UNIT-V

Image Manipulations:

Image manipulation, at its core, is the art and science of altering or modifying an image to achieve a desired result. It's a practice that has been in existence for as long as images have been produced, evolving from physical alterations made to photographs to the digital tweaks we see today. Commonly referred to as "Photo shopping," this term underscores how Adobe Photoshop has become synonymous with the art of image editing, showcasing its widespread adoption and cultural significance.

Image editing software has made it possible to manipulate images, thanks to their myriad of tools and features. From simple tasks like cropping and color correction to more complex operations like cloning, layer blending, and digital painting, the boundaries of image manipulation have expanded exponentially.

Position an Image:

The position of an image in CSS, properties like object position and float are used to control the placement of an image within its container.

Using object-position Property

The object-position property in CSS is used to set the position of an image within its container when using the object-fit property. It allows you to adjust how the image is displayed inside the box by specifying the alignment in terms of x and y coordinates.

Syntax

```
object-position: <x-position> <y-position>;
```

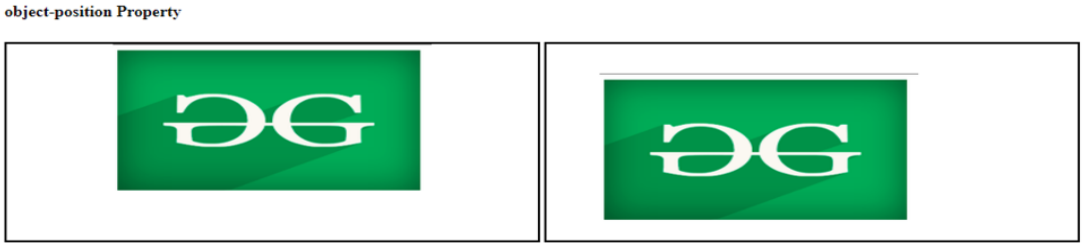
Property Values

- **x-position:** The horizontal alignment (distance from the left of the content box).
- **y-position:** The vertical alignment (distance from the top of the content box).

```
<h4>object-position Property</h4>
<img src="https://media.geeksforgeeks.org/wp-content/uploads/20240117104925/1.png"
    style="width: 500px; height: 200px;
        object-fit: none; object-position: center top;
        border: 2px solid black;" />
```

```
<img src="https://media.geeksforgeeks.org/wp-content/uploads/20240117104925/1.png"
    style="width: 500px; height: 200px;
        object-fit: none; object-position: 50px 30px;
        border: 2px solid black;" />
```

Output



How to display a shortcut icon in a browser's tab area:

A browser tab icon, also known as a **favicon** is a small image that appears before the title of a webpage in the browser tab or bookmark bar to represent a website. It helps in brand recognition, improves user experience, and makes the website stand out among others. Adding a favicon can help users easily identify your website and differentiate it from other open tabs.
In this article, we will discuss how to add a favicon to a website using HTML. A favicon is a small image that is always displayed next to the page title in the browser tab.

The part which is highlighted with yellow color shows the favicon or browser icon. The small GeeksforGeeks image shown in the tab is the favicon we are talking about.

To add a favicon to a website, you need to create an icon file in .ico or .png format, and then reference it in the HTML code:

Approach: Use an existing icon: If you have an existing icon in .ico, .png, or .gif format, you can use it as a favicon. To use an existing icon as a favicon, add the following code to the head section of your HTML document:

```
<link rel="icon" type="image/png" href="path-to-favicon">
```

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8" />
        <title>
                GeeksforGeeks
        </title>
        <!-- add icon link -->
        <link rel="icon" href=
"https://media.geeksforgeeks.org/wp-content/cdn-uploads/gfg_200X200.png"
                type="image/x-icon">
</head>
<body>
        <h1 style="color:green;">
                GeeksforGeeks
        </h1>

        <p>
                Welcome to my website
        </p>
</body>
</html>
```

IFrame:

An iframe, or Inline Frame, is an HTML element represented by the <iframe> tag. It functions as a 'window' on your webpage through which visitors can view and interact with another webpage from a different source.

Iframes are used for various purposes like:

- **Embedding Multimedia**: Easily integrate videos, audio, or animations from platforms like YouTube, etc.
- **Including Maps**: Embed maps from services like Google Maps directly into your site.
- **Loading Forms and Widgets**: Incorporate forms or widgets from other sources without writing complex code.

Syntax:

```
<iframe src="URL" title="description"></iframe>
```

- The **src** attribute specifies the **URL** of the document you want to embed.
- Iframes can include **videos**, **maps**, or **entire web pages** from other sources.

Example:

```
<!DOCTYPE html>
<html>

<head>
   <title>HTML iframe Tag</title>
</head>

<body style="text-align: center">
   <h2>HTML iframe Tag</h2>
   <iframe src=
"https://media.geeksforgeeks.org/wp-content/uploads/20240206111438/uni2.html"
         height="370"
         width="400">
   </iframe>
</body>
</html>
```

Create an Image Sprite File:

CSS Image Sprites are nothing but a way to reduce the HTTP requests from the image resources. A CSS Image Sprite is a single image file containing all images on a document page. Image sprites are advantageous since image resources will only have to be loaded once.

Approach: At first we need to create an image that will have all the images we want to combine into one using any image editor. At first, we need to add all the divs or anchor tags, the number of them will be the same as the number of images in that sprite(generally). Then in the style part, we need to specify the **background-position** of the divs or anchor tags with the width or height of the images in the sprite. In this article, we have made a navbar using the image sprite below.

Properties used:

- **background:** It is a property that is used to add a background to any element, it can be a simple color or it can even be an image.
- **background-position:** This specifies the initial image position of the background image.
- **position:** This property specifies the type of positioning an element will have. It takes values such as static, relative, absolute, fixed, or sticky.
- **display:** This property defines how the element will be shown on the webpage.
- **left:** This class is used to position elements horizontally, this class only works on an element whose position is set to some value.
- **width:** This class is used to specify the total width of an element.

Example:

```
<!DOCTYPE html>
<html>
<head>
   <style type="text/css">
      .sprite {
        background: url("sprite.png") no-repeat;
        width: 280px;
        height: 200px;
        display: inline-block;
      }
      .logo1 {
         left: 0px;
         width: 100px;
         background-position: 0px 0px;
      }
      .logo2 {
         left: 100px;
         width: 111px;
         background-position: -100px 0px;
      }
      .logo3 {
         left: 200px;
         width: 100px;
         background-position: -211px 0px;
      }
      .logo4 {
         left: 300px;
         width: 100px;
         background-position: -311px 0px;
      }
      body {
        text-align: center;
      }
      h1{
         color: green;
         text-align: center;
      }
   </style>
</head>
<body>
   <h1>GeeksForGeeks</h1>
   <div><h2>How to create and use CSS Image Sprites ?</h2></div>
   <div class="sprite logo1"></div>
   <div class="sprite logo2"></div>
   <div class="sprite logo3"></div>
```

```
    <div class="sprite logo4"></div>
</body>
</html>
```

Implement an audio player using the audio Element:

To implement a basic audio player using HTML's <audio> element, include the element in your HTML, set the src attribute to the audio file's path, and add the controls attribute to display playback controls.

Here's a basic example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple Audio Player</title>
</head>
<body>
  <audio controls>
    <source src="path/to/your/audio.mp3" type="audio/mpeg">
    Your browser does not support the audio element.
  </audio>
</body>
</html>
```

Explanation:

- <audio> element: This element is used to embed audio content.
- controls attribute: This attribute displays the default audio controls (play, pause, volume, etc.).
- src attribute: This attribute specifies the path or URL of the audio file.
- <source> element: This element is used to provide multiple audio file sources for browser compatibility. The browser will use the first supported source.
- type attribute (in <source>): This attribute indicates the type of audio file (e.g., audio/mpeg for MP3).
- **Fallback text:** The text between the opening and closing <audio> tags will be displayed in browsers that do not support the <audio element.

  Optional Attributes:

- autoplay: Starts playing the audio automatically.
- loop: Repeats the audio playback.
- muted: Starts playing automatically but muted.

Handle different audio file formats:

To handle different audio file formats in HTML, use the <audio> tag with multiple <source> tags, each specifying a different audio file format and type (e.g., audio/mpeg for MP3, audio/ogg for Ogg) to ensure browser compatibility.

