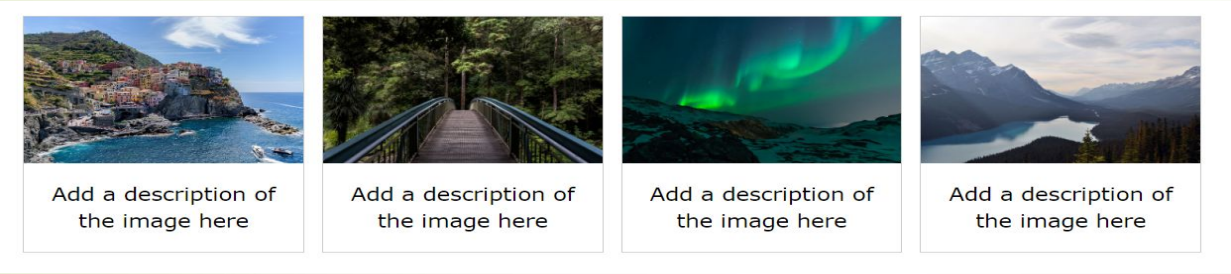
Create a dropdown box that appears when the user moves the mouse over an element.

Example

```
<style>
.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px
rgba(0,0,0,0.2);
  padding: 12px 16px;
  z-index: 1;
}

.dropdown:hover .dropdown-content {
  display: block;
}
</style>

<div class="dropdown">
  <span>Mouse over me</span>
  <div class="dropdown-content">
    <p>Hello World!</p>
  </div>
</div>
```

CSS-CSS Image Gallery

CSS can be used to create an image gallery.



CSS-CSS Website Layout

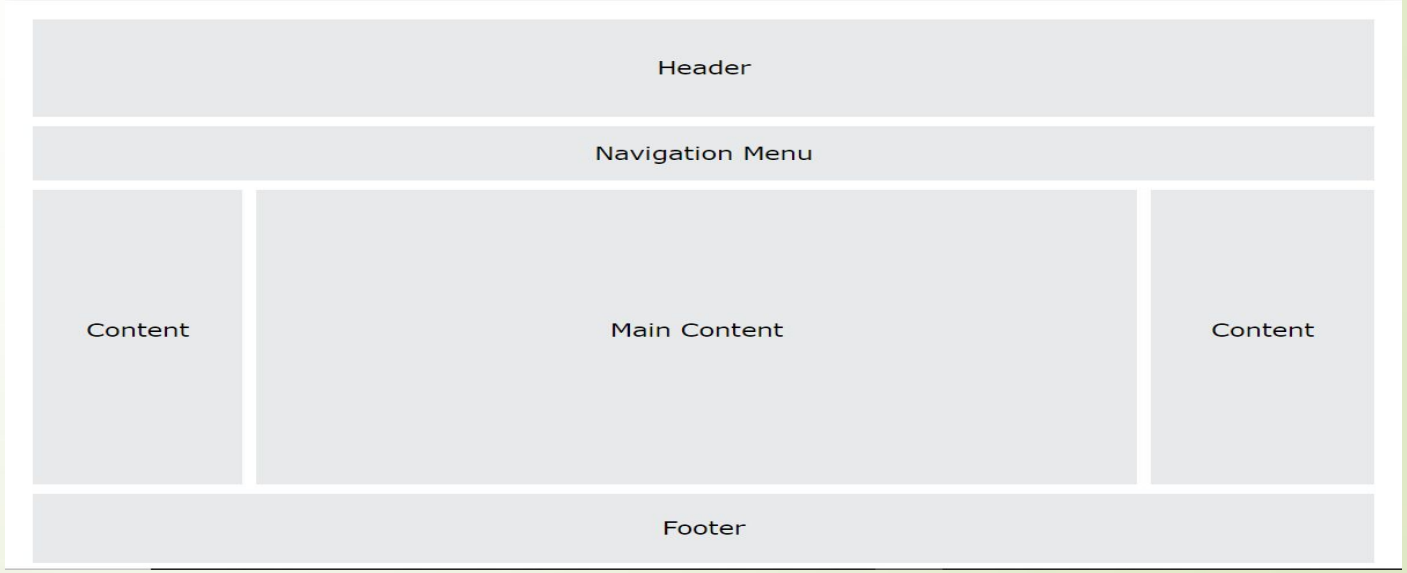
- A website is often divided into headers, menus, content and a footer:
- Different layout designs to choose form. However, the structure below, is one of the most common, and we will take a closer look at it in this tutorial.

Header

- A header is usually located at the top of the website (or right below a top navigation menu). It often contains a logo or the website name:

- **Example**

```
.header {
  background-color: #F1F1F1;
  text-align: center;
  padding: 20px;
}
```



CSS-CSS The !important Rule

What is !important?

- The !important rule in CSS is used to add more importance to a property/value than normal.
- In fact, if you use the !important rule, it will override ALL previous styling rules for that specific property on that element!

Example

```
#myid {
  background-color: blue;
}

.myclass {
  background-color: gray;
}

p {
  background-color: red !important;
}
```

CSS-CSS Math Functions

The CSS math functions allow mathematical expressions to be used as property values. Here, we will explain the calc(), max() and min() functions.

The calc() Function

The calc() function performs a calculation to be used as the property value.

CSS Syntax

calc(expression)

Value	**Description**
expression	Required. A mathematical expression. The result will be used as the value.

CSS-CSS Math Functions

The following operators can be used: + - * /

Example

Use calc() to calculate the width of a element:

```
#div1 {
  position: absolute;
  left: 50px;
  width: calc(100% / 2);
  width: calc(100% - 100px);
  width: calc(50% * 2);
  width: calc(50% + 50%);
  border: 1px solid black;
  background-color: yellow;
  padding: 5px;
}
```

Example

Use max() to set the width of #div1 to whichever value is largest, 50% or 300px:

```
#div1 {
  background-color: yellow;
  height: 100px;
  width: max(50%, 300px);
}
```

CSS 2D Transforms

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

CSS 2D Transforms Functions

With the CSS transform property you can use the following 2D transformation functions:

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()

CSS 2D Transforms

The translate() Function

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

The following example moves the element 50 pixels to the right, and 100 pixels down from its current position:

Example

```
div {
  transform: translate(50px, 100px);
}
```

CSS 2D Transforms

The rotate() Function

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

The following example rotates the element clockwise with 20 degrees:

Example

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

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

CSS 2D Transforms

The scale() Function

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

The following example increases the element to be two times of its original width, and three times of its original height:

Example

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

The following example decreases the element to be half of its original width and height:

Example

```
div {
  transform: scale(0.5, 0.5);
}
```

CSS 2D Transforms

CSS 2D Transform Functions

Function	Description
translate()	Defines a 2D translation, moving the element along the X & Y axis
translateX()	Defines a 2D translation, moving the element along the X-axis
translateY()	Defines a 2D translation, moving the element along the Y-axis
scale()	Defines a 2D scale transformation, scaling the elements w and h.
scaleX()	Defines a 2D scale transformation, scaling the element's width
scaleY()	Defines a 2D scale transformation, scaling the element's height
rotate()	Defines a 2D rotation, the angle is specified in the parameter
skew()	Defines a 2D skew transformation along the X- and the Y-axis
skewX()	Defines a 2D skew transformation along the X-axis
skewY()	Defines a 2D skew transformation along the Y-axis

CSS 3D Transforms

CSS 3D Transforms
CSS also supports 3D transformations.

Mouse over the elements below to see the difference between a 2D and a 3D transformation:

CSS 3D Transforms Functions
With the CSS transform property you can use the following 3D transformation functions:

- rotateX()
- rotateY()
- rotateZ()

The rotateX() function rotates an element around its X-axis at a given degree:

Example

```
#myDiv {
  transform: rotateX(150deg);
}
```

CSS 3D Transforms

CSS 3D Transforms
CSS also supports 3D transformations.

Mouse over the elements below to see the difference between a 2D and a 3D transformation:

CSS 3D Transforms Functions
With the CSS transform property you can use the following 3D transformation functions:

- rotateX()
- rotateY()
- rotateZ()

The rotateX() function rotates an element around its X-axis at a given degree:

Example

```
#myDiv {
  transform: rotateX(150deg);
}
```

CSS 3D Transforms

The following table lists all the 3D transform properties:

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space
perspective	Specifies the perspective on how 3D elements are viewed
perspective-origin	Specifies the bottom position of 3D elements
backface-visibility	Defines whether or not an element should be visible when not facing the screen

CSS 3D Transforms

CSS 3D Transform Functions

Function	Description
translate3d()	Defines a 3D translation
translateZ()	Defines a 3D translation, using only the value for the Z-axis
scale3d()	Defines a 3D scale transformation
scaleZ()	Defines a 3D scale transformation by giving a value for the Z-axis
rotate3d()	Defines a 3D rotation
rotateX()	Defines a 3D rotation along the X-axis
rotateY()	Defines a 3D rotation along the Y-axis
rotateZ()	Defines a 3D rotation along the Z-axis
perspective()	Defines a perspective view for a 3D transformed element

CSS Animations

Following properties:

- @keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

CSS Animations

What are CSS Animations?

An animation lets an element gradually change from one style to another.
You can change as many CSS properties you want, as many times as you want.
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.

The following example binds the "example" animation to the element. The animation will last for 4 seconds, and it will gradually change the background-color of the element from "red" to "yellow":

CSS Animations

Note: The animation-duration property defines how long an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

CSS Animations

```
/* The animation code */
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}

@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
</style>
</head>
<body>

<h1>CSS Animation</h1>

<div></div>

<p><b>Note:</b> When an animation is finished, it goes back to its original style.</p>

</body>
</html>
```

Flexbox

CSS3 Flexbox

- Flexible boxes, or flexbox, is a new layout mode in CSS3
- Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices
- For many applications, the flexible box model provides an improvement over the block model in that it does not use floats, nor do the flex container's margins collapse with the margins of its contents

