HTML 5

The term HTML5 is essentially a buzzword that refers to a set of modern web technologies. This includes the HTML Living Standard, along with JavaScript APIs to enhance storage, multimedia, and hardware access.

Advantages of HTML 5:

1. Cleaner markup/ Improved code: HTML5 will enable web designers to use cleaner, neater code. We can remove div tags and replace them with semantic HTML5 elements.

2. Elegant forms: HTML5 enables designer to use fancier forms. There will be different type of inputs, search and different fields for different purpose.

3. Consistency: As websites will adopt the new HTML5 elements we will see more consistency in terms of HTML used to code a web page on one site compared to another. This will make it much easier for designers and developers to immediately understand how a web page is created.

4. Supports rich media elements: HTML5 has an inbuilt capability to play audio and video and so we can bid goodbye to those plugin tags.

5. Offline Application Cache: The Application Cache concept means that a web application is cached. It can be accessible without the need for internet connection. Some advantages of Application Cache:

1. Offline browsing – Web users can also use the application when they are offline.
2. Speed – Cached resources load quicker
3. Reduce the server load – The web browser will only download updated resources from the server.

6. No more cookies – As HTML 5 uses web storage, there is no need of using cookies.

HTML 5 features:

Html 5 has introduced some new features they are:

1.semantic elements
2. forms
3.web storage
4. canvas,
5. audio and video,
6. Geolocation
7.web socket
8. server sent events (SSE),
9.micro data
10.drag and drop.

New semantic elements: A semantic element clearly describes its meaning to both the browser and the developer.
The example of the non-semantic elements are 1. , 2. they tells nothing about its contents.
The examples of semantic elements are 1.<form>,2.<table>,3.<article>they clearly define its content.

Forms: uses forms 2.0 to facilitate UI design. Forms are used to collect the data from the user. In HTML file form has have more semantic elements. In form 2.0 in addition to form input types and several new input types are added.

The following new input types are added 1. datetime-local ,2. month, 3. week, 4. date, 5. time, 6. number, 7. range, 8. email, 9.url.

The <output> tag is introduced to display the output. The attributes placeholder, autofocus and required are added.

Web storage API (1. local storage 2. session storage)
Web storage API provides mechanisms by which browsers can store key/value pairs in a much better way than using cookies. the two mechanisms within web storage are as follows:

1.**SessionStorage:** maintenance a separate area for each given origin that's available for the duration of the page session (as long as the browser is open including page reloaded and restores). Stores data only for a session meaning that the data is stored until the browser (or tab) is closed. the storage limit is larger than a cookie (at most 5MB).

2.**localStorage:** does the same thing but persists even when the browser is closed and reopened. Stores data with no expiration date and gets clear only through java script or cleaning the browser cache /locally stored data. Storage limit is larger than the cookie and session storage.

Two main benefits of HTML5 Web Storage:

- It can store up to 10 MB data which is certainly more than what cookies have.
- Web storage data cannot be transferred with the HTTP request. It helps to increase the performance of the application.

window.localStorage and window.sessionStorage are the two methods in which methods will be present in HTML5.

- **window.localStorage:** stores data with no expiration date.
- **window.sessionStorage:** stores data for one session

Canvas: it allows drawing the user interface. Compare to SVG, it is more effective and user friendly. It allows java script code to perform 2D drawing. <canvas> is the tag to be used in association with java script.

Audio & video: HTML 5 supports <audio> and <video> tags for involving multimedia content in the HTML pages.

Geolocation API: The HTML Geolocation API is used to get the geographical position of a user. Since this can compromise privacy, the position is not available unless the user approves it.

Web socket: Web Sockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers.

Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a send() method, and receive data from server to browser by an onmessage() event handler.

Server sent events (SSE): A server-sent event is when a web page automatically gets updates from a server. This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.

Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

Micro data: Microdata is a standardized way to provide additional semantics in your web pages. Microdata lets you define your own customized elements and start embedding custom properties in your web pages.

At a high level, microdata consists of a group of name-value pairs. The groups are called items, and each name-value pair is a property. Items and properties are represented by regular elements.

Drag and drop API: In HTML, any element can be dragged and dropped. Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location. This feature is basically related to data transfer feature in HTML5. It is implemented by associating HTML with java script.

