Full Stack Development

Unit - I

Dr. Y. J. Nagendra Kumar

Professor of IT

Dean - Technology and Innovation Cell - GRIET

Table of Contents

01 HTML 5

02 CSS 3

HTML 5

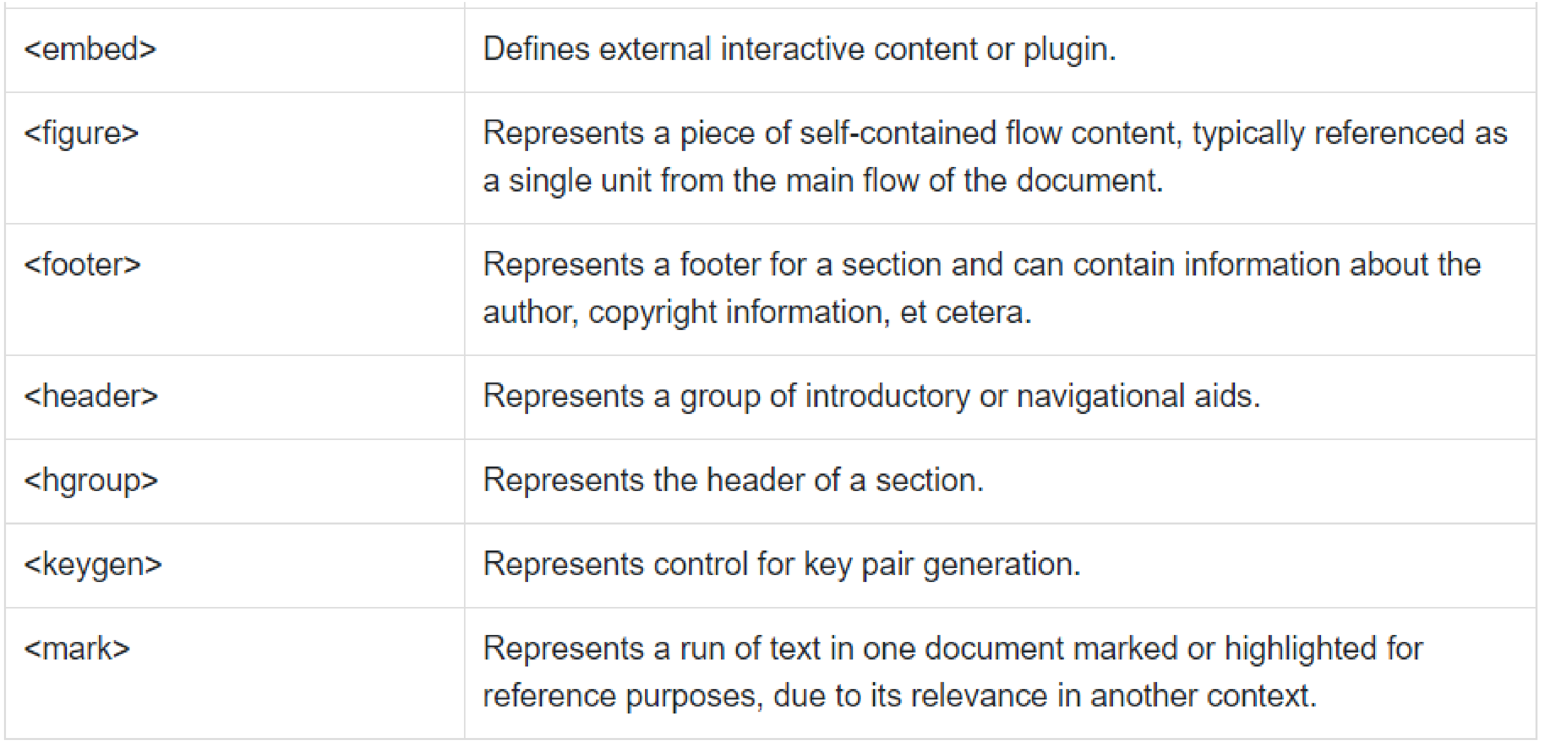
HTML 5 – New Elements

- Full stack development is the end-to-end development of applications. It includes both the front end and back end of an application. The front end is usually accessed by a client, and the back end forms the core of the application where all the business logic is applied.
- In HTML 5, there are lots of new elements are added which provides some extra functionality to create an attractive and dynamic website.
- With the help of these elements, you can make your code easy and quick.

HTML 5 – New Elements

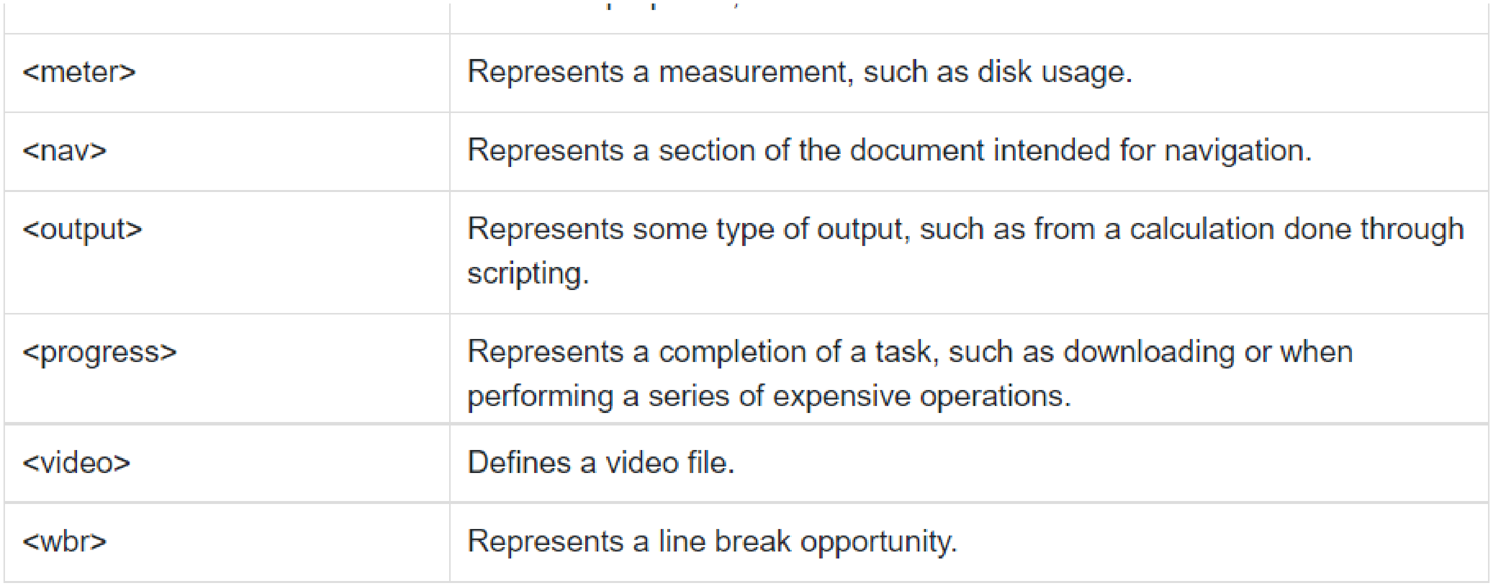
Tags (Elements)	Description
<article>	Represents an independent piece of content of a document, such as a blog entry or newspaper article
<aside >	Represents a piece of content that is only slightly related to the rest of the page.
<audio>	Defines an audio file.
<canvas>	This is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
<datalist>	Together with the a new list attribute for input can be used to make comboboxes
<details>	Represents additional information or controls which the user can obtain on demand

HTML 5



<embed>	Defines external interactive content or plugin.
<figure>	Represents a piece of self-contained flow content, typically referenced as a single unit from the main flow of the document.
<footer>	Represents a footer for a section and can contain information about the author, copyright information, et cetera.
<header>	Represents a group of introductory or navigational aids.
<hgroup>	Represents the header of a section.
<keygen>	Represents control for key pair generation.
<mark>	Represents a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.

HTML 5



<meter>	Represents a measurement, such as disk usage.
<nav>	Represents a section of the document intended for navigation.
<output>	Represents some type of output, such as from a calculation done through scripting.
<progress>	Represents a completion of a task, such as downloading or when performing a series of expensive operations.
<video>	Defines a video file.
<wbr>	Represents a line break opportunity.

HTML 5 – Article Tag

- <article> : Represents an independent piece of content of a document, such as a blog entry or newspaper article.
- A potential source for Article Element are:
  - A magazine/newspaper article
  - A blog entry
  - A forum post
  - A user-submitted a comment
- The <article> element does not render as anything special in a browser. However, you can use CSS to style the <article> element

HTML 5 – Article Tag

```
<html>
<body>

<h1>The article element</h1>

<article>
  <h2>Google Chrome</h2>
  <p>Google Chrome is a web browser developed by
Google, released in 2008. Chrome is the world's
most popular web browser today!</p>
</article>

<h2>Mozilla Firefox</h2>
  <p>Mozilla Firefox is an open-source web browser
developed by Mozilla. Firefox has been the second
most popular web browser since January,
2018.</p>

<article>
  <h2>Microsoft Edge</h2>
  <p>Microsoft Edge is a web browser
developed by Microsoft, released in 2015.
Microsoft Edge replaced Internet
Explorer.</p>
</article>

</body>
</html>
```

HTML 5 – Aside Tag

- <aside> tag defines some content aside from the content it is placed in.
- The aside content should be indirectly related to the surrounding content.
- The <aside> content is often placed as a sidebar in a document.
- The <aside> element does not render as anything special in a browser. However, you can use CSS to style the <aside> element

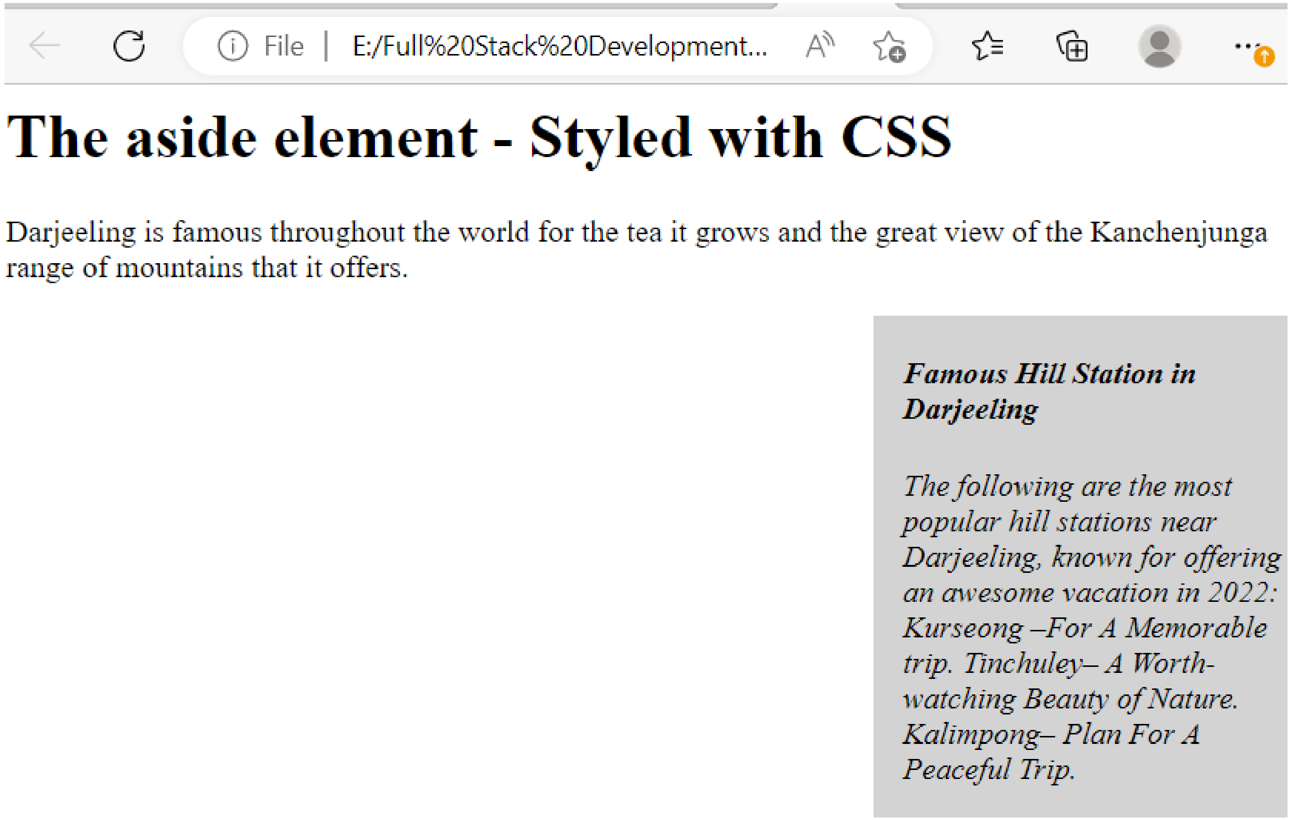
HTML 5 - Aside Tag

```
<html>
<body>

<h1>The aside element</h1>

<p>Darjeeling is famous throughout the world for the tea it grows and the great view of the Kanchenjunga
range of mountains that it offers. </p>

<aside>
  <h4>Famous Hill Station in Darjeeling</h4>
  <p>The following are the most popular hill stations near Darjeeling, known for offering an awesome
vacation in 2022: Kurseong –For A Memorable trip. Tinchuley– A Worth-watching Beauty of Nature.
Kalimpong– Plan For A Peaceful Trip.</p>
</aside>

</body>
</html>
```

HTML 5 – Aside with CSS

```
<html> <head>
<style>
aside {
  width: 30%;
  padding-left: 15px;
  margin-left: 15px;
  float: right;
  font-style: italic;
  background-color: lightgray;
}
</style> </head>
<body>
<h1>The aside element - Styled with CSS</h1>

<p>Darjeeling is famous throughout the world for
the tea it grows and the great view of the
Kanchenjunga range of mountains that it offers.
</p>

<aside>
  <h4>Famous Hill Station in Darjeeling</h4>
  <p>The following are the most popular hill
stations near Darjeeling, known for offering an
awesome vacation in 2022: Kurseong –For A
Memorable trip. Tinchuley– A Worth-watching
Beauty of Nature. Kalimpong– Plan For A Peaceful
Trip.</p>
</aside>

</body>
</html>
```

HTML 5 – Aside with CSS



The aside element - Styled with CSS

Darjeeling is famous throughout the world for the tea it grows and the great view of the Kanchenjunga range of mountains that it offers.

Famous Hill Station in Darjeeling

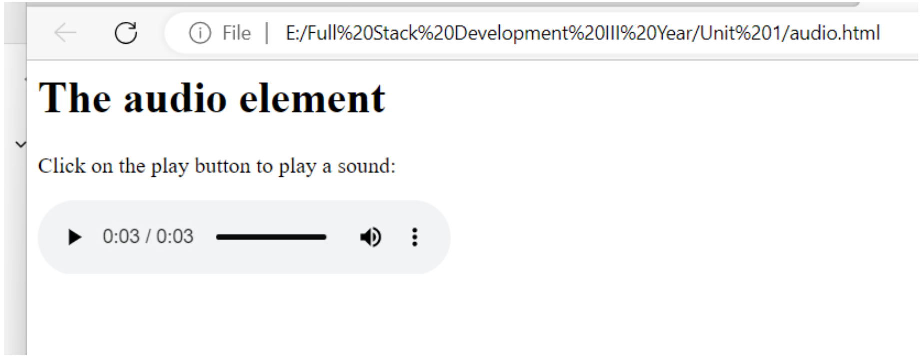
The following are the most popular hill stations near Darjeeling, known for offering an awesome vacation in 2022: Kurseong –For A Memorable trip. Tinchuley– A Worth-watching Beauty of Nature. Kalimpong– Plan For A Peaceful Trip.

HTML 5 – Audio Tag

- The <audio> tag is used to embed sound content in a document, such as music or other audio streams.
- The <audio> tag contains one or more <source> tags with different audio sources. The browser will choose the first source it supports.
- The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.
- There are three supported audio formats in HTML: MP3, WAV, and OGG.

HTML 5 – Audio Tag

```
<html>
<body>

<h1>The audio element</h1>

<p>Click on the play button to play a sound:</p>

<audio controls>
  <source src="Test_MP3.mp3" type="audio/mpeg">

  Your browser does not support the audio element.
</audio>

</body>
</html>
```


File | E:/Full%20Stack%20Development%20III%20Year/Unit%201/audio.html

The audio element

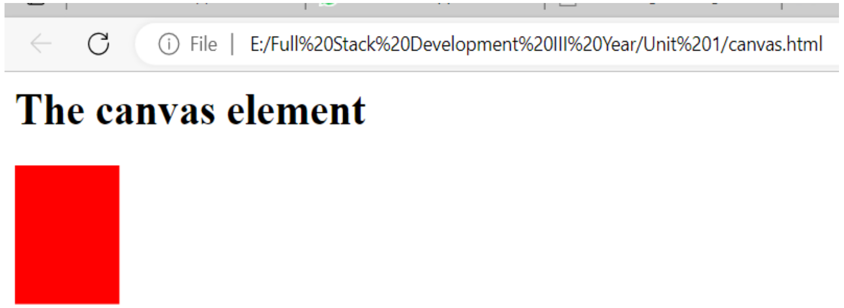
Click on the play button to play a sound:

HTML 5 – Canvas Tag

- The <canvas> tag is used to draw graphics, on the fly, via scripting (usually JavaScript).
- The <canvas> tag is transparent, and is only a container for graphics, you must use a script to actually draw the graphics.

