

OVER
40
YEARS
OF ACADEMIC
WISDOM



PRESIDENCY UNIVERSITY

CSE3150 – Front-end Full Stack Development



Department of Computer Science Engineering
School of Engineering

Module I - Syllabus

HTML5 – Syntax, Attributes, Events, Web Forms 2.0, Web Storage, Canvas, Web Sockets; CSS3 – Colors, Gradients, Text, Transform

Assignment: Develop a website for managing HR policies of a department.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



INTRODUCTION TO HTML5

What is HTML5?

HTML5 is the latest version of HTML, designed to structure and present web content effectively.

Key Features:

- **Simplified Syntax:** `<!DOCTYPE html>`
- **Semantic Elements:** `<header>`, `<footer>`, `<section>`, `<article>`.
- **Multimedia Support:** `<audio>` and `<video>` tags for media playback.
- **Form Enhancements:** New input types like email, date, range.
- **Offline Support:** Using the Cache API for offline functionality.

Why HTML5?

- Better structure and readability.
- Native support for multimedia and graphics.
- Mobile-friendly and responsive design.

EDITOR

- **Visual Studio Code (VSCode):**
- **Sublime Text:**
- **Atom:**
- **Brackets:**
- **Notepad++:**
- **WebStorm:**
- **Adobe Dreamweaver:**
- **CodeSandbox:**



HTML SYNTAX

HTML5 is the latest version of HTML, designed for modern web development.

Key Features of HTML5 Syntax

- **Doctype:** Simplified as `<!DOCTYPE html>`.
- **Multimedia:** Supports `<audio>` and `<video>` tags.
- **Self-Closing Tags:** Optional closing slash (e.g., `` vs. ``).
- **Global Attributes:** Attributes like `id`, `class`, `style`, and `data-`
- **Tags:** Semantic tags for better structure.
- **Form Controls:** Improved form controls with new input types



Basic HTML5 Structure

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 Example</title>
  </head>
  <body>
    <header>
      <h1>Welcome to HTML5</h1>
    </header>
    <footer>
      <p>© 2025 My Website</p>
    </footer>
  </body>
</html>
```



HTML5 & HTML 4

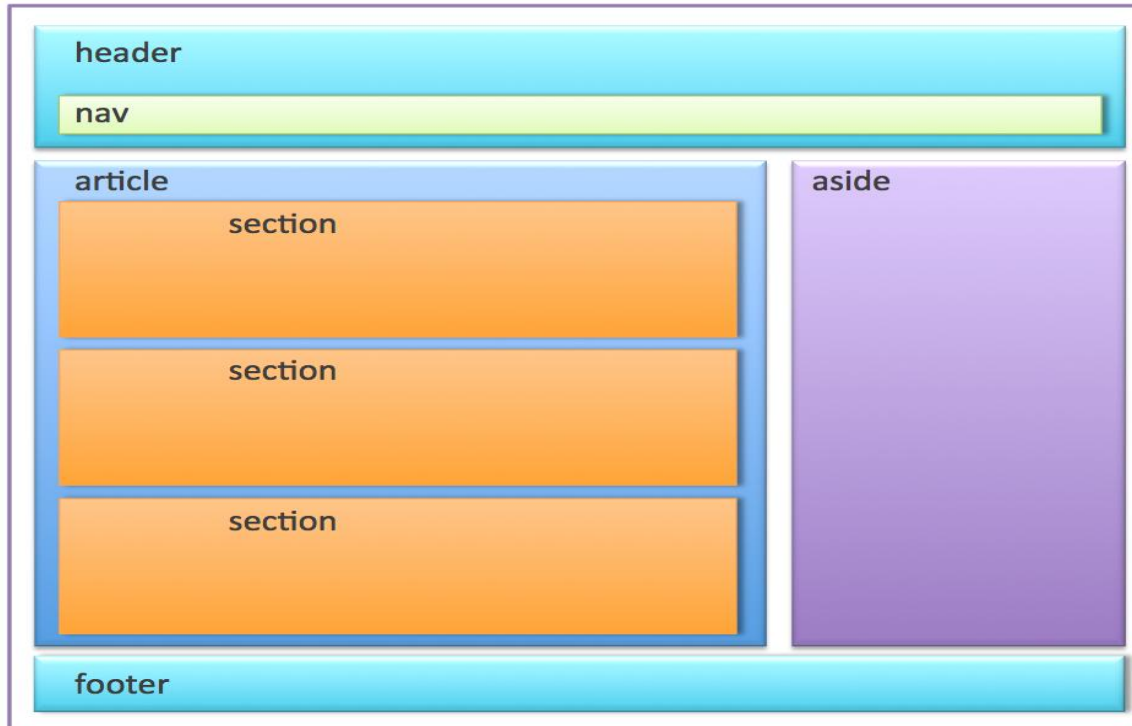
HTML 4

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.d
td">
<html>
<head> <title>HTML 4 Example</title>
</head>
<body>
<h1>Hello, HTML 4!</h1>
<p>This is a simple HTML 4
example.</p>
<ul> <li>Item 1</li> <li>Item 2</li>
<li>Item 3</li> </ul>
</body>
</html>
```

