History of HTML5:

As far as HTML5 vs HTML goes, unlike the latter, HTML5 has been developed by two separate groups: the **World Wide Web Consortium (W3C)** and the **Web Hypertext Application Technology Working Group (WHATWG)**. To understand the advancements of HTML5, it's helpful to have a brief overview of the general HTML timeline:

- **1991–1999:** HTML was created by web legend Tim Berners-Lee in 1991, and HTML versions 1–4 were developed throughout the 1990s by W3C. In these early days of widespread internet use, HTML efficiently displays the vast majority of web content, since at this time it largely consists of static, non-interactive sites.
- **2000:** W3C recommends XHTML 1.0 – an XML-based markup language that mirrors/extends HTML. Previous versions of HTML are now showing their age, struggling to handle the latest generation of multimedia, interactive sites. To get the best results, developers are resorting to third-party plugins.
- **2004:** Development of HTML is closed by W3C, who instead decide to focus on XHTML. WHATWG is formed to develop HTML further, with the aim of reflecting the modern dynamic web, while keeping backwards compatibility with existing HTML code.
- **2004–2006:** WHATWG gains support from major web browser developers. In 2006, W3C also announced its support for the project.
- **2008:** The first public draft of HTML5 is released by WHATWG.
- **2012:** W3C and WHATWG decide to separate development of HTML5. W3C would work on a definitive standard of HTML5, while WHATWG would pursue development of a 'living standard' – a continual evolution with ongoing improvements.
- **2014:** The official HTML5 release date, according to W3C recommendations.

New standard for HTML

HTML5 is the next major revision of the HTML standard superseding HTML 4.01, XHTML 1.0, and XHTML 1.1. HTML5 is a standard for structuring and presenting content on the World Wide Web.

HTML5 is cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).

The new standard incorporates features like video playback and drag-and-drop that have been previously dependent on third-party browser plug-ins such as Adobe Flash, Microsoft Silverlight, and Google Gears.

HTML5 brought significant improvements to the web development landscape by introducing new elements, attributes, and APIs that enhance the functionality, semantics, and multimedia capabilities of web pages.

These objectives primarily include:

1. Encouraging semantic (meaningful) markup
2. Separating design from content
3. Promoting accessibility and design responsiveness
4. Reducing the overlap between HTML, CSS, and JavaScript
5. Supporting rich media experiences while eliminating the need for plugins such as Flash or Java

1. Encouraging semantic (meaningful) markup:

HTML5 introduces a range of new semantic elements like `<header>`, `<nav>`, `<article>`, `<section>`, and `<footer>`. These elements provide a clearer and more structured way to define different parts of a webpage's content. For example, the `<header>` element represents the header section of a document or a section of content, `<nav>` represents navigation links, `<article>` represents a self-contained composition, and `<footer>` represents the footer section. This semantic markup makes it easier for developers to create well-organized and meaningful content that is more understandable by both humans and search engines.

2. Separating design from content:

HTML5 maintains the separation of concerns principle by focusing on the structural and semantic markup of content while leaving design and styling to CSS (Cascading Style Sheets). This clear separation allows developers to define the structure and meaning of content using HTML5 elements and attributes, while CSS is used to control the visual presentation, layout, and styling. This approach makes it easier to update and modify the design without affecting the underlying content structure.

3. Promoting accessibility and design responsiveness:

HTML5 includes features that promote accessibility and responsive design. The semantic elements help screen readers and assistive technologies better understand the content's structure, improving accessibility for users with disabilities. Additionally, HTML5's `<meta>` tags allow developers to specify viewport settings, which is crucial for creating responsive designs that adapt to various screen sizes and devices. The use of media queries in CSS, often used in conjunction with HTML5, further enhances responsive design by allowing developers to apply different styles based on device characteristics.

4. Reducing the overlap between HTML, CSS, and JavaScript:

While HTML, CSS, and JavaScript have distinct roles, HTML5 provides some features that reduce the need for extensive JavaScript or complex CSS styling. For example, the `<input>` element now supports various types like "email," "number," and "url," which provide native validation and reduce the need for JavaScript-based validation. Additionally, HTML5's form enhancements, semantic elements, and media elements contribute to reducing the necessity of overlapping technologies.