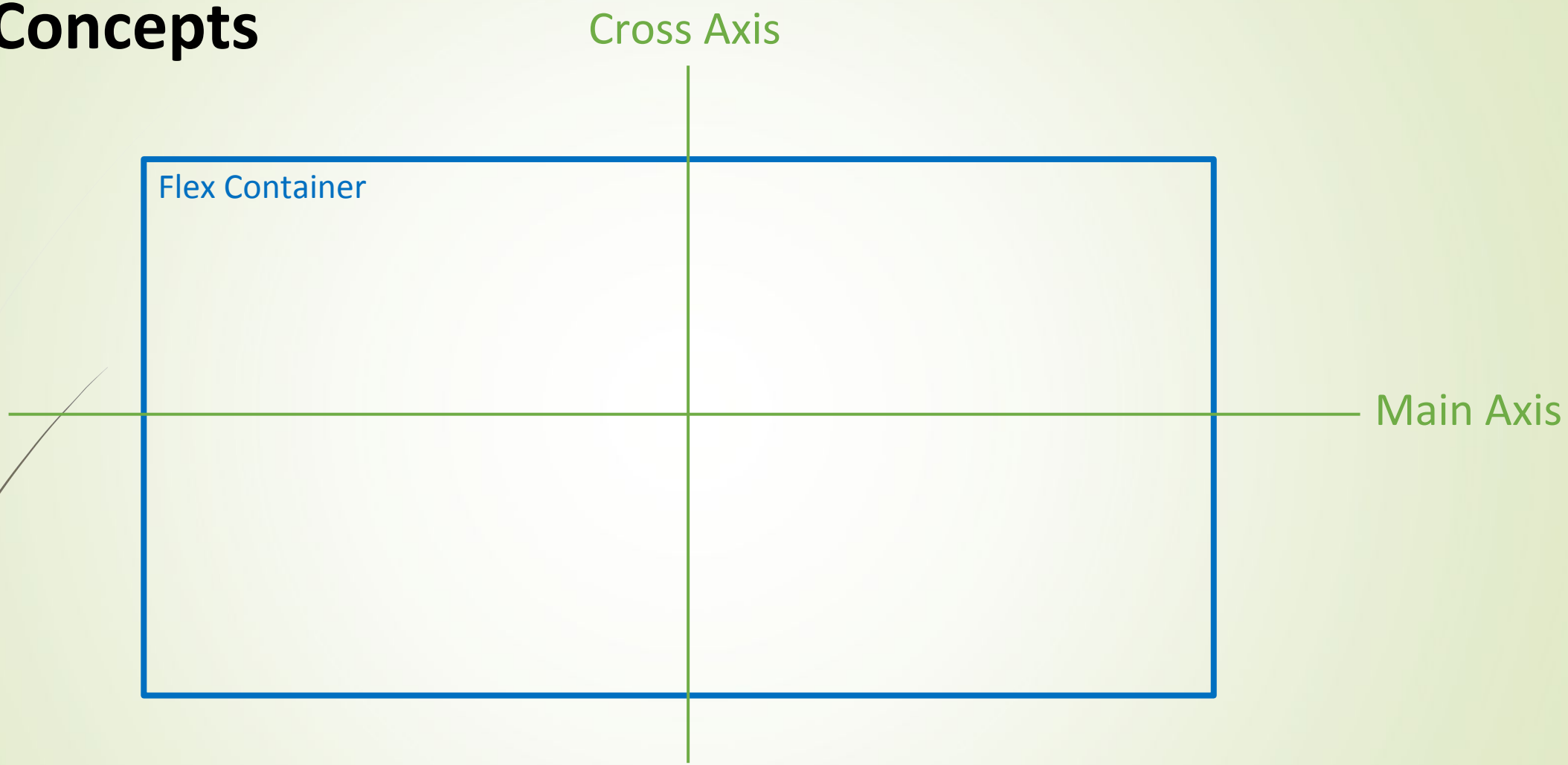
CSS3 Flexbox - Concepts

- Flexbox consists of flex containers and flex items
- A flex container is declared by setting the display property of an element to either flex (rendered as a block) or inline-flex (rendered as inline)
- Inside a flex container there is one or more flex items

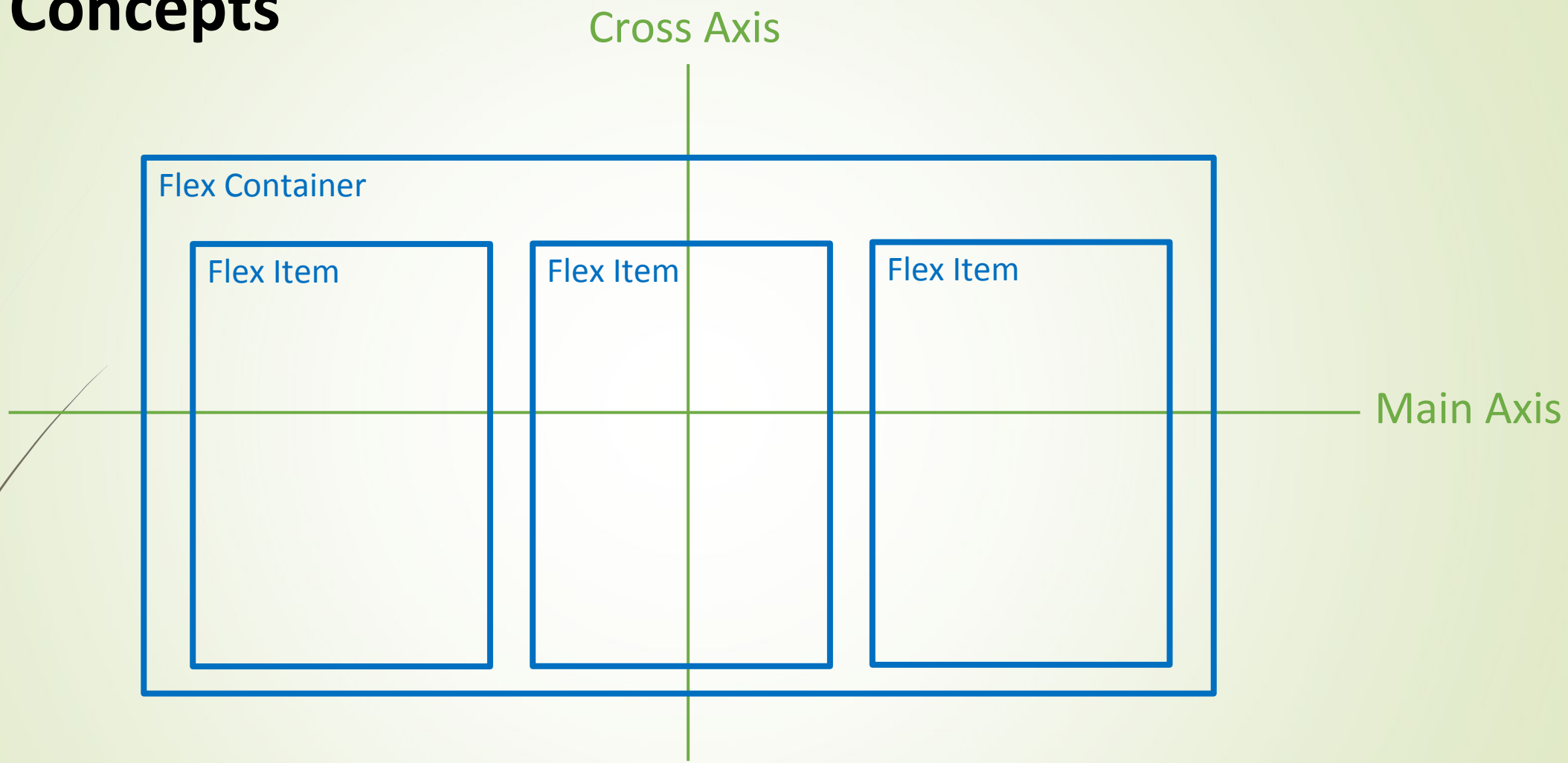
CSS3 Flexbox - Concepts

- Note: Everything outside a flex container and inside a flex item is rendered as usual
- Flexbox defines how flex items are laid out inside a flex container
- Flex items are positioned inside a flex container along a flex line
- By default there is only one flex line per flex container

CSS3 Flexbox - Concepts



CSS3 Flexbox - Concepts



CSS3 Flexbox - Code

Why do we need this?

CSS:

```css
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}

.flex-item {
    background-color: cornflowerblue;
    width: 100px;
    height: 100px;
    margin: 10px;
}
```

You need to set display: -webkit-box on an element to be able to use line-clamp or -webkit-line-clamp with it. Those two properties set the maximum number of lines of text a box can be

CSS3 Flexbox - Code

HTML:

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

CSS3 Flexbox - Code

Result:

CSS3 Flexbox - Code

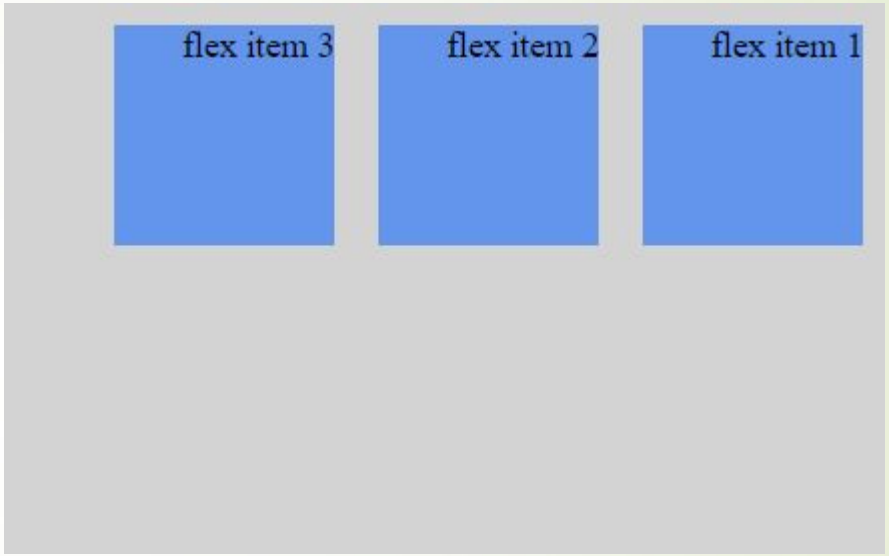
It is also possible to change the direction of the flex line

If we set the direction property to rtl (right-to-left), the text is drawn right to left, and also the flex line changes direction, which will change the page layout

```css
body {
    direction: rtl;
}

.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}

.flex-item {
    background-color: cornflowerblue;
    width: 100px;
    height: 100px;
    margin: 10px;
}
```

CSS3 Flexbox - Code

Result:



CSS3 Flexbox - Flex Direction

- The flex-direction property specifies the direction of the flexible items inside the flex container. The default value of flex-direction is row (left-to-right, top-to-bottom)

- The other values are as follows:
  - row-reverse - If the writing-mode (direction) is left to right, the flex items will be laid out right to left
  - column - If the writing system is horizontal, the flex items will be laid out vertically
  - column-reverse - Same as column, but reversed

CSS3 Flexbox - Flex Direction

Example:

(Row-reverse

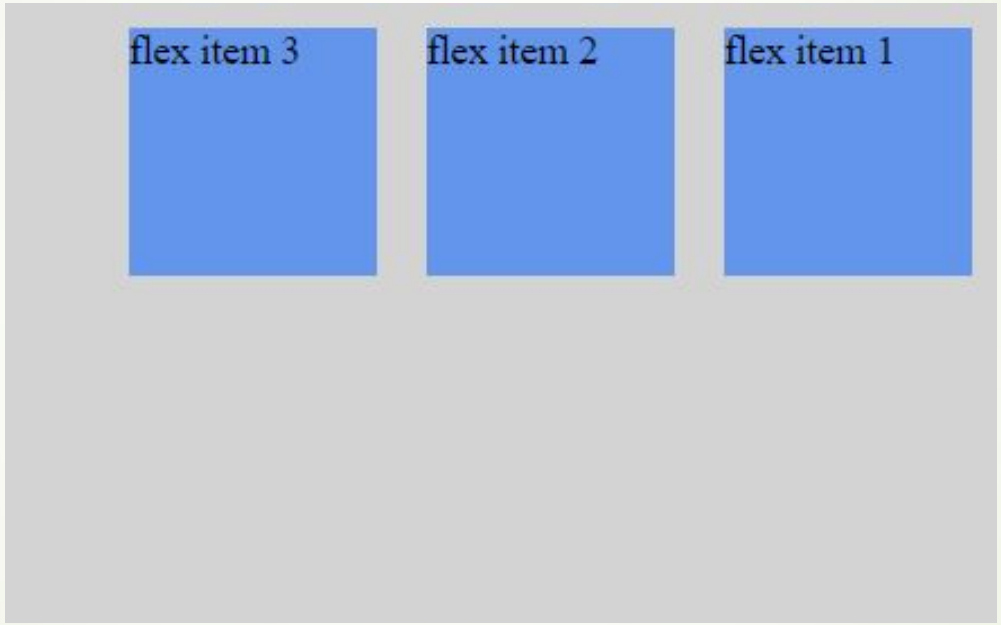
)

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-direction: row-reverse;
    flex-direction: row-reverse;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