HTML 5

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
  <title>HTML5 Example</title>
</head>
<body>
  <header>
    <h1>Hello, HTML5!</h1>
  </header>
  <nav>
    <ul>
      <li>Home</li>
      <li>About</li>
      <li>Contact</li>
    </ul>
  </nav>
  <section>
    <article>
```

HTML5 Semantic Elements



This image is part of the Bioinformatics Web Development tutorial at http://www.cellbiol.com/bioinformatics_web_development/
© cellbiol.com, all rights reserved

HTML5 Semantic Elements

Why are they needed?

One substantial problem with modern, pre-HTML5 semantic markup:

most complex web sites are absolutely packed solid with `<div>` elements.

Unfortunately, all these `<div>` elements can make the resulting markup confusing and hard to modify.

Developers typically try to bring some sense and order to the `<div>` chaos by using id or class names that provide some clue as to their meaning.



Semantic Elements in HTML5

HTML5 introduces **semantic elements** for better structure and readability:

`<header>`: Defines a header for a document or section.

`<nav>`: Defines navigation links.

`<section>`: Represents a thematic grouping of content.

`<article>`: Represents standalone content.

`<footer>`: Defines a footer for a document or section.

Semantic Elements in HTML5

```
<header>
```

```
  <h1>My Blog</h1>
```

```
  <nav>
```

```
    <ul>
```

```
      <li><a href="#home">Home</a></li>
```

```
      <li><a href="#about">About</a></li>
```

```
    </ul>
```

```
  </nav>
```

```
</header>
```



Header and Footer

Most web site pages have a recognizable header and footer section.

Typically the **header** contains

- the site logo
- title (and perhaps additional subtitles or taglines)
- horizontal navigation links, and
- perhaps one or two horizontal banners.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Header and Footer

The typical footer contains less important material, such as

- smaller text versions of the navigation,
- copyright notices,
- information about the site's privacy policy, and
- perhaps twitter feeds or links to other social sites.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Header and Footer

Both the HTML5 <header> and <footer> element can be used not only for *page* headers and footers, they can also be used for header and footer elements within other HTML5 containers, such as <article> or <section>.

<header>

```

<h1>Fundamentals of Web Development</h1>
...
```

</header>

```
<article>
```

<header>

```
  <h2>HTML5 Semantic Structure Elements </h2>
  <p>By <em>Randy Connolly</em></p>
  <p><time>September 30, 2012</time></p>
```

</header>

```
...
```

```
</article>
```



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Heading Groups

The <hgroup> element can be used to group related headings together within one container.

```
<header>
  <hgroup>
    <h1>Chapter Two: HTML 1</h1>
    <h2>An Introduction</h2>
  </hgroup>
</header>
<article>
  <hgroup>
    <h2>HTML5 Semantic Structure Elements </h2>
    <h3>Overview</h3>
  </hgroup>
</article>
```



Navigation

The **<nav>** element represents a section of a page that contains links to other pages or to other parts within the same page.

Like the other new HTML5 semantic elements, the browser does not apply any special presentation to the **<nav>** element.

The **<nav>** element was intended to be used for major navigation blocks, presumably the global and secondary navigation systems.



Navigation