Here's a more detailed explanation:

- The <audio> Tag: This tag is used to embed audio content in an HTML document.
- The <source> Tag: You can include multiple <source> tags inside the <audio> tag, each pointing to a different audio file format.
- src Attribute: The src attribute of the <source> tag specifies the URL of the audio file.
- type Attribute: The type attribute specifies the MIME type of the audio file (e.g., audio/mpeg for MP3, audio/ogg for Ogg).
- **Browser Compatibility:** Browsers will choose the first audio source format they support from the <source> tags.

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

Other Attributes: The <audio> tag also supports other attributes like controls (to display default controls), autoplay (to start playing automatically), and loop (to repeat the audio).

Cover a web page's background with an image:

HTML background images are graphics applied to the background of HTML elements, often used in webpage design for aesthetic or branding purposes. They're set using CSS background-image property, allowing for single or repeating images, gradients, or patterns to enhance visual appeal.

Understanding HTML Background Images

HTML Background Images allows the developers to set images as backgrounds for HTML elements, enhancing visual appeal and customization within web pages.

Using Background-Image Attribute Inside Body Tag

In this method, we will add a background image using the background image attribute inside the <body> tag.

Syntax:

```
<body background="image.png"></body>
```

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Background Image</title>
</head>
<body background=
"https://media.geeksforgeeks.org/wp-content/cdn-uploads/20210203170945/HTML-
Tutorials.png">
   <h1 style="color: green;
            text-align: center;">
      Welcome to GeeksforGeeks
   </h1>
</body>
</html>
```

Web Fonts:

Web fonts allow Web designers to use fonts that are not installed on the user's computer.

When you have found/bought the font you wish to use, just include the font file on your web server, and it will be automatically downloaded to the user when needed.

Your "own" fonts are defined within the CSS @font-face rule.

Different Font Formats

TrueType Fonts (TTF)

TrueType is a font standard developed in the late 1980s, by Apple and Microsoft. TrueType is the most common font format for both the Mac OS and Microsoft Windows operating systems.

OpenType Fonts (OTF)

OpenType is a format for scalable computer fonts. It was built on TrueType, and is a registered trademark of Microsoft. OpenType fonts are used commonly today on the major computer platforms.

The Web Open Font Format (WOFF)

WOFF is a font format for use in web pages. It was developed in 2009, and is now a W3C Recommendation. WOFF is essentially OpenType or TrueType with compression and additional metadata. The goal is to support font distribution from a server to a client over a network with bandwidth constraints.

The Web Open Font Format (WOFF 2.0)

TrueType/OpenType font that provides better compression than WOFF 1.0.

SVG Fonts/Shapes

SVG fonts allow SVG to be used as glyphs when displaying text. The SVG 1.1 specification define a font module that allows the creation of fonts within an SVG document. You can also apply CSS to SVG documents, and the @font-face rule can be applied to text in SVG documents.

Embedded OpenType Fonts (EOT)

EOT fonts are a compact form of OpenType fonts designed by Microsoft for use as embedded fonts on web pages.

Implement a video player using the video element:

The <video> element in HTML is used to add video content to web pages. It supports various video formats, including MP4, WebM, and Ogg. Video and audio tags are introduced in HTML5.

Syntax:

```
<video src="" controls>   </video>
```

- The **src attribute s**pecifies the URL of the video file.
- The controls attribute adds default video controls (play, pause, volume, etc.).

```
<html >
<body>
   <video width="320" height="240" controls>
      <source src=
"https://media.geeksforgeeks.org/wp-content/uploads/20190616234019/Canvas.move_.mp4"
         type="video/mp4">
      Sample Vedio
   </video>
</body>
</html>
```

- The <video> tag defines the video player, with width and height attributes setting its dimensions.
- The controls attribute adds playback controls like play, pause, and volume.

HTML Video Tags

Here are the HTML tags used for adding video content:

- **<video>:** Defines a video or movie on a webpage.
- **<source>:** Specifies multiple media resources for video or audio elements (e.g., different video formats).
- Adds text tracks (like subtitles or captions) to a video or audio element.

Center a web page's content:

The tag in HTML was used to horizontally center content on a webpage. However, it is **deprecated** in HTML5 and should no longer be used. Alternatively, we will use **the text-align property**.