In fact, some of the above feature's HTML are implemented through API. They are:

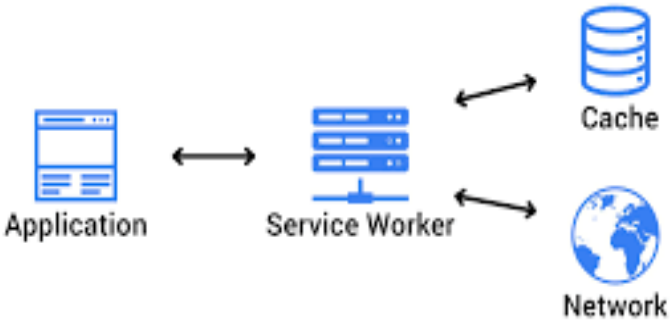
- Geolocation API
- Drag/Drop API
- Web storage API
- Web workers API
- Web SSE

Web workers: Web workers make it possible to run a script operation in a background third separate from the main execution thread of a web application. Advantage of this is that laborious processing can be performed in a separate thread along the main (usually the UI) thread to run without being blocked/slowed down.

There are two different types of web workers in HTML5 i.e., Dedicated Workers and Shared Workers.

Service workers:

Service workers essentially act as proxy servers that sit between web applications, the browser, and the network (when available). They are intended, among other things, to enable the creation of effective offline experiences, intercept network requests and take appropriate action based on whether the network is available, and update assets residing on the server. They will also allow access to push notifications and background sync APIs.



Some of the tags that are removed from the HTML 5 are:

<applet>, <font>, <frame>, <frameset>, <noframe>, <big>, <acronym>, , <tt>.

HTML5 tags are classified into Semantic and Non-semantic elements.

1. Non-semantic elements are: , and <p>.
2. Semantic elements have meaningful names which tells about type of content. For example, header, footer, table, ... etc. HTML5 introduces many semantic elements as mentioned below which make the code easier to write and understand for the developer as well as instructs the browser on how to treat them.

    - article
    - aside
    - details
    - figcaption
    - figure
    - footer
    - header
    - main
    - mark
    - nav
    - section

header: The **<header>** HTML element represents introductory content, typically a group of introductory or navigational aids. It may contain some heading elements but also a logo, a search form, an author name, and other elements.

```
<header>
  <h1>Main Page Title</h1>
  <img src= “college.png" alt="My College logo">
</header>
```

Footer: The **<footer>** HTML element represents a footer for its nearest sectioning content or sectioning root element. A <footer> typically contains information about the author of the section, copyright data or links to related documents.

```
<article>
   <h1>How to be a Professional</h1>
   <ol>
      <li>Be strong in Academics</li>
      <li>Be strong in Problem Solving</li>
      <li>Be an active member in SAC, IUCEE, EWB, IEC, UIF</li>
   </ol>
   <footer>
      <p>© 2022 VVIT, Nambur</p>
   </footer>
</article>
```

article: The **<article>** HTML element represents a self-contained composition in a document, page, application, or site, which is intended to be independently distributable or reusable (e.g., in syndication). Examples include: a forum post, a magazine or newspaper article, or a blog entry, a product card, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.
Ex:

```
<article>

  <header>
   <h2>The Planet Earth</h2>
   <p>
         Posted on Wednesday,
         <time datetime="2022-03-22">22nd March 2022</time>
         by CSE Dept</p>
  </header>

  <p>We live on a planet with so many things still unseen</p>
```

```
  <p><a href="https://example.com/the-planet-earth/">
          Continue reading....
  </a></p>

</article>
```

Aside: The **<aside>** HTML element represents a portion of a document whose content is only indirectly related to the document's main content. Asides are frequently presented as sidebars or call-out boxes.

Ex:

```
<p>Vasireddy Venkatadri Institute of Technology (VVIT) was established in the year 2007, with an intake of 240 students in four B. Tech programs under Social Educational Trust in Nambur village, Guntur, AP, by Er. Vasireddy Vidya Sagar
</p>

<aside>

   <p>VVIT is one among the top colleges in A.P region in securing more placements</p>

</aside>

<p>Realizing the slogan that “Everyone Counts,” VVIT has the unique achievement of having placed most its students in MNCs like Tech Mahindra, Infosys, TCS, Wipro, etc. (almost 50 companies) every year</p>
```

nav: The **<nav>** HTML element represents a section of a page whose purpose is to provide navigation links, either within the current document or to other documents. Common examples of navigation sections are menus, tables of contents, and indexes.

```
<nav class="menu">
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>
```

CSS File:

```
aside {
   width: 40%;
   padding-left: .5rem;
   margin-left: .5rem;
   float: right;
```

```
    box-shadow: inset 5px 0 5px -5px #29627e;
    font-style: italic;
    color: #29627e;
}
aside > p {
    margin: .5rem;
}
p {
    font-family: 'Fira Sans', sans-serif;
}
```

Section: The **<section>** HTML element represents a generic standalone section of a document, which doesn't have a more specific semantic element to represent it. Sections should always have a heading, with very few exceptions.