5. Supporting rich media experiences while eliminating the need for plugins:

HTML5 introduces native support for multimedia elements like `<audio>` and `<video>`. This eliminates the need for third-party plugins such as Adobe Flash or Java applets to embed audio and video content. The `<canvas>` element allows developers to create graphics, animations, and interactive applications directly within the browser using JavaScript. These features provide the foundation for creating engaging and rich media experiences directly through HTML5 and JavaScript, without relying on external plugins that might require separate installations or updates.

Browser Support

The latest versions of Apple Safari, Google Chrome, Mozilla Firefox, and Opera all support many HTML5 features and Internet Explorer 9.0 will also have support for some HTML5 functionality.

The mobile web browsers that come pre-installed on iPhones, iPads, and Android phones all have excellent support for HTML5.

Key features and enhancements of HTML5 include:

1. Semantics: HTML5 introduces new semantic elements like `<header>`, `<nav>`, `<article>`, `<section>`, `<footer>`, and more. These elements provide a clearer structure for defining different parts of a webpage's content, making it easier for both humans and search engines to understand the meaning of each section.

2. Multimedia Support: HTML5 includes native support for audio and video elements, eliminating the need for third-party plugins like Adobe Flash. This enables developers to embed audio and video directly into web pages without relying on external plugins.

3. Canvas: The `<canvas>` element provides a programmable space for rendering graphics and animations directly within the browser using JavaScript. This feature is widely used for creating interactive games, data visualizations, and other dynamic content.

4. Offline Web Applications: HTML5 introduces the concept of "application cache" that allows web applications to be accessible even when the user is offline. This feature is particularly useful for creating web apps that continue to function without an active internet connection.

5. Geo location: HTML5 provides APIs (Application Programming Interfaces) for obtaining the user's geographical location through the browser. This feature enables location-based applications and services.

6. Form Enhancements: HTML5 introduces new input types (e.g., email, URL, number) and attributes (e.g., required, placeholder) that enhance the way forms are designed and validated. This results in better user experiences and improved data collection.

7. Responsive Design: While responsive design is more of a design and CSS concept, HTML5's structural elements and media queries have greatly facilitated the creation of websites that adapt to various screen sizes and devices.

8. Web Workers: Web Workers allow developers to run scripts in the background, freeing up the main user interface thread. This enables better performance and responsiveness, especially in complex web applications.

9. Web Storage: HTML5 provides mechanisms for storing data on the client side, such as local Storage and session Storage. These are useful for storing small amounts of data without relying solely on cookies.

XHTML

What is XHTML?

- XHTML stands for EXtensible HyperText Markup Language
- XHTML is a stricter, more XML-based version of HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers

Why XHTML?

- XML is a markup language where all documents must be marked up correctly (be "well formed").
- XHTML was developed to make HTML more extensible and flexible to work with other data formats (such as XML). In addition, browsers ignore errors in HTML pages, and try to display the website even if it has some errors in the markup. So XHTML comes with a much stricter error handling.

The Most Important Differences from HTML

- <!DOCTYPE> is mandatory
- The xmlns attribute in <html> is mandatory
- <html>, <head>, <title>, and <body> are mandatory
- Elements must always be properly nested
- Elements must always be closed
- Elements must always be in lowercase
- Attribute names must always be in lowercase
- Attribute values must always be quoted
- Attribute minimization is forbidden