```
<!DOCTYPE html>
<html>
<body>
   <center>
      <h3>Welcome to GeeksforGeeks</h3>
      <p>Hi Geeks</p>
   </center>
</body>
</html>
```

Cover a web page's background with a color gradient:

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.

Syntax

background-image: linear-gradient(*direction, color-stop1, color-stop2, ...*);

Direction - Top to Bottom (this is default)

The following example shows a linear gradient that starts at the top. It starts red, transitioning to yellow:

top to bottom (default)

Example

```
#grad {
  background-image: linear-gradient(red, yellow);
}
```

Try it Yourself »

Direction - Left to Right

The following example shows a linear gradient that starts from the left. It starts red, transitioning to yellow:

left to right

Example

```
#grad {
        background-image: linear-gradient(to right, red , yellow);
}
```

Try it Yourself »

Direction - Diagonal

You can make a gradient diagonally by specifying both the horizontal and vertical starting positions.

The following example shows a linear gradient that starts at top left (and goes to bottom right). It starts red, transitioning to yellow:

top left to bottom right

Example

```
#grad {
        background-image: linear-gradient(to bottom right, red, yellow);
}
```

Introduction to Java Script:

JavaScript is a versatile, dynamically typed programming language used for interactive web applications, supporting both client-side and server-side development, and integrating seamlessly with HTML, CSS, and a rich standard library.

- JavaScript is a single-threaded language that executes one task at a time.
- It is an Interpreted language which means it executes the code line by line.
- The data type of the variable is decided at run-time in JavaScript that's why it is called dynamically typed.

"Hello, World!" Program in Browser Console

A "Hello, World!" program is the simplest way to get started with any programming language. Here's how you can write one using JavaScript.

```
<html>
<head></head>
<body>
  <h1>Check the console for the message!</h1>
  <script>
    // This is our first JavaScript program
    console.log("Hello, World!");
  </script>
</body>
</html>
```

In this example

- The<script> tag is used to include JavaScript code inside an HTML document.
- console.log() prints messages to the browser's developer console. Open the browser console to see the "Hello, World!" message.

Button control with an event Handler:

avaScript Events are **actions or occurrences** that happen in the browser. They can be triggered by various user interactions or by the browser itself.

```
<html>
<script>
   function myFun() {
      document.getElementById(
         "gfg").innerHTML = "GeeksforGeeks";
   }
</script>
<body>
   <button onclick="myFun()">Click me</button>
   <p id="gfg"></p>
</body>
</html>
```

- The onclick attribute in the <button> calls the myFun() function when clicked.
- The myFun() function updates the <p> element with id=”gfg” by setting its innerHTML to “GeeksforGeeks”.
- Initially, the <p> is empty, and its content changes dynamically on button click.

Event Types

JavaScript supports a variety of event types. Common categories include:

- **Mouse Events:** click, dblclick, mousemove, mouseover, mouseout
- **Keyboard Events:** keydown, keypress, keyup
- **Form Events:** submit, change, focus, blur
- **Window Events:** load, resize, scroll

Common JavaScript Events

Event Attribute	Description
onclick	Triggered when an element is clicked.
onmouseover	Fired when the mouse pointer moves over an element.
onmouseout	Occurs when the mouse pointer leaves an element.
onkeydown	Fired when a key is pressed down.

Event Attribute	Description
onkeyup	Fired when a key is released.
onchange	Triggered when the value of an input element changes.
onload	Occurs when a page has finished loading.
onsubmit	Fired when a form is submitted.
onfocus	Occurs when an element gets focus.
onblur	Fired when an element loses focus.

Syntax rules for functions:

Functions in JavaScript are reusable blocks of code designed to perform specific tasks. They allow you to organize, reuse, and modularize code. It can take inputs, perform actions, and return outputs.