HTML 5 – Canvas Tag

```
<html>
<body>

<h1>The canvas element</h1>

<canvas id="myCanvas">Canvas Tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0, 0, 80, 100);
</script>

</body>
</html>
```


File
E:/Full%20Stack%20Development%20III%20Year/Unit%201/canvas.html

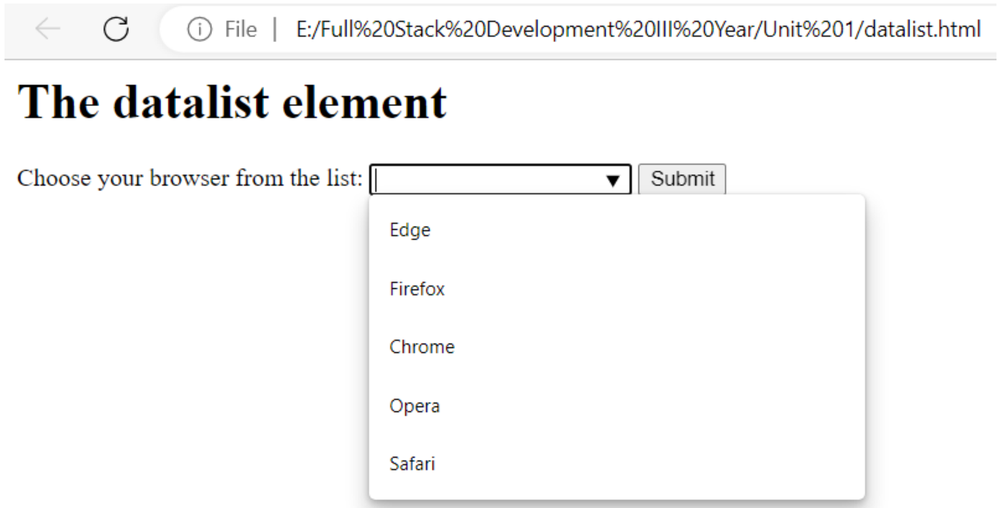
The canvas element

HTML 5 – Data List

- The <datalist> tag specifies a list of pre-defined options for an <input> element.
- The <datalist> tag is used to provide an "autocomplete" feature for <input> elements.
- Users will see a drop-down list of pre-defined options as they input data.
- The <datalist> element's id attribute must be equal to the <input> element's list attribute

HTML 5 – DataList Tag

```
<html>
<body>
<h1>The datalist element</h1>
<form action="/action_page.php" method="get">
 <label for="browser">Choose your browser from the list:</label>
 <input list="browsers" name="browser" id="browser">

 <datalist id="browsers">
  <option value="Edge">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
 </datalist>
 <input type="submit">
</form>

</body>
</html>
```


File
E:/Full%20Stack%20Development%20III%20Year/Unit%201/datalist.html

The datalist element

Choose your browser from the list: [▼] Submit

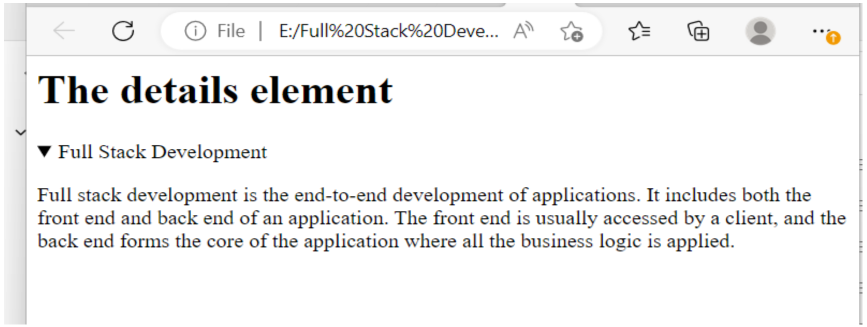
Edge
Firefox
Chrome
Opera
Safari

HTML 5 – details tag

- The <details> tag specifies additional details that the user can open and close on demand.
- The <details> tag is often used to create an interactive widget that the user can open and close. By default, the widget is closed. When open, it expands, and displays the content within.
- Any sort of content can be put inside the <details> tag.

HTML 5 – details tag

```
<html>
<body>

<h1>The details element</h1>

<details>
  <summary>Full Stack Development</summary>
  <p>Full stack development is the end-to-end development of applications. It includes both the front end and back end of an application. The front end is usually accessed by a client, and the back end forms the core of the application where all the business logic is applied.</p>
</details>

</body>
</html>
```



The details element

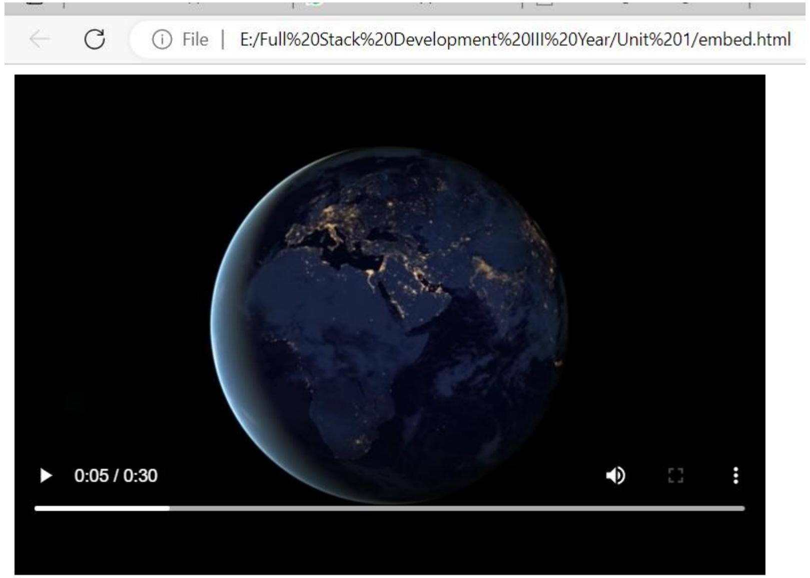
▼ Full Stack Development

Full stack development is the end-to-end development of applications. It includes both the front end and back end of an application. The front end is usually accessed by a client, and the back end forms the core of the application where all the business logic is applied.

HTML 5 - embed

- The <embed> tag defines a container for an external resource, such as a web page, a picture, a media player, or a plug-in application.

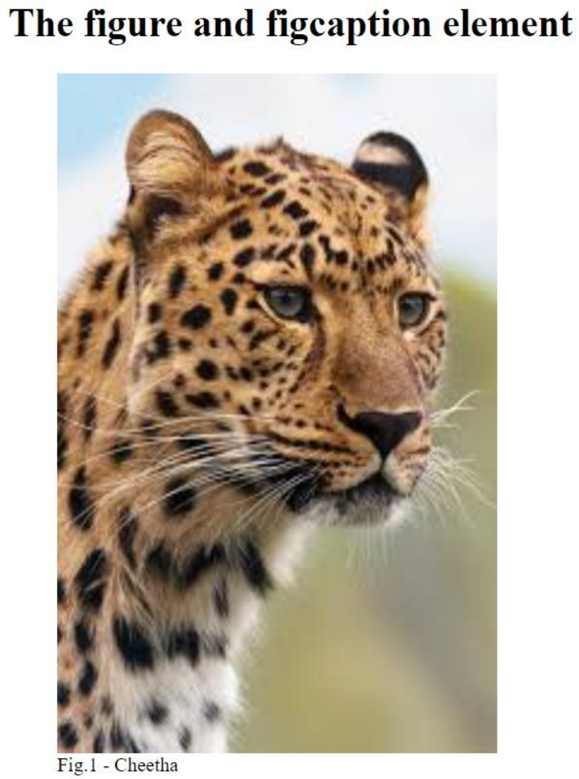
```
<html>
<body>


<embed type="video/webm"
src="Movie.mp4" width="600"
height="400">


</body>
</html>
```


File
E:/Full%20Stack%20Development%20III%20Year/Unit%201/embed.html
0:05 / 0:30

HTML 5 - Figure

- The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
- The <figcaption> element is used to add a caption for the <figure> element.

```
<html> <body>
<h1>The figure and figcaption element</h1>
<figure>
  <img src="Image.jfif" alt="Trulli"
style="width:25%">
  <figcaption>Fig.1 - Cheetha</figcaption>
</figure>
</body> </html>
```



The figure and figcaption element

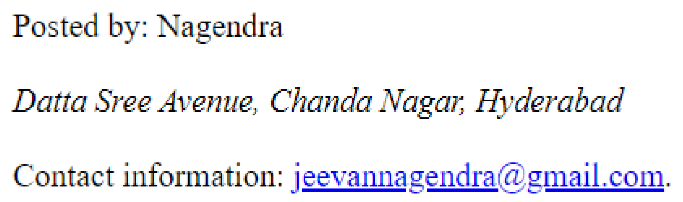
Fig.1 - Cheetha

HTML 5 – Footer

- The <footer> tag defines a footer for a document or section.
- A <footer> element typically contains:
    - authorship information
    - copyright information
    - contact information
    - sitemap
    - back to top links
    - related documents

HTML 5 - Footer

```
<html>
<body>
<footer>
<p>Posted by: Nagendra</p>
<p>
<address> Datta Sree Avenue, Chanda Nagar,
Hyderabad
</address>
</p>
<p>Contact information:
<a
href="mailto:jeevannagendra@gmail.com">jeevann
agendra@gmail.com</a>.
</p>
</footer>
</body>
</html>
```



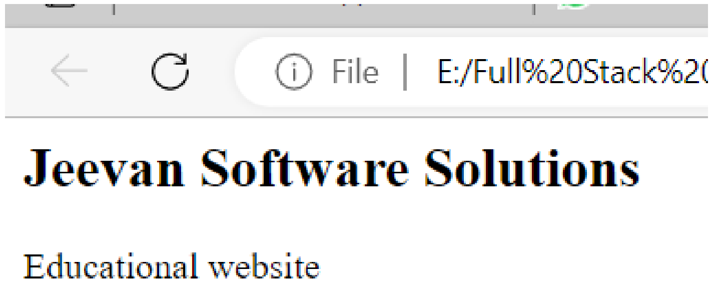
```
Posted by: Nagendra

Datta Sree Avenue, Chanda Nagar, Hyderabad

Contact information: jeevannagendra@gmail.com.
```

HTML 5 - Header

- The <header> element represents a container for introductory content or a set of navigational links.
- A <header> element typically contains:
    - one or more heading elements (<h1> - <h6>)
    - logo or icon
    - authorship information

HTML 5 - Header

```
<html>

<body>

<header>

<h2>Jeevan Software Solutions</h2>

<p> Educational website</p>

</header>

</body>

</html>
```

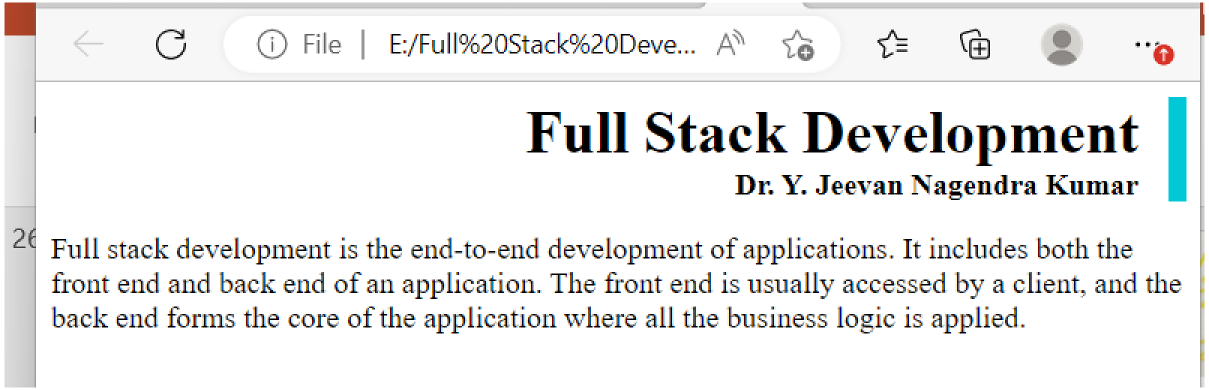


Jeevan Software Solutions

Educational website

HTML 5 - hgroup

- The hgroup element represents the heading of a section.
- The element is used to group a set of h1 – h6 elements when the heading has multiple levels, such as subheadings, alternative titles, or taglines.



Full Stack Development

Dr. Y. Jeevan Nagendra Kumar

Full stack development is the end-to-end development of applications. It includes both the front end and back end of an application. The front end is usually accessed by a client, and the back end forms the core of the application where all the business logic is applied.

HTML 5 - hgroup

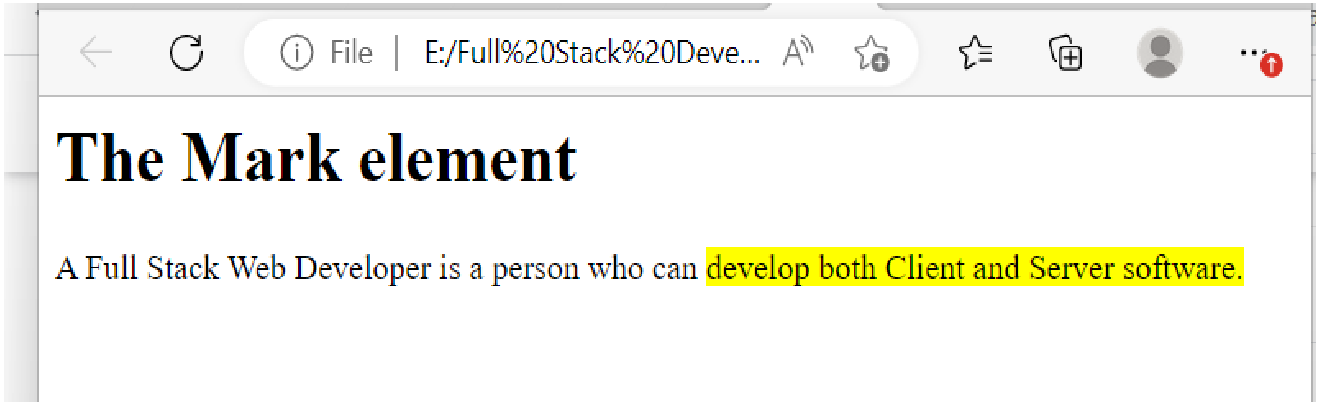
```
<html>
 <head>    <title>hgroup</title>
  <style>
          hgroup
          {           text-align: right;
                      padding-right: 16px;
          border-right: 10px solid #00c8d7;
          }
          hgroup h1
          {           margin-bottom: 0;
          }
          hgroup p
          {           margin: 0;
                      font-weight: bold;

          }
  </style>
 </head>
```

```
<body>
  <hgroup>
   <h1>Full Stack Development</h1>
   <p>Dr. Y. Jeevan Nagendra Kumar <p>
  </hgroup>

  <p>
   Full stack development is the end-to-end
development of applications. It includes both the
front end and back end of an application. The
front end is usually accessed by a client, and the
back end forms the core of the application where
all the business logic is applied.
  </p>
 </body>
</html>
```

HTML 5 - Mark

- The <mark> tag defines text that should be marked or highlighted.

```
<html> <body>
<h1>The Mark element</h1>
<p> A Full Stack Web Developer is a person who can <mark>develop both Client and Server software </mark> </p>
</body> </html>
```



The Mark element

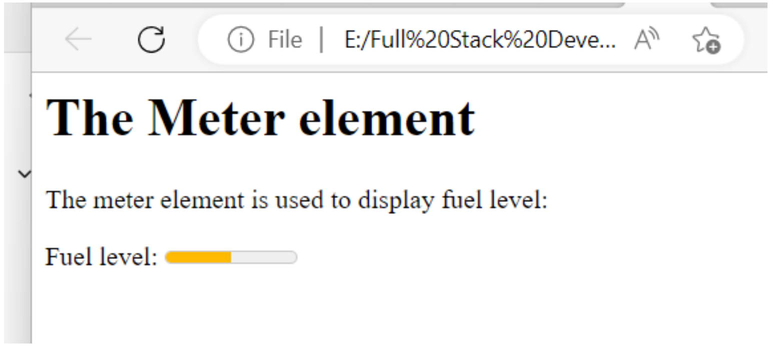
A Full Stack Web Developer is a person who can develop both Client and Server software.

HTML 5 - Meter

- The <meter> tag defines a scalar measurement within a known range, or a fractional value. This is also known as a gauge.
- Examples: Disk usage, Fuel status etc.

HTML 5 - Meter

```
<html>
<body>

<h1>The Meter element</h1>

<p>The meter element is used to display fuel level:</p>

<label for="fuel">Fuel level:</label>

<meter id="fuel"
      min="0" max="100"
      low="33" high="66" optimum="80"
      value="50">
   at 50/100
</meter>

</body>
</html>
```



File | E:/Full%20Stack%20Deve...