```
<header>
  
  <h1>Fundamentals of Web Development</h1>
  <nav role="navigation">
    <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="about.html">About Us</a></li>
      <li><a href="browse.html">Browse</a></li>
    </ul>
  </nav>
</header>
```



Articles and Sections

<article> <section>

The **<article>** element represents a section of content that forms an independent part of a document or site; for example, a magazine or newspaper article, or a blog entry.

The **<section>** element represents a section of a document, typically with a title or heading.



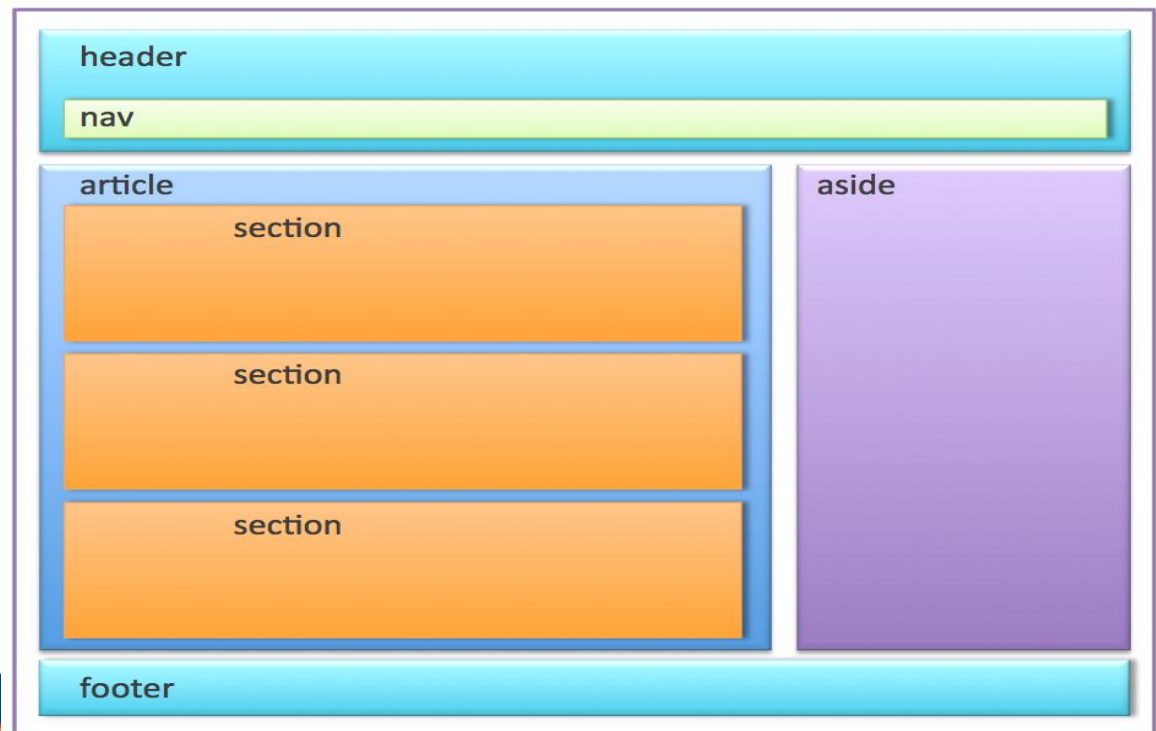
**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Articles and Sections

According to the W3C, **<section>** is a much broader element, while the **<article>** element is to be used for blocks of content that could potentially be read or consumed independently of the other content on the page.



This image is part of the Bioinformatics Web Development tutorial at http://www.cellbiol.com/bioinformatics_web_development/
© cellbiol.com, all rights reserved



Sections versus Divs

How to decide which to use

The WHATWG(Web Hypertext Application Technology Working Group) specification warns readers that the `<section>` element is **not** a generic container element. HTML already has the `<div>` element for such uses.

When an element is needed only for styling purposes or as a convenience for scripting, it makes sense to use the `<div>` element instead.

Another way to help you decide whether or not to use the `<section>` element is to ask yourself if it is appropriate for the element's contents to be listed explicitly in the document's outline.

If so, then use a `<section>`; otherwise use a `<div>`.