```
function sum(x, y) {
   return x + y;
}
console.log(sum(6, 9));
```

Function Syntax and Working

A function definition is sometimes also termed a function declaration or function statement. Below are the rules for creating a function in JavaScript:

- Begin with the keyword **function** followed by,
- A user-defined function name (In the above example, the name is **sum**)
- A list of parameters enclosed within parentheses and separated by commas (In the above example, parameters are **x** and **y**)
- A list of statements composing the body of the function enclosed within curly braces {} (In the above example, the statement is "return x + y").

Return Statement

In some situations, we want to return some values from a function after performing some operations. In such cases, we make use of the return. This is an optional statement. In the above function, "sum()" returns the sum of two as a result.

Function Parameters

Parameters are input passed to a function. In the above example, sum() takes two parameters, x and y.

Calling Functions

After defining a function, the next step is to call them to make use of the function. We can call a function by using the function name separated by the value of parameters enclosed between the parenthesis.

```
// Function Definition
function welcomeMsg(name) {
   return ("Hello " + name + " welcome to GeeksforGeeks");
}

let nameVal = "User";

// calling the function
console.log(welcomeMsg(nameVal));
```

Variables:

Variables in JavaScript can be declared using var, let, or const. JavaScript is dynamically typed, so variable types are determined at runtime without explicit type definitions.

- JavaScript var keyword
- JavaScript let keyword
- JavaScript const keyword

```
var a = 10     // Old style
let b = 20;    // Prferred for non-const
const c = 30;  // Preferred for const (cannot be changed)
console.log(a);
console.log(b);
console.log(c);
```

Declaring Variables in JavaScript

1. JavaScript var keyword

var is a keyword in JavaScript used to declare variables and it is Function-scoped and hoisted, allowing redeclaration but can lead to unexpected bugs.

```
var a = "Hello Geeks";
var b = 10;
console.log(a);
console.log(b);
```

2. JavaScript let keyword

let is a keyword in JavaScript used to declare variables and it is Block-scoped and not hoisted to the top, suitable for mutable variables

```
let a = 12
let b = "gfg";
console.log(a);
console.log(b);
```

3. JavaScript const keyword

const is a keyword in JavaScript used to declare variables and it is Block-scoped, immutable bindings that can't be reassigned, though objects can still be mutated.