Ex:

```
<h1>Choosing a right JOB </h1>
<section>
    <h2>Introduction</h2>
    <p>This document provides a guide to help with the important task of choosing the correct Job</p>
</section>
<section>
    <h2>Criteria</h2>
    <p>There are many different criteria to be considered when choosing a JOB — Salary, working hours, place, etc...</p>
</section>
```

Video & Audio: Before HTML 5 came into existence, videos could only be played in a browser using a plugin like flash. But after the release of HTML 5, adding a video and an audio to a webpage is as easy as adding an image. There are three different formats that are commonly supported by web browsers – mp4, Ogg, and WebM.

Syntax: <video src="" controls> </video>

Video Ex:

```
<!DOCTYPE html>
<html>
<body>
 <p>Adding Video on my webpage </p>
    <video width="400" height="350" controls>
```

```
    <source src="myvid.mp4" type="video/mp4">
    <source src="myvid.ogg" type="video/ogg">
  </video>
</body>
</html>
```

Attributes that can be used with the "video" tag are listed below:

1. **Autoplay:** It tells the browser to immediately start downloading the video and play it as soon as it can.
2. **Preload:** It intends to provide a hint to the browser about what the author thinks will lead to the best user experience.
3. **Loop:** It tells the browser to automatically loop the video.
4. **height:** It sets the height of the video in CSS pixels.
5. **width:** It sets the width of the video in CSS pixels.
6. **Controls:** It shows the default video controls like play, pause, volume, etc.
7. **Muted:** It mutes the audio from the video.
8. **Poster:** It loads an image to preview before the loading of the video.
9. **src:** It is used to specify the URL of the video file.

Audio: The HTML <audio> element is used to play an audio file on a web page.

Syntax/ex:

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
   Your browser does not support the audio element.
</audio>
```

Canvas: The HTML <canvas> element is used to draw graphics on a web page.

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.

Ex-1: <canvas id="myCanvas" width="200" height="100"></canvas>

Ex-2: <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;"> </canvas>

SVG: Earlier <svg> is used. The HTML <svg> element is a container for SVG graphics. SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

Ex:

```
<!DOCTYPE html>
<html>
<body>
<svg width="100" height="100">
<circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
</svg>
</body>
</html>
```

SVG Vs canvas:

Canvas	SVG
1.pixel graphics.	1.vector graphics.
2.Resolution dependent.	2.Resolution independent.
3.drawn with java script.	3.XML format.
4.modified through script only.	4.modified through css and script.
5.not suitable for printing at high resolution.	5.high quality print at any resolution.
6.suitable for games.	6.not suitable for games.

******

CSS3 (Cascading Style Sheets)

CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. In other words, it's a style sheet language that determines how the elements/contents in the page are looked/shown. CSS is used to develop a consistent look and feel for all the pages.

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS enables the separation of the content from the presentation. This separation provides a lot of flexibility and control over how the website has to look like. This is the main advantage of using CSS.

A CSS rule consists of a selector and a declaration block.

CSS syntax:

```
Selector {
            Property1: value1;
            Property2: value2;
            …..
      }
```

The selector can be a HTML element that we want to style.

Styling can be applied in 3 different ways:

1. Inline style sheets
2. Internal style sheets
3. External style sheets

Or

Local style, Page-Level style, and External Styles.

1. Local styles

Also known as inline. This form is defined within your HTML tags/elements. It's mostly used to style specific elements in your code.