The Meter element

The meter element is used to display fuel level:

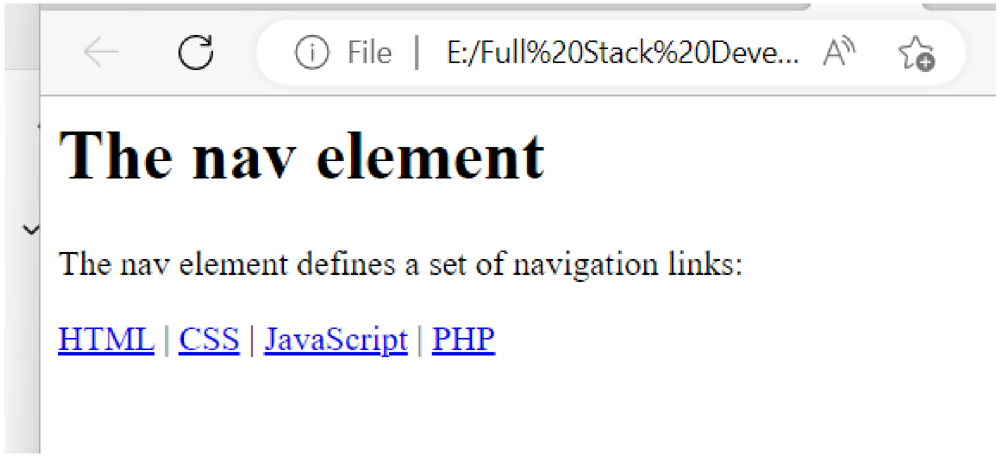
Fuel level:

HTML 5 - Nav

- The <nav> tag defines a set of navigation links.
- Notice that NOT all links of a document should be inside a <nav> element.
- The <nav> element is intended only for major blocks of navigation links.

HTML 5 - Nav

```
<html> <body>
<h1>The nav element</h1>
<p>The nav element defines a set of
navigation links:</p>

<nav>
<a href="/html/">HTML</a> |
<a href="/css/">CSS</a> |
<a href="/js/">JavaScript</a> |
<a href="/php/">PHP</a>
</nav>

</body> </html>
```



The nav element

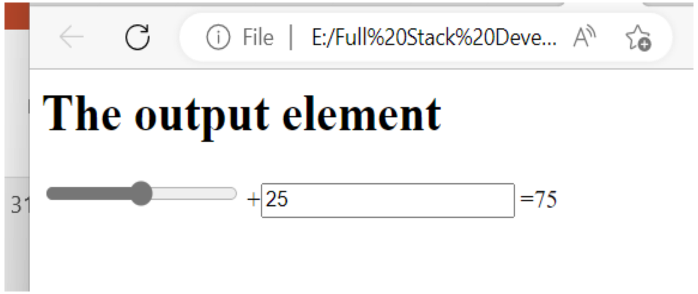
The nav element defines a set of navigation links:

HTML | CSS | JavaScript | PHP

HTML 5 - Output

- The <output> tag is used to represent the result of a calculation.
- The form provides a slider whose value can range between 0 and 100, and an <input> element into which you can enter a second number. The two numbers are added together, and the result is displayed in the <output> element each time the value of any of the controls changes.

```
<html> <body>
<h1>The output element</h1>

<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">
<input type="range" id="a" value="50">
+<input type="number" id="b" value="25">
=<output name="x" for="a b">75</output>
</form>

</body> </html>
```



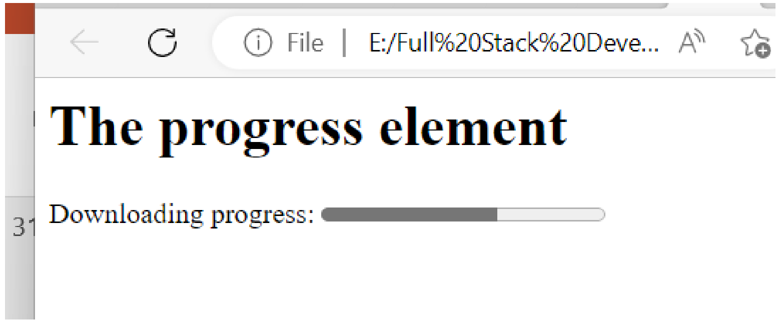
The output element

+25 =75

HTML 5 - Progress

- The <progress> tag represents the completion progress of a task.

```
<html><body>


<h1>The progress element</h1>


<label for="file">Downloading progress:</label>
<progress id="file" value=“62" max="100"> 62%
</progress>


</body></html>
```



File | E:/Full%20Stack%20Deve...

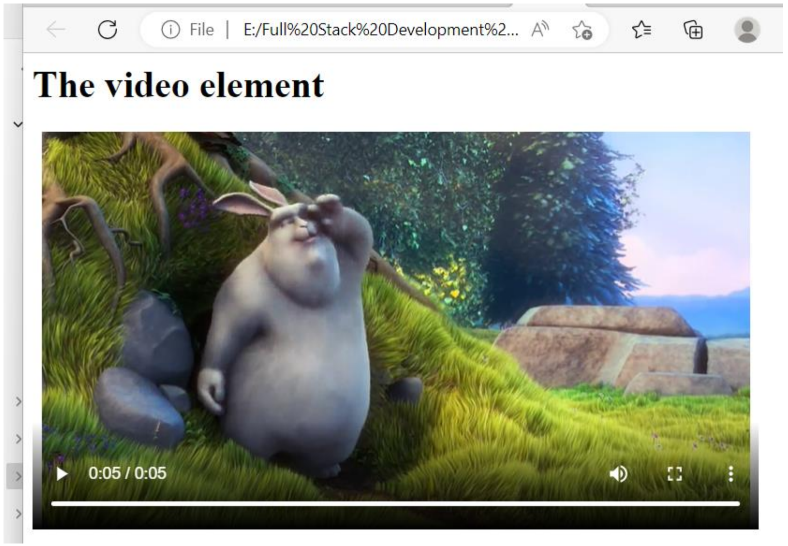
The progress element

Downloading progress:

HTML 5 - Video

- The <video> tag is used to embed video content in a document, such as a movie clip or other video streams.
- The <video> tag contains one or more <source> tags with different video sources.
- There are three supported video formats in HTML: MP4, WebM, and OGG.

```
<html>
<body>

<h1>The video element</h1>

<video width="620" height="340" controls>
  <source src="video1.mp4" type="video/mp4">
</video>

</body>
</html>
```

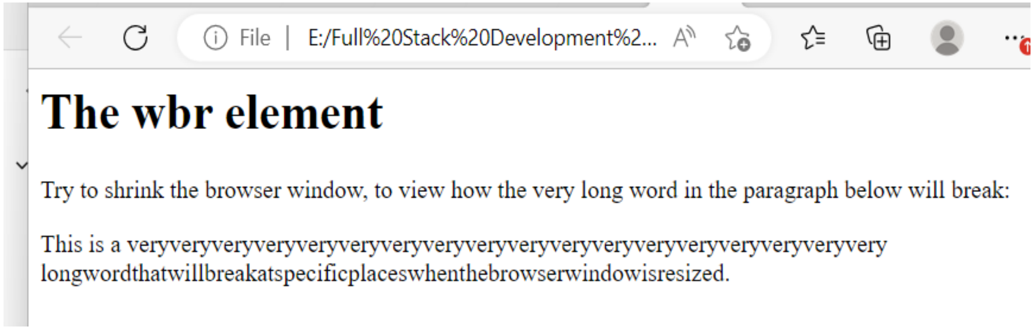


The video element

HTML 5 - WBR

- The <wbr> (Word Break Opportunity) tag specifies where in a text it would be ok to add a line-break.
- Tip: When a word is too long, the browser might break it at the wrong place. You can use the <wbr> element to add word break opportunities.

```
<html><body> <h1>The wbr element</h1>
<p>Try to shrink the browser window, to view how the very long word in the paragraph below will
break: </p>

<p>This is a
veryveryveryveryveryveryveryveryveryveryveryveryveryveryveryveryveryvery<wbr>longwordthatwi
llbreakatspecific<wbr>placeswhenthebrowserwindowisresized.</p>

</body></html>
```



The wbr element

Try to shrink the browser window, to view how the very long word in the paragraph below will break:

This is a veryveryveryveryveryveryveryveryveryveryveryveryveryveryveryveryveryvery
longwordthatwillbreakatspecificplaceswhenthebrowserwindowisresized.

HTML 5

Audio & Video

Audio & Video

- The HTML5 <audio> and <video> tags make it simple to add media to a website.
- We need to set src attribute to identify the media source and include a controls attribute so the user can play and pause the media.
- **Embedding Video**
- Most commonly used video formats are – MPEG, OGG
- The controls attribute adds video controls, like play, pause, and volume.
- It is a good idea to always include width and height attributes.
- The <source> element allows you to specify alternative video files which the browser may choose from.
- The browser will use the first recognized format.
- The text between the <video> and </video> tags will only be displayed in browsers.

HTML <video> Autoplay Muted

- Autoplay Muted: It will automatically play the video as soon as the video is downloaded fully.
- Users will not have to press the play button because the video will get started on its own.
- The only limitation here is you have to use the "muted" attribute along with the "autoplay" attribute.

```
<html>
<body>

<video width="1020" height="840" autoplay muted>

        <source src="Bells.mp4" type="video/mp4">

</video>

</body>
</html>
```

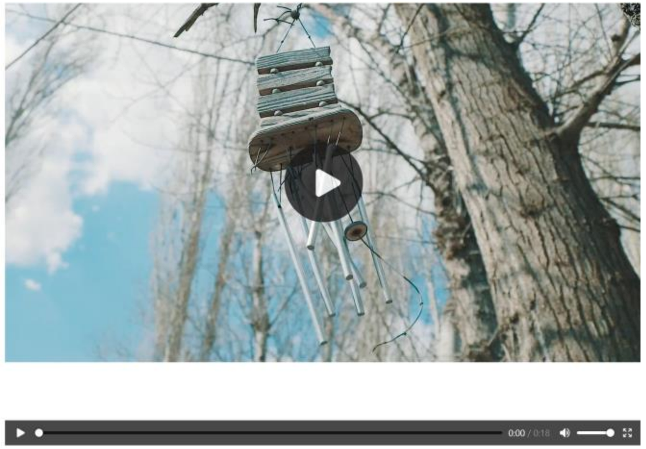
HTML <video> Attributes

- Width and Height - Height and Width attributes are used to specify the height and width of the video's size in HTML.

```
<video width="320" height="240">
 <source src="Bells.mp4" type="video/mp4">
</video>
```

- Controls: This attribute allows us to show various controls like play, pause, volume adjustments, full-screen icon

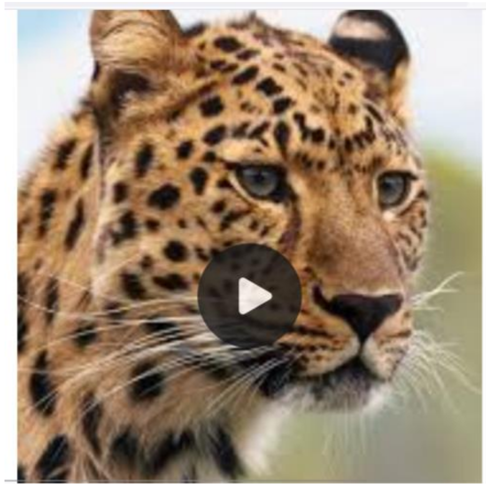
```
<video width="320" height="240" controls>
 <source src="Bells.mp4" type="video/mp4">
</video>
```



HTML <video> Attributes

- Poster: Whenever a video is getting displayed on the screen, it takes some time to download and then display. If the file size is large and the connectivity is low, you might see the loading icon.
- Instead, you can use the poster attribute to add an image over the video, which gets disappear after the video gets fully loaded.

```
<video controls poster="Image.jfif">
        <source src=“Bells.mp4" type="video/mp4">
</video>
```



HTML <video> Attributes

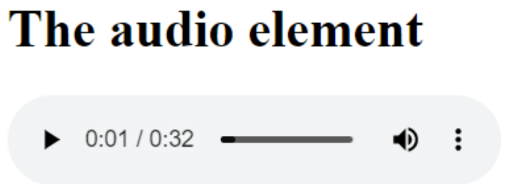
- Loop

The loop attribute allows us to repeat a particular video an infinite number of times. As soon as the video ends, it will start all over again.

```
<video controls loop>
  <source src=“Bells.mp4" type="video/mp4">
</video>
```

HTML <Audio> Tag

- Just like the <video> tag is used to add video files to your website, the <audio> tag is used to add audio files to your webpage.
- HTML5 provides this tag where you can embed music/audio files to your HTML document.
- There are 3 types of files that you can add to the audio tag: .mp3, .wav, .ogg
- <audio controls src="AUD.mp3"></audio>
- You will have to add another attribute known as 'controls,' which will provide you the controls like play, pause, and audio buttons, just like the video tag.



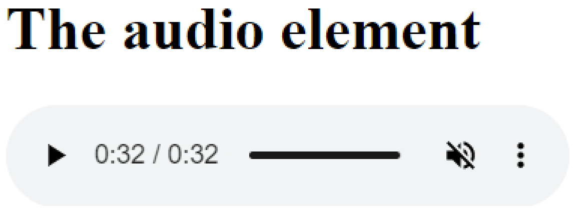
The audio element

HTML <Audio> Tag

- You would need to have a backup file that will get played if the first one is not supported.
- For example, if the .mp3 file is not supported, then I will add another file (let's say .ogg). This file will act as a backup file if .mp3 does not get played.
- At this time, you cannot use the src attribute inside the <audio> tag because "src" can have only one file path. We need to add more than the 1 file path. That's why you have to specify a separate tag for that called <source>.
- <audio controls>
-     <source src="AUD.mp3" type="audio/mp3">
-     <source src="AUD.ogg" type="audio/ogg">
- </audio>

HTML <Audio> Tag

- Autoplay: This attribute is used to play the audio automatically as soon as the page loads or as soon as the audio file gets fully downloaded.
- Muted: The muted attribute is used to mute the sound of the audio when the page loads. You can see a cross next to the volume bar that indicates that the audio is muted; click on that icon to hear the sound.
- <audio controls autoplay muted>
-     <source src=“AUD.mp3" type="audio/mp3">
-     <source src=“AUD.ogg" type="audio/ogg">
- </audio>


The audio element

0:32 / 0:32

HTML <Video> &<Audio> Tag

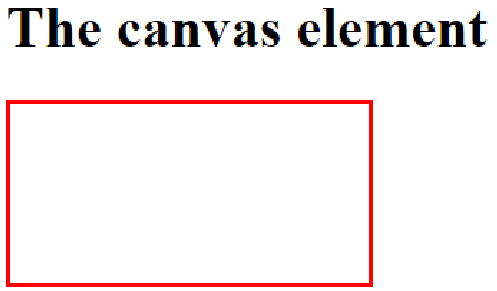
- The <video> tag is used to embed video content in a document, such as a movie clip or other video streams, and its main attributes are:
    - Width and Height
    - controls
    - poster
    - muted etc
- The <audio> tag is used to embed sound content in a document, such as music or other audio streams. Attributes are
    - src
    - controls
    - autoplay
    - muted etc

HTML <Canvas> Tag

- The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.
- The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.
- Canvas has several methods for drawing paths, boxes, circles, text, and adding images.
- <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;"> </canvas
- Always specify an id attribute, and a width and height attribute to define the size of the canvas. To add a border, use the style attribute.

HTML <Canvas> Tag

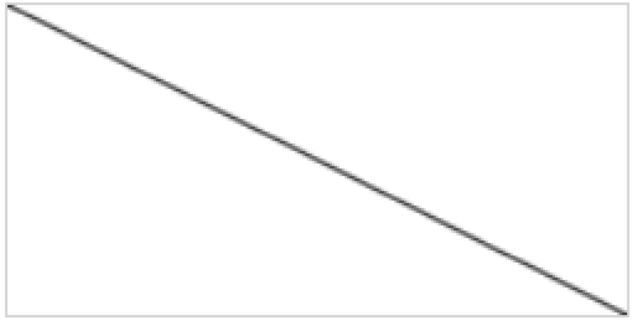
- <html>
- <body>
- <h1>The canvas element</h1>
- <canvas id="myCanvas" width="200" height="100" style="border:3px solid #ff0000;"> </canvas>
- </body>
- </html>
- Always specify an id attribute, and a width and height attribute to define the size of the canvas. To add a border, use the style attribute.


The canvas element

HTML 5 – Canvas - Line

- After creating the rectangular canvas area, you must add a JavaScript to do the drawing.

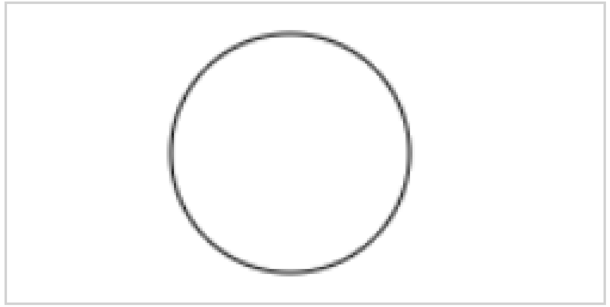
```
<html> <body>
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
Canvas Tag</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>
</body> </html>
```



HTML 5 – Canvas - Circle

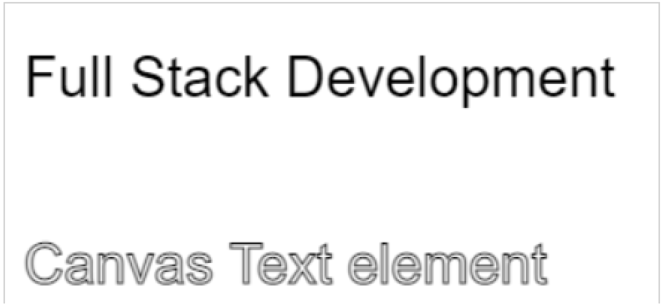
If you want to draw a circle on the canvas, you can use the arc() method: arc(x, y, r, start, stop)

```
<html><body>
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas
tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
</script>
</body></html>
```



HTML 5 – Canvas - Text

- There are property and methods used for drawing text on the canvas.
- font property: It is used to define the font property for the text.
- fillText(text,x,y) method: It is used to draw filled text on the canvas.
- strokeText(text,x,y) method: It is also used to draw text on the canvas, but the text is unfilled.



