Prachi Shahare

Assistant Professor

Computer Department

Durga Mahavidyalaya, Raipur

Topic: Javascript

JAVASCRIPT

Introducing JavaScript

JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive (e.g., having complex animations, clickable buttons, popup menus, etc.). There are also more advanced server side versions of JavaScript such as Node.js, which allow you to add more functionality to a website than downloading files (such as realtime collaboration between multiple computers). Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them.

JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements. Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects; for example:

- *Client-side JavaScript* extends the core language by supplying objects to control a browser and its *Document Object Model* (DOM). For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.
- *Server-side JavaScript* extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application to communicate with a database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server.

JavaScript can be added to your HTML file in two ways:

- **Internal JS:** We can add JavaScript directly to our HTML file by writing the code inside the <script> tag. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.
- **External JS:** We can write JavaScript code in other file having an extension.js and then link this file inside the <head> tag of the HTML file in which we want to add this code.

Syntax:

```
<script>
   // JavaScript Code
</script>
```

Example

```
<!DOCTYPE html>
<html lang="en">

<head>
   <title>
      Basic Example to Describe JavaScript
   </title>
</head>

<body>

   <!-- JavaScript code can be embedded inside
      head section or body section -->
   <script>
      console.log("Welcome to Javascript");
   </script>
```

</body>
</html>
Output: The output will display on the console.
Welcome to Javascript

Features of JavaScript:

According to a recent survey conducted by **Stack Overflow**, JavaScript is the most popular language on earth. With advances in browser technology and JavaScript having moved into the server with Node.js and other frameworks, JavaScript is capable of so much more. Here are a few things that we can do with JavaScript:

- JavaScript was created in the first place for DOM manipulation. Earlier websites were mostly static, after JS was created dynamic Web sites were made.
- Functions in JS are objects. They may have properties and methods just like another object. They can be passed as arguments in other functions.
- Can handle date and time.
- Performs Form Validation although the forms are created using HTML.
- No compiler is needed.

Applications of JavaScript:

- **Web Development:** Adding interactivity and behavior to static sites JavaScript was invented to do this in 1995. By using AngularJS that can be achieved so easily.
- **Web Applications:** With technology, browsers have improved to the extent that a language was required to create robust web applications. When we explore a map in Google Maps then we only need to click and drag the mouse. All detailed view is just a click away, and this is possible only because of JavaScript. It uses Application Programming Interfaces(APIs) that provide extra power to the code. The Electron and React is helpful in this department.
- **Server Applications:** With the help of Node.js, JavaScript made its way from client to server and node.js is the most powerful on the server-side.
- **Games:** Not only in websites, but JavaScript also helps in creating games for leisure. The combination of JavaScript and HTML 5 makes JavaScript popular in game development as well. It provides the EaseJS library which provides solutions for working with rich graphics.
- **Smartwatches:** JavaScript is being used in all possible devices and applications. It provides a library PebbleJS which is used in smartwatch applications. This framework works for applications that require the internet for its functioning.
- **Art:** Artists and designers can create whatever they want using JavaScript to draw on HTML 5 canvas, and make the sound more effective also can be used **p5.js** library.
- **Machine Learning:** This JavaScript ml5.js library can be used in web development by using machine learning.
- **Mobile Applications:** JavaScript can also be used to build an application for non-web contexts. The features and uses of JavaScript make it a powerful tool for creating mobile applications. This is a Framework for building web and mobile apps using JavaScript. Using React Native, we can build mobile applications for different operating systems. We do not require to write code for different systems. Write once use it anywhere!

Limitations of JavaScript:

- **Security risks:** JavaScript can be used to fetch data using AJAX or by manipulating tags that load data such as <img>, <object>, <script>. These attacks are called cross site script attacks. They inject JS that is not the part of the site into the visitor's browser thus fetching the details.
- **Performance:** JavaScript does not provide the same level of performance as offered by many traditional languages as a complex program written in JavaScript would be comparatively slow. But as JavaScript is used to perform simple tasks in a browser, so performance is not considered a big restriction in its use.
- **Complexity:** To master a scripting language, programmers must have a thorough knowledge of all the programming concepts, core language objects, client and server-side objects otherwise it would be difficult for them to write advanced scripts using JavaScript.

- **Weak error handling and type checking facilities:** It is weakly typed language as there is no need to specify the data type of the variable. So wrong type checking is not performed by compile.

Embedding JavaScript in HTML page:

JavaScript adds a breath of fresh air to a static HTML file. You can embed JavaScript **directly or as an external file** in HTML.

HTML, CSS, and JavaScript are the three basic pillars of the modern world wide web. If you want to create a modern, interactive website, you can hardly avoid **enlivening HTML text files by incorporating CSS and JavaScript**. HTML files are plain text documents for creating and structuring website content. HTML is easy to learn, and you can get by without formatting. It's also user-friendly thanks to free code editors such as Notepad++ or Kate. On the other hand, creativity is provided by CSS text files embedded in HTML embedded CSS text files, which define the layout, color scheme, typographies, and other design elements of a website.