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Title of document</title>
</head>
```

<body>
some content here...
</body>
</html>

Some text
This is a paragraph (Correct) This is another paragraph (Wrong)
A break:
 An image:
<BODY>
<P>This is a paragraph</P> wrong
</BODY> ---- Wrong

Visit our HTML tutorial
</html>

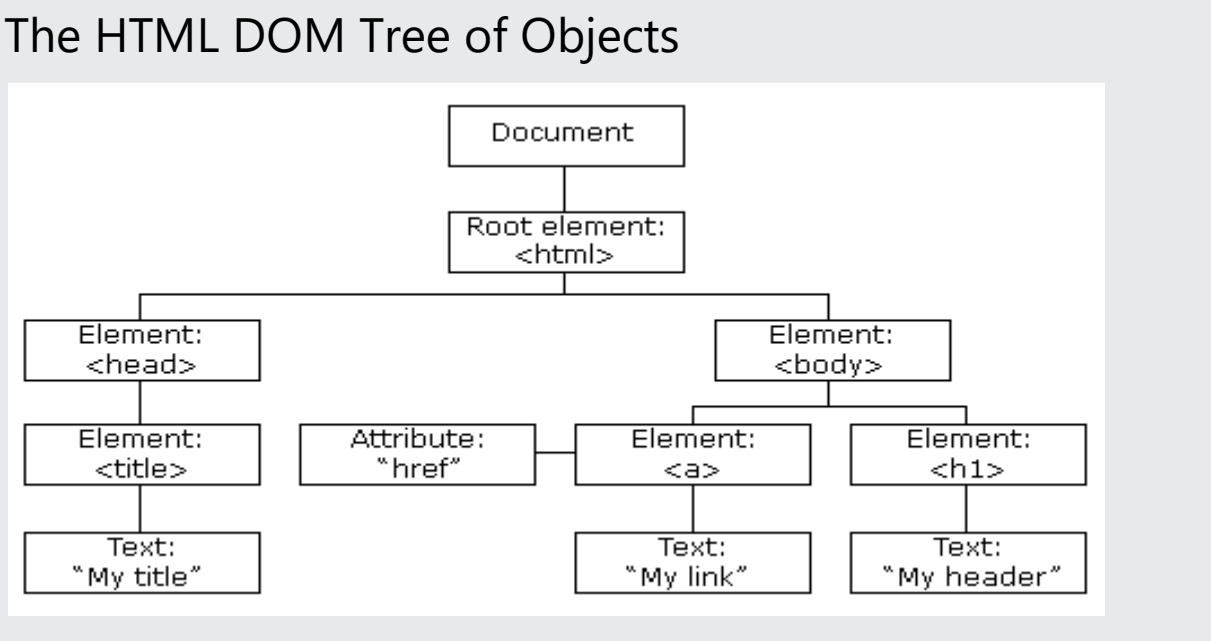
Example : 2