Full Stack Development

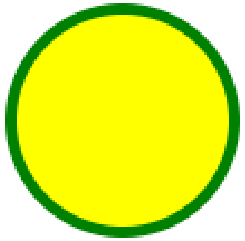
Canvas Text element

```
<html><body>
<canvas id="myCanvas" width="350" height="200"
style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas
tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Full Stack Development",10,50);
ctx.strokeText("Canvas Text element",10,150);
</script>

</body></html>
```

HTML 5 – Vector Graphics - Circle

- SVG stands for Scalable Vector Graphics
- SVG is used to define 2D graphics for the Web

```
<html>
<body>
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40"
  stroke="green" stroke-width="4" fill="yellow" />
Sorry, your browser does not support inline SVG.
</svg>
 </body>
</html>
```



HTML 5 – Vector Graphics – SVG Rectangle

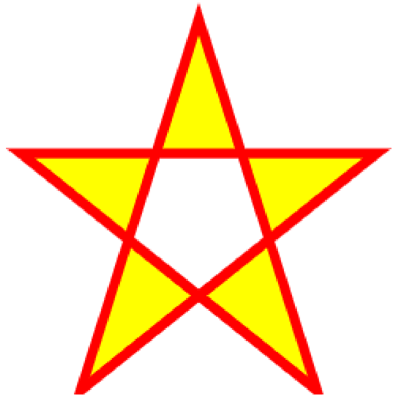
```
<html><body>

<svg width="400" height="100">
  <rect width="400" height="100"
  style="fill:rgb(0,100,255);stroke-
width:10;stroke:rgb(255,0,0)" />
Sorry, your browser does not support inline SVG.
</svg>

</body></html>
```

HTML 5 – Vector Graphics – Rounded Rectangle

```
<html> <body>

<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150"
height="150"
  style="fill:green;stroke:yellow;stroke-
width:5;opacity:0.5" />
Sorry, your browser does not support inline SVG.
</svg>

</body> </html>
```

HTML 5 – Vector Graphics – SVG Star

```
<html><body>

<svg width="300" height="200">

  <polygon points="100,10 40,198 190,78 10,78 160,198"

  style="fill:yellow;stroke:red;stroke-width:5;fill-rule:evenodd;" />

Sorry, your browser does not support inline SVG.

</svg>

 </body></html>
```



Comparison b/w Canvas & SVG

Canvas	Scalable Vector Graphics
Draw 2D Graphics with a Java Script	Describing 2D Graphics in XML
Resolution Dependent	Resolution Independent
No support for event handlers	Support for event handlers
Poor text rendering capabilities	Best suited for applications with large rendering areas (Google Maps)
Well suited for Graphic-intensive games	Not suited for Game applications
Ex: JPEG, BMP, GIF etc.,	Ex: PDF, CDR etc.,

HTML 5 – Web Storage

- With web storage, web applications can store data locally within the user's browser.
- Before HTML5, application data had to be stored in cookies, included in every server request.
- Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
- Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
- Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

HTML 5 – Web Storage Objects

- HTML web storage provides two objects for storing data on the client:
- window.localStorage - stores data with no expiration date
- window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)

HTML 5 – Local Storage Object

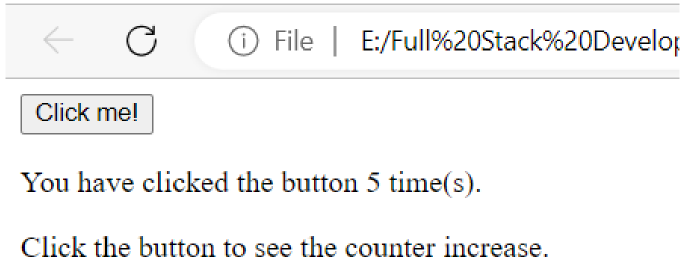
The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

```
<html> <body>
<div id="Output"></div>
<script>
if (typeof(Storage) !== "undefined") {
  localStorage.setItem("firstname", "Nagendra");
  document.getElementById("Output").innerHTML = localStorage.getItem("firstname");
}
else {
  document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Storage...";
}
</script>
</body></html>
```

HTML 5 – Local Storage Object

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

```
<html> <head>
<script>
function clickCounter() {
 if (typeof(Storage) !== "undefined") {
  if (localStorage.clickcount) {
   localStorage.clickcount = Number(localStorage.clickcount)+1;
  } else {     localStorage.clickcount = 1;     }
  document.getElementById("result").innerHTML = "You have clicked the button " + localStorage.clickcount + "
time(s).";
 }
```



You have clicked the button 5 time(s).

Click the button to see the counter increase.

HTML 5 – Local Storage Object

```
else {
  document.getElementById("result").innerHTML = "Sorry, your browser does not support web storage...";
 }
}
</script> </head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter will continue to count (is not reset).</p>
</body>
</html>
```

HTML 5 – Session Storage Objects

- The sessionStorage object is equal to the localStorage object, except that it stores the data for only one session. The data is deleted when the user closes the specific browser tab.

```
<html><head>
<script>
function clickCounter() {
 if (typeof(Storage) !== "undefined") {
  if (sessionStorage.clickcount) {
   sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;
  } else {     sessionStorage.clickcount = 1;    }
  document.getElementById("result").innerHTML = "You have clicked the button " +
sessionStorage.clickcount + " time(s) in this session.";
```

HTML 5 – Session Storage Objects

```
 } else {    document.getElementById("result").innerHTML = "Sorry, your browser does not support web
storage...";  }
}
</script></head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter is reset.</p>

</body> </html>
```

HTML 5 – Drag and Drop

- In HTML, any element can be dragged and dropped.
- Drag and Drop (DnD) is powerful User Interface concept which makes it easy to copy, reorder and deletion of items with the help of mouse clicks.
- This allows the user to click and hold the mouse button down over an element, drag it to another location, and release the mouse button to drop the element there.

HTML 5 – Drag and Drop Events

- Dragstart: Fires when the user starts dragging of the object.
- Dragenter: Fired when the mouse is first moved over the target element while a drag is occurring.
- Dragover: This event is fired as the mouse is moved over an element when a drag is occurring.
- Dragleave: This event is fired when the mouse leaves an element while a drag is occurring.
- Drag: Fires every time the mouse is moved while the object is being dragged.
- Drop: The drop event is fired on the element where the drop was occurred at the end of the drag operation.
- Dragend: Fires when the user releases the mouse button while dragging an object.

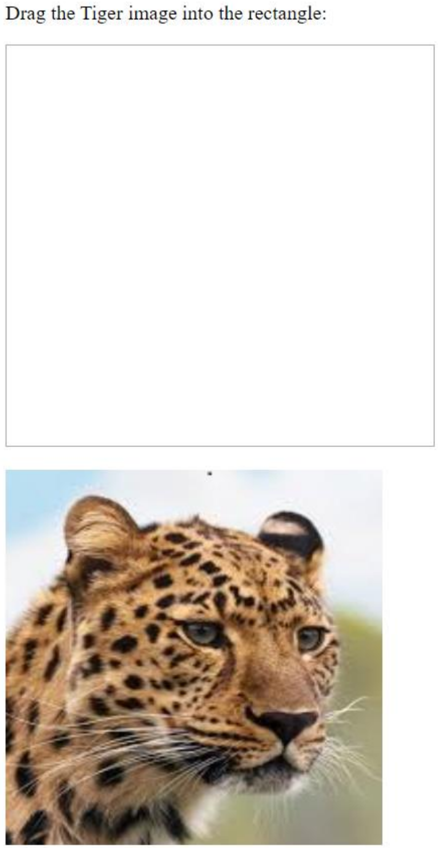
HTML 5 – Data Transfer Attributes

- dataTransfer.clearData([format])
- dataTransfer.setData(format, data)
- data = dataTransfer.getData(format)
- dataTransfer.setDragImage(element, x, y)
- dataTransfer.addElement(element)

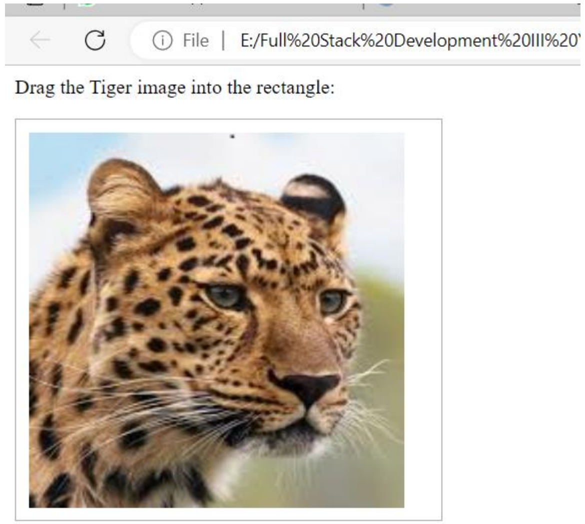
HTML 5

```
<html><head>
<style>
#div1 {
  width: 320px;
  height: 300px;
  padding: 10px;
  border: 1px solid #aaaaaa;
}
</style>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}
function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id); }
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data)); }
</script> </head>
<p>Drag the Tiger image into the rectangle:</p>
<div id="div1" ondrop="drop(event)"
ondragover="allowDrop(event)"></div>
<br> <img id="drag1" src="img1.jfif" draggable="true"
ondragstart="drag(event)" width="300" height="300">
</body></html>
```

HTML 5



Drag the Tiger image into the rectangle:



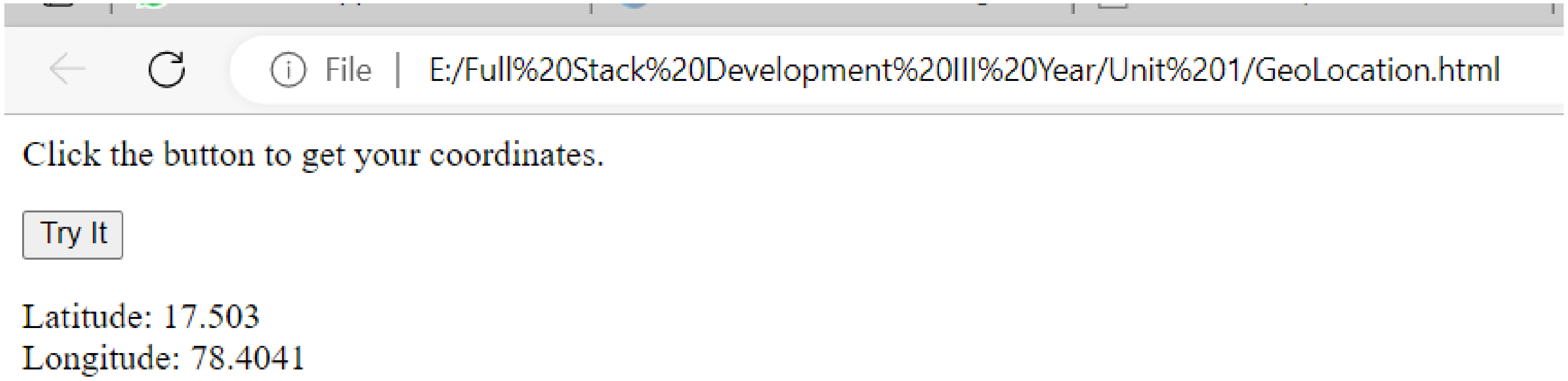
Drag the Tiger image into the rectangle:

HTML 5 – Geo Location

- The HTML Geolocation API is used to get the geographical position of a user.
- The getCurrentPosition() method is used to return the user's position.

```
<html><body>
<p>Click the button to get your coordinates.</p>
<button onclick="getLocation()">Try It</button>
<p id="demo"></p>
<script> var x = document.getElementById("demo");
function getLocation() {
 if (navigator.geolocation) {
   navigator.geolocation.getCurrentPosition(showPosition);
 } else {
   x.innerHTML = "Geolocation is not supported by this
browser.";
 } }
function showPosition(position) {
 x.innerHTML = "Latitude: " +
position.coords.latitude +
 "<br>Longitude: " +
position.coords.longitude;
}
</script>

</body>
</html>
```

HTML 5 – Geo Location


File | E:/Full%20Stack%20Development%20III%20Year/Unit%201/GeoLocation.html

```
Click the button to get your coordinates.

Try It

Latitude: 17.503
Longitude: 78.4041
```

CSS

CSS 3

- CSS3 is the latest version of the CSS specification.
- CSS3 adds several new styling features and improvements to enhance the web presentation capabilities.

Features of CSS 3

- You can easily apply same style rules on multiple elements.
- You can control the presentation of multiple pages of a website with a single style sheet.
- You can present the same page differently on different devices.
- You can style dynamic states of elements such as hover, focus, etc.
- You can change the position of an element on a web page without changing the markup.
- You can alter the display of existing HTML elements.
- You can transform elements like scale, rotate, skew, etc. in 2D or 3D space.
- You can create animations and transitions effects without using any JavaScript.

Advantages of CSS 3

CSS Save Lots of Time: CSS gives lots of flexibility to set the style properties of an element. You can write CSS once; and then the same code can be applied to the groups of HTML elements, and can also be reused in multiple HTML pages.

Easy Maintenance : CSS provides an easy means to update the formatting of the documents, and to maintain the consistency across multiple documents.

Superior Styles to HTML : CSS has much wider presentation capabilities than HTML and provide much better control over the layout of your web pages.

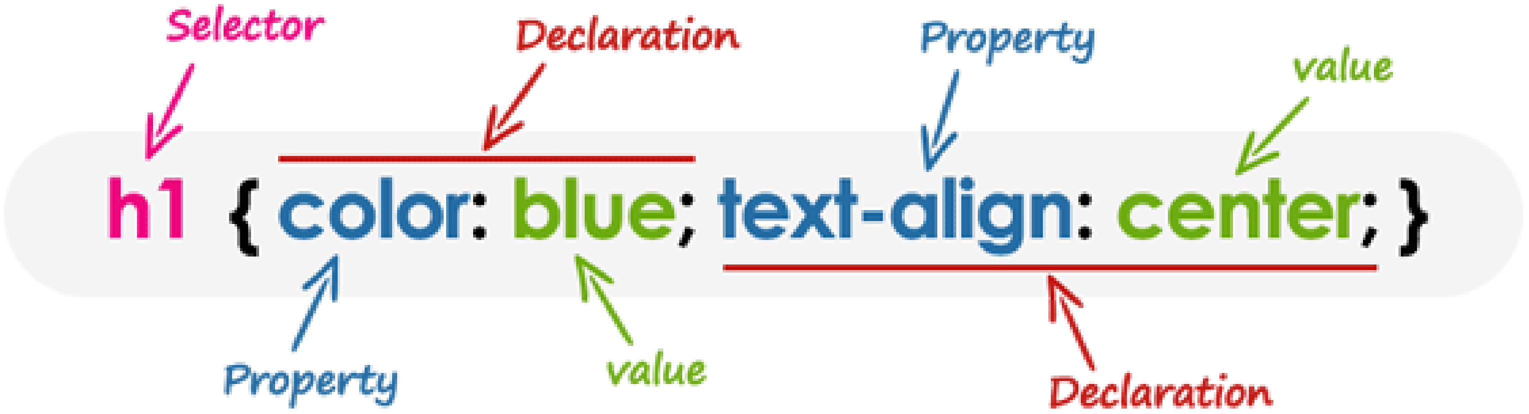
Advantages of CSS 3

Pages Load Faster: CSS enables multiple pages to share the formatting information, which reduces complexity and repetition in the structural contents of the documents. It significantly reduces the file transfer size, which results in a faster page loading.

Multiple Device Compatibility: Using CSS the same HTML document can be presented in different viewing styles for different rendering devices such as desktop, cell phones, etc.

CSS Syntax

- A CSS stylesheet consists of a set of rules that are interpreted by the web browser and then applied to the corresponding elements such as paragraphs, headings, etc. in the document.
- A CSS rule have two main parts, a selector and one or more declarations:



CSS Syntax

- The selector specifies which element or elements in the HTML page the CSS rule applies to.
- The declarations within the block determines how the elements are formatted on a webpage. Each declaration consists of a property and a value separated by a colon (:) and ending with a semicolon (;) and the declaration groups are surrounded by curly braces { }

Types of Styles

CSS can either be attached as a separate document or embedded in the HTML document itself. There are three methods of including CSS in an HTML document:

- **Inline styles** — Using the `style` attribute in the HTML start tag.
- **Embedded styles** — Using the `<style>` element in the head section of a document.
- **External style sheets** — Using the `<link>` element, pointing to an external CSS file.

In this tutorial we will cover all these three methods for inserting CSS one by one.

Note: The inline styles have the highest priority, and the external style sheets have the lowest. It means if you specify styles for an element in both *embedded* and *external* style sheets, the conflicting style rules in the embedded style sheet would override the external style sheet.

CSS – Static Positioning

- Positioning elements appropriately on the web pages is a necessity for a good layout design. There are several methods in CSS that you can use for positioning elements.

Static Positioning

- A static positioned element is always positioned according to the normal flow of the page. HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.