CSS3 Flexbox - Flex Direction

Example:



CSS3 Flexbox - Flex Direction

Example:
(Column)

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-direction: column;
    flex-direction: column;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

CSS3 Flexbox - Flex Direction

Example:

CSS3 Flexbox - Flex Direction

Example: (Column-reverse)

```css
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-direction: column-reverse;
    flex-direction: column-reverse;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```html
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

CSS3 Flexbox - Flex Direction

Example:

justify-content

- The justify-content property horizontally aligns the flexible container's items when the items do not use all available space on the main-axis
- The possible values are as follows:
  - flex-start - Default value. Items are positioned at the beginning of the container
  - flex-end - Items are positioned at the end of the container
  - center - Items are positioned at the center of the container
  - space-between - Items are positioned with space between the lines
  - space-around - Items are positioned with space before, between, and after the lines

justify-content Property

Example:
(Flex-end
)

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-justify-content: flex-end;
    justify-content: flex-end;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

justify-content Property

Example (Flex-end) :

justify-content Property

Example: (center)

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-justify-content: center;
    justify-content: center;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

justify-content Property

Example:
(center)

justify-content Property

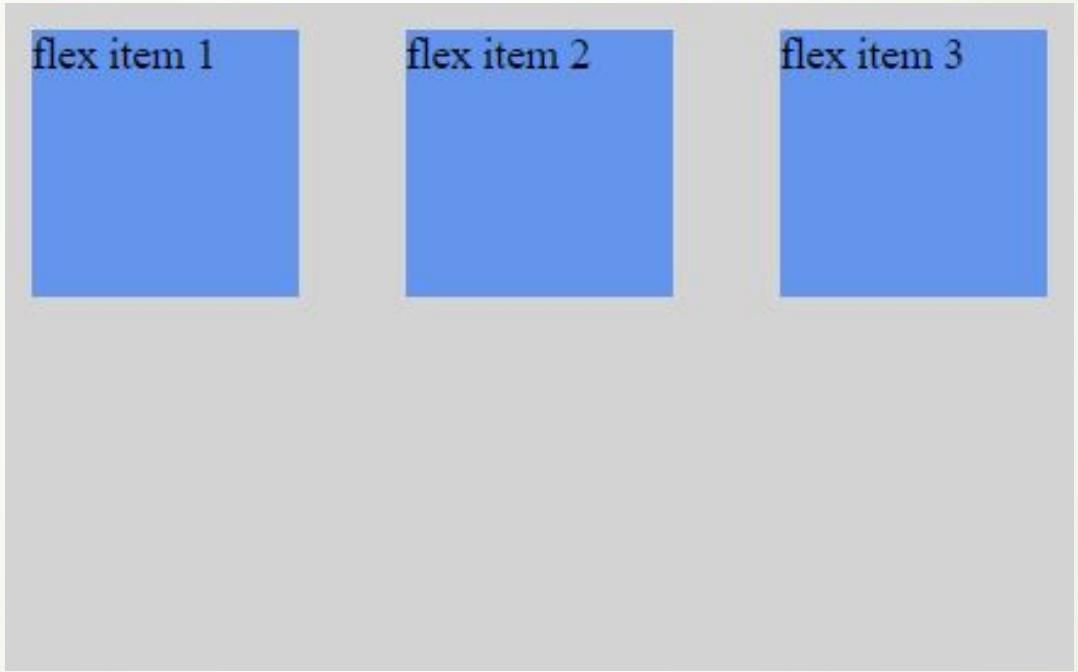
Example: (space-between)

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-justify-content: space-between;
    justify-content: space-between;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

justify-content Property

Example: (space-between)



align-items

- The align-items property vertically aligns the flexible container's items when the items do not use all available space on the cross-axis
- The possible values are as follows:
- stretch - Default value. Items are stretched to fit the container flex-start - Items are positioned at the top of the container flex-end - Items are positioned at the bottom of the container
- center - Items are positioned at the center of the container (vertically)
- baseline - Items are positioned at the baseline of the container

align-items Property

Example: (Stretch)

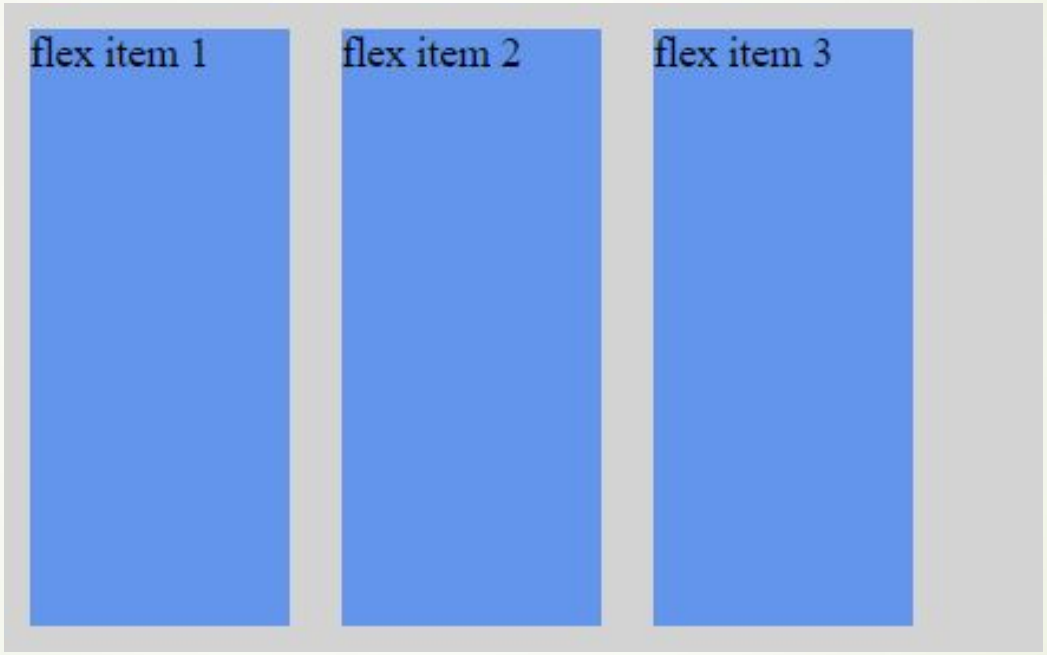
```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-align-items: stretch;
    align-items: stretch;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

`align-items` Property

Example:

(Stretch)



align-items Property

Example:

(Flex-start

)

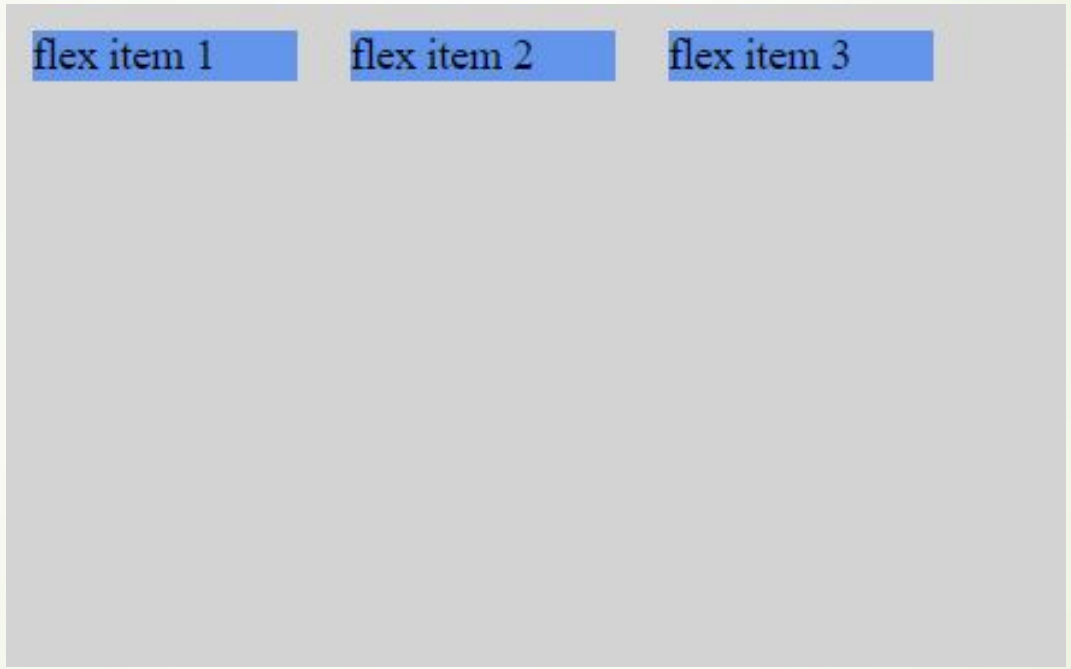
```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-align-items: flex-start;
    align-items: flex-start;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

align-items Property

Example:

(Flex-start)



align-items Property

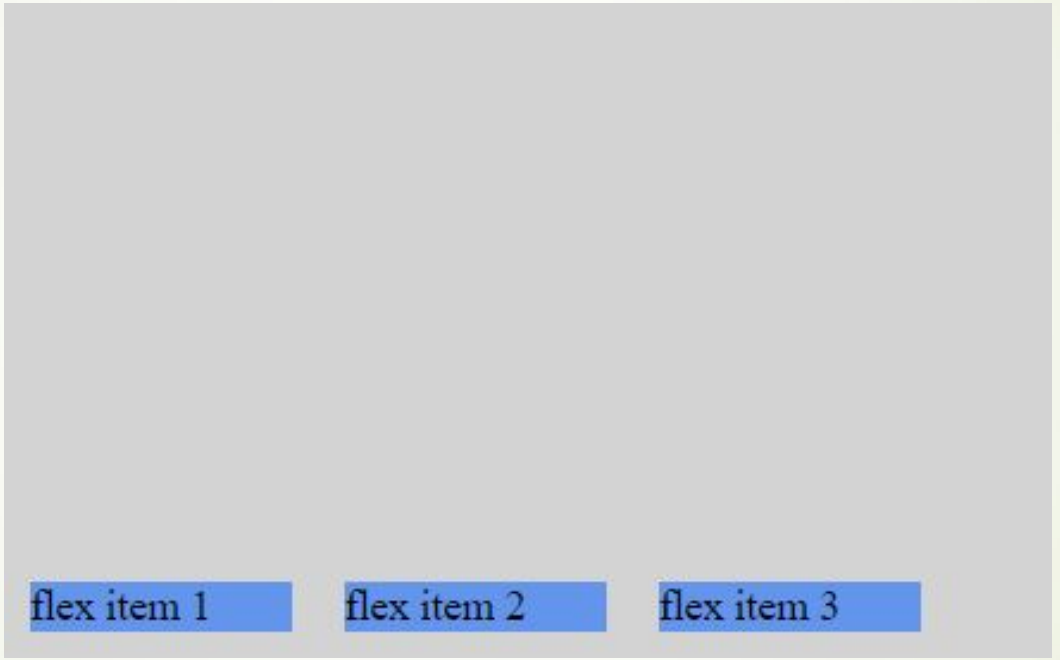
Example: (Flex-end)

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-align-items: flex-end;
    align-items: flex-end;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

`align-items` Property

Example:
(Flex-end
)



align-items Property

Example:

(Center)

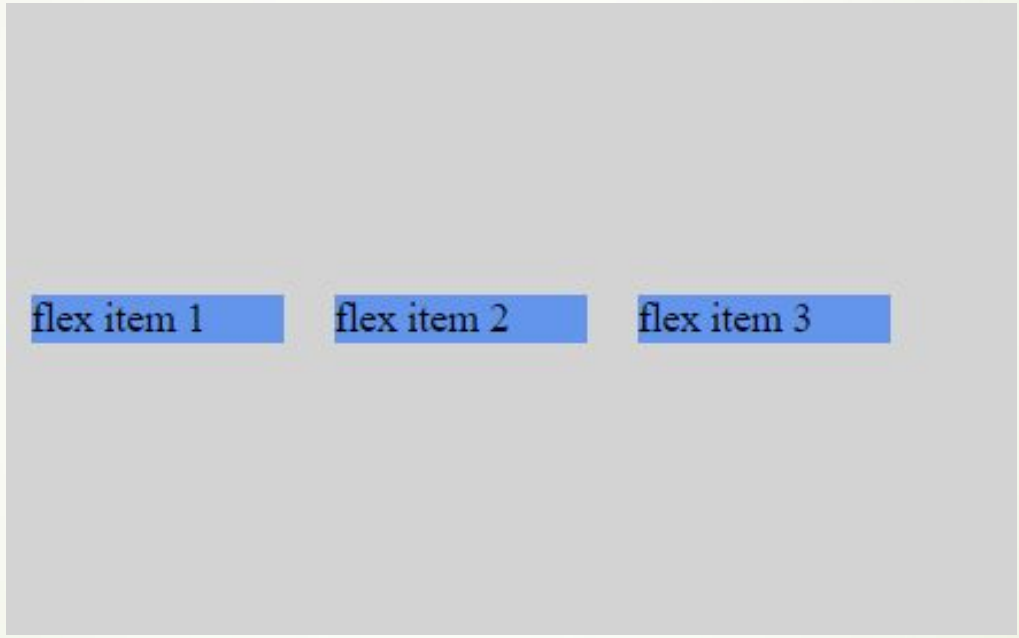
```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-align-items: center;
    align-items: center;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