```

1
2 <!-- Headings & Sections: http://www.w3.org/TR/2014/REC-html5-
20141028/sections.html#headings-and-sections -->
3 <section>
4   <h2>Article List Heading</h2>
5   <article>
6     <h3>Article 1 Heading</h3>
7     
8     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque ac
lacus ante.</p>
9     <section>
10      <h4>Article 1 Subsection Heading</h4>
11      <p>Praesent scelerisque arcu eu dui molestie tincidunt.</p>
12    </section>
13  </article>
14  <article>
15    <h3>Article 2 Heading</h3>
16    
17    <p>Curabitur ornare ex sit amet tortor rhoncus, sit amet imperdiet nisl
posuere.</p>
18    <section>
19      <h4>Article 2 Subsection Heading</h4>
20      <p>Nam pretium ultrices mauris sit amet dignissim.</p>
21    </section>
22  </article>
23 </section>
24

```



Example

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<article>
```

```
  <h1>What Does WWF Do?</h1>
```

```
  <p>WWF's mission is to stop the degradation of our planet's  
  natural environment, and build a future in which humans  
  live in harmony with nature.</p>
```

```
</article>
```

```
</body> </html>
```



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Figure and Figure Captions

The **<figure>** element should **not** be used to wrap every image.

For instance, it makes no sense to wrap the site logo or non-essential images such as banner ads and graphical embellishments within **<figure>** elements.

Instead, only use the **<figure>** element for circumstances where the image (or other content) has a caption and where the figure is essential to the content but its position on the page is relatively unimportant.



Aside

The **<aside>** element is similar to the **<figure>** element in that it is used for marking up content that is separate from the main content on the page.

But while the **<figure>** element was used to indicate important information whose location on the page is somewhat unimportant, the **<aside>** element “represents a section of a page that consists of content that is tangentially related to the content around the aside element.”

The **<aside>** element could thus be used for sidebars, pull quotes, groups of advertising images, or any other grouping of non-essential elements.



Header

[HOME](#)[OUR TEAM](#)[PROJECTS](#)[CONTACT](#)[Go!](#)

Article heading

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Donec a diam lectus. Set sit amet ipsum mauris. Maecenas congue ligula as quam viverra nec consectetur ant hendrerit. Donec et mollis dolor. Praesent et diam eget libero egestas mattis sit amet vitae augue. Nam tincidunt congue enim, ut porta lorem lacinia consectetur.

subsection

Donec ut libero sed accu vehicula ultricies a non tortor. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aenean ut gravida lorem. Ut turpis felis, pulvinar a semper sed, adipiscing id dolor.

Pelientesque auctor nisi id magna consequat sagittis. Curabitur dapibus, enim sit amet elit pharetra tincidunt feugiat nist imperdiet. Ut convallis libero in urna ultrices accumsan. Donec sed odio eros.

Another subsection

Donec viverra mi quis quam pulvinar at malesuada arcu rhoncus. Cum soclis natoque penatibus et manis dis parturient montes, nascetur ridiculus mus. In rutrum accumsan ultricies. Mauris vitae nisi at sem facilisis semper ac in est.

Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare, ligula semper consectetur sagittis, nisi diam iaculis velit, is fringille sem nunc vet mi.

Related

- [Oh I do like to be beside the seaside](#)
- [Oh I do like to be beside the sea](#)
- [Although in the North of England](#)
- [It never stops raining](#)
- [Oh well...](#)

©Copyright 2050 by nobody. All rights reversed.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



INTRODUCTION TO HTML5 EVENTS

Events are actions or occurrences detected by the browser, like user interactions or system-generated updates. HTML5 provides a rich set of events to enhance web interactivity.

Types of Events:

Mouse Events: Clicks, hovering.

Keyboard Events: Key presses.

Form Events: Submissions, changes.

Window Events: Resize, load.



Common HTML5 Events

Categories of HTML5 Events

1. Mouse Events:

onclick: Triggered when an element is clicked.

onmouseover: Triggered when the mouse hovers over an element.

2. Keyboard Events:

onkeydown: Triggered when a key is pressed down.

onkeyup: Triggered when a key is released.

3. Form Events:

onsubmit: Triggered when a form is submitted.

onchange: Triggered when a form element changes.

4. Window Events:

onload: Triggered when the page finishes loading.

onresize: Triggered when the window is resized.

Event Attributes

HTML has the ability to let events trigger actions in a browser, like starting a JavaScript when a user clicks on an element.