```
const a = 5
let b = "gfg";
console.log(a);
console.log(b);
```

Rules for Naming Variables

When naming variables in JavaScript, follow these rules

- Variable names must begin with a letter, underscore (_), or dollar sign ($).
- Subsequent characters can be letters, numbers, underscores, or dollar signs.
- Variable names are case-sensitive (e.g., age and Age are different variables).
- Reserved keywords (like function, class, return, etc.) cannot be used as variable names.

Identifiers:

In JavaScript, an identifier is simply a name used to identify a variable, function, class, or other elements within your code, following specific rules like starting with a letter, underscore, or dollar sign.

- Identifiers are names you give to variables, functions, classes, and other elements in your JavaScript code.
- They are used to refer to these elements and perform operations on them.
- **Rules for Creating Identifiers:**
- **First Character:** An identifier must start with a letter (A-Z, a-z), an underscore (_), or a dollar sign ($).

- **Subsequent Characters:** After the first character, identifiers can contain letters, digits (0-9), underscores, or dollar signs.
- **Case Sensitivity:** JavaScript identifiers are case-sensitive, meaning "myVariable" and "myvariable" are treated as different identifiers.
- **Reserved Words:** You cannot use JavaScript reserved words (like var, let, const, function, return, etc.) as identifiers.
- **Examples of Valid Identifiers:**
- myVariable
- _myFunction
- $myObject
- userName123
- **Examples of Invalid Identifiers:**
- 123Variable (starts with a digit)
- my-variable (contains a hyphen)
- function (reserved word)
- **Purpose of Identifiers:**
- **Variable Naming:** Identifiers are used to name variables, which store data.
- **Function Naming:** Identifiers are used to name functions, which perform specific tasks.
- **Class Naming:** Identifiers are used to name classes, which are blueprints for creating objects.
- **Property Naming:** Identifiers are used to name properties of objects.

Assignments:

In JavaScript, assignments use the = operator to assign a value to a variable, and there are also compound assignment operators that combine assignment with other operations like addition, subtraction, etc.

Here's a breakdown of assignments in JavaScript:

1. Basic Assignment Operator (=):

- The = operator assigns the value on the right-hand side to the variable on the left-hand side.
- **Example:**

JavaScript
```
let x = 5; // Assigns the value 5 to the variable 'x'
let y = "Hello"; // Assigns the string "Hello" to the variable 'y'
```

2. Compound Assignment Operators:

- These operators combine an arithmetic or bitwise operation with the assignment operator.
- **Examples:**
  - += (addition and assignment):

JavaScript
```
let a = 10;
a += 5; // Equivalent to a = a + 5;  'a' becomes 15
```

-= (subtraction and assignment).

JavaScript
```
let b = 20;
b -= 3; // Equivalent to b = b - 3; 'b' becomes 17
```

*= (multiplication and assignment).

JavaScript
```
let c = 4;
c *= 2; // Equivalent to c = c * 2; 'c' becomes 8
```

/= (division and assignment).

JavaScript
```
let d = 16;
d /= 4; // Equivalent to d = d / 4; 'd' becomes 4
```

%= (remainder and assignment).

JavaScript
```
let e = 10;
e %= 3; // Equivalent to e = e % 3; 'e' becomes 1
```

**= (exponentiation and assignment).

JavaScript
```
let f = 2;
f **= 3; // Equivalent to f = f ** 3; 'f' becomes 8
```

<<= (left shift and assignment).

JavaScript
```
let g = 5;
g <<= 2; // Equivalent to g = g << 2; 'g' becomes 20
```

>>= (right shift and assignment).

JavaScript
```
let h = 20;
h >>= 2; // Equivalent to h = h >> 2; 'h' becomes 5
```

&= (bitwise AND and assignment).

JavaScript
```
let i = 15;
i &= 5; // Equivalent to i = i & 5; 'i' becomes 5
```

|= (bitwise OR and assignment).

JavaScript
```
let j = 5;
j |= 3; // Equivalent to j = j | 3; 'j' becomes 7
```

^= (bitwise XOR and assignment).

JavaScript
```
let k = 10;
k ^= 3; // Equivalent to k = k ^ 3; 'k' becomes 9
```

3. Destructuring Assignment:

- Destructuring allows you to unpack values from arrays or properties of objects into distinct variables.
- **Example (array destructuring):**

JavaScript
```
const myArray = [10, 20, 30];
const [a, b, c] = myArray; // a = 10, b = 20, c = 30
```

Example (object destructuring).

JavaScript
```
const myObject = { name: "Alice", age: 30 };
const { name, age } = myObject; // name = "Alice", age = 30
```

4. Logical OR Assignment (||=):

- Assigns the right-hand value to the left-hand variable only if the left-hand variable is falsy (e.g., false, 0, "", null, undefined).
- **Example:**

JavaScript
```
let x = null;
x ||= 10; // x becomes 10 (because null is falsy)
```

DOM(Document Object Model):

The HTML DOM (Document Object Model) is a programming interface that represents the structure of a web page in a way that programming languages like JavaScript can understand and manipulate.

The DOM, or Document Object Model, is a programming interface for web documents. It represents the structure of a document as a tree of objects, where each object corresponds to a part of the document, such as elements, attributes, and text. The DOM provides a way for programs to manipulate the structure, style, and content of web documents dynamically.

Key points about the DOM:

- **Tree Structure:** The DOM represents an HTML or XML document as a tree structure, with each node in the tree corresponding to an object in the document.
- **Objects:** Each element, attribute, and piece of text in the document is represented by a specific object in the DOM. These objects can be manipulated using programming languages like JavaScript.
- **Dynamic Interaction:** The DOM enables dynamic interaction with web pages. JavaScript can be used to access, modify, and update the content and structure of a document in real-time, allowing for interactive and responsive user interfaces.
- **Platform-Neutral:** The DOM is platform-neutral, meaning that it provides a standardized way to access and manipulate document content regardless of the underlying operating system or browser.
- **Event Handling:** The DOM allows the registration of event handlers, enabling developers to respond to user actions (such as clicks or key presses) and update the document accordingly.

Form with a text control and button:

HTML forms and buttons are powerful tools for interacting with a website's users. Most commonly, they provide users with controls to manipulate a user interface (UI) or input data when required.

In this article, we provide an introduction to the basics of forms and buttons. There is a lot more to know — a lot of input types and form features are not mentioned — but this article will give you a solid foundation for most cases. You can learn the advanced or specialized uses on a need-to-know basis as part of the constant learning you'll do throughout your career.

Prerequisites: Basic HTML familiarity, as covered in Basic HTML Syntax. Text-level semantics such as headings and paragraphs and lists. Structural HTML.

Learning outcomes:

- An appreciation that forms and buttons are the main tools for users to interact with a website, along with links.
- Different button types.
- Common <input> types.
- Common attributes such as name and value.
- The <form> element, and the basics of form submission.
- Making forms accessible with labels and correct semantics.
- Other control types: <textarea>, <select>, and <option>.
- Client-side validation basics.

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

Example

First name:
John
Last name:
Doe

Submit

Try it Yourself »

HTML <button> Tag

The <button> tag defines a clickable button.

Inside a <button> element you can put text (and tags like <i>, <b>, ,
, <img>, etc.). That is not possible with a button created with the <input> element!

Tip: Always specify the type attribute for a <button> element, to tell browsers what type of button it is.

Tip: You can easily style buttons with CSS! Look at the examples below or visit our CSS Buttons .

Example

A clickable button is marked up as follows:

```
<button type="button">Click Me!</button>
```

Event –Handler attributes:

HTML event attributes are used to define actions that should occur when specific events are triggered on a webpage.

- They enable interaction between the user and the system, like clicking a button or resizing a window.
- Event attributes are added to HTML elements to specify the event type and action.
- They enhance dynamic behavior and improve user experience on a webpage.

```
<html>

<head>

   <script>

      function displayMessage() {

         alert("Button was clicked!");

      }

   </script>

</head>

<body>

   <button onclick="displayMessage()">Click Me!</button>

</body>

</html>
```

- <button onclick=”displayMessage()”>Click Me!</button>: The <button> element includes the onclick attribute, which is set to call the displayMessage() function when the button is clicked.

- function displayMessage() { alert("Button was clicked!"); }: This JavaScript function displays an alert box with the message "Button was clicked!" when invoked.

The complete list of Event Attributes are given below:

Window Event Attributes

Window Event Attributes	Description	Example
onafterprint	Used with onbeforeprint to perform actions after the page is printed.	Try
onbeforeprint	The alert message display before the print dialogue box appears.	Try
onbeforeunload	The onbeforeunload event run when the document is about to be unloaded.	Try
onerror	This attribute works when an error occurs while loading an external file.	Try
onhashchange	This attribute works when there has been changes to the anchor part.	Try
onload	This attribute works when an object has been loaded.	Try
onoffline	The onoffline event attribute works when the browser work in offline mode.	
ononline	The ononline event attribute works when the browser starts working in online mode.	
onpageshow	This event occurs when a user navigates to a website.	
onresize	The onresize event attribute is triggered each time when resize the browser window size.	Try
onunload	Fired when the document is unloaded (e.g., when navigating away).	

Rollover using mouse events:

Events in JavaScript provide a dynamic interface to the webpage. There are wide variety of events such as user **clicking**, **moving** the mouse over an element, etc. Events that occur when the mouse interacts with the HTML document falls under the category of MouseEvent property.

- **mouseover:** The onmouseover event triggers when the mouse pointer enters an element or any one of its child elements.

```
<element onmouseover="myfunction()">
```

- **mouseenter:** The onmouseenter event is triggered only when the mouse pointer hits the element.

```
<element onmouseenter="myfunction()">
```

- **mousemove:** The onmousemove event is triggered each time the mouse pointer is moved when it is over an element.

```
<element onmousemove="myfunction()">
```

The MouseEvent Object handles events that occur when the mouse interacts with the HTML document.

Mouse Events

Event	Occurs When
onclick	A user clicks on an element
oncontextmenu	A user right-clicks on an element
ondblclick	A user double-clicks on an element

onmousedown	A mouse button is pressed over an element
onmouseenter	The mouse pointer moves into an element
onmouseleave	The mouse pointer moves out of an element
onmousemove	The mouse pointer moves over an element
onmouseout	The mouse pointer moves out of an element
onmouseover	The mouse pointer moves onto an element
onmouseup	A mouse button is released over an element