`align-items` Property

Example:

(Center)



align-items Property

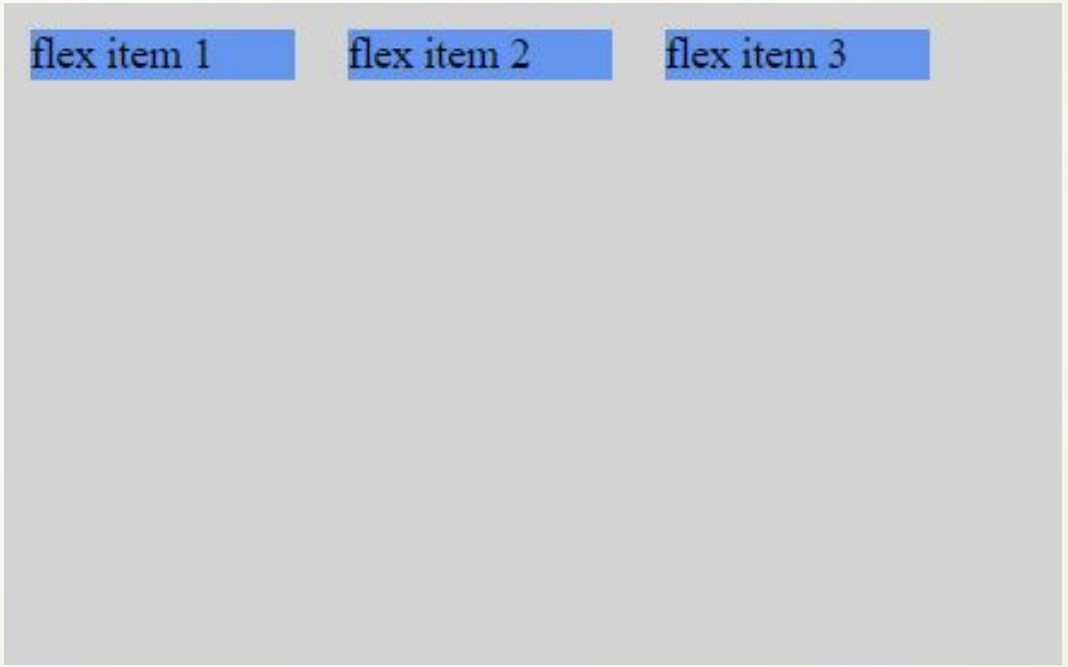
Example:
(Baseline)

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-align-items: baseline;
    align-items: baseline;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

`align-items` Property

Example:
(Baseline)



flex-wrap

- The flex-wrap property specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
- The possible values are as follows:
  - nowrap - Default value. The flexible items will not wrap
  - wrap - The flexible items will wrap if necessary
  - wrap-reverse - The flexible items will wrap, if necessary, in reverse order

flex-wrap Property

Example: (Nowrap)

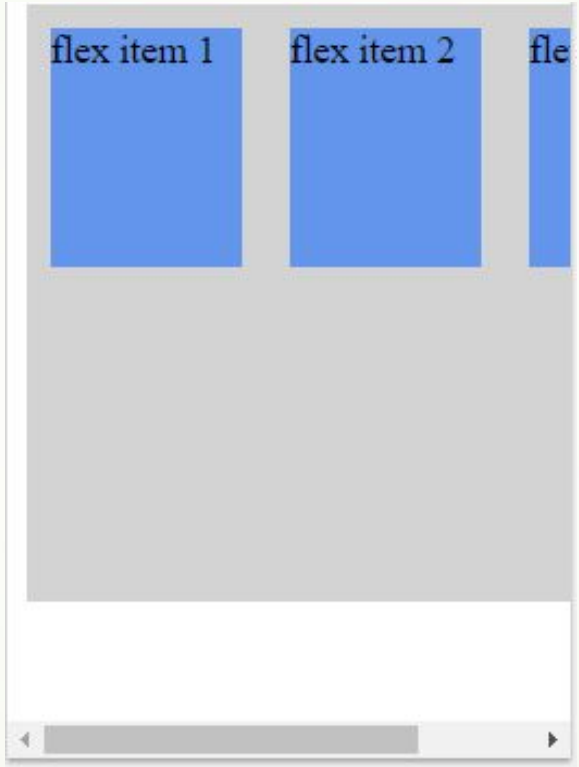
```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: nowrap;
    flex-wrap: nowrap;
    width: 300px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

flex-wrap

Property

Example:
(Nowrap
)



flex-wrap Property

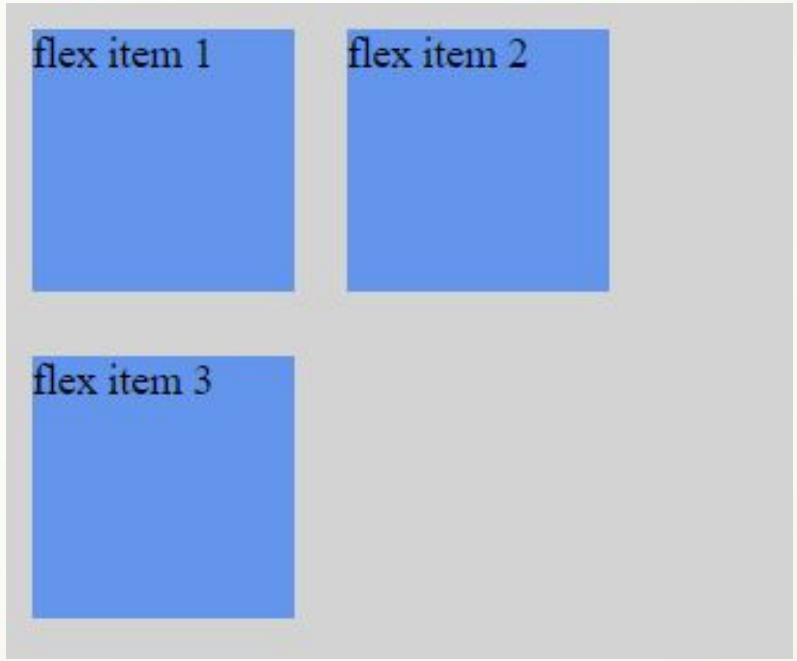
Example: (Wrap)

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: wrap;
    flex-wrap: wrap;
    width: 300px;
    height: 250px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

`flex-wrap` Property

Example:
(wrap)



align-content

- The align-content property modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
- The possible values are as follows:
  - stretch - Default value. Lines stretch to take up the remaining space
  - flex-start - Lines are packed toward the start of the flex container
  - flex-end - Lines are packed toward the end of the flex container

align-content

- The align-content property modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
- The possible values are as follows:
  - center - Lines are packed toward the center of the flex container
  - space-between - Lines are evenly distributed in the flex container
  - space-around - Lines are evenly distributed in the flex container, with half-size spaces on either end

align-content Property

Example:

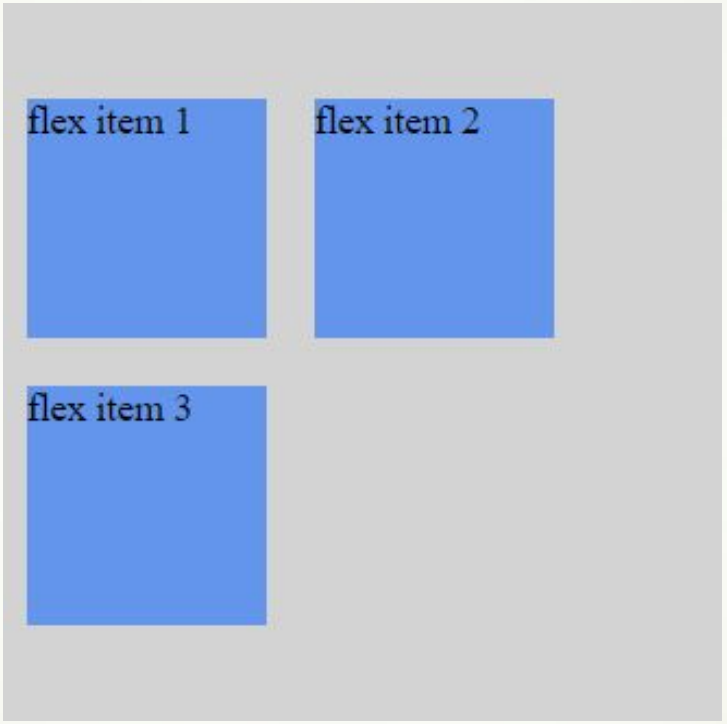
(Center)

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: wrap;
    flex-wrap: wrap;
    -webkit-align-content: center;
    align-content: center;
    width: 300px;
    height: 300px;
    background-color: lightgrey;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

align-content Property

Example:
(Center)



align-content Property

Example: (Flex-end)

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: wrap;
    flex-wrap: wrap;
    -webkit-align-content: flex-end;
    align-content: flex-end;
    width: 300px;
    height: 300px;
    background-color: lightgrey;
}
```