Attribute	Description
<u>onafterprint</u>	<i>Script to be run after the document is printed</i>
<u>onbeforeprint</u>	<i>Script to be run before the document is printed</i>
<u>onbeforeunload</u>	<i>Script to be run when the document is about to be unloaded</i>
<u>onerror</u>	<i>Script to be run when an error occurs</i>
<u>onhashchange</u>	<i>Script to be run when there has been changes to the anchor part of the a URL</i>
<u>onload</u>	<i>Fires after the page is finished loading</i>
<u>onmessage</u>	<i>Script to be run when the message is triggered</i>



Event Attributes

<u>onoffline</u>	<i>Script to be run when the browser starts to work offline</i>
<u>ononline</u>	<i>Script to be run when the browser starts to work online</i>
<u>onpagehide</u>	<i>Script to be run when a user navigates away from a page</i>
<u>onpageshow</u>	<i>Script to be run when a user navigates to a page</i>
<u>onpopstate</u>	<i>Script to be run when the window's history changes</i>
<u>onresize</u>	<i>Fires when the browser window is resized</i>
<u>onstorage</u>	<i>Script to be run when a Web Storage area is updated</i>
<u>onunload</u>	<i>Fires once a page has unloaded (or the browser window has been closed)</i>



Try to resize the browser window.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body onresize="myFunction()">
```

```
<p>Try to resize the browser window.</p>
```

```
<script>
```

```
function myFunction() {
```

```
    alert("You have changed the size of the browser window!");
```

```
}
```

```
</script> </body> </html>
```



WEB FORMS 2.0

- An HTML form constitutes a distinct section of a webpage, encompassing specific controls such as labels, text fields, password fields, hidden fields (utilized by software), radio buttons, checkboxes, fieldsets, legends, and submit buttons.
- Users engage with these forms to furnish essential information for server processing.
- Presently, client-side scripting, commonly employing JavaScript, is employed for effects and basic validation.
- HTML5 is expected to significantly diminish the reliance on client-side scripting in the foreseeable future.

Key features of Web Forms 2.0 include:

- *New Input Types*: Introduces specialized input types like date, email, and range, providing users with intuitive ways to input data.
- *Input Validation*: Built-in validation mechanisms ensure data integrity, reducing errors and enhancing form submission reliability.
- *Placeholder Attribute*: Enables developers to provide hints or examples within form fields, guiding users and improving usability.
- *Autofocus Attribute*: Automatically focuses on specific form fields upon page load, enhancing user convenience and efficiency.

Advantages of Web Forms 2.0

- Improved user experience with enhanced form controls and validation
- Better accessibility and device compatibility
- Simplified development process with built-in functionalities
- Increased security with built-in input validation
- Use graphics or charts to illustrate the advantages



Web Forms 2.0

HTML5 Input Element (Text)

E-mail

- It will only accept email values.
- Input fields that need to contain an email address should use this type.

<form>

<label for="myemail">Enter Email Address:</label>

<input type="email" id="myemail" required>

</form>

Web Forms 2.0

Number

- This field accepts only numerical values. The step attribute specifies the precision, which defaults to 1.

```
<form>
```

```
<label for="mynumber">Enter a Number:</label>
```

```
<input type="number" min="1" max="10" step="0.5" id="mynumber">
```

```
</form>
```

Web Forms 2.0

Time

- The time (hour, minute, second, fractional second) is encoded according to ISO 8601.

```
<form>
```

```
<label for="mytime">Select Time:</label>
```

```
<input type="time" id="mytime">
```

```
</form>
```

Week

- A date that is composed of a weekday and a year is encoded according to ISO 8061.

```
<form>
```

```
<label for="myweek">Select Week:</label>
```

```
<input type="week" id="myweek">
```

```
</form>
```

Web Forms 2.0

Date: A date (year, month, day) is encoded using the ISO 8601 standard.

```
<form>
```

```
<label for="mydate">Select Date:</label>
```

```
<input type="date" value="2019-04-15" id="mydate">
```

```
</form>
```

range: For input fields, the range type is used to represent a range of values.

```
<form>
```

```
<label for="mynumber">Select a Number:</label>
```

```
<input type="range" min="1" max="10" step="0.5" id="mynumber">
```

```
</form>
```

Web Forms 2.0

URL: It can only accept URL values. In this type of field, URL addresses should be entered. Those who submit simple text entries must specify the URL, either `http://www.example.com` or <http://example.com>

```
<form>
```

```
<label for="myurl">Enter Website URL:</label>
```

```
<input type="url" id="myurl" required>
```

```
</form>
```

Web Forms 2.0

Placeholder attribute

- HTML5 introduced a new attribute called placeholder.
- With placeholder attributes on <input> and <textarea> elements, users are able to know what they can enter in the field.
- The placeholder text cannot contain line-feeds or carriage returns.