```
<html>
   <head>
     <title>Cascading Style Sheets</title>
   </head>
   <body>
     <p style = "font-family: sans-serif;
        font-size: 1.2em
        font-style: italic;">
```

```
        This paragraph is an example of a local style.
      </p>
      <p>This is an Unaffected paragraph</p>
  </body>
</html>
```

This code snippet edits the font of the p tag within the body. However, it only changes the contents of the first p tag. The second p tag maintains the webpage's default style.

Pros:

- Easy to test designs on individual elements.
- You can use one document to load CSS styles.

Cons:

- It can take a long time to load or render a page since multiple individual elements are styled within.
- This final file will look disorganized, thus making it harder to read.

2. Page-Level styles

Page-level styles are defined at the header area of the HTML file. All similar tags, whether elements members of the class or ID selector within the body of the HTML will undergo the changes at once. An ID selector can only identify one element each while Class selectors can identify more than just one at a time.

```
<html>
  <head>
    <title>Cascading Style Sheets</title>
    <meta charset="utf-8">
    <style type="text/css">
      body {
        color: yellow;
        background-color: red;
      }
      p {
        color: red;
        background-color: yellow;
      }
    </style>
  </head>
  <body>
    <h1>Heading</h1>
    <p>This paragraph has been styled using page level styling</p>
  </body>
</html>
```

Pros:

- Easy to manage during initial development stages.

- The programmer doesn't need to keep switching files to make minor adjustments.

Cons:

- Adding code to the document increases the size of the file.
- In order to manipulate a website design, you would have to acquire all the files that contain CSS.

3. External Styles

The styles used for the webpage are located in a completely different file. This other file purely contains CSS code and is saved with a .css extension. The .HTML file is linked to the .css file that can be imported to modify the webpage style.

```
<!DOCTYPE html>
 <html>
  <head>
    <meta charset="utf-8">
    <title>External Styles</title>
    <link rel="stylesheet" type="text/css" href="css/myStyle.css"/>
  </head>
  <body>
   <h1>Heading</h1>
    <p>This is an example of External CSS</p>
  </body>
 </html>

/*This is the CSS file called myStyle.css*/

 body {
  background-color: black;
  color: white;
 }

 p {
  color: purple;
 }
```

Pros:

- One style sheet controls many pages.
- Separation of content and design.

Cons:

- You cannot render the page until the entire CSS file is downloaded.
- Linking multiple CSS files to the page can lead to downtime.

Features of CSS3:

1. CSS Animations and Transitions

2. Calculating values with calc()
3. Advanced Selectors
4. Generated Content and Counters
5. Gradient
6. Web fonts
7. Box sizing
8. Border images
9. Media queries
10. Multiple backgrounds
11. CSS columns
12. CSS 3D transforms

Some of the key modules are:

- Box model
- Image values and replaced content
- Text effects
- Selectors
- Backgrounds and borders
- Animations
- User interface (UI)
- Multiple column layouts
- 2D/3D transformations

Advantages of CSS3

- CSS3 provides a consistent and precise positioning of navigable elements.
- It is easy to customize a web page as it can be done by merely altering a modular file.
- Graphics are easier in CSS3, thus making it easy to make the site appealing.
- It permits online videos to be seen without using third-party plug-ins.
- CSS3 is economical, time-saving, and most browsers support it.

CSS frameworks are the pre-planned libraries which make easy and more standard compliant web page styling. The frequently used CSS frameworks are: -

- Bootstrap
- Foundation
- Semantic UI
- Gumby
- Ulkit

CSS selectors:

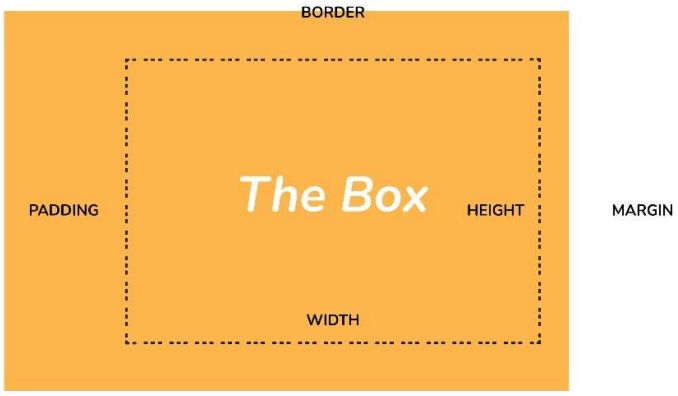
A CSS selector selects the HTML element(s) you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

CSS Box Model and the related properties:

A rectangle box is wrapped around every HTML element. The box model is used to determine the height and width of the rectangular box. The CSS Box consists of Width and height (or in the absence of that, default values and the content inside), padding, borders, margin.



- **Content:** Actual Content of the box where the text or image is placed.
- **Padding:** Area surrounding the content (Space between the border and content).
- **Border:** Area surrounding the padding.
- **Margin:** Area surrounding the border.

Ex:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 100px;
  border: 15px solid green;
  padding: 50px;
  margin: 15px;
  font-size: 20px;
}
</style>
</head>
```

```
<body>
<h2>Demonstrating the Box Model</h2>
<div>CSE</div>
<div>AI and DS</div>
<div>AI and ML</div>
</body>
</html>
```

Specificity:

If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.