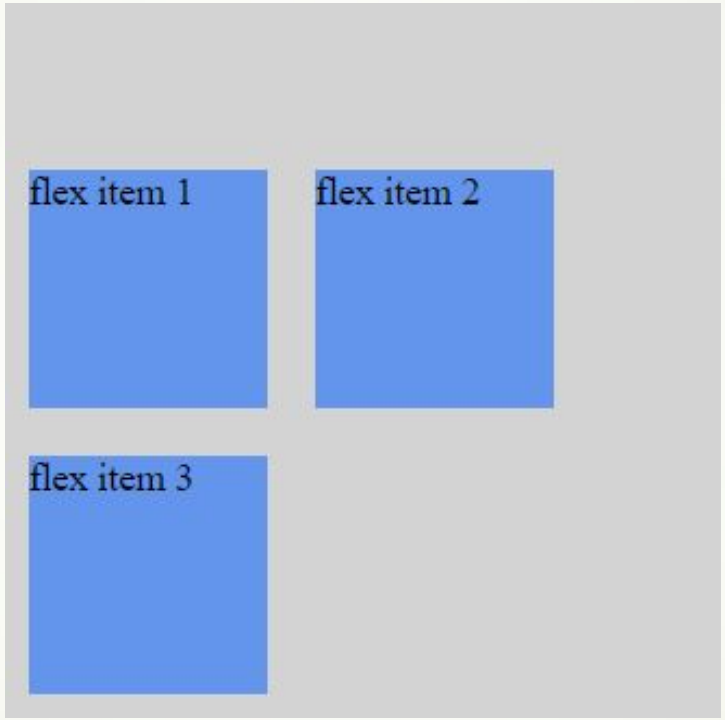
```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

align-content Property

Example:

(Flex-end

)



Flex Item Properties - Ordering

The order property specifies the order of a flexible item relative to the rest of the flexible items inside the same container

```
#main div {
  width: 70px;
  height: 70px;
}

div#myRedDIV   {order: 2;}
div#myBlueDIV  {order: 4;}
div#myGreenDIV {order: 3;}
div#myPinkDIV  {order: 1;}
</style>
</head>
<body>

<h1>The order Property</h1>

<div id="main">
  <div style="background-color:coral;" id="myRedDIV">1</div>
  <div style="background-color:lightblue;" id="myBlueDIV">2</div>
  <div style="background-color:lightgreen;" id="myGreenDIV">3</div>
  <div style="background-color:pink;" id="myPinkDIV">4</div>
</div>
```

The order Property

4 1 3 2

```
  width: 70px;
  height: 70px;
}

div#myRedDIV   {order: -1;}
div#myBlueDIV  {order: 2;}
div#myGreenDIV {order: -2;}
div#myPinkDIV  {order: 1;}
</style>
</head>
<body>

<h1>The order Property</h1>

<div id="main">
  <div style="background-color:coral;" id="myRedDIV">1</div>
  <div style="background-color:lightblue;" id="myBlueDIV">2</div>
  <div style="background-color:lightgreen;" id="myGreenDIV">3</div>
  <div style="background-color:pink;" id="myPinkDIV">4</div>
</div>
```

The order Property

3 1 4 2

Flex Item Properties - Ordering

Example:
(Ordering)

```
.flex-item {
    background-color: cornflowerblue;
    width: 100px;
    height: 100px;
    margin: 10px;
}

.first {
    -webkit-order: -1;
    order: -1;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item first">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

Item Properties - Ordering

Example:
(Ordering)

Flex Item Properties - **Margin**

Setting margin: auto; will absorb extra space

It can be used to push flex items into different positions

Flex Item Properties - Margin

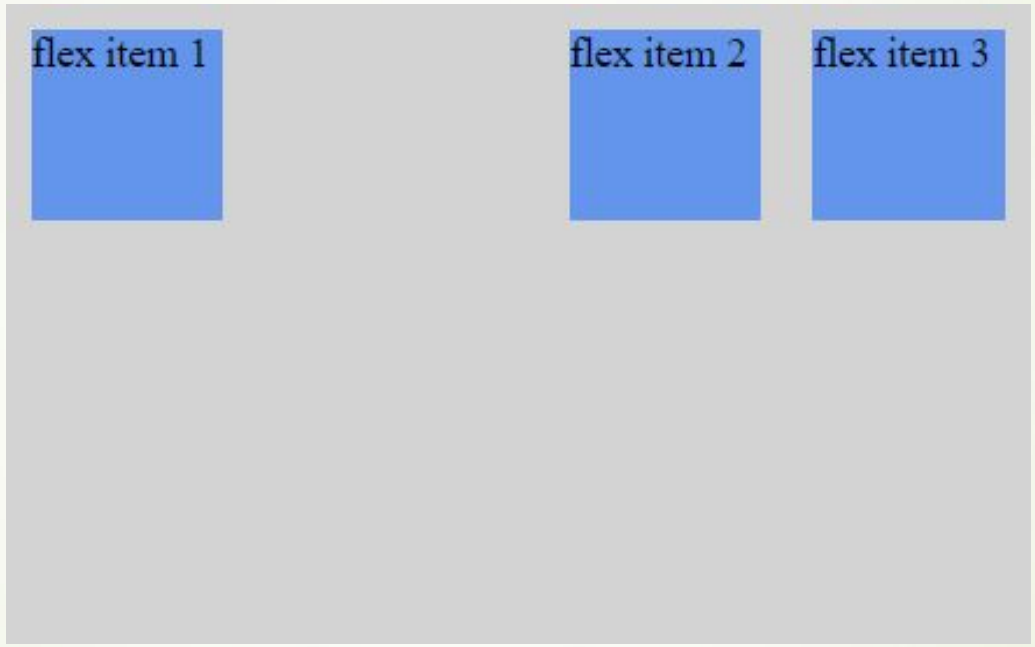
Example:
(Margin)

```
.flex-item {
    background-color: cornflowerblue;
    width: 75px;
    height: 75px;
    margin: 10px;
}

.flex-item:first-child {
    margin-right: auto;
}
```

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>
```

Flex Item Properties - **Margin**

Example:
(Margin)



Flex Item Properties - Perfect Centering

Setting `margin: auto;` will make the item perfectly centered in both axis

Flex Item Properties - **Margin**

Example:
(Margin)

```
.flex-item {
    background-color: cornflowerblue;
    width: 75px;
    height: 75px;
    margin: auto;
}
```

```
<div class="flex-container">
  <div class="flex-item">Perfect centering!</div>
</div>
```

Flex Item Properties - **Margin**

Example:
(Margin)

Flex Item Properties - Align Self

- The align-self property of flex items overrides the flex container's align-items property for that item
- It has the same possible values as the align-items property

Flex Item Properties - Align Self

Example:

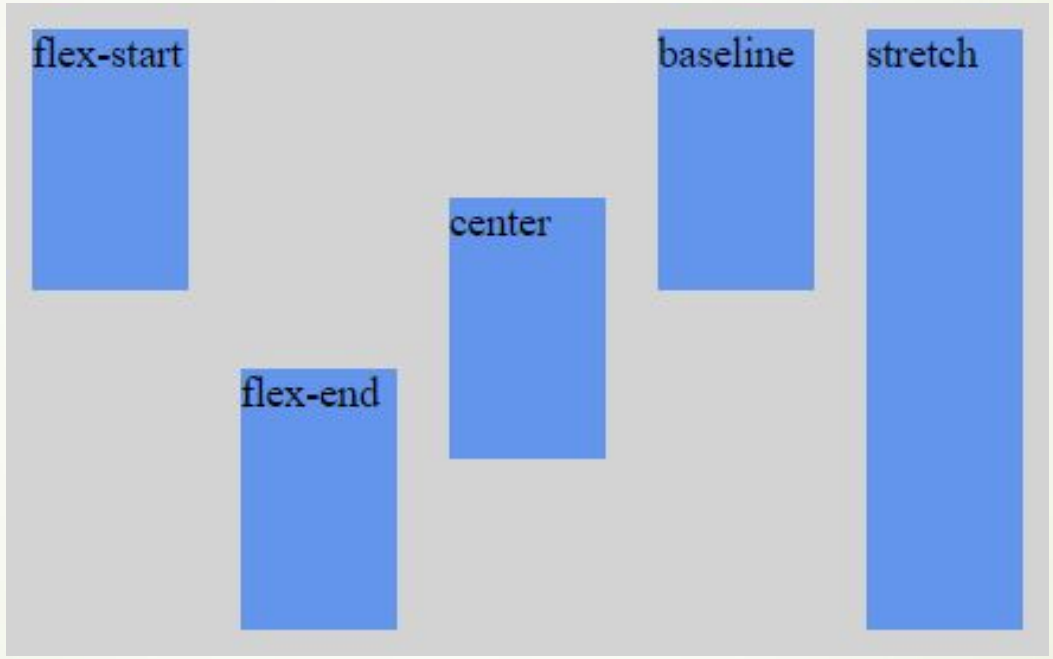
(Align self)

```
<div class="flex-container">
  <div class="flex-item item1">flex-start</div>
  <div class="flex-item item2">flex-end</div>
  <div class="flex-item item3">center</div>
  <div class="flex-item item4">baseline</div>
  <div class="flex-item item5">stretch</div>
</div>
```

```
.flex-item {
    background-color: cornflowerblue;
    width: 60px;
    min-height: 100px;
    margin: 10px;
}

.item1 {
    -webkit-align-self: flex-start;
    align-self: flex-start;
}
.item2 {
    -webkit-align-self: flex-end;
    align-self: flex-end;
}

.item3 {
    -webkit-align-self: center;
    align-self: center;
}

.item4 {
    -webkit-align-self: baseline;
    align-self: baseline;
}

.item5 {
    -webkit-align-self: stretch;
    align-self: stretch;
}
```

Flex Item Properties - Align Self

Example:
(Align self)



Flex Item Properties - **Flex**

The flex property specifies the length of the flex item, relative to the rest of the flex items inside the same container

Flex Item Properties - Flex

Example:

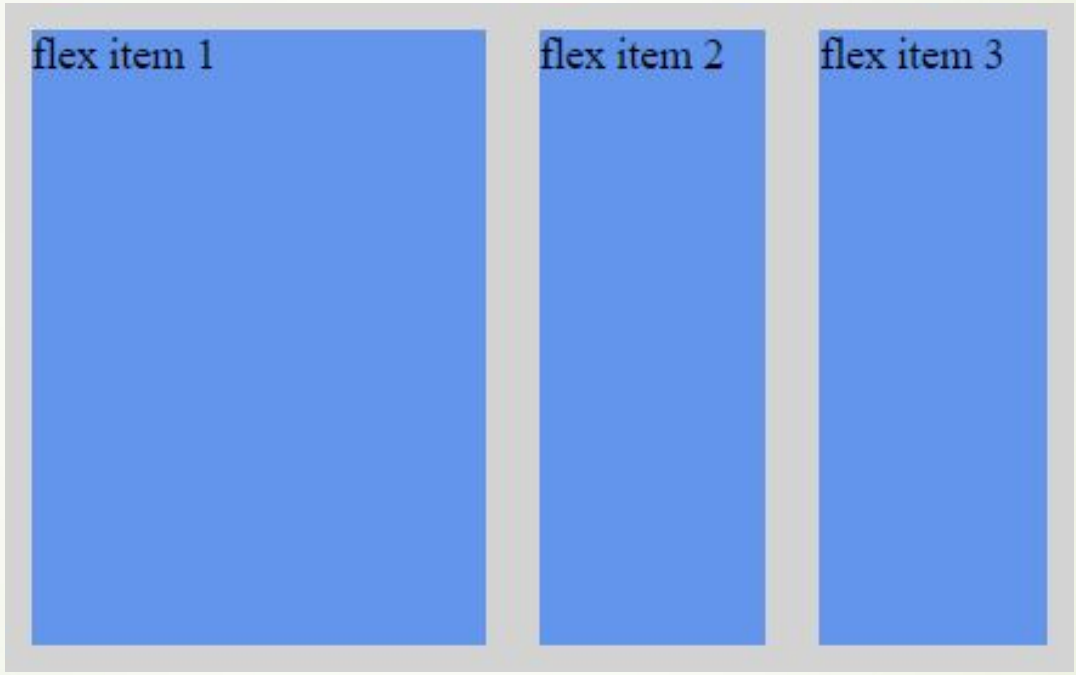
(Flex)

```
<div class="flex-container">
  <div class="flex-item item1">flex item 1</div>
  <div class="flex-item item2">flex item 2</div>
  <div class="flex-item item3">flex item 3</div>
</div>
```