CSS – Static Positioning

```
<html>

<head>


<style>


div.static {

  position: static;

  border: 3px solid #73AD21;

}

</style>

</head
```

```
<body>

<h2>position: static;</h2>

<p>An element with position: static; is not
positioned in any special way; it is always
positioned according to the normal flow of
the page:</p>


<div class="static">

This div element has position: static;

</div>

</body></html>
```

CSS – Relative Positioning

- An element with position: relative; is positioned relative to its normal position.
- Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.

CSS – Relative Positioning

```
<html>
<head>

<style>

div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
</style>

</head>
```

```
<body>

<h2>position: relative;</h2>


<p>An element with position: relative; is
positioned relative to its normal position:</p>


<div class="relative">
This div element has position: relative;
</div>


</body></html>
```

CSS – Fixed Positioning

- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.
- The top, right, bottom, and left properties are used to position the element.

CSS – Fixed Positioning

```
<html> <head>
<style>
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
</style>
</head>
```

```
<body>

<h2>position: fixed;</h2>

<p>An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:</p>

<div class="fixed">
This div element has position: fixed;
</div>

</body>
</html>
```

CSS – Absolute Positioning

- An element with position: absolute; is positioned relative to the nearest positioned ancestor.
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

CSS – Absolute Positioning

```
<html><head>
<style>
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
</style> </head>
```

```
<body>

<h2>position: absolute;</h2>

<p>An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):</p>

<div class="relative">This div element has position: relative;
 <div class="absolute">This div element has position: absolute;</div>
</div>

</body>
</html>
```

CSS – Sticky Positioning

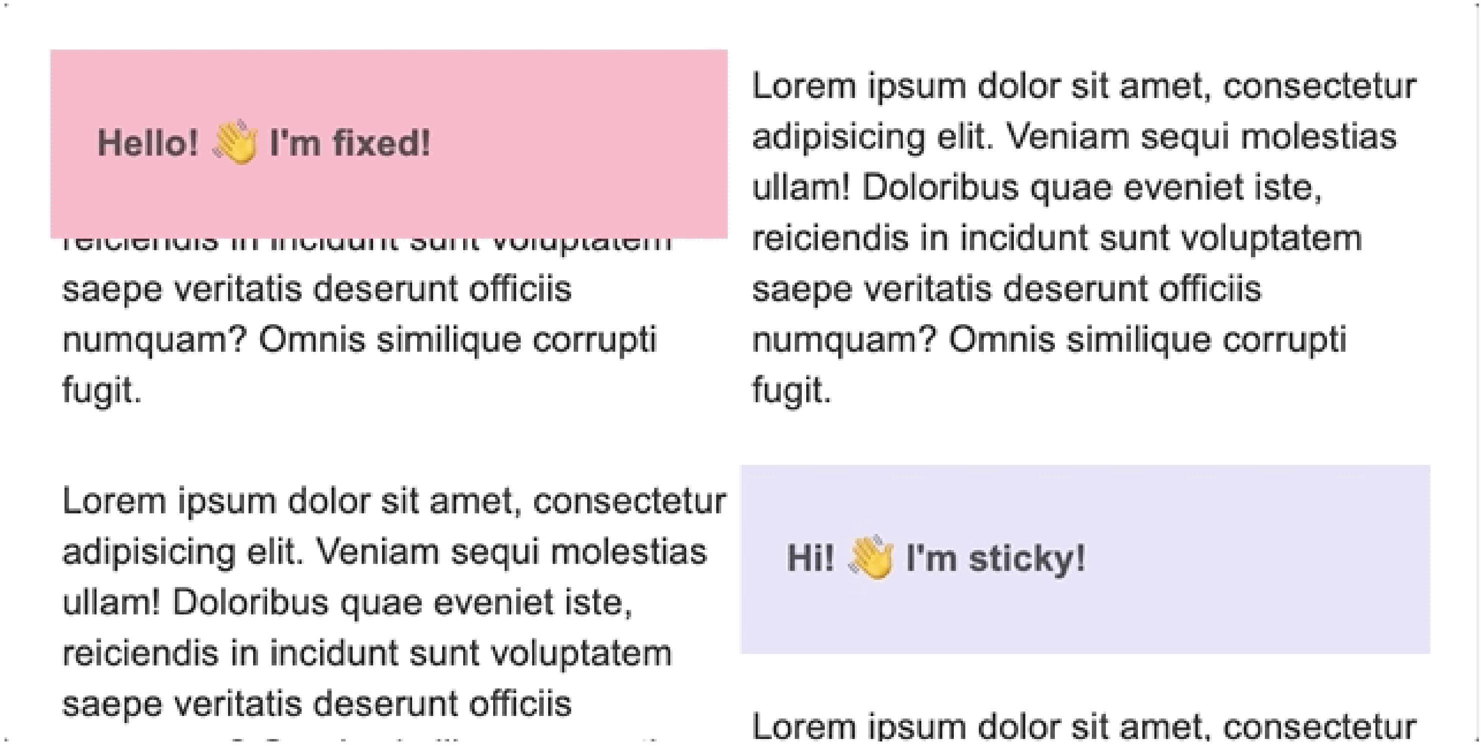
- An element with position: sticky; is positioned based on the user's scroll position.

CSS – Sticky Positioning

```
<html><head>
<style>
div.sticky {
 position: sticky;
  top: 0;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
</style></head>
```

```
<body>
<p>Try to <b>scroll</b> inside this frame to understand how sticky positioning works.</p>

<div class="sticky">I am sticky!</div>

<div style="padding-bottom:2000px">
  <p>In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.</p>
  <p>Scroll back up to remove the stickyness.</p>
<p>HTML 5: New Elements, Video & Audio, Canvas, Vector Graphics, Web Storage, Drag & Drop, Geolocation. </p>
</div>
</body></html>
```

Diff b/w Fixed and Sticky



Hello! 👋 I'm fixed!

reiciendis in incidunt sunt voluptatem
saepe veritatis deserunt officiis
numquam? Omnis similique corrupti
fugit.

Lorem ipsum dolor sit amet, consectetur
adipisicing elit. Veniam sequi molestias
ullam! Doloribus quae eveniet iste,
reiciendis in incidunt sunt voluptatem
saepe veritatis deserunt officiis

Lorem ipsum dolor sit amet, consectetur
adipisicing elit. Veniam sequi molestias
ullam! Doloribus quae eveniet iste,
reiciendis in incidunt sunt voluptatem
saepe veritatis deserunt officiis
numquam? Omnis similique corrupti
fugit.

Hi! 👋 I'm sticky!

Lorem ipsum dolor sit amet, consectetur

CSS – Background Images

- An element with background-image; property sets one or more background images for an element.
- By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally.
- The background of an element is the total size of the element, including padding and border (but not the margin).
- Always set a background-color to be used if the image is unavailable.

CSS – Background Images

- Set a background-image for the <body> element:

```
<html><head>
<style>
body  {
  background-image: url("Picture2.jpg");
  background-color: #cccccc;
}
</style></head>
<body>

<h1>The background-image Property</h1>

<p>Cascading Style Sheets</p>

</body></html>
```

CSS – Background Images

- Set a background-images for the <body> element:

```
<html><head>
<style>
body  {
  background-image: url("Picture1.jpg"),url("Picture4.jpg");
  background-color: #ccccff;
}
</style></head>
<body>

<h1>The background-image Property</h1>

<p>Cascading Style Sheets</p>

</body></html>
```

CSS-Pseudo Classes

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 mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

The CSS pseudo-classes allow you to style the dynamic states of an element such as hover, active and focus state, as well as elements that are existing in the document.

A pseudo-class starts with a colon (:)

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

CSS – Anchor Pseudo Classes

Using anchor pseudo-classes links can be displayed in different ways:

These pseudo-classes let you style unvisited links differently from visited ones. The most common styling technique is to remove underlines from visited links.

```
<html><head>
<title>Example of Anchor Pseudo-classes</title>
<style>
   a:link {
      color: blue;
   }
   a:visited {
      text-decoration: red;
   }
</style></head>
<body>
   <p>Visit <a href="https://www.griet.ac.in" target="_blank">www.griet.ac.in</a></p>
</body>
</html>
```

CSS – Dynamic Anchor Pseudo Classes

Some anchor pseudo-classes are dynamic — they're applied as a result of user interaction with the document like on hover, or on focus etc.

```
<html>
<head>

<title>Example of Dynamic Anchor Pseudo-classes</title>
<style>
   a:link {
      color: blue;
   }
   a:visited {
      text-decoration: none;
   }
   a:hover {
      color: red;
      font-size: 22px;
   }
 a:focus {
       color: green;
   }
</style>
</head>
<body>
   <p>Visit <a href="https://www.griet.ac.in"
target="_blank">www.griet.ac.in</a></p>
</body>
</html>
```

CSS – First Child Pseudo Classes

The first-child pseudo-class matches a specified element that is the first child of another element.

```
<html><head>
<title>Example of CSS :first-child Pseudo-class</title>
<style>
   ol{
      padding: 0;
      list-style: none;
   }
   ol li{
      padding: 10px;
      border-top: 1px solid #ff0000;
   }
   li:first-child {
      border-top: none;
      color:blue
   }
</style></head>
```

```
<body>   <h1>Full Stack Development</h1>
   <ol>
      <li>HTML 5</li>
      <li>CSS3</li>
      <li>Java Script</li>
      <li>JQuery</li>
      <li>Node JS</li>
      <li>PHP MySQL</li>
   </ol>
</body></html>
```

CSS – Last Child Pseudo Classes

The last-child pseudo-class matches an element that is the last child element of some other element.

```
<html> <head>
<title>Example of CSS :last-child Pseudo-class</title>
<style>
   ul{
      padding: 0;
      list-style: none;
   }
   ul li{
      display: inline;
      padding: 20px;
      border-right: 1px solid #ff0000;
   }
   li:last-child {
      border-right: none;
   }
</style></head>
```

```
<body>   <h1>Full Stack Development</h1>
   <ul>
      <li>HTML 5</li>
      <li>CSS3</li>
      <li>Java Script</li>
      <li>JQuery</li>
      <li>Node JS</li>
      <li>PHP MySQL</li>
   </ul>
</body></html>
```

CSS - Colors

- Colors are specified using predefined color names
- Colors in CSS most often specified in the following formats:
- a color keyword - like "red", "green", "blue", "transparent", etc.
- a HEX value - like "#ff0000", "#00ff00", etc.
- an RGB value - like "rgb(255, 0, 0)"

```
<html> <body>
<h1 style="background-color:Tomato;">Tomato</h1>
<h1 style="background-color:Orange;">Orange</h1>
<h1 style=" background-color :#92a8d1;">#92a8d1</h1>
<h1 style=" background-color :#00ff00"> #00ff00</h1>
<h1 style=" background-color :rgb(100,100,255);">
rgb(100,100,255)</h1>
<h1 style=" background-color :rgb(201,76,76);">
rgb(201,76,76)</h1>
</body> </html>
```

CSS - Colors

Tomato

Orange

#92a8d1

#00ff00

rgb(100,100,255)

rgb(201,76,76)

CSS - TextColor

```
<html><body>
<h3 style="color:Tomato;">CSS - Colors</h3>

<p style="color:DodgerBlue;"> Colors are specified using predefined color names
Colors in CSS most often specified in the following formats:
</p>

<p style="color:MediumSeaGreen;">a color keyword - like "red", "green", "blue", "transparent", etc. <br>
a HEX value - like "#ff0000", "#00ff00", etc. <br>
an RGB value - like "rgb(255, 0, 0)"
</p>
</body></html>
```

CSS - TextColor

CSS - Colors

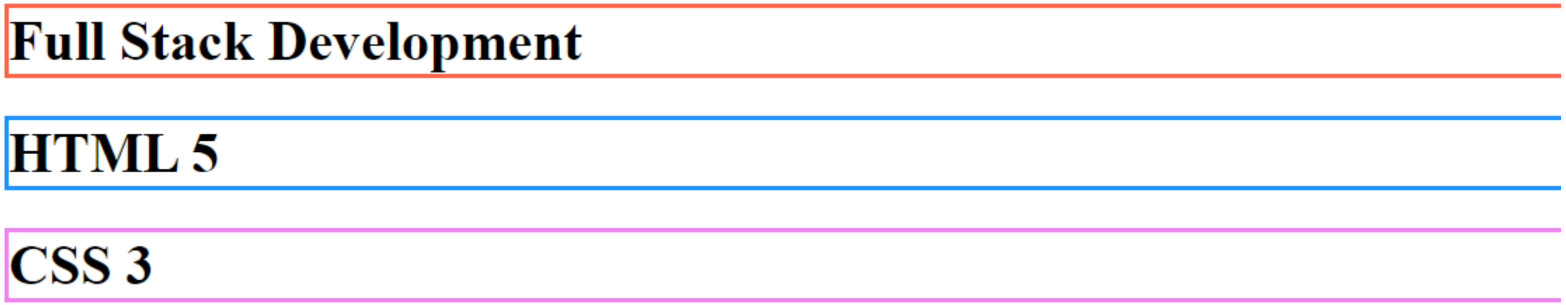
Colors are specified using predefined color names Colors in CSS most often specified in the following formats:

a color keyword - like "red", "green", "blue", "transparent", etc.
a HEX value - like "#ff0000", "#00ff00", etc.
an RGB value - like "rgb(255, 0, 0)"

CSS – Border Color

```
<html>
<body>

<h1 style="border: 3px solid Tomato;">Full Stack Development</h1>

<h1 style="border: 3px solid DodgerBlue;">HTML 5</h1>

<h1 style="border: 3px solid Violet;">CSS 3</h1>

</body>
</html>
```

CSS – Border Color



Full Stack Development

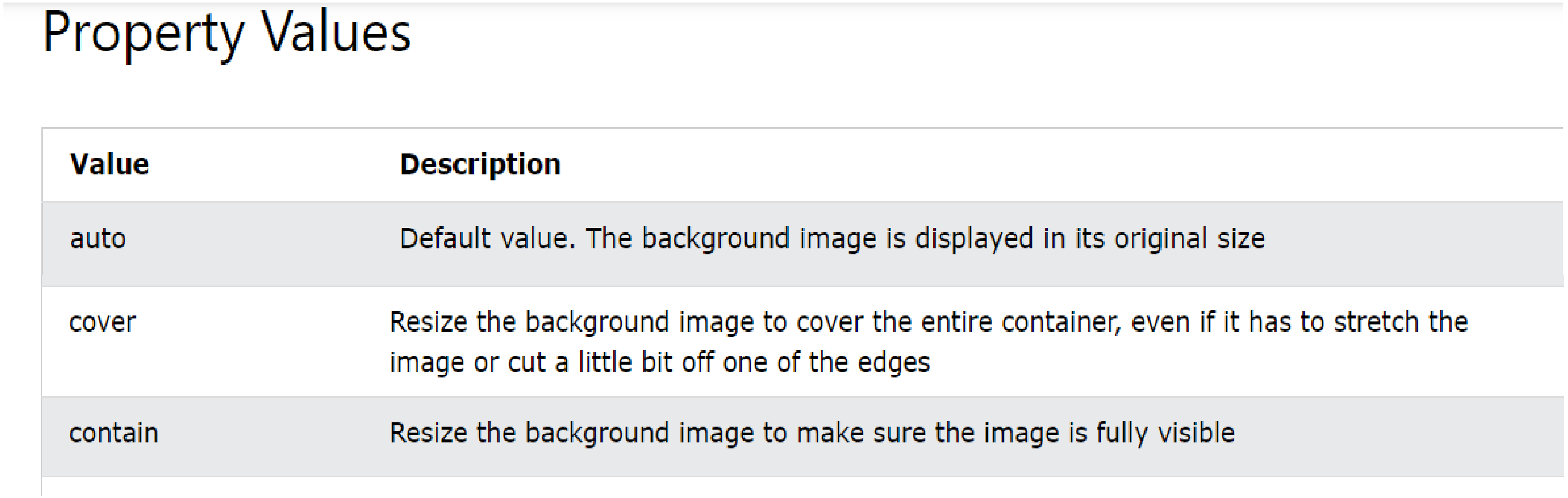
HTML 5

CSS 3

CSS 3 – Background size Property

- The CSS3 provides several new properties to manipulate the background of an element like background clipping, multiple backgrounds, and the option to adjust the background size.
- The background-size property can be used to specify the size of the background images.
- The background image size can be specified using the pixels or percentage values as well as the keywords auto, contain, and cover.