A website only becomes truly interactive with JavaScript elements, which **add dynamic behavior to the content**. JavaScript files embedded in HTML, for example, enable the active modification of website content such as automatic date display, day-dependent coloring, or automatically displayed messages when the website is visited. With enabled JavaScript, JavaScript content can be executed directly in most browsers. This saves processing power and improves the loading speed of interactive content and animations on a website.

What options are there for embedding JavaScript?

You can **note or reference** JavaScript elements in HTML source code as script elements as follows.

```
<script> <script>
JavaScript-Element
```

Script elements are usually embedded in the **body or head area** of an HTML document.

Depending on how you include JavaScript in HTML, you can choose from the following options:

- **Note JavaScript directly in an HTML page**: Direct notation in HTML is done in the head element and ensures that JavaScript files load as quickly as possible via direct notation. The downside is that direct notations must be made for each HTML document on a website to load content.
- **Referencing JavaScript as an external file in HTML**: Embedding an external JavaScript file references the JavaScript file in the HTML text. **Embedding as a reference** allows externally noted JavaScript files to be loaded on multiple pages without having to note them directly as verbose JavaScript elements in HTML files.

How to embed JavaScript in HTML: practical examples

You can choose between two different methods to include JavaScript. We'll introduce you to both.

Embedding JavaScript in HTML directly

To load JavaScript elements as quickly as possible, place the script element **in the HTML head or body**. In new browsers, a relatively simple JS source code is sufficient. In the following example, the message "Hello friend" is written on a website via JavaScript integration:

```
<!DOCTYPE html>
<html>
<head>
```

```
  <title>JavaScript: Hello Friend</title>
  <script>
    alert(“Hello Friend!“);
  </script>
</head>
<body>
  <p>This website only displays a message box.</p>
</body>
</html>
```

Complex visualizations, animations, or interactive elements can also be noted directly in HTML. The advantage of direct embedding is that JavaScript elements are edited directly in the HTML file. The disadvantage is that you have to note JavaScript functions separately in the source code for each HTML document. This leads to **increased effort when maintaining the source code**.

Embedding JavaScript as an external file

It’s more efficient and more common to note down **JavaScript as an external file** and reference it in the HTML document. In this way, JavaScript can be included in HTML as if the file were noted directly in the source code.

In the HTML document it looks like this:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>Include external JavaScript file</title>
  <script src="filename.js"></script>
</head>
</body>
```

Embedding JavaScript as an external file offers the **advantage that only the link to the external file** is referenced. So, all desired HTML pages with a corresponding reference can access the file and load it faster. In addition, the source code is easier to maintain, since it’s not several HTML documents, but only one JavaScript file.

Special features in different HTML versions

When embedding JavaScript in HTML, note that **older browsers** require more verbose source code. Instead of the simpler HTML 5 <script>...</script> tag, the JavaScript inclusion would look like this:

```
<script type="text/javascript">
JavaScript-Element
</script>
```

If you want to display a placeholder message for **visitors who have JavaScript disabled**, use the following noscript tag:

```
<head>
  <script>
    JavaScript-Element
  </script>
  <noscript>
    Please enable JavaScript to see the JavaScript element.
  </noscript>
</head>
```

Event Handling

- Event Handling is a software routine that processes actions, such as keystrokes and mouse movements.
- It is the receipt of an event at some event handler from an event producer and subsequent processes.

Functions of Event Handling

- Event Handling identifies where an event should be forwarded.
- It makes the forward event.
- It receives the forwarded event.
- It takes some kind of appropriate action in response, such as writing to a log, sending an error or recovery routine or sending a message.
- The event handler may ultimately forward the event to an event consumer.

Event Handlers

Event Handler	Description
onclick	When a click action occurs on an HTML element, this event handler runs a JavaScript script. For example, the onClick event handler may be called when a button is clicked, a link is clicked, a checkbox is checked, or an image map is selected.
onmouseover	When the mouse is over a particular link or object, this event handler runs a JavaScript script.
onload	When a window or image has fully loaded, this event handler executes some JavaScript code.
onkeypress	When a user presses a key, this event handler executes JavaScript code.
onmouseout	When the mouse leaves a specific link or object, this event handler runs a JavaScript script.
onkeyup	When a key is released during a keyboard action, this event handler executes JavaScript code.
onkeydown	When a key is pressed on the keyboard during an action, this event handler runs a JavaScript script.

onclick Event Type

This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

Example

Try the following example.

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function sayHello() {
          alert("Hello World")
        }
      //-->
    </script>
  </head>
```

```
  <body>
    <p>Click the following button and see result</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
    </form>
  </body>
</html>
```

Output

Click the following button and see result

onsubmit Event Type

onsubmit is an event that occurs when you try to submit a form. You can put your form validation against this event type.

Example

The following example shows how to use onsubmit. Here we are calling a **validate()** function before submitting a form data to the webserver. If **validate()** function returns true, the form will be submitted, otherwise it will not submit the data.