```
<!DOCTYPE html PUBLIC "-// W //DT XHTML 1.2 //EN"
" http : // www . myblogpost .org/T /xhtml12 / DT / xhtml12.dtd">
<html xmlns=http://www. myblogpost . org / 199 / xhtml >
<head>
<title> XHTML document </title>
</head>
<body>
Wrong XHTML rule<br>
Correct XHTML rule<br />
Wrong XHTML rule <hr>
Correct XHTML rule <hr />
Wrong XHTML rule
<img src=" pic/ document / mypic.gif" alt=" picture of a boy playing in a field ">
Correct XHTML rule
<img src="pic/ document / mypic.gif" alt="picture of a boy playing in a field " />
</body>
</html>
```

The HTML DOM (Document Object Model) is a programming interface for web documents. It represents the structure of an HTML or XML document as a tree of objects, where each object corresponds to a part of the document, such as elements, attributes, text content, and more. The DOM provides a way for programming languages, like JavaScript, to interact with and manipulate the content, structure, and style of web documents.

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

Finding HTML Elements

When you want to access HTML elements with JavaScript, you have to find the elements first.

- Finding HTML elements by id
- Finding HTML elements by tag name
- Finding HTML elements by class name
- Finding HTML elements by CSS selectors
- Finding HTML elements by HTML object collections

New Elements/New Mark up Elements in HTML5: HTML5 introduced several new markup elements that provide enhanced functionality and better semantics for structuring web content. These elements help web developers create more meaningful and organized web pages. Here are some of the new markup elements introduced in HTML5:

1. `<header>`: Represents the header section of a document or a section within a document. It's typically used to contain introductory content, headings, logos, and navigation menus.
2. `<nav>`: Represents a section of a document that contains navigation links, such as menus or lists of links to other pages or sections within the same page.
3. `<article>`: Represents a self-contained composition within a document, such as a blog post, news article, or forum post. It should make sense on its own and be able to be distributed and reused independently.
4. `<section>`: Represents a thematic grouping of content within a document. It's used to structure and organize related content, often with a heading to describe its purpose.
5. `<aside>`: Represents content that is tangentially related to the main content, such as sidebars, pull quotes, or advertisements. It's usually presented as a separate block of content.
6. `<footer>`: Represents the footer section of a document or a section within a document. It's typically used to contain copyright information, author details, and related links.
7. `<main>`: Represents the main content of the document. There should be only one `<main>` element in a document, and it's used to encapsulate the primary content.
8. `<figure>` and `<figcaption>`: The `<figure>` element is used to encapsulate media content, such as images, videos, diagrams, and code examples. The `<figcaption>` element is used to provide a caption or description for the media within the `<figure>`.
9. `<time>`: Represents a specific point in time or a range of time. It's used to mark up dates, times, and durations in a structured way.
10. `<mark>`: Highlights text within the document, often used to indicate search results or important terms.
11. `<progress>`: Represents the progress of a task, such as the completion percentage of an operation.
12. `<meter>`: Represents a scalar measurement within a known range, such as ratings, scores, or other types of measurements.
13. `<datalist>`: Provides a predefined list of options that can be associated with an `<input>` element to provide auto completion or suggestions.
14. `<summary>` and `<details>`: The `<summary>` element provides a visible heading for the `<details>` element, which can be toggled to reveal or hide additional content, creating an accordion-like effect.

15. `<dialog>`: Represents a dialog box or modal window that prompts the user for input or displays information.

These new markup elements introduced in HTML5 promote better document structure, improved semantics, and accessibility, making it easier to create well-organized and meaningful web content.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>All HTML5 Tags</title>
</head>
<body>
   <header>
      <h1>Welcome to My Website</h1>
      <nav>
         <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Services</a></li>
            <li><a href="#">Contact</a></li>
         </ul>
      </nav>
   </header>
   <main>
      <section>
         <h2>About Us</h2>
         <p>This is the main content of the about section.</p>
      </section>
<article>
         <h2>Article Title</h2>
         <p>This is an independent piece of content.</p>
      </article>
     <aside>
         <p>This is additional information related to the main content.</p>
```

```
    </aside>
     <figure>
        <img src="image.jpg" alt="An image">
        <figcaption>Caption for the image</figcaption>
    </figure>
     <details>
        <summary>Click to show more</summary>
        <p>Hidden content that can be shown on demand.</p>
    </details>
     <dialog open>
        <p>This is a dialog box content.</p>
        <button>Close</button>  </dialog>
<figcaption>Figure caption</figcaption>

    <footer>
        <p>&copy; 2023 My Website. All rights reserved.</p>
    </footer>

    <header>
        <h1>Header Title</h1>
        <p>Header content for navigation or introduction.</p>
    </header>

    <main>
        <h1>Main Content</h1>
        <p>This is the main content of the page.</p>
    </main>

    <mark>This text is highlighted.</mark>

    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">Products</a></li>
            <li><a href="#">Services</a></li>
        </ul>
```

```
    </nav>

    <progress value="90" max="100">90%</progress>

            <time datetime="2023-08-17">August 17, 2023</time>
    <br>
    <wbr>Insert a line break opportunity here.<wbr>
  </main>

  <footer>
    <p>&copy; 2023 My Website. All rights reserved.</p>
  </footer>
</body>
</html>
```

svg

The svg element is used for the scalable vector graphics in the html 5, we can create any kind of graphical animation with use of the svg tag, such as circle, square or rectangle etc.

1. Circle

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

2. Rectangle

```
<svg width="400" height="110">
  <rect width="300" height="100" stroke=”blue”/>
</svg>
```

New Media Elements

HTML5 introduced several new media elements that allow developers to embed various types of multimedia content directly into web pages without relying on external plugins. These elements enhance the user experience by providing native support for audio, video, and other multimedia content.

Here are some of the new media elements introduced in HTML5:

1. <audio>: This element allows you to embed audio content, such as music or spoken words, directly into a web page. You can use it to play sound files in different formats.

Example:

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

2. <video>: The `<video>` element enables you to embed video content into a web page. You can specify multiple video sources in different formats to ensure compatibility across various browsers.

Example:

```
  <video controls>
  <source src="video.mp4" type="video/mp4">
  <source src="video.webm" type="video/webm">
  Your browser does not support the video element.
</video>
```

Adding youtube videos:

Youtube videos can also be directly added to your web page using the embed video option on any youtube video.