CSS 3 – Background size Property



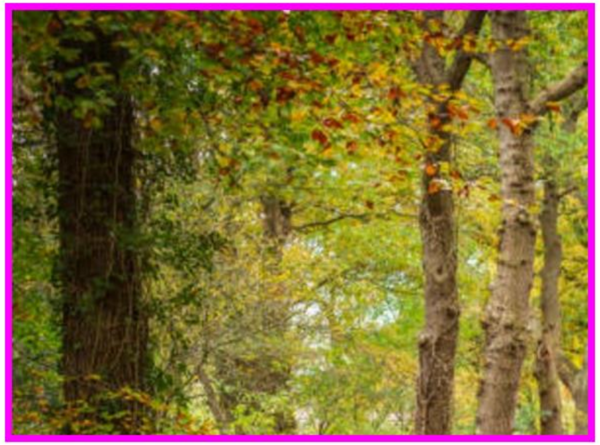
Property Values

Value	Description
auto	Default value. The background image is displayed in its original size
cover	Resize the background image to cover the entire container, even if it has to stretch the image or cut a little bit off one of the edges
contain	Resize the background image to make sure the image is fully visible

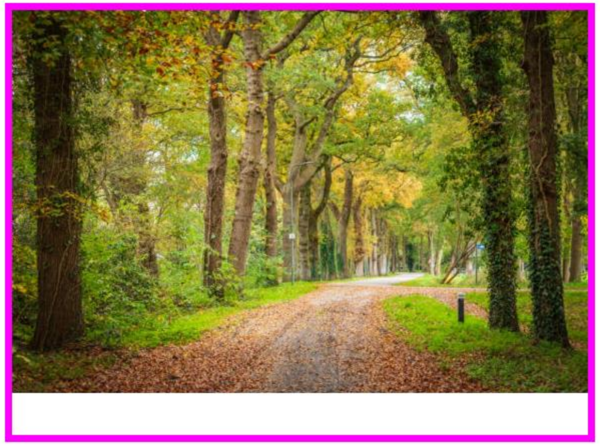
CSS 3 – Background size Property

```
<html>
<head>
<title>Setting background-size of an Element</title>
<style>
            .box {
            width: 460px;
            height: 340px;
background: url("Picture3.jpg") no-repeat;
            background-size: contain;
            border: 6px solid #ff00ff;
            }
</style>
</head>
<body>
  <div class="box"></div>
</body>
</html>
```

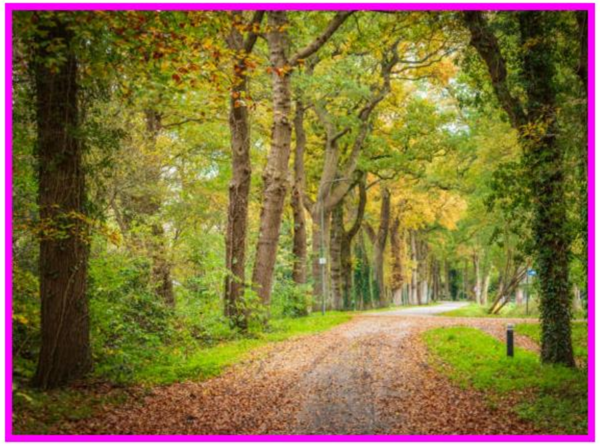
CSS 3 – Background size Property



Auto Property



Contain Property



Cover Property

CSS 3 – Background Clip Property

- The background-clip property defines how far the background (color or image) should extend within an element.

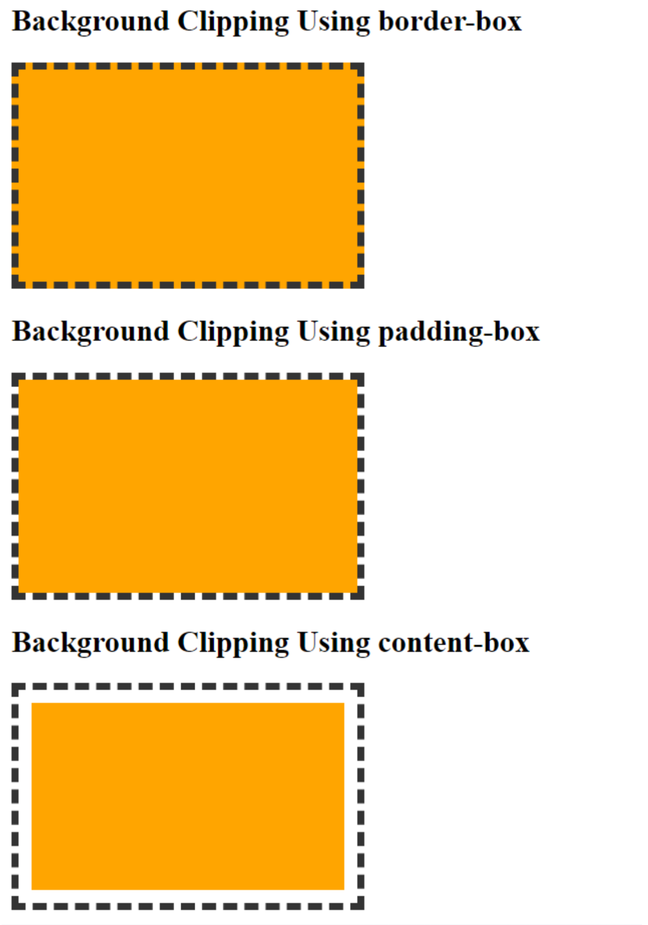
border-box	Default value. The background extends behind the border
padding-box	The background extends to the inside edge of the border
content-box	The background extends to the edge of the content box

CSS 3 – Background Clip Property

```
<html>
<head>
<title>Example of CSS3 Background Clipping</title>
<style>       .box {
                        width: 250px;
                        height: 150px;
                        padding: 10px;
                        border: 6px dashed #333;
                        background: orange;
              }
              .clip1 {
                        background-clip: border-box;
              }
              .clip2 {
                        background-clip: padding-box;
              }
              .clip3 {
                        background-clip: content-box;
              }
</style></head>
```

```
<body>
  <h2>Background Clipping Using border-box</h2>
   <div class="box clip1"></div>
   <h2>Background Clipping Using padding-box</h2>
   <div class="box clip2"></div>
   <h2>Background Clipping Using content-box</h2>
   <div class="box clip3"></div>
</body>
</html>
```

CSS 3 – Background Clip Property



Background Clipping Using border-box

Background Clipping Using padding-box

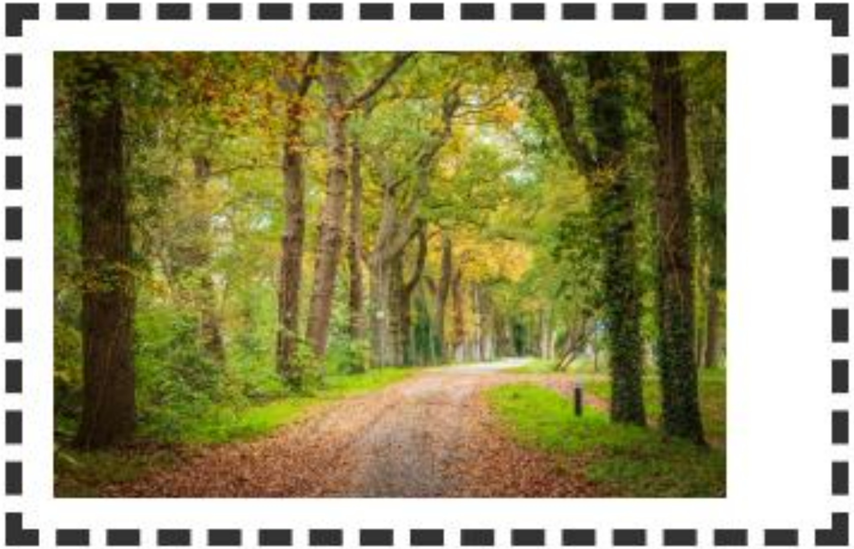
Background Clipping Using content-box

CSS 3 – Background-Origin Property

- The background-origin property can be used to specify the positioning area of the background images.
- It can take the same values as background-clip property: border-box, padding-box, content-box.

```
<html><head>
<title>Example of Setting background-origin of an Element</title>
<style>       .box {
                     width: 250px;
                     height: 150px;
                     padding: 10px;
                     border: 6px dashed #333;
                     background: url("Picture3.jpg") no-repeat;
                     background-size: contain;
                     background-origin: content-box;
                     }
</style></head>
<body>
  <div class="box"></div>
</body></html>
```

CSS 3 – Background-Origin Property



CSS 3 – Multiple Backgrounds

- CSS3 gives you ability to add multiple backgrounds to a single element.
- The backgrounds are layered on the top of one another.
- The number of layers is determined by the number of comma-separated values in the background-image.

CSS 3 – Multiple Backgrounds

```
<html> <head>
<title>Example of CSS3 Multiple Backgrounds</title>
<style>
        .box {
                width: 100%;
                height: 500px;
background: url("1.jpg") no-repeat right,  url("2.jpg")  no-repeat left, url("3.jpg")  no-repeat center;
        }
</style>
</head>
<body>
  <div class="box"></div>
</body>
</html>
```

CSS 3 – Multiple Backgrounds

CSS 3 – Gradients

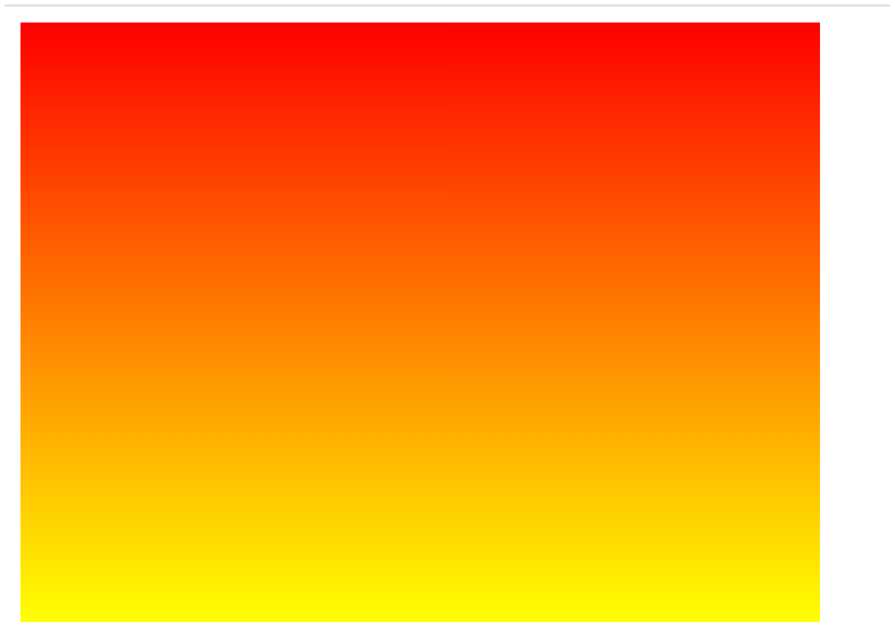
- The CSS3 gradient feature allows you to create a gradient from one color to another without using any images.
- Provides a flexible solution to generate smooth transitions between two or more colors.
- The elements with gradients can be scaled up or down to any extent without losing the quality.
- Gradients are available in two styles: *linear* and *radial*.

CSS 3 Linear Gradients

- To create a linear gradient you must define at least two color stops.
- However to create more complex gradient effects you can define more color stops.
- Color stops are the colors you want to render smooth transitions among.
- You can also set a starting point and a direction (or an angle) along which the gradient effect is applied.
- The basic syntax of creating the linear gradients using the keywords can be given with:
    - *linear-gradient(direction, color-stop1, color-stop2, ...)*

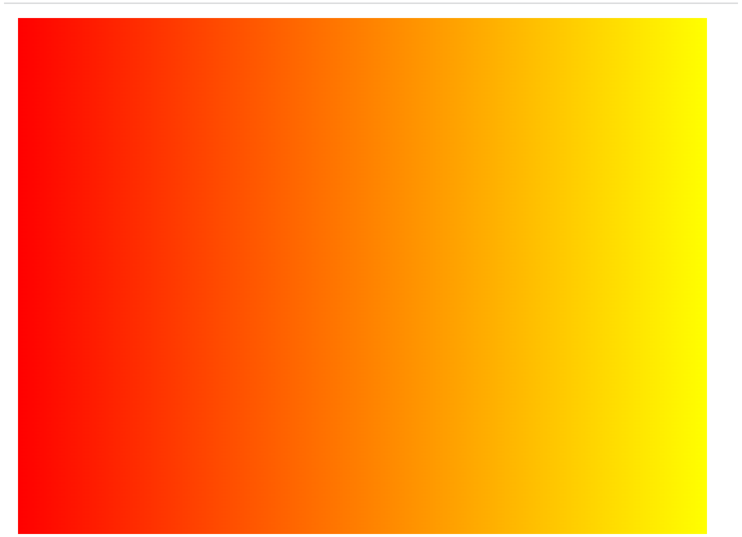
CSS 3 – Linear Gradients – Top to Bottom

```
<html>
<head>
<title>Example of Linear Gradients from Top to
Bottom</title>
<style>
            .gradient {
                        width: 400px;
                        height: 300px;

                        background: red;

            background: linear-gradient(red, yellow);
            }
</style>
</head>
<body>
  <div class="gradient"></div>
</body>
</html>
```



CSS 3 – Linear Gradients – Left to Right

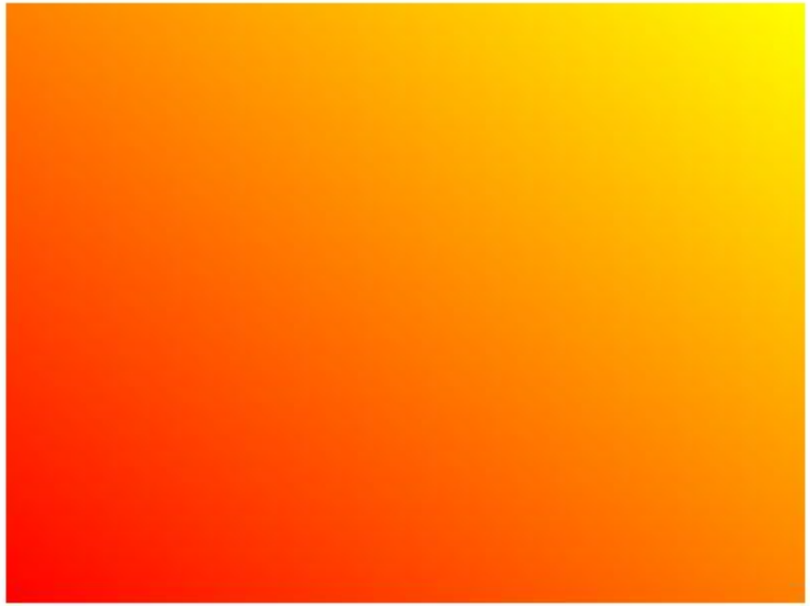
```
<html>
<head>
<title>Example of Linear Gradients from Left to
Right</title>
<style>
                .gradient {
                                width: 400px;
                                height: 300px;

                                background: red;

background: linear-gradient(to right, red, yellow);
</style>
</head>
<body>
   <div class="gradient"></div>
</body>
</html>
```



CSS 3 – Linear Gradients – Diagonal

The following example will create a linear gradient from the bottom left corner to the top right corner of the element's box.

```
<html>
<head>
<title>Example of Linear Gradients - Gradients</title>
<style>
            .gradient {
                        width: 400px;
                        height: 300px;

                        background: red;

background: linear-gradient(to top right, red, yellow);
</style>
</head>
<body>
  <div class="gradient"></div>
</body>
</html>
```

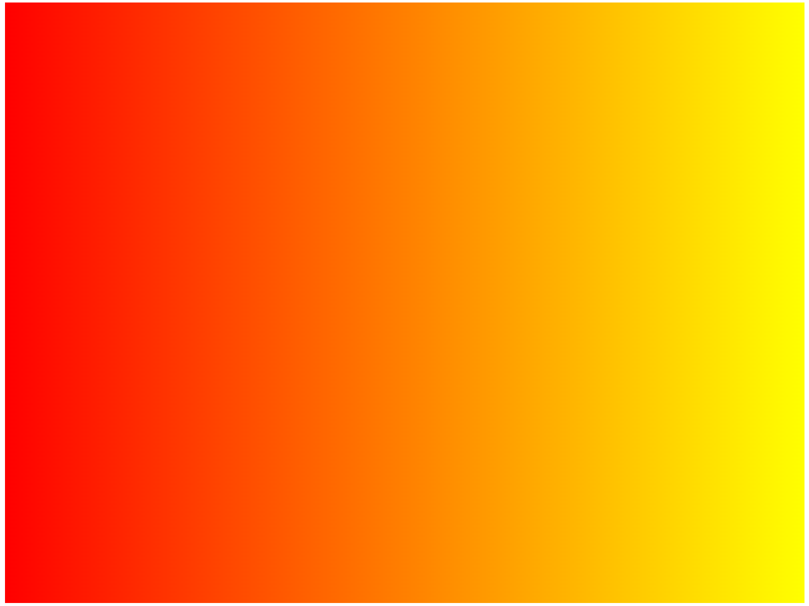


CSS 3 – Linear Gradients – Using Angles

- If you want more control over the direction of the gradient, you can set the angle.
- The angle 0deg creates a bottom to top gradient, and positive angles represent clockwise rotation, that means the angle 90deg creates a left to right gradient.
- The basic syntax of creating the linear gradients using angle can be given with:
    - *linear-gradient(angle, color-stop1, color-stop2, ...)*

CSS 3 – Linear Gradients – Direction

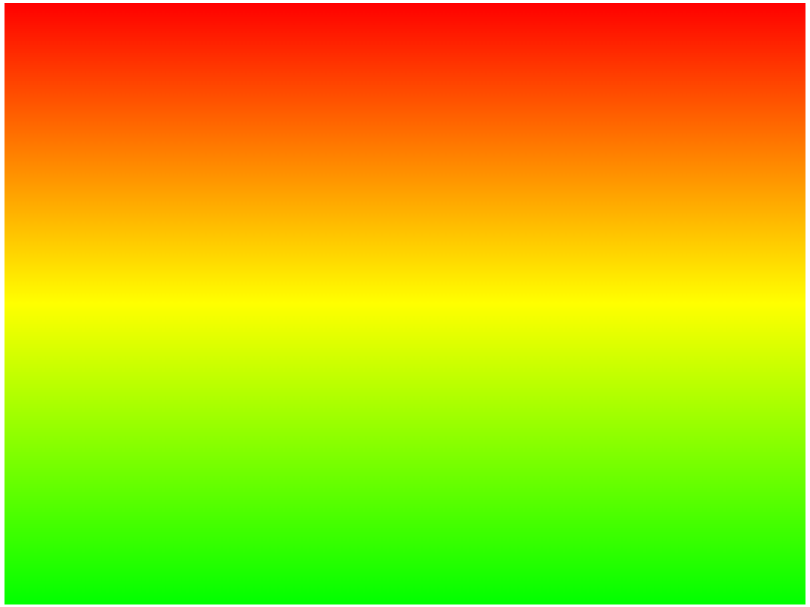
```
<html><head>
<title>Example of Linear Gradients - Direction</title>
<style>
		.gradient {
				width: 400px;
				height: 300px;

				background: red;

background: linear-gradient(90deg, red, yellow);
</style>
</head>
<body>
  <div class="gradient"></div>
</body>
</html>
```