```
.flex-item {
    background-color: cornflowerblue;
    margin: 10px;
}

.item1 {
    -webkit-flex: 2;
    flex: 2;
}

.item2 {
    -webkit-flex: 1;
    flex: 1;
}

.item3 {
    -webkit-flex: 1;
    flex: 1;
}
```

Flex Item Properties - **Flex**

Example:
(Flex)



CSS3 Flexbox Properties

Property	Description
display	Specifies the type of box used for an HTML element
flex-direction	Specifies the direction of the flexible items inside a flex container
justify-content	Horizontally aligns the flex items when the items do not use all available space on the main-axis
align-items	Vertically aligns the flex items when the items do not use all available space on the cross-axis
flex-wrap	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
align-content	Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
flex-flow	A shorthand property for flex-direction and flex-wrap
order	Specifies the order of a flexible item relative to the rest of the flex items inside the same container
align-self	Used on flex items. Overrides the container's align-items property
flex	Specifies the length of a flex item, relative to the rest of the flex items inside the same container

CSS Layout: Flexbox

CSS Layout Techniques

Flexible Box Layout Module - Flexbox

CSS Layout Module – offers greater control over arranging components of web page items **along one axis** (Ex: menu bars, product listings, galleries)

Advantages of Flexbox

- Ability to "flex" – stretch or shrink inside their containers
- Ability to make all neighboring items the same height
- Easy horizontal and vertical centering
- Ability to change the display order of items independent of the html source

*Browser Support – older browsers require prefixes/additional code for flexbox to work properly

CSS Layout - Flexbox

Flexbox Container

You create a flex container by setting the element's **display** pr

Figure 16-2. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

```
.flex-container {
    display: flex;
}
```

/* or */

```
.flex-container {
    display: inline-flex;
}
```

Display: flex creates a block-level flex container

Display: inline-flex creates an inline-level flex container

CSS Layout - Flexbox

Flex container - Flex items

flex applying the flex display mode turns the child elements of the flex container into flex items

Figure 16-2. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

Note: You can turn any flex item into a flex container by setting its **display** to **flex** (resulting in nested flexbox)

CSS Layout - Flexbox

Flex container - Properties

flex container properties allow you to control how items appear within the container

flex-direction

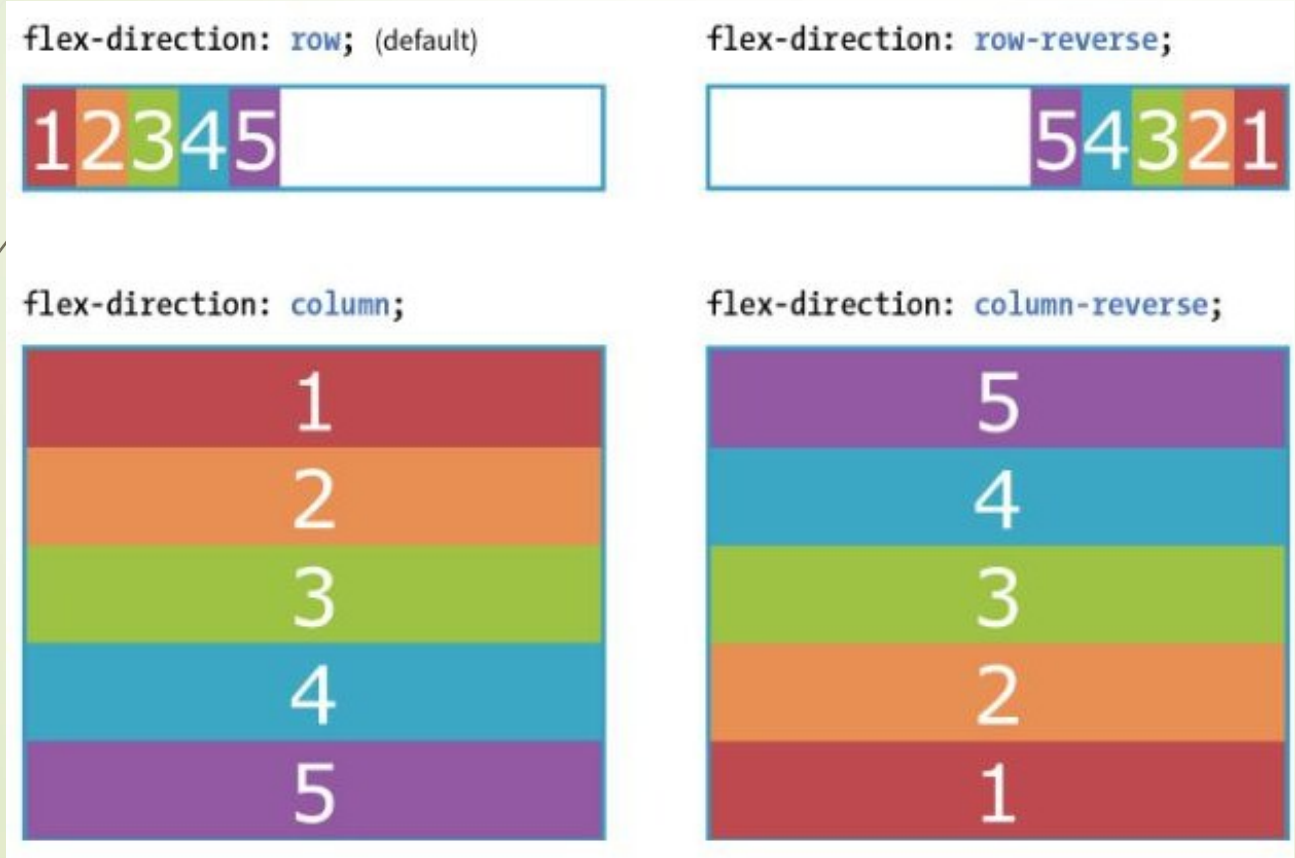


Figure 16-3. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

CSS Layout - Flexbox

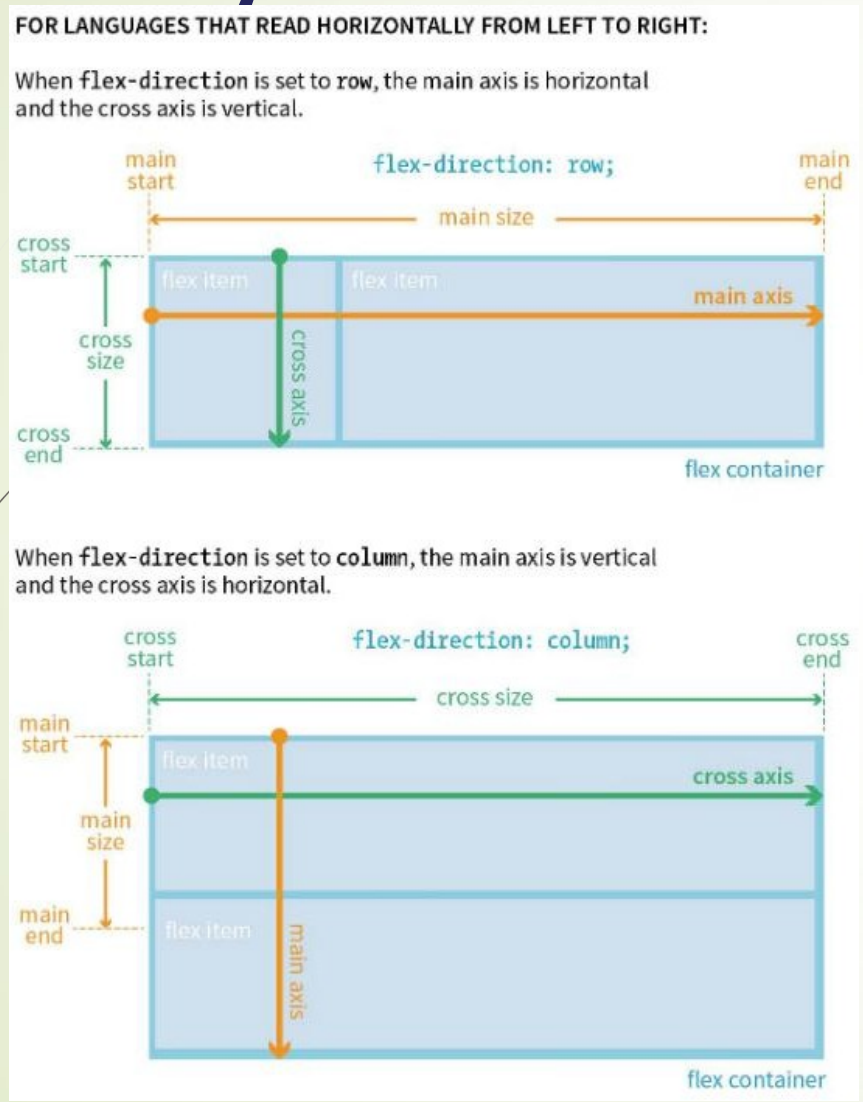


Figure 16-4. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

flex items

flex items line up along one axis

main axis -or- cross axis

The main axis is the flow direction you've specified for the flex container. The cross axis is perpendicular to the main axis

Note: axis direction is specific to the direction of the writing system in use.
For example: In horizontally oriented languages – "row" would align items horizontally
Vertically oriented languages – "row" would align items vertically

CSS Layout - Flexbox

flex-wrap

nowrap | wrap | wrap-reverse

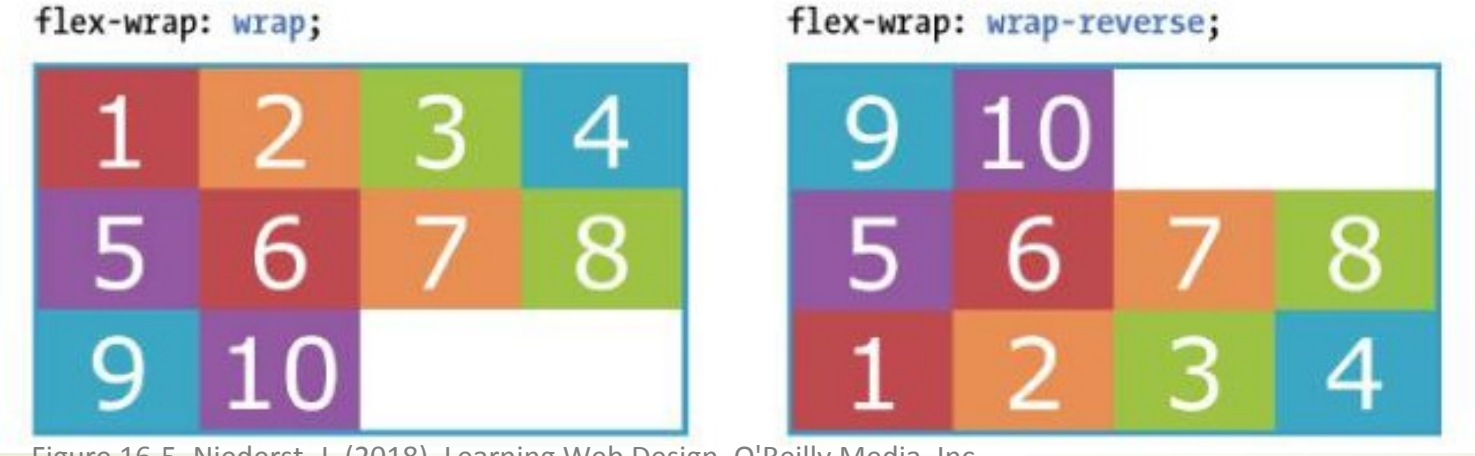


Figure 16-5. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

THE MARKUP