<iframe> element is used to add the video with the src attribute but a shortcut to that is simply to go to any youtube video and right-click on the video, then select the copy embed code option.

```
<iframe width="942" height="530"
        src="https://www.youtube.com/embed/mzPxo7Y6JyA?si=7vmr5BEWaJ8q9Yg6"
        title="Introduction to HTML 5">
 </iframe>
```

3. <source>: Although not a media element itself, the `<source>` element is used within `<audio>` and `<video>` elements to specify alternative media sources in different formats. This helps ensure compatibility across various browsers and devices.

Example:

```
  <video controls>
  <source src="video.mp4" type="video/mp4">
  <source src="video.webm" type="video/webm">
  Your browser does not support the video element.
</video>
```

4. <track>: The `<track>` element is used to provide captions or subtitles for `<video>` and `<audio>` elements. It allows you to include text-based content that enhances the accessibility of multimedia content.

Example:

```
<video controls>
  <source src="video.mp4" type="video/mp4">
  <track src="captions.vtt" kind="subtitles" srclang="en" label="English">
  Your browser does not support the video element.
</video>
```

5. **<embed>:** It is used as a container for embedding plug-ins such as flash animations.

Syntax:

```
<embed attributes>
```

```
<embed src="https://media.com " width="300px" height="300px">
```

These new media elements introduced in HTML5 offer native support for embedding audio, video, and interactive graphics directly into web pages, enhancing the multimedia capabilities of the web platform.

The Canvas Element

The `<canvas>` element is a powerful HTML5 element that provides a drawing surface for rendering graphics, animations, and interactive content using JavaScript. It allows developers to create dynamic and visually engaging elements directly within a web page.

The `<canvas>` element is particularly useful for creating games, data visualizations, image manipulation, and other types of dynamic content.

Here's an overview of how the `<canvas>` element works:

1. Canvas Creation:

To create a canvas element, you simply use the `<canvas>` tag in your HTML code and provide an `id` attribute to uniquely identify the canvas. You can also set the canvas dimensions using the `width` and `height` attributes or CSS.

```
<canvas id="myCanvas" width="800" height="600"></canvas>
```

2. Getting the Canvas Context:

To work with the `<canvas>` element, you need to obtain a rendering context using JavaScript. The context provides methods and properties for drawing and interacting with the canvas content.

```
const canvas = document.getElementById("myCanvas");

const context = canvas.getContext("2d"); // "2d" context for 2D graphics
```

3. Drawing on the Canvas:

The context provides a variety of methods for drawing shapes, paths, text, and images on the canvas. Some common drawing methods include `fillRect()`, `strokeRect()`, `arc()`, `lineTo()`, `fillText()`, and more.

```
context.fillStyle = "blue";

context.fillRect(50, 50, 100, 100);

context.strokeStyle = "red";

context.lineWidth = 3;

context.strokeRect(75, 75, 50, 50);
```

4. Animating the Canvas:

The `<canvas>` element can be used to create animations by repeatedly updating the canvas content. This is often done using techniques like the `requestAnimationFrame()` function, which helps achieve smooth animations.

```
function animate() {
  // Clear the canvas
  context.clearRect(0, 0, canvas.width, canvas.height);
  // Update and draw your animation elements here
  requestAnimationFrame(animate);
}
animate();
```

5. Interactivity:

The canvas element can respond to user interactions, such as mouse clicks and movements. You can attach event listeners to the canvas to trigger actions based on user input.

The `<canvas>` element provides a way to create custom graphics and interactive content, but it requires JavaScript to achieve its full potential. If you're looking to create dynamic visual experiences on the web, the `<canvas>` element is a valuable tool for achieving that goal.