CSS 3 – Linear Gradients – Multiple Color Stops

You can also create gradients for more than two colors. All colors are evenly spaced.

```
<html>
<head>
<title>Example of Linear Gradients – Multiple Color
Stops</title>
<style>
		.gradient {
				width: 400px;
				height: 300px;

				background: red;

background: linear-gradient(red, yellow, lime);
</style>
</head>
<body>
   <div class="gradient"></div>
</body>
</html>
```



CSS 3 Radial Gradients

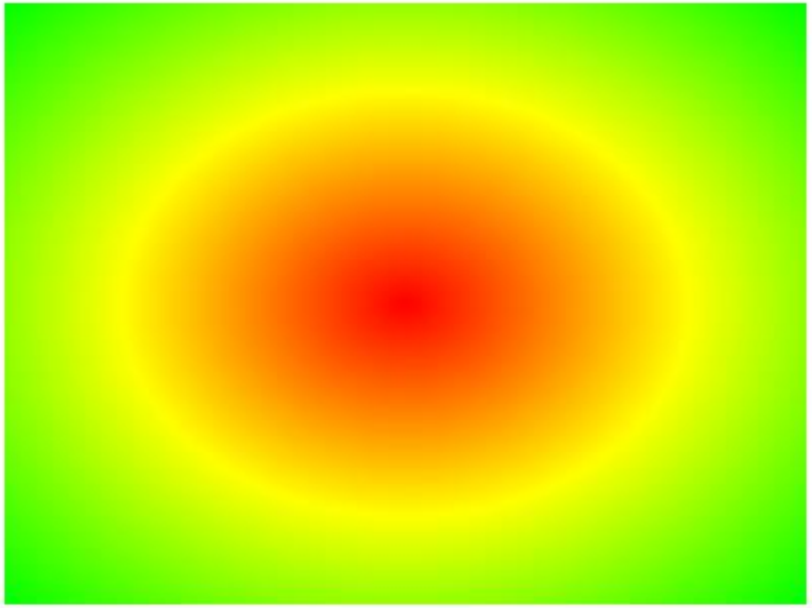
- In a radial gradient color emerge from a single point and smoothly spread outward in a circular or elliptical shape rather than fading from one color to another in a single direction as with linear gradients. The basic syntax of creating a radial gradient can be given with:

- *radial-gradient(shape size at position, color-stop1, color-stop2, ...);*

- **position** — Specifies the starting point of the gradient
- **shape** — Specifies the shape of the gradient's ending shape. It can be circle or ellipse.

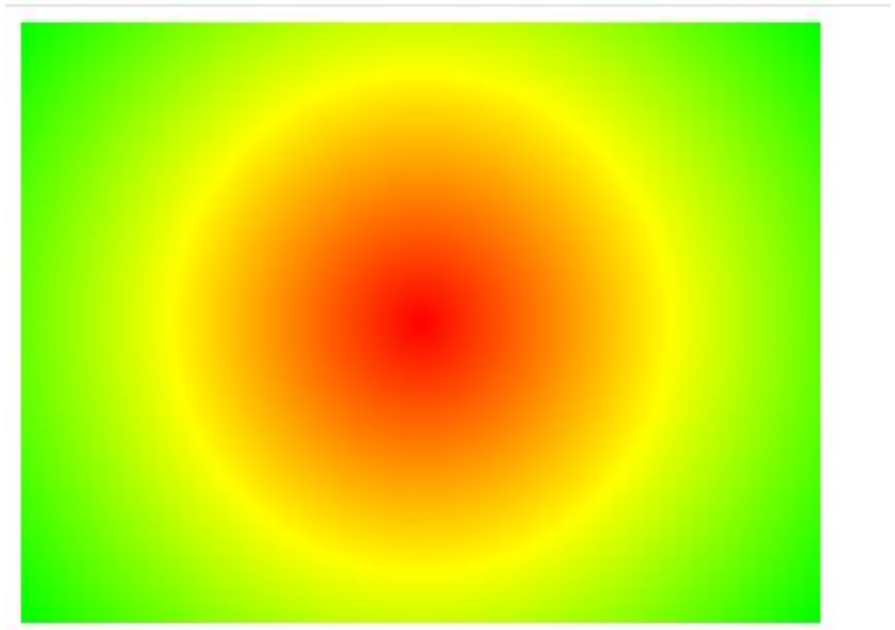
CSS 3 – Radial Gradients

```
<html>
<head>
<title>Example of Radial Gradients </title>
<style>
            .gradient {
                        width: 400px;
                        height: 300px;

                        background: red;

background: radial-gradient(red, yellow, lime);
</style>
</head>
<body>
   <div class="gradient"></div>
</body>
</html>
```



CSS 3 – Radial Gradients - Shape

```
<html><head>
<title>Example of Radial Gradients -
Shape</title>
<style>
          .gradient {
                    width: 400px;
                    height: 300px;

                    background: red;

background: radial-gradient(circle, red, yellow,
lime);
</style>
</head>
<body>
  <div class="gradient"></div>
</body></html>
```



CSS Shadow Effects

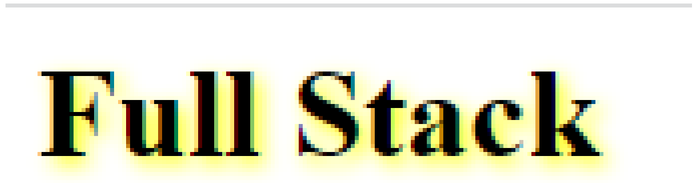
With CSS you can add shadow to text and to elements.

- text-shadow
- box-shadow

CSS Text Shadow

- The CSS text-shadow property applies shadow to text.
- Specify the horizontal shadow (2px) and the vertical shadow (2px): Blur effect (5px) and add color yellow

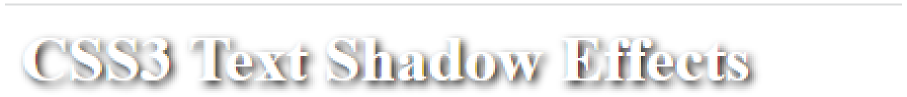
```
<head>
<style>
h1 {
  text-shadow: 2px 2px 5px yellow;
}
</style>
</head>
<body>

<h1>Full Stack</h1>

</body> </html>
```



CSS Text Shadow

- White text with black shadow

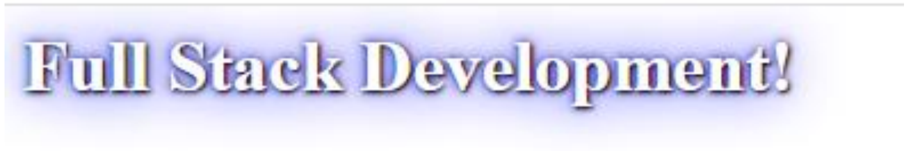
```
<html> <head>
<style>
h1 {
  color: white;
  text-shadow: 2px 2px 4px #000000;
}
</style>
</head>
<body>

<h1>CSS3 Text Shadow Effects</h1>

</body> </html>
```



CSS Text Shadow

- Shows a white text with black, blue, and darkblue shadow

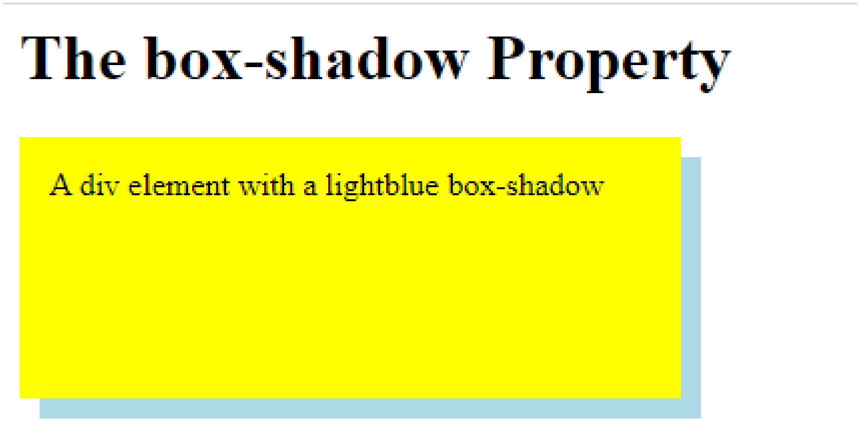
```
<html> <head>
<style>
h1 {
  color: white;
  text-shadow: 1px 1px 2px black, 0 0
25px blue, 0 0 5px darkblue;
}
</style> </head>
<body>
<h1>Full Stack Development!</h1>
</body></html>
```


Full Stack Development!

CSS Box Shadow

```
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  padding: 15px;
  background-color: yellow;
  box-shadow: 10px 10px lightblue;
}
</style>
</head>
```

```
<body><h1>The box-shadow Property</h1>

<div>A div element with a lightblue box-
shadow</div>

</body>
</html>
```



The box-shadow Property

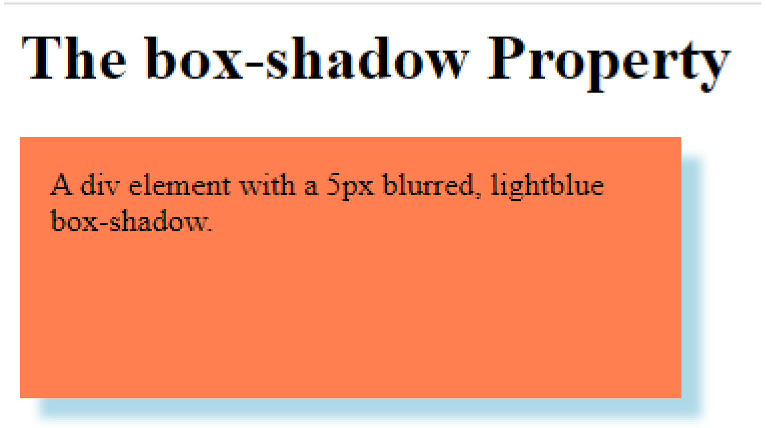
A div element with a lightblue box-shadow

CSS Box Shadow

Add blur effect to the Shadow

```
<html><head>
<style>
div {
  width: 300px;
  height: 100px;
  padding: 15px;
  background-color: coral;
  box-shadow: 10px 10px 5px lightblue;
}
</style>
</head>
```

```
<body>
<h1>The box-shadow Property</h1>

<div>A div element with a 5px blurred,
lightblue box-shadow.</div>

</body> </html>
```



The box-shadow Property

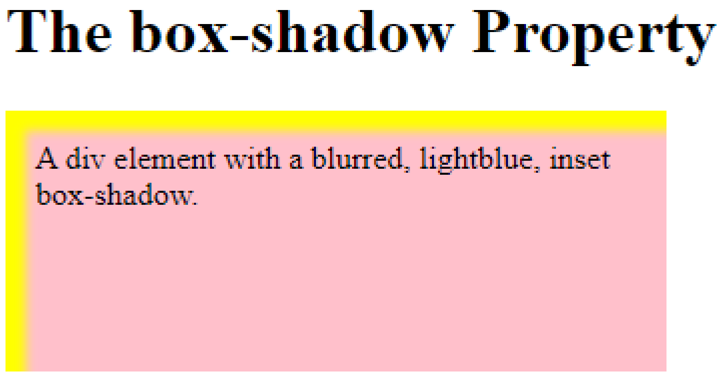
A div element with a 5px blurred, lightblue box-shadow.

CSS Box Shadow

Set the inset parameter: It changes the shadow from an outer shadow (outset) to an inner shadow

```
<html><head>
<style>
div {
  width: 300px;
  height: 100px;
  padding: 15px;
  background-color: pink;
 box-shadow: 10px 10px 5px yellow inset;
}
</style></head>
```

```
<body>
<h1>The box-shadow Property</h1>
<div>A div element with a blurred, lightblue,
inset box-shadow.</div>
</body></html>
```



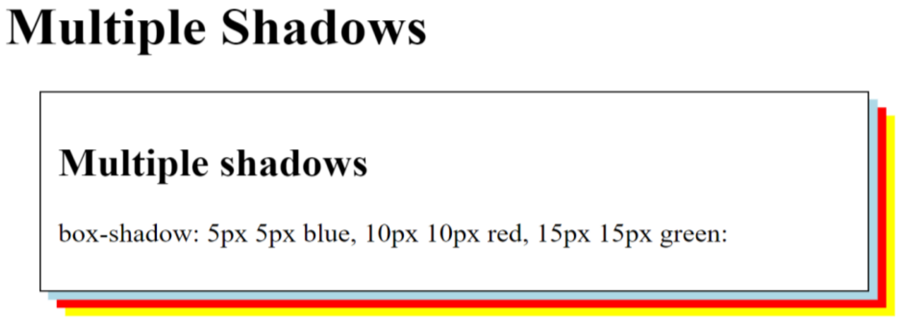
The box-shadow Property

A div element with a blurred, lightblue, inset box-shadow.

CSS Box Shadow – Multiple Shadows

```
<html><head>
<style>
#example1 {
  border: 1px solid;
  padding: 10px;
  box-shadow: 5px 5px lightblue, 10px
10px red, 15px 15px yellow;
  margin: 20px;
}
</style></head>

<body>
<h1>Multiple Shadows</h1>
```

```
<div id="example1">
  <h2>Multiple shadows</h2>
  <p>box-shadow: 5px 5px blue, 10px 10px
red, 15px 15px green:</p>
</div>
<br>
</body> </html>
```



Multiple Shadows

Multiple shadows

box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green:

CSS - Transitions

- CSS transitions allows you to change property values smoothly, over a given duration.
- We will learn about the following properties:

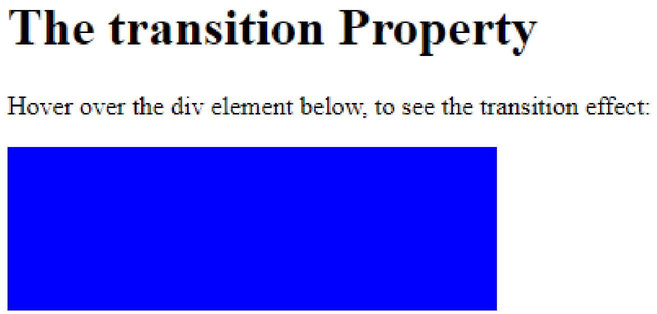
- **Transition:** A shorthand property for setting the four transition properties into a single property
- **transition-delay:** Specifies a delay (in seconds) for the transition effect
- **transition-duration:** Specifies how many seconds or milliseconds a transition effect takes to complete
- **transition-property:**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

CSS - Transition

- To create a transition effect, you must specify two things:
  - the CSS property you want to add an effect to
  - the duration of the effect
- The transition effect will start when the specified CSS property (width) changes value.

```
<html><head>
<style>
div {
  width: 100px;
  height: 100px;
  background: blue;
  transition: width 2s;
}
div:hover {
  width: 300px;
}
</style></head>
```

```
<body>
<h1>The transition Property</h1>
<p>Hover over the div element below, to see
the transition effect:</p>
<div></div>
</body> </html>
```



The transition Property

Hover over the div element below, to see the transition effect:

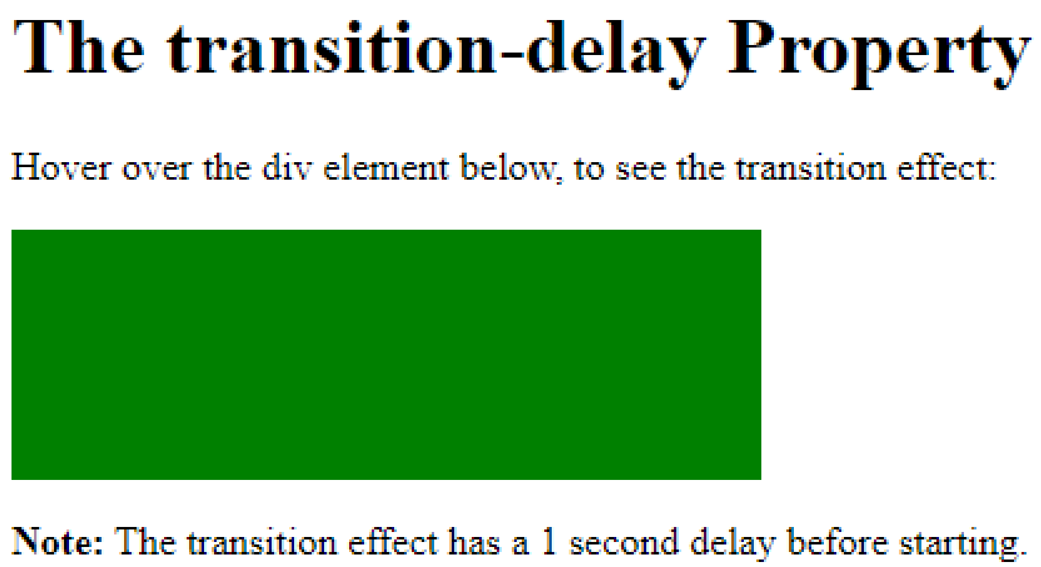
CSS – Transition-Delay

The transition-delay property specifies a delay (in seconds) for the transition effect.

```
<html><head>
<style>
div {
  width: 100px;
  height: 100px;
  background: green;
  transition: width 3s;
  transition-delay: 1s;
}
div:hover {
  width: 300px;
}
</style>
</head>
```

```
<body>

<h1>The transition-delay Property</h1>

<p>Hover over the div element below, to see
the transition effect:</p>

<div></div>

<p><b>Note:</b> The transition effect has a
1 second delay before starting.</p>

</body>
</html>
```

CSS – Transition-Delay



The transition-delay Property

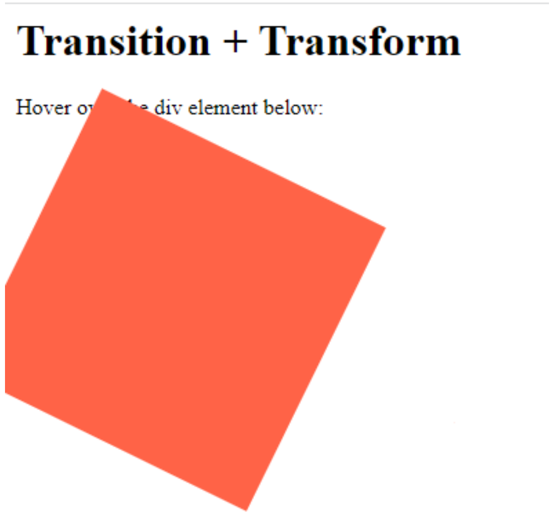
Hover over the div element below, to see the transition effect:

Note: The transition effect has a 1 second delay before starting.