```
<div id="container">
  <div class="box box1">
  1</div>
<!– more boxes here
 <div class="box box10"> 10
</div>
```

THE STYLES

```
#container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}

.box {
  width: 25%;
}
```

CSS Layout - Flexbox

flex-wrap

nowrap | wrap | wrap-reverse

`flex-wrap: nowrap;` (default)

When wrapping is disabled, flex items squish if there is not enough room, and if they can't squish any further, may get cut off if there is not enough room in the viewport.

Figure 16-5. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

CSS Layout - Flexbox

flex-wrap

nowrap | wrap | wrap-reverse

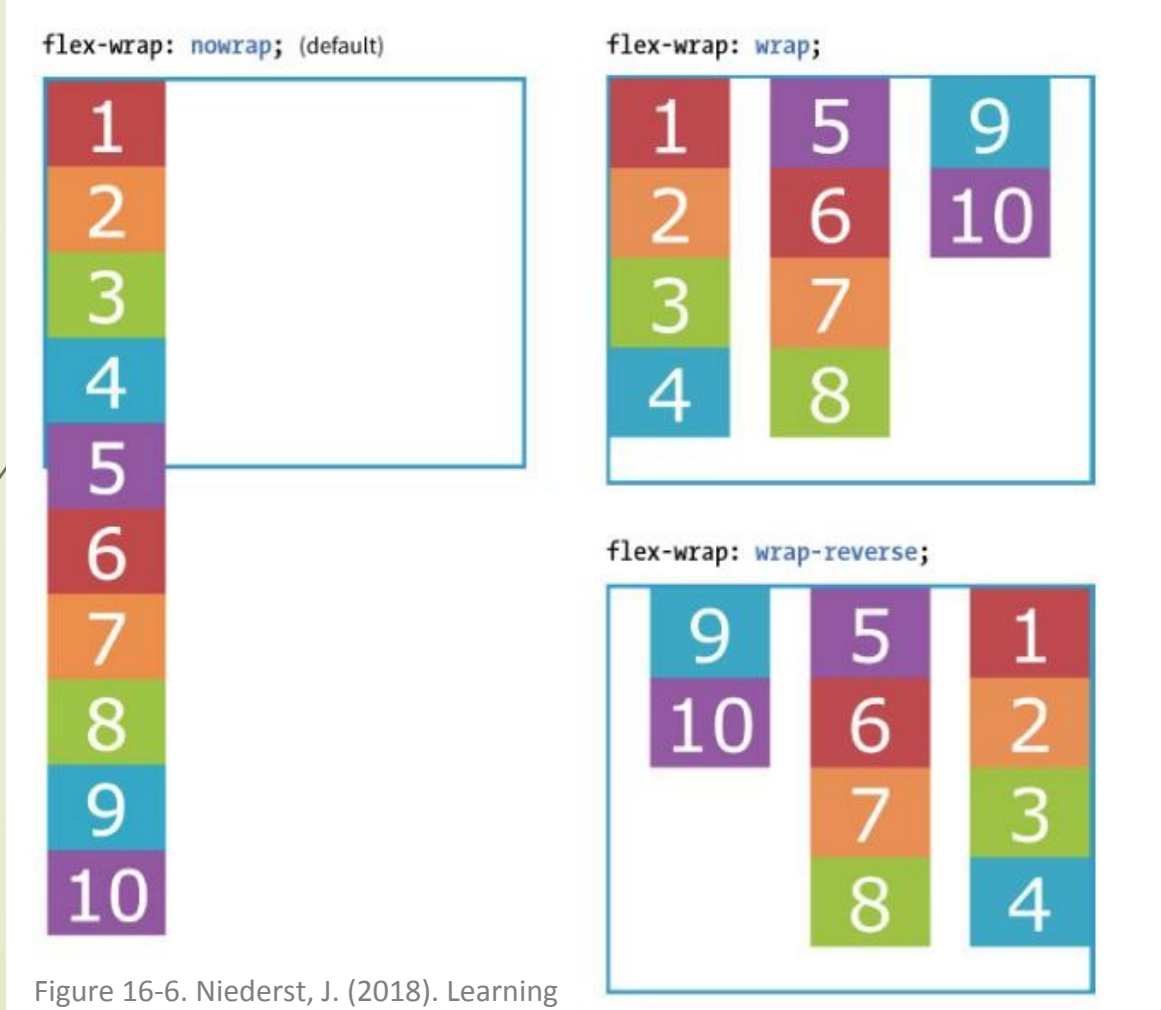


Figure 16-6. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

THE MARKUP

```
<div id="container">
  <div class="box box1"> 1</div>
  <!– more boxes here □
 <div class="box box10"> 10 </div>
</div>
```

THE STYLES

```
#container {
  display: flex;
  height: 350px;
  flex-direction: column;
  flex-wrap: wrap;
}.box {
   width: 25%;
 }
```

CSS Layout - Flexbox

flex-flow (shorthand)

flex-direction flex-wrap

Using flex-direction & flex-wrap

```
#container {
  display: flex;
  height: 350px;
  flex-direction: column;
  flex-wrap: wrap;
}
```

Using **flex-flow**