Think of specificity as a score/rank that determines which style declaration are ultimately applied to an element.

Look at the following examples: In this example, we have added a class selector (named "test"), and specified a green color for this class. The text will now be green (even though we have specified a red color for the element selector "p". This is because the class selector is given higher priority:

```
<!DOCTYPE html>
<html>
<head>
<style>
.test {color: green;}
    p {color: red;}
</style>
</head>
<body>
<p class="test">Hello World! </p>
</body>
</html>
```

Advanced Features:

1) Rounder corners to the elements:

```
<!DOCTYPE html>
<html>
<head>
<style>
#rcorners1 {
  border-radius: 25px;
  background: #73AD21;
```

```
  padding: 20px;
  width: 100px;
  height: 50px;
  text-align: center;
}

#rcorners2 {
  border-radius: 25px;
  background: url(paper.gif);
  background-position: left top;
  background-repeat: repeat;
  padding: 20px;
  width: 100px;
  height: 50px;
}
</style>
</head>
<body>

<h1>The border-radius Property</h1>
<p>Rounded corners for an element with a specified background color:</p>
<p id="rcorners1">Rounded corners</p>
<p id="rcorners2">Rounded corners</p>
</body>
</html>
```

2) CSS Colors:

RGBA Colors: RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color. An RGBA color value is specified with: rgba(red, green, blue, alpha). The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Ex:

```
<!DOCTYPE html>
<html>
<head>
<style>
#p1 {background-color:rgba(255,0,0,0.3);}
#p2 {background-color:rgba(0,255,0,0.3);}
#p3 {background-color:rgba(0,0,255,0.3);}
</style>
</head>
<body>
```

```
<h1>Define Colors With RGBA Values</h1>

<p id="p1">Red</p>
<p id="p2">Green</p>
<p id="p3">Blue</p>

</body>
</html>
```

HSL Colors: HSL stands for Hue, Saturation and Lightness. An HSL color value is specified with: hsl(hue, saturation, lightness).

1. Hue is a degree on the color wheel (from 0 to 360):
    - 0 (or 360) is red
    - 120 is green
    - 240 is blue
2. Saturation is a percentage value: 100% is the full color.
3. Lightness is also a percentage; 0% is dark (black) and 100% is white.

Ex:

```
<!DOCTYPE html>
<html>
<head>
<style>
#p1 {background-color:hsl(120,100%,50%);}
#p2 {background-color:hsl(120,100%,75%);}
#p3 {background-color:hsl(120,100%,25%);}
</style>
</head>
<body>

<h1>Define Colors With HSL Values</h1>

<p id="p1">Green</p>
<p id="p2">Light green</p>
<p id="p3">Dark green</p>

</body>
</html>
```

Opacity

The CSS opacity property sets the opacity for the whole element (both background color and text will be opaque/transparent).

The opacity property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
<!DOCTYPE html>
<html>
<head>
<style>
#p1 {background-color:rgb(255,0,0);opacity:0.6;}
#p2 {background-color:rgb(0,255,0);opacity:0.6;}
#p3 {background-color:rgb(0,0,255);opacity:0.6;}

</style>
</head>
<body>

<h1>Define Colors With Opacity</h1>

<p id="p1">Red</p>
<p id="p2">Green</p>
<p id="p3">Blue</p>

</body>
</html>
```

CSS Backgrounds: The CSS background properties are used to add background effects for elements. The following are the CSS background properties:

- `background-color`
- `background-image`
- `background-repeat`
- `background-attachment`
- `background-position`
- `background` (shorthand property)

background-color: with CSS, a color is most often specified by:

- a valid color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

background-image: The `background-image` property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element.