CSS – Transition-Transform

```
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: tomato;
  transition: width 2s, height 2s, transform 2s;
}

div:hover {
  width: 300px;
  height: 300px;
  transform: rotate(180deg);
}
</style> </head>
```

```
<body>
<h1>Transition + Transform</h1>
<p>Hover over the div element below:</p>
<div></div>
</body> </html>
```


Transition + Transform

Hover o… …e div element below:

CSS – Transition-timing-function

- The transition-timing-function property specifies the speed curve of the transition effect.

- The transition-timing-function property can have the following values:

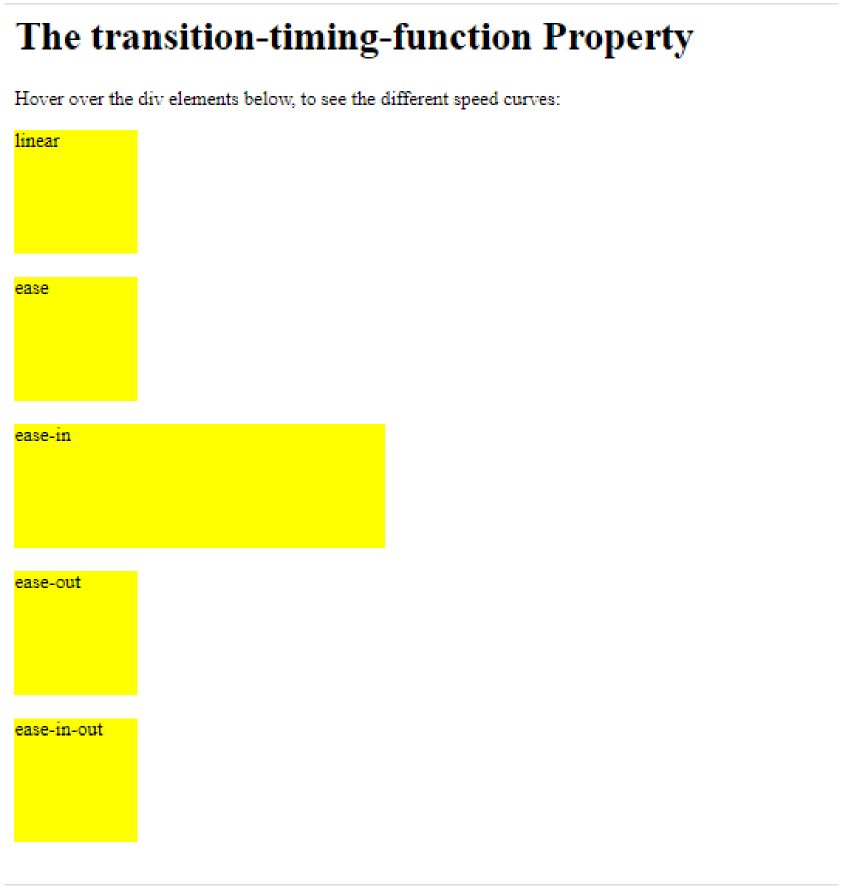
- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- linear - specifies a transition effect with the same speed from start to end
- ease-in - specifies a transition effect with a slow start
- ease-out - specifies a transition effect with a slow end
- ease-in-out - specifies a transition effect with a slow start and end

CSS – Transition-timing-function

```
<head> <style>
div {
  width: 100px;
  height: 100px;
  background: yellow;
  transition: width 2s;
}
#div1 {transition-timing-function: linear;}
#div2 {transition-timing-function: ease;}
#div3 {transition-timing-function: ease-in;}
#div4 {transition-timing-function: ease-out;}
#div5 {transition-timing-function: ease-in-out;}

div:hover {
  width: 300px;
}
</style> </head>
```

```
<body>

<h1>The transition-timing-function
Property</h1>

<p>Hover over the div elements below, to see
the different speed curves:</p>

<div id="div1">linear</div><br>
<div id="div2">ease</div><br>
<div id="div3">ease-in</div><br>
<div id="div4">ease-out</div><br>
<div id="div5">ease-in-out</div><br>

</body>
</html>
```

CSS – Transition-timing-function



The transition-timing-function Property

Hover over the div elements below, to see the different speed curves:

linear

ease

ease-in

ease-out

ease-in-out

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.
- 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.

CSS Animations – Keyframes Rule

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

```
<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>
```

CSS Animations – Keyframes Rule

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

```
<html> <head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}
</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>
```

CSS Animations – Keyframes Rule

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

```
<html><head>
<style>
div {  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;  }
@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  75%  {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
</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>
```

CSS Animations – Animation Delay

The animation-delay property specifies a delay for the start of an animation.

```
<html><head>
<style>
div {  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
  animation-delay: 2s; }
@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  75%  {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>

<h1>CSS Animation</h1>

<p>The animation-delay property
specifies a delay for the start of an
animation. The following example has a 2
seconds delay before starting the
animation:</p>

<div></div>

</body></html>
```

CSS Animations – Animation Direction

- The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles.

- The animation-direction property can have the following values:
- normal - The animation is played as normal (forwards). This is default
- reverse - The animation is played in reverse direction (backwards)
- alternate - The animation is played forwards first, then backwards
- alternate-reverse - The animation is played backwards first, then forwards

CSS Animations – Iteration Count, Reverse

- The animation-iteration-count property specifies the number of times an animation should run.
- The following example will run the animation 3 times before it stops:

```
<html><head>
<style> div {  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
  animation-iteration-count: 3; //infinite
  animation-direction: reverse;  }
@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  75%  {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
```

```
</style> </head>
<body>

<h1>CSS Animation</h1>

<p>The animation-iteration-count property specifies the number of times an animation should run. The following example will run the animation 3 times before it stops:</p>

<div></div>

</body>
</html>
```

CSS – Speed Curve of the Animation

- The animation-timing-function property specifies the speed curve of the animation.
- The animation-timing-function property can have the following values:
  - ease - Specifies an animation with a slow start, then fast, then end slowly (this is default)
  - linear - Specifies an animation with the same speed from start to end
  - ease-in - Specifies an animation with a slow start
  - ease-out - Specifies an animation with a slow end
  - ease-in-out - Specifies an animation with a slow start and end

CSS Animations – Timing Function

```
<html><head>
<style> div {
  width: 100px;
  height: 50px;
  background-color: yellow;
  font-weight: bold;
  position: relative;
  animation: mymove 5s infinite; }
#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}
@keyframes mymove {
  from {left: 0px;}
  to {left: 300px;} }
</style></head>
```

```
<body>

<h1>CSS Animation</h1>

<p>The animation-timing-function property
specifies the speed curve of the animation. The
following example shows some of the different
speed curves that can be used:</p>

<div id="div1">linear</div>
<div id="div2">ease</div>
<div id="div3">ease-in</div>
<div id="div4">ease-out</div>
<div id="div5">ease-in-out</div>

</body>
</html>
```

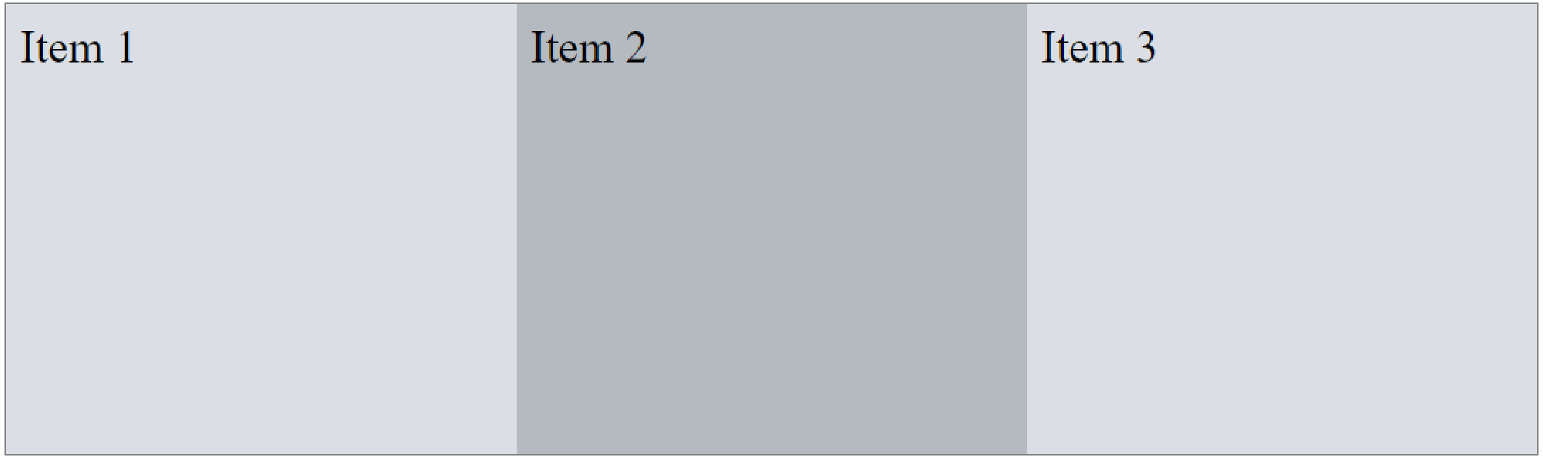
CSS Flexbox

- Flexible box, commonly referred to as flexbox, is a new layout model introduced in CSS3 for creating the flexible user interface design with multiple rows and columns without using the percentage or fixed length values.
- The CSS3 flex layout model provides a simple and powerful mechanism for handling the distributing of space and alignment of content automatically.
- There were four layout modes:
    - Block, for sections in a webpage
    - Inline, for text
    - Table, for two-dimensional table data
    - Positioned, for explicit position of an element

CSS Columns-Flexbox

```
<html><head>
<title>Example of CSS3 Three Equal Flex Column Layout</title>
<style>
.flex-container {
            width: 80%;
            min-height: 300px;
            margin: 0 auto;
            font-size: 32px;
            display: flex;
            border: 1px solid #808080;
}
.flex-container div {
            padding: 10px;
            background: #dbdfe5;
            flex: 1;
}
.flex-container div.bg-alt{
            background: #b4bac0;
}
</style>
</head>
<body>
  <div class="flex-container">
    <div class="item1">Item 1</div>
    <div class="item2 bg-alt">Item 2</div>
    <div class="item3">Item 3</div>
  </div>
  <p><strong>Note:</strong> If you change the width of the flex-container the width of the flex items will automatically adjusted without doing anything.</p>
</body>
</html>
```

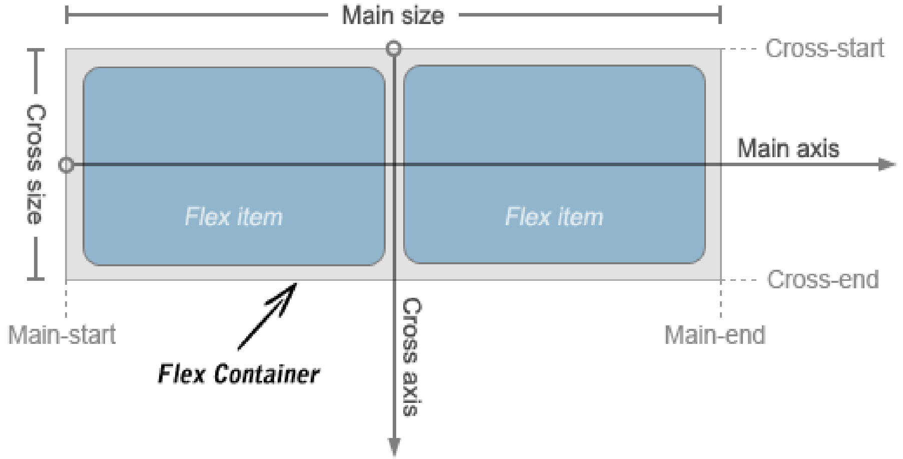
CSS Columns-Flexbox



Note: If you change the width of the flex-container the width of the flex items will automatically adjusted without doing anything.

CSS Flex Layout

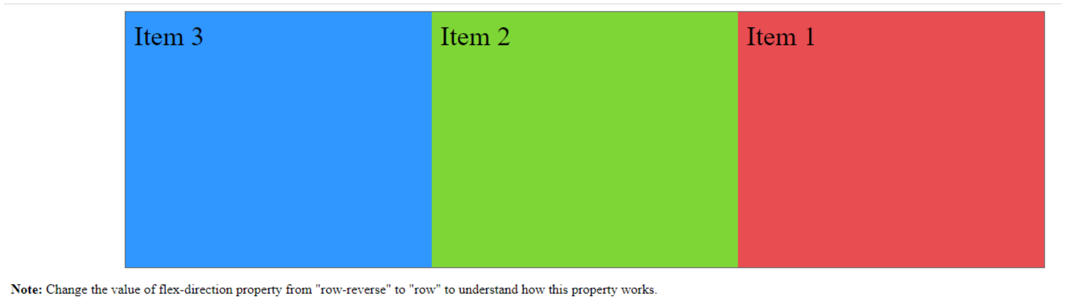
- Flexbox consists of flex containers and flex items.
- A flex container can be created by setting the display property.
- All child elements of flex container automatically become flex items and are laid out using the flex layout model.
- Flex items are positioned inside a flex container along a flex line controlled by the flex-direction property.



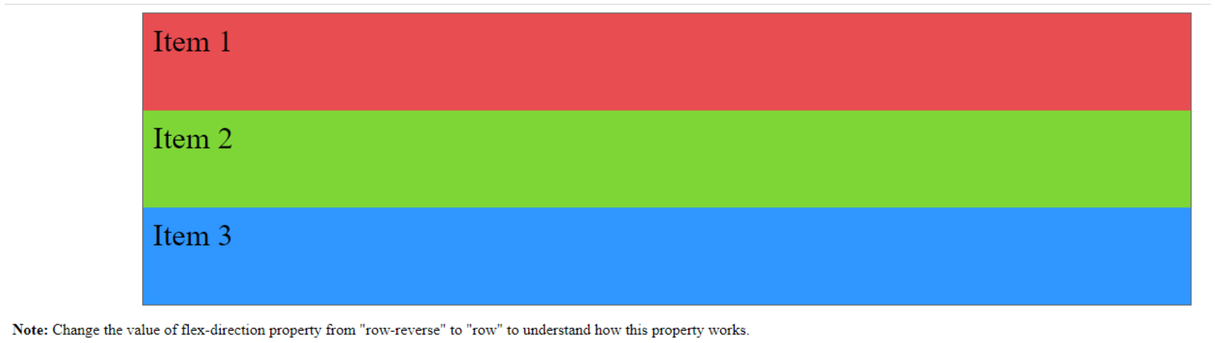
CSS Columns-Flexbox – Direction

```
<html><head>
<title>Example of Controlling Flow inside Flex
Container along Main Axis</title>
<style>
.flex-container {
   width: 80%;
   min-height: 300px;
   margin: 0 auto;
   font-size: 32px;
   display: flex;
   flex-direction: row-reverse; //column
   border: 1px solid #666;
}
.flex-container div {
              padding: 10px;
              flex: 1;
}
.item1 {      background: #e84d51; }
.item2 {      background: #7ed636; }
.item3 {      background: #2f97ff; }
</style>
</head>
<body>
   <div class="flex-container">
      <div class="item1">Item 1</div>
      <div class="item2">Item 2</div>
      <div class="item3">Item 3</div>
   </div>
   <p><strong>Note:</strong> Change the value of
flex-direction property from "row-reverse" to
"row" to understand how this property works.</p>
</body>
</html>
```

CSS Columns-Flexbox – Direction



Note: Change the value of flex-direction property from "row-reverse" to "row" to understand how this property works.



Note: Change the value of flex-direction property from "row-reverse" to "row" to understand how this property works.

End of Unit I