```
#container {
  display: flex;
  height: 350px;
  flex-flow: column wrap;
}
```

CSS Layout - Flexbox

Flexbox Alignment Properties

justify-content

flex-start | flex-end | center | space-between | space-around

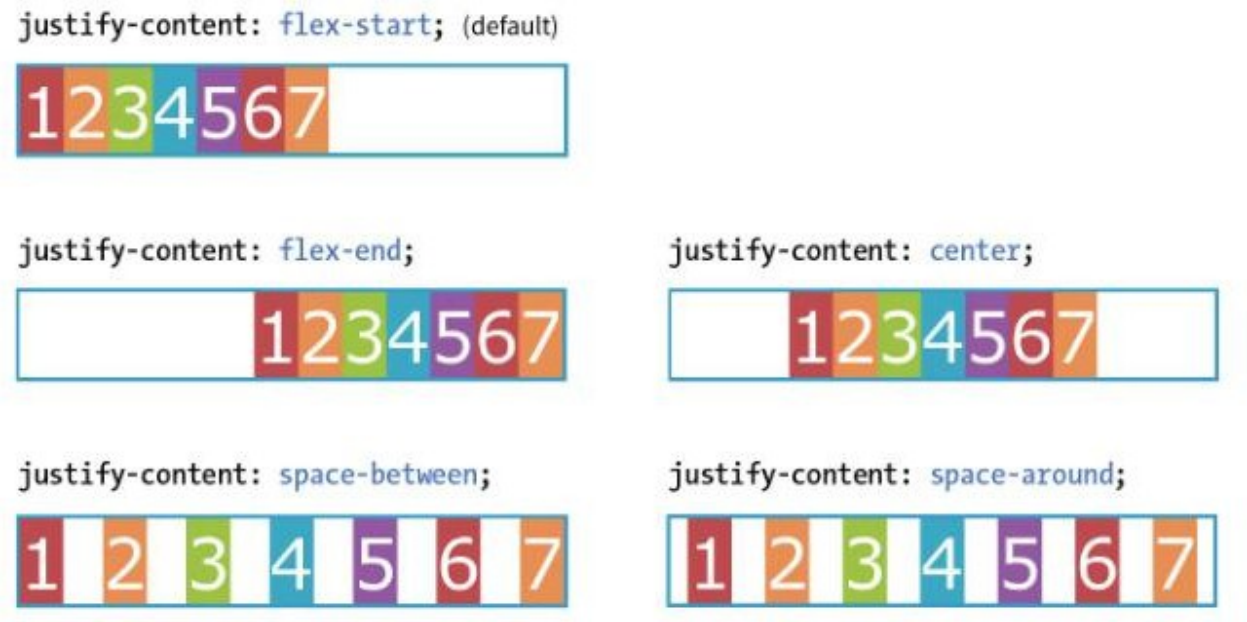


Figure 16-8. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

Flex items are, by default, as wide as they need to be to contain the element's content.

The **justify-content** property defines how extra space is distributed around or between items

CSS Layout - Flexbox

justify-content

flex-start | flex-end | center | space-between | space-around

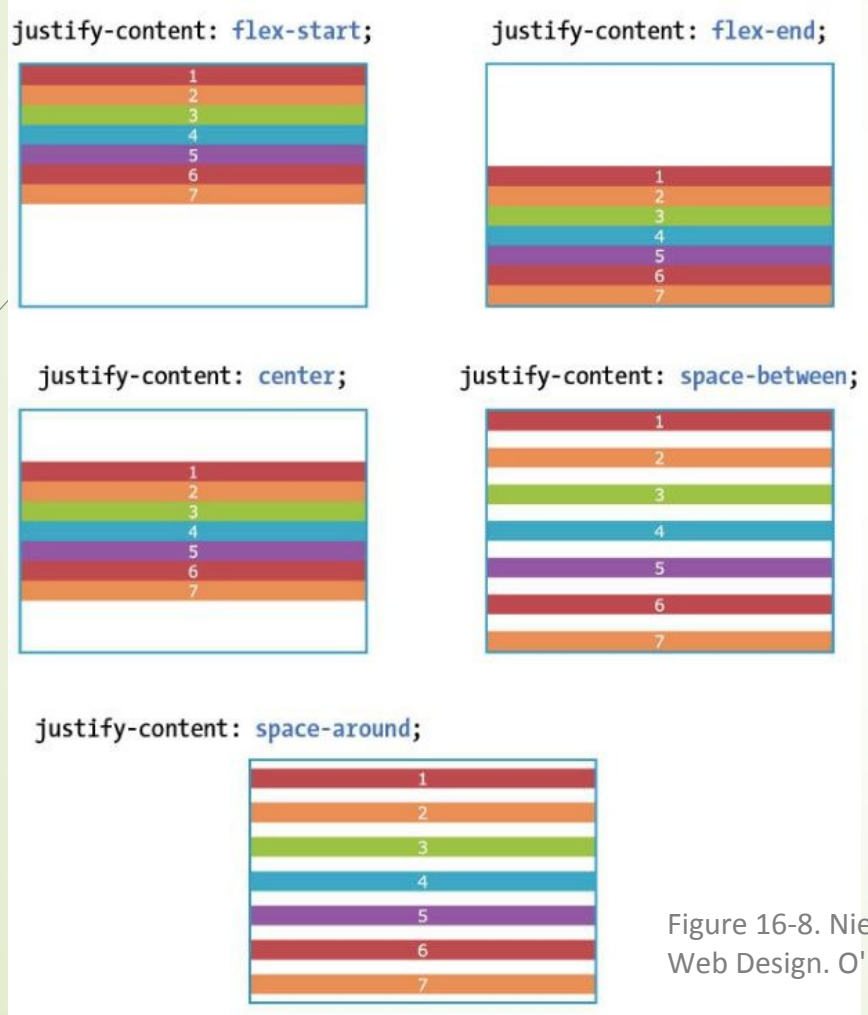


Figure 16-8. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

Flex items are, by default, as wide as they need to be to contain the element's content.

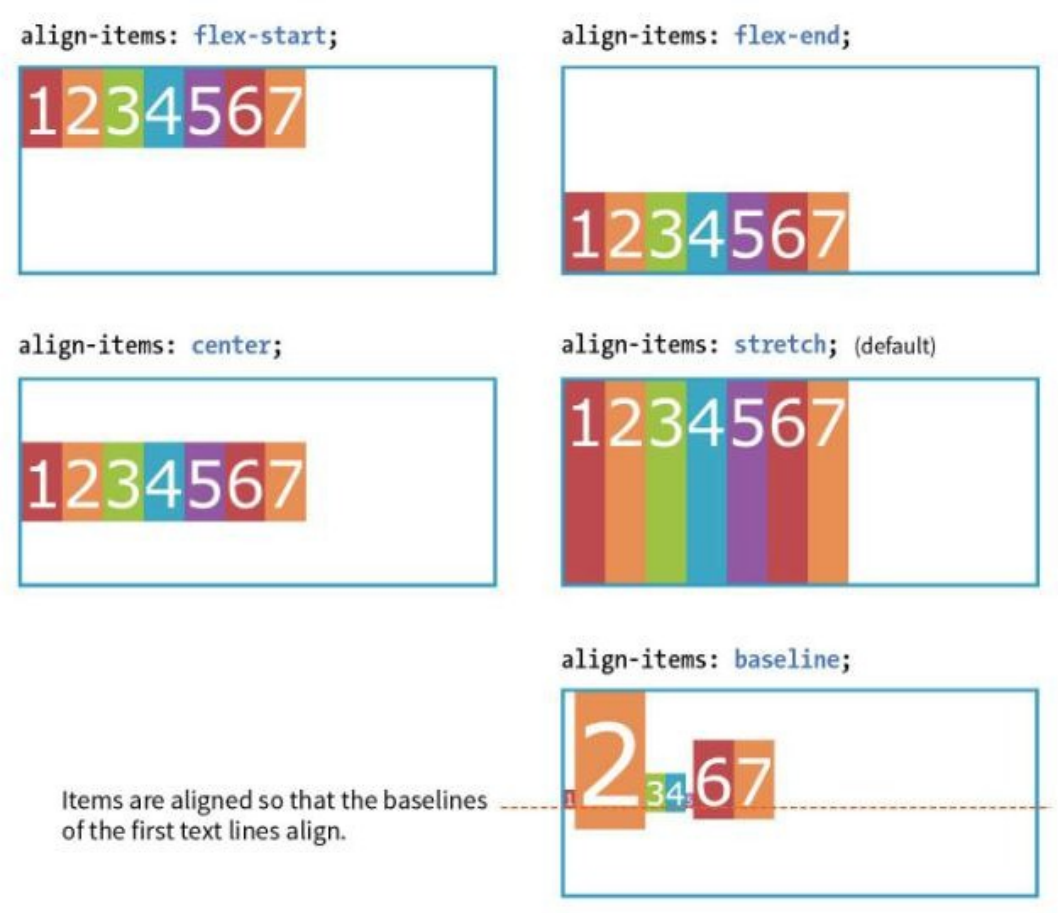
The **justify-content** property defines how extra space is distributed around or between items

CSS Layout - Flexbox

Flexbox Alignment Properties

align-items

flex-start | flex-end | center | baseline | stretch



The **align-items** property allows you to arrange items on the cross axis (up and down when the direction is **row**, left and right if the direction is **column**)

*Note: you must specify a container height

Figure 16-10. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

CSS Layout - Flexbox

Flexbox Alignment Properties

align-content

flex-start | flex-end | center | space-around | space-between | stretch

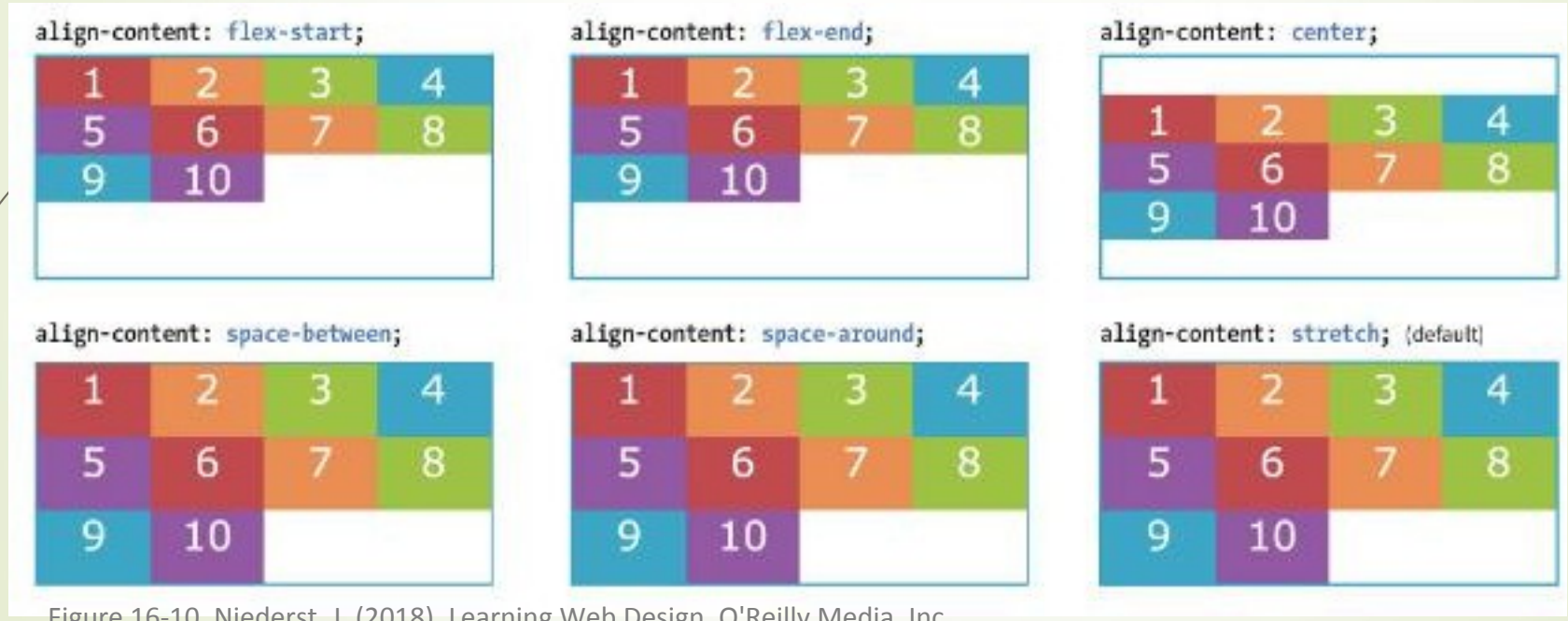


Figure 16-10. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

The **align-content** property applies only when there are multiple wrapped flex lines

CSS Layout - Flexbox

f

F l ex ite m - P r o p e r ti e s

le x i tem pr ope rt ie s de te r m i n e s how space is distributed ***within*** items

flex

None | *'flex-grow flex-shrink flex-basis'*

Example
Flex: *flex-grow flex-shrink flex-basis*

```
li {
  flex: 1 0 200px
}
```

Example

This list item starts at 200px wide is allowed to grow to fill extra space is NOT allowed to shrink below 200px

Values of 1 and 0 work as on/off switch

1 "turns on" or allows an item to grow or shrink

0 "turns off" prevents item from growing or shrinking

CSS Layout - Flexbox

Flex item - Properties

flex-grow

Value: *number* **Default:** *0*

Figure 16-18. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

THE MARKUP

```
<div id="container">
 <div class="box box1">
 1</div>
 <div class="box box2">
 2</div>
 <div class="box box3">
 3</div>
 <div class="box box4">
 4</div>
 <div class="box box5"> 5
 </div>
</div>
```

THE STYLES

```
.box {
 …
 flex: 1 1 auto;
}
```

CSS Layout - Flexbox

Flex item - Properties

flex-grow

Value: *number* **Default:** *0*

Assigning a higher integer to an item – applies more space within that item.

```
.box4 {
   flex: 3 1 auto;
}
```

A flex-grow value of “3” means the item receives three times the amount of space than the remaining items set to flex-grow: 1

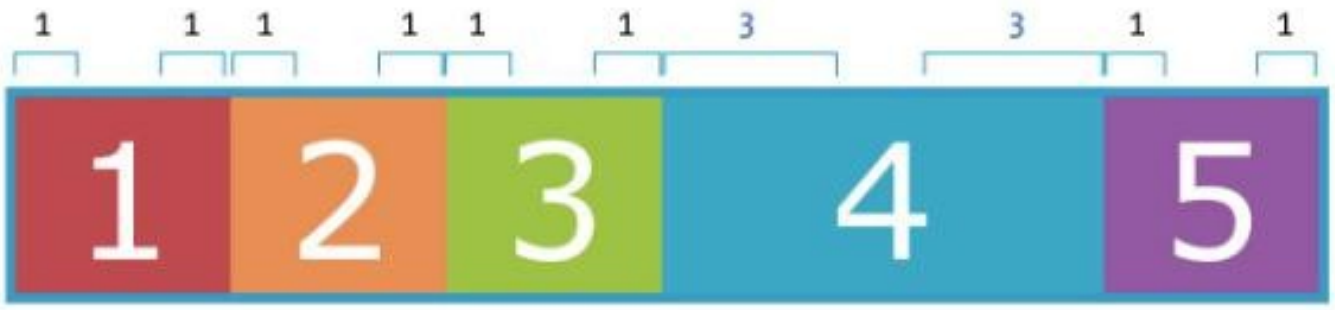


Figure 16-19. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

CSS Layout - Flexbox

Flex item - Properties

flex-shrink

Value: *number* **Default:** *1*

When flex-shrink is 0, items are not permitted to shrink and may hang out of their containing element Flex items stop shrinking when they reach their minimum size (defined by min-width/min-height).

```
box {
  flex: 0 1 100px;
}
```

flex: 0 1 100px;

When the container is wide, the items will not grow wider than their **flex-basis** of 100 pixels because **flex-grow** is set to 0.

When the container is narrow, the items are allowed to shrink to fit (**flex-shrink: 1**).

Figure 16-20. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

CSS Layout - Flexbox

Flex item - Properties

flex-basis

Value: *length* | *percentage* | *content* | *auto*

```
box {
  flex: 0 1 100px;
}
```

flex: 0 1 100px;

When the container is wide, the items will not grow wider than their **flex-basis** of 100 pixels because **flex-grow** is set to 0.

When the container is narrow, the items are allowed to shrink to fit (**flex-shrink: 1**).

Flex-basis defines the starting size of an item before any wrapping, growing or s
occurs. It may be used instead of the width property (or height for columns) for f

By default, flex-basis is set to auto (width/height value of item size)

If item's size is not defined or set to auto – flex-basis uses the content width.

Figure 16-20. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

CSS Layout - Flexbox

Flex item - Properties

order

Value: *integer*

The order property give the ability to display items in an order that is different from the order in the HTML source code.

If items have the same order value = laid out in order they appear in code

If items have different order values = arranged from the lowest order value to highest.

```
.box3 {
  order: 1;
}
```

Figure 16-22. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

CSS Layout - Flexbox

Flex item - Properties

order

Value: *integer*

```
.box2, .box3 {
  order: 1
}
```

Figure 16-23&24. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

```
.box5 {
  order: -1
}
```

CSS Layout - Flexbox

Flex item - Properties

order

Value: *integer*

```
main {
  display: flex;
}
article {
  flex: 1 1 50%;
  order: 2;
}
#news {
  flex: 1 1 25%;
  order: 3;
}
#contact {
  flex: 1 1 25%;
  order: 1;
}
```

Figure 16-25. Niederst, J. (2018). Learning Web Design. O'Reilly Media, Inc.

```
<header>…</header>
<main>
  <article><h2>Where It's At</h2></article>
  <aside id="news"><h2>News></h2></aside>
  <aside id="contact"><h2>Contact</h2><aside>
</main>
<footer>…<footer>
```

Sources

https://www.w3schools.com/css/css3_flexbox.asp

Thank You.