```
<input type="text" name="search" placeholder="search the internet"/>
```

Autofocus attribute

- This is a simple one-step pattern that can be easily programmed in JavaScript as soon as the document loads.
- When the form loads, it automatically focuses on a particular field in the document.

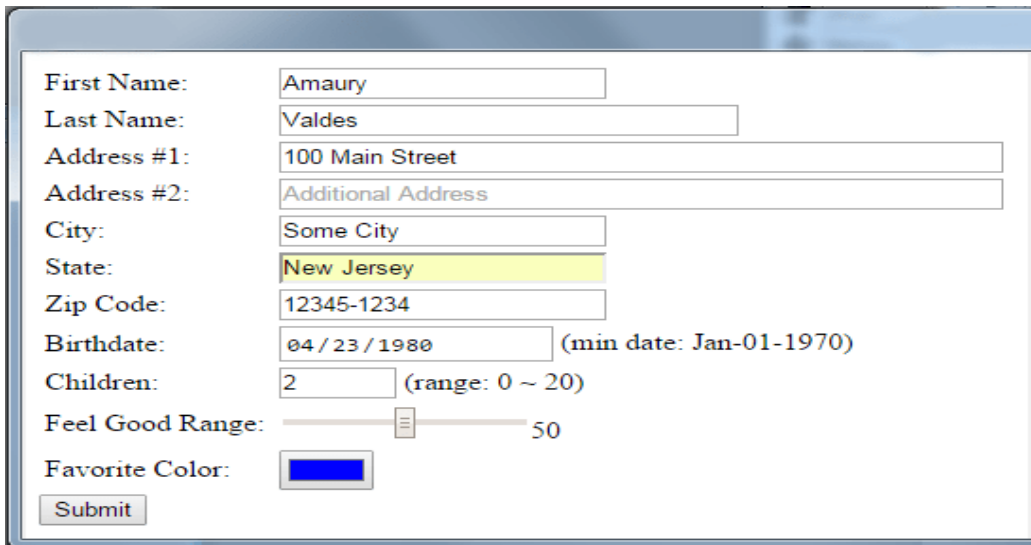
```
<input type="text" name="search" autofocus/>
```

Web Forms 2.0

Required attribute

- The required attribute is used in place of Javascript validations.
- Due to this attribute, Javascript is now only required for client-side validations where an empty text box cannot be submitted.

`<input type="text" name="search" required>`



A screenshot of a web form titled "Form". The form contains the following fields and controls:

- First Name:
- Last Name:
- Address #1:
- Address #2:
- City:
- State: - Zip Code:
- Birthdate: (min date: Jan-01-1970)
- Children: (range: 0 ~ 20)
- Feel Good Range: 50
- Favorite Color:
-

Overall, Web Forms 2.0 aims to provide developers with a comprehensive set of tools and features to create more user-friendly, accessible, and interactive web forms, ultimately enhancing the overall browsing experience for users.

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

Web Storage

- The web storage is more secure and large amounts of data can be stored locally on the client-side web browser. All the data is stored in key-value pairs.
- In HTML5 there are two types of web storage API.
- localStorage
- SessionStorage

Local Storage v/s Session Storage

Local Storage:

It is used to store data on the client side. It has no expiration time, so the data in the LocalStorage exists always till the user manually deletes it.

Syntax:

For storing data in web storage: The key and value both should be string or number; `LocalStorage.setItem("key", "value");` **For getting data from web storage:** We will pass the key and it will return value. `LocalStorage.getItem("key");`

Session Storage:

It is used to store data on the client-side. Data in the SessionStorage exist till the current tab is open, if we close the current tab then our data will also erase automatically from the SessionStorage.

Syntax:

For storing data in web storage: `SessionStorage.setItem("key", "value");` **For getting data from web storage:** `SessionStorage.getItem("key");`

HTML CANVAS

- HTML canvas is an HTML element that **provides a drawing surface for creating graphics and animations with JavaScript.**
- Canvas elements have **attributes such as width and height that define the size** of the drawing area.
- The Canvas API provides a set of methods and properties for drawing shapes, text, and images on the canvas, as well as manipulating colors, gradients, and patterns.
- Canvas can be used to create a wide range of visual content, such as **charts, graphs, diagrams, animations, games, and more.**
- Canvas is highly customizable with CSS styles that control the appearance of the canvas element and its contents.



HTML CANVAS (Contd)

- Canvas can be integrated with other web technologies, such as HTML, CSS, and JavaScript, to create sophisticated web applications.
- Canvas supports interactivity and animation, allowing you to create dynamic, user-driven content that responds to user input.
- Canvas is supported by most modern web browsers, including Chrome, Firefox, Safari, and Edge.
- Canvas requires some programming knowledge, as you need to write JavaScript code to create and manipulate graphics on the canvas.
- There are also many libraries and frameworks available that can help you create more complex and advanced canvas projects, such as Three.js, D3.js, and p5.js.



Example

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="300" height="150"
style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas
tag.</canvas>
```

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(20, 20, 150, 100);
</script> </body> </html>
```



WEBSOCKETS

WebSockets 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.

WEBSOCKETS

Once the socket is created, we should listen to events on it.
There are totally 4 events:

- open** – connection established,
- message** – data received,
- error** – websocket error,
- close** – connection closed.

And if we'd like to send something, then **socket.send(data)** will do that.



WEBSOCKETS

Following is the API which creates a new WebSocket object.

To open a websocket connection, we need to create new WebSocket using the special protocol **ws in the url**:

```
var Socket = new WebSocket(url, [protocol] );
```

Here first argument, url, specifies the URL to which to connect. The second attribute, protocol is optional, and if present, specifies a sub-protocol that the server must support for the connection to be successful

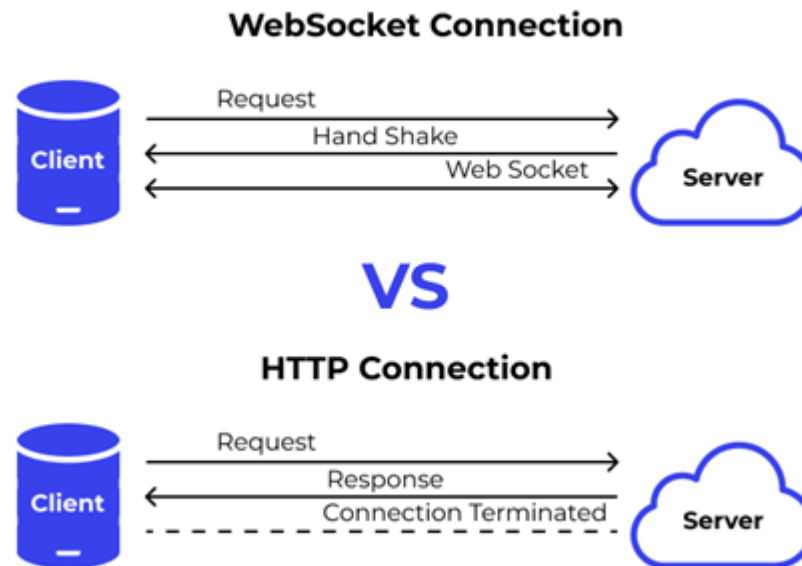


When not to use WebSocket:

WebSocket can be used if we want any **real-time updated or continuous streams** of data that are being transmitted over the network.

If we want to fetch old data, or want to get the data only once to process it with an application we should go with **HTTP protocol**, old data which is not required very frequently or fetched only once can be queried by the simple HTTP request, so in this scenario, it's better not use WebSocket.

Difference between WebSockets and HTTP



Difference between WebSockets and HTTP

Feature	WebSockets	HTTP
Communication Type	Full-duplex (two-way communication).	Half-duplex (client sends a request, server sends a response).
Connection Lifecycle	Persistent connection after the handshake.	Stateless; each request-response is independent.
Use Case	Real-time applications (e.g., chat apps, live updates, gaming).	General client-server interactions (e.g., web browsing, API calls).
Protocol	Uses TCP with its own protocol on top after an initial HTTP handshake.	Application-layer protocol based on request-response over TCP or other.
Efficiency	Low overhead for ongoing communication.	Higher overhead due to repeated headers in each request-response cycle.
Data Format	Binary or text frames (efficient for frequent updates).	Usually textual (e.g., JSON, HTML, XML).
Real-Time Support	Designed for low-latency, real-time communication.	Requires workarounds (e.g., long polling or HTTP/2) for real-time apps.
Security	Often used with WebSocket Secure (WSS), which is encrypted via TLS.	Secure via HTTPS, encrypted with TLS.
Scalability	More complex to scale due to persistent connections.	Easier to scale with stateless nature.



CSS3

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. CSS3 is a latest standard of css earlier versions(CSS2).

Some of the CSS3 modules are shown below –

- Selectors
- Box Model
- Backgrounds
- Image Values and Replaced Content
- Text Effects
- 2D Transformations
- 3D Transformations
- Animations
- Multiple Column Layout

Transform Property

The transform property applies a 2D or 3D transformation to an element.

Some common transform functions that can be used with the transform property include:

rotate: Rotates the element by a specified angle.

scale: Scales the element by a specified factor.

skew: Skews the element by a specified angle.

translate: Moves the element by a specified distance.

The translate() Method

The translate() method moves an element from its current position:

```
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
  transform: translate(50px,100px);
}
</style>
</head>
```



```
<body>
```

```
<h1>The translate() Method</h1>
```

```
<p>The translate() method moves an element from  
its current position:</p>
```

```
<div>
```

```
This div element is moved 50 pixels to the right,  
and 100 pixels down from its current position.
```

```
</div>
```

```
</body>
```

```
</html>
```



The scale() Method

transform: scaleY(3):

This div element is three times
of its original height.

transform: scaleX(0.5):

This div element is half of its
original width.

The skew() Method

- Skew effects create slanted or tilted designs, commonly used for visual emphasis.

- **Skew()**

Skews an element along both the X and Y axes.

- **skewX()**

Skews an element along the X-axis (horizontal skew).

- **skewY()**

Skews an element along the Y-axis (vertical skew).

CSS Gradients

- CSS gradients let you display smooth transitions between two or more specified colors.
- CSS defines three types of gradients:
- **Linear Gradients (goes down/up/left/right/diagonally)**
- **Radial Gradients (defined by their center)**
- **Conic Gradients (rotated around a center point)**



CSS Linear Gradients

- To create a linear gradient you must define at least two 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 with the gradient effect.



Direction - Top to Bottom (this is default)

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
  height: 200px;
  background-color: red; /* For browsers that do not support gradients */
  background-image: linear-gradient(red, yellow);
}
</style> </head> <body>
<h1>Linear Gradient - Top to Bottom</h1>
<p>This linear gradient starts red at the top, transitioning to yellow at the
  bottom:</p>

<div id="grad1">< /div> </body> </html>
```

Radial gradients

- Radial gradients are visual effects where colors smoothly transition in a circular pattern from a central point outward. They are defined by a center point and extend radially, often used in design to create depth or focus.
- **Key Characteristics:**
- **Center:** The starting point of the gradient.
- **Radius:** Determines how far the gradient extends outward.
- **Color Stops:** Points along the radius where specific colors are defined. Colors blend smoothly between stops.

CSS:

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

Conic gradients

Conic gradients are a type of CSS gradient where colors transition around a central point in a circular, conical pattern (like a pie chart). Instead of radiating outward or transitioning linearly, the gradient follows a sweeping, angular direction.

Key Characteristics of Conic Gradients

- **Center Point:** The point around which the gradient rotates.
- **Angle:** Colors transition based on angular positions (measured in degrees).
- **Color Stops:** Specific angles where colors are explicitly defined. Transitions occur between these stops.

CSS

`background: conic-gradient(from angle at position, color-stop1, color-stop2, ...);`

CSS3-Color

CSS3 builds on previous capabilities by:

- Adding transparency support with RGBA and HSLA.
- Introducing gradients for smoother color transitions.
- Enhancing flexibility with the `currentColor` keyword and opacity settings.
- Supporting more color formats and named colors.



Feature	HTML4	HTML5
Color Names	16 standard names (e.g., red, blue, green).	140 standard names (e.g., orange, crimson, gold).
Hexadecimal Colors	Supported (e.g., #RRGGBB).	Supported (e.g., #RRGGBB).
RGB Colors	Supported (e.g., rgb(255, 0, 0)).	Supported (e.g., rgb(255, 0, 0)).
RGBA Colors	Not supported.	Supported (e.g., rgba(255, 0, 0, 0.5)).
HSL Colors	Not supported.	Supported (e.g., hsl(120, 100%, 50%)).
HSLA Colors	Not supported.	Supported (e.g., hsla(120, 100%, 50%, 0.5)).
Transparency	No direct support (workarounds like opacity).	Supported via RGBA and HSLA.
CSS Integration	Limited by older CSS capabilities.	Enhanced by modern CSS features.
Browser Support	Widely supported but limited features.	Supported in all modern browsers.

END OF MODULE 1



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013