Try the following example.

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function validation() {
          all validation goes here
          .........
          return either true or false
        }
      //-->
    </script>
  </head>

  <body>
    <form method = "POST" action = "t.cgi" onsubmit = "return validate()">
      .......
      <input type = "submit" value = "Submit" />
    </form>
  </body>
</html>
```

---

onmouseover and onmouseout

These two event types will help you create nice effects with images or even with text as well. The **onmouseover** event triggers when you bring your mouse over any element and the **onmouseout** triggers when you move your mouse out from that element. Try the following example.

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function over() {
```

```
            document.write ("Mouse Over");
         }
         function out() {
            document.write ("Mouse Out");
         }
      //-->
   </script>
</head>

<body>
   <p>Bring your mouse inside the division to see the result:</p>
   <div onmouseover = "over()" onmouseout = "out()">
      <h2> This is inside the division </h2>
   </div>
</body>
</html>
```

HTML 5 Standard Events

The standard HTML 5 events are listed here for your reference. Here script indicates a Javascript function to be executed against that event.

Attribute	Value	Description
Offline	script	Triggers when the document goes offline
Onabort	script	Triggers on an abort event
onafterprint	script	Triggers after the document is printed
onbeforeonload	script	Triggers before the document loads
onbeforeprint	script	Triggers before the document is printed
onblur	script	Triggers when the window loses focus
oncanplay	script	Triggers when media can start play, but might has to stop for buffering
oncanplaythrough	script	Triggers when media can be played to the end, without stopping for buffering
onchange	script	Triggers when an element changes
onclick	script	Triggers on a mouse click
oncontextmenu	script	Triggers when a context menu is triggered
ondblclick	script	Triggers on a mouse double-click
ondrag	script	Triggers when an element is dragged
ondragend	script	Triggers at the end of a drag operation
ondragenter	script	Triggers when an element has been dragged to a valid drop target

ondragleave	script	Triggers when an element is being dragged over a valid drop target
ondragover	script	Triggers at the start of a drag operation
ondragstart	script	Triggers at the start of a drag operation
ondrop	script	Triggers when dragged element is being dropped
ondurationchange	script	Triggers when the length of the media is changed
onemptied	script	Triggers when a media resource element suddenly becomes empty.
onended	script	Triggers when media has reach the end
onerror	script	Triggers when an error occur
onfocus	script	Triggers when the window gets focus
onformchange	script	Triggers when a form changes
onforminput	script	Triggers when a form gets user input
onhaschange	script	Triggers when the document has change
oninput	script	Triggers when an element gets user input
oninvalid	script	Triggers when an element is invalid
onkeydown	script	Triggers when a key is pressed
onkeypress	script	Triggers when a key is pressed and released
onkeyup	script	Triggers when a key is released
onload	script	Triggers when the document loads
onloadeddata	script	Triggers when media data is loaded
onloadedmetadata	script	Triggers when the duration and other media data of a media element is loaded
onloadstart	script	Triggers when the browser starts to load the media data
onmessage	script	Triggers when the message is triggered
onmousedown	script	Triggers when a mouse button is pressed
onmousemove	script	Triggers when the mouse pointer moves
onmouseout	script	Triggers when the mouse pointer moves out of an element
onmouseover	script	Triggers when the mouse pointer moves over an element
onmouseup	script	Triggers when a mouse button is released

onmousewheel	script	Triggers when the mouse wheel is being rotated
onoffline	script	Triggers when the document goes offline
onoine	script	Triggers when the document comes online
ononline	script	Triggers when the document comes online
onpagehide	script	Triggers when the window is hidden
onpageshow	script	Triggers when the window becomes visible
onpause	script	Triggers when media data is paused
onplay	script	Triggers when media data is going to start playing
onplaying	script	Triggers when media data has start playing
onpopstate	script	Triggers when the window's history changes
onprogress	script	Triggers when the browser is fetching the media data
onratechange	script	Triggers when the media data's playing rate has changed
onreadystatechange	script	Triggers when the ready-state changes
onredo	script	Triggers when the document performs a redo
onresize	script	Triggers when the window is resized
onscroll	script	Triggers when an element's scrollbar is being scrolled
onseeked	script	Triggers when a media element's seeking attribute is no longer true, and the seeking has ended
onseeking	script	Triggers when a media element's seeking attribute is true, and the seeking has begun
onselect	script	Triggers when an element is selected
onstalled	script	Triggers when there is an error in fetching media data
onstorage	script	Triggers when a document loads
onsubmit	script	Triggers when a form is submitted
onsuspend	script	Triggers when the browser has been fetching media data, but stopped before the entire media file was fetched
ontimeupdate	script	Triggers when media changes its playing position
onundo	script	Triggers when a document performs an undo
onunload	script	Triggers when the user leaves the document
onvolumechange	script	Triggers when media changes the volume, also when volume is set to "mute"

onwaiting	script	Triggers when media has stopped playing, but is expected to resume