```
Body { background-image: url("paper.gif");

}
```

background-repeat: By default, the `background-image` property repeats an image both horizontally and vertically. Some images should be repeated only horizontally or vertically, or they will look strange, like this:

repeat-x, repeat-y, no-repeat

```
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
}
```

background-position

The `background-position` property is used to specify the position of the background image.

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

background-attachment

The `background-attachment` property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
}
```

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: scroll;
}
```

CSS Animation:

Ex:

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

@keyframes sample {
  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 Transitions

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

Mouse over the element below to see a CSS transition effect:

- transition
- transition-delay
- transition-duration

- transition-property
- transition-timing-function

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

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

Ex:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  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>
```

CSS 2D Transforms: CSS transforms allow you to move, rotate, scale, and skew elements. Mouse over the element below to see a 2D transformation. With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()

- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()
- matrix()

Ex:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
div#one {
  transform: rotate(20deg);
}
div#two {
 transform: rotate(-20deg);
}
div#three {
 transform: scale(2, 3);
}
div#four {
 transform: scale(0.5, 0.5);
}
div#five {
 transform: scaleX(2);
}
</style>
</head>
<body>
<h1>The rotate() Method</h1>
<p>The rotate() method rotates an element clockwise or counter-clockwise.</p>

<div>
This a normal div element.
```

```
</div>
<div id="one">
This div element is rotated clockwise 20 degrees.
</div>
<div id="two">
This div element is rotated counter clockwise 20 degrees.
</div>
<div id="three">
This div element is scaled
</div>
<div id="four">
This div element is reduced
</div>
<div id="five">
This div scaling in X direction
</div>
</body>
</html>
```

CSS Multiple Columns:

Ex:

```
<!DOCTYPE html>
<html>
<head>
<style>
.newspaper {
  column-count: 3;
}
</style>
</head>
<body>

<h1>Javascript notes in Create Multiple Columns</h1>

<div class="newspaper">
```

JavaScript is a lightweight, cross-platform, and interpreted scripting language. It is well-known for the development of web pages; many non-browser environments also use it. JavaScript can be used for Client-side developments as well as Server-side developments. JavaScript contains a standard library of objects, like Array, Date, and Math, and a core set of language elements like operators, control structures, and statements.

```
</div>

</body>
</html>
```

CSS Flex: Before the Flexbox Layout module, 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

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

The flex container properties are:

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

Ex:

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  background-color: #FF0000;
}

.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
</style>
</head>
```

```
<body>
<h1>Create a Flex Container</h1>
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
</body>
</html>
```

Web Workers in ReactJS are used to run tasks in the background without blocking the main UI thread. This can be useful for CPU-intensive operations, such as large computations, parsing large datasets, or handling complex animations. Here's a simple ReactJS example to demonstrate how to use Web Workers.

Steps:
1.Create a Web Worker file (worker.js).
2.Use the Web Worker in your React component (App.js).

1. worker.js

```
// worker.js
// A simple worker that calculates the nth Fibonacci number
self.onmessage = function (e) {
  const number = e.data;
  const result = fibonacci(number);
  postMessage(result);
};

// A recursive function to calculate Fibonacci
function fibonacci(n) {
  if (n <= 1) return n;
  return fibonacci(n - 1) + fibonacci(n - 2);
}
```

2. App.js (React component)

```
import React, { useState, useEffect } from 'react';
function App() {
  const [worker, setWorker] = useState(null);
  const [inputValue, setInputValue] = useState(0);
  const [result, setResult] = useState(null);
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    // Create a new worker instance
    const newWorker = new Worker(new URL('./worker.js', import.meta.url));

    // Listen to messages from the worker
    newWorker.onmessage = (e) => {
      setResult(e.data);  // Get the result from the worker
      setLoading(false);   // Stop loading when we have the result
    };

    // Set the worker in the state
```

```
    setWorker(newWorker);

    // Clean up the worker on component unmount
    return () => {
      if (worker) {
        worker.terminate();
      }
    };
  }, []);

  const calculateFibonacci = () => {
    if (worker) {
      setLoading(true);
      worker.postMessage(inputValue); // Send input to the worker
    }
  };

  return (
    <div style={{ padding: "20px" }}>
      <h2>Web Worker Fibonacci Calculator</h2>
      <input
        type="number"
        value={inputValue}
        onChange={(e) => setInputValue(e.target.value)}
        placeholder="Enter a number"
      />
      <button onClick={calculateFibonacci}>Calculate Fibonacci</button>

      {loading ? <p>Calculating...</p> : result !== null && <p>Result: {result}</p>}
    </div>
  );
}

export default App;
```