Draw a Line:

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

Draw a Circle :

```
<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>
```

Draw a Text :

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
</script>
```

Stroke Text :

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World", 10, 50);
</script>
```

Draw Linear Gradient :

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
// Create gradient
var grd = ctx.createLinearGradient(0, 0, 200, 0);
grd.addColorStop(0, "red");
grd.addColorStop(1, "green");
// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
</script>
```

Draw Image

```
<!DOCTYPE html>
<html>
<body>
<p>Image to use:</p>
<img id="scream" src="img_the_scream.jpg" alt="The Scream" width="220"
height="277">
<p>Canvas to fill:</p>
```

```
<canvas id="myCanvas" width="250" height="300" style="border:1px solid #d3d3d3;">

Your browser does not support the HTML canvas tag.</canvas>

<p><button onclick="myCanvas()">Try it</button></p>

<script>

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

var img = document.getElementById("scream");

ctx.drawImage(img, 10, 10);

</script>
```

New Form Elements

HTML5 introduced several new form elements that enhance the way forms are designed, validated, and interacted with on web pages. These elements provide better user experiences and simplify the process of collecting and processing user input. Here are some of the new form elements introduced in HTML5:

1. `<input>` Types:

HTML5 introduced several new `type` attributes for the `<input>` element, offering more specialized input fields.

- `type="email"`: For entering email addresses. It includes basic email validation.

- `type="url"`: For entering URLs. It helps ensure valid URLs are entered.

- `type="tel"`: For entering telephone numbers. It supports various phone number formats.

- `type="number"`: For entering numeric values. It can include min, max, and step attributes for control.

- `type="date"`, `type="time"`, `type="datetime-local"`: For entering date, time, and combined date-time values.

- `type="color"`: For selecting colors using a color picker.

2. `<input>` Attributes:

HTML5 introduced new attributes that improve the behavior and validation of `<input>` elements.

- `required`: Specifies that an input field must be filled out before submitting the form.

- `placeholder`: Provides a hint or example value within an input field.

- `pattern`: Allows you to specify a regular expression pattern for input validation.

- `autocomplete`: Indicates whether a form should have autocomplete enabled or disabled.

- `autofocus`: Automatically focuses on the input field when the page loads.

- `min`, `max`, `step`: Applicable to numeric input types (`number`, `range`, etc.), these attributes define valid value ranges and increments.

3. `<textarea>` Attributes:

HTML5 introduced new attributes for the `<textarea>` element to enhance its behavior and appearance.

- `placeholder`: Provides a hint or example text within the text area.

- `maxlength`: Specifies the maximum number of characters allowed in the text area.

4. `<select>` Attributes:

The `<select>` element, used for creating dropdown lists, gained new attributes for better control.

- `size`: Allows you to display multiple options at once, creating a scrolling list.

- `multiple`: Enables users to select multiple options from the list.

5. `<datalist>` Element:

The `<datalist>` element introduces a predefined list of options that can be associated with an `<input>` element, providing suggestions as users type.

```
<input list="colors">
<datalist id="colors">
  <option value="Red">
  <option value="Blue">
  <option value="Green">
</datalist>
```

6. `<output>` Element:

The `<output>` element is used to display the result of a calculation or the outcome of a form submission. It's often associated with other form elements.

```
<input type="number" id="num1">
<input type="number" id="num2">
```

```
  <output for="num1 num2" id="result"></output>
```

These new form elements and attributes introduced in HTML5 improve the user experience, enhance input validation, and provide developers with more options for creating user-friendly and efficient web forms.

Example

```
<!DOCTYPE html>

<html>

<head>

   <title>New HTML5 Form Elements</title>

</head>

<body>

   <form>

        <label for="color">Choose a color:</label>

      <input type="color" id="color" name="color"><br>

      <label for="birthdate">Birthdate:</label>

      <input type="date" id="birthdate" name="birthdate"><br>

      <label for="email">Email:</label>

      <input type="email" id="email" name="email"><br>


      <label for="quantity">Quantity:</label>

      <input type="number" id="quantity" name="quantity" min="1" max="10" step="1"><br>

      <label for="search">Search:</label>

      <input type="search" id="search" name="search"><br>

      <label for="username">Username:</label>

      <input type="text" id="username" name="username" placeholder="Enter your username"><br>

      <label for="fullname">Full Name:</label>
```

```
    <input type="text" id="fullname" name="fullname" required><br>
    <label for="zipcode">Zip Code:</label>
    <input type="text" id="zipcode" name="zipcode" pattern="[0-9]{5}"><br>
    <label for="browser">Choose a browser:</label>
    <input type="text" id="browser" name="browser" list="browsers">
    <datalist id="browsers">
      <option value="Chrome">
      <option value="Firefox">
      <option value="Safari">
      <option value="Edge">
      <option value="Opera">
    </datalist><br>
    <label for="volume">Volume:</label>
    <input type="range" id="volume" name="volume" min="0" max="100" step="1"><br>
<input type="submit" value="Submit">
  </form>
</body>
</html>
```

New Input Type Attribute Values

HTML5 introduced several new `type` attribute values for the `<input>` element, enabling developers to create specialized input fields with built-in validation and user interface enhancements. These attributes help improve user experience and simplify data entry. Here are some of the new `type` attribute values introduced in HTML5:

1. `type="email"`: Creates an input field specifically designed for entering email addresses. It includes basic email validation to ensure that the entered value has a valid email format.

```
<input type="email" name="user_email">
```

2. `type="url"`: Generates an input field optimized for entering URLs. It validates that the entered value is a valid web URL.

```
<input type="url" name="website_url">
```

3. `type="tel"`: Creates an input field tailored for entering telephone numbers. It can help enforce specific phone number formats, but it doesn't verify the validity of the number itself.

```
<input type="tel" name="phone_number">
```

4. `type="number"`: Generates an input field for numeric values. It can include optional attributes like `min`, `max`, and `step` to specify valid value ranges and increments.

```
<input type="number" name="quantity" min="1" max="10" step="1">
```

5. `type="date"`, `type="time"`, `type="datetime-local"`: These input types allow users to enter date, time, or combined date-time values. They provide built-in date and time pickers on supporting browsers.

```
<input type="date" name="event_date">

<input type="time" name="event_time">

<input type="datetime-local" name="event_datetime">
```

6. `type="month"`, `type="week"`: These input types let users select a specific month or week, respectively, using built-in controls.

```
<input type="month" name="selected_month">

<input type="week" name="selected_week">
```

7. `type="color"`: Generates a color picker control, allowing users to select a color from a predefined range.

```
<input type="color" name="selected_color">
```

8. `type="range"`: Creates a slider input control that lets users select a value from a range. It can have optional `min` and `max` attributes.

```
<input type="range" name="volume" min="0" max="100">
```

9. `type="search"`: Generates a search input field with added styling and functionality, such as clearing the field.

```
<input type="search" name="search_query">
```

These new `type` attribute values for the `<input>` element provide specialized input fields that enhance user experience, improve data validation, and offer features like built-in pickers and controls for specific data types.

Example

```
<!DOCTYPE html>
<html>
<head> <title>New Input Types in HTML5</title>
</head>
<body>
   <form>
      <label for="colorInput">Color:</label>
      <input type="color" id="colorInput" name="colorInput"><br>
 <label for="dateInput">Date:</label>
      <input type="date" id="dateInput" name="dateInput"><br>
 <label for="datetime-localInput">Datetime-local:</label>
      <input type="datetime-local" id="datetime-localInput" name="datetime-
localInput"><br>
 <label for="emailInput">Email:</label>
      <input type="email" id="emailInput" name="emailInput"><br>
 <label for="urlInput">Color:</label>
      <input type="url" id="urlInput" name="urlInput"><br>
 <label for="telInput">Telephone:</label>
      <input type="tel" id="telInput" name="telInput"><br>
 <label for="monthInput">Month:</label>
      <input type="month" id="monthInput" name="monthInput"><br>
 <label for="weekInput">Week:</label>
      <input type="week" id="weekInput" name="weekInput"><br>
 <label for="numberInput">Number:</label>
      <input type="number"min="1" max="10" step="0.5" id="numberInput"
name="numberInput"><br>
 <label for="rangeInput">Range:</label>
```

```
    <input type="range" min="10" max="100" step="1" id="rangeInput"
name="rangeInput"><br>

 <label for="searchInput">Search:</label>

    <input type="search" id="searchInput" name="searchInput"><br>

 <label for="timeInput">Time:</label>

    <input type="time" id="timeInput" name="timeInput"><br>

    <input type="submit" value="Submit">

   </form>

</body></html>
```

Video Formats

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. The browser will choose the first source it supports. There are three supported video formats in HTML: **MP4, WebM, and OGG**.

Browser	MP4	WebM	Ogg
Edge	YES	YES	YES
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES