



JavaScript and Angular JS

जावास्क्रिप्ट एण्ड एनगुलर जेएस

Web Designing and Publishing

6.0 JavaScript and Angular JS - जावास्क्रिप्ट एण्ड एनगुलर जेएस

JavaScript - जावास्क्रिप्ट

JavaScript is an object-oriented program scripting language which is also referred as multi-paradigm, functional and event-driven language. It is symbolized as JS and is used for developing web applications with HTML and CSS. It uses '.js' filename extension.

जावास्क्रिप्ट एक ऑब्जेक्ट-ओरिएंटेड प्रोग्रामिंग स्क्रिप्टिंग भाषा है। इसे मल्टी-पैराडिगम (बहु-प्रतिमान), फंक्शनल एवं इवेंट ड्रिवेन लैंग्वेज के रूप में भी जाना जाता है। यह मुख्य रूप से JS के रूप में सिंबॉलाइज्ड है। यह HTML और CSS के साथ वेब एप्लिकेशन विकसित करने के लिए सबसे अधिक उपयोग की जाने वाली तकनीकों में से एक है। जावास्क्रिप्ट का फाइलनेम एक्सटेंशन '.js' है।

Designing of javascript is done by Brendan Eich but it is developed by Netscape, Mozilla and Ecma organizations. It is released in 1995 and is interpreted and used as client side scripting language with Java. It can be run on the browser without recompiling of the code. It makes the webpages more dynamic and more attractive.

जावास्क्रिप्ट ब्रेंडन ईच द्वारा डिजाइन किया गया है लेकिन यह नेटस्केप, मोजिला फाउंडेशन और एक्मा इंटरनेशनल द्वारा विकसित किया गया। यह पहली बार वर्ष 1995 में जारी किया गया था। जावास्क्रिप्ट को क्लाइंट साइड स्क्रिप्टिंग भाषा के रूप में जावा के साथ इंटरप्रिट और उपयोग किया जाता है, इसके कोड को फिर से बिना रीकम्पाइल किए ब्राउजर पर चलाया जा सकता है। यह वेब पेज या एप्लिकेशन को अधिक गतिशील और इंटरैक्टिव बनाने में मदद करता है।

JavaScript has features like dynamic typing which make it better and safe programming language. It can be manipulated in a browser for doing changes in webpage or web applications. It will take care of security and performance. JavaScript also helps in adding a new HTML

Angular JS - एनगुलर जेएस

AngularJS is a JavaScript-based front-end web framework. It is an open source and gives the dynamic quality to web pages.

AngularJS एक जावास्क्रिप्ट बेस्ड फ्रंट एंड वेब फ्रेमवर्क है, यह ओपेन सोर्स होता है और वेब पेजेस को डायनैमिक क्वालिटी प्रदान करता है।

It is developed by **Google** and initially released in the year 2010. It uses the markup language that is HTML language to extend its syntax.

AngularJS Google द्वारा विकसित किया गया था और वर्ष 2010 में जारी किया गया था। यह अपने सिंटैक्स को एक्सटेंड करने के लिए मार्कअप लैंग्वेज जोकि एचटीएमएल है को यूज करता है।

AngularJS (Dynamic language) and HTML language (Static language) both are used in developing web applications more efficiently and dynamically.

AngularJS (डायनैमिक भाषा) और HTML भाषा (स्टैटिक भाषा) दोनों का उपयोग वेब एप्लिकेशन को अधिक कुशलता और गतिशील रूप से विकसित करने में किया जाता है।

AngularJS mainly works on the model view controller i.e. MVC concept, which defines the behavior of data, logic and view layer. These layers help in fetching the data, modifying the data and

page and change the existing content and styles.

जावास्क्रिप्ट में डायनामिक टाइपिंग जैसे कई फीचर हैं जो इसे अच्छा एवं सेफ प्रोग्रामिंग लैंग्वेज बनाते हैं। वेबपेज या वेब एप्लिकेशन में परिवर्तन करने के लिए इसे वेब ब्राउजर में मैनिपुलेट किया जा सकता है यह सिक्यूरिटी और पेरफोमेंस को भी बनाये रखता है। जावास्क्रिप्ट नए एचटीएमएल पेज को जोड़ने एवं एक्जिस्टिंग कंटेंट एवं एस्टाइल को भी चेंज करने में मदद करता है।

JavaScript is referred as Unique because of its full integration with HTML/CSS. JavaScript also allows simple things to be done simply only. It is also supported by the major and modern browsers, and is being enabled by default as well.

HTML/CSS के साथ पूर्ण एकीकरण के कारण जावास्क्रिप्ट को अद्वितीय रूप में संदर्भित किया जाता है। जावास्क्रिप्ट भी सरल चीजों को सरलता से करने की अनुमति देता है। यह प्रमुख और आधुनिक ब्राउजरों द्वारा भी सपोर्ट किया जाता है, और यह डिफॉल्ट रूप से भी इनेबल होता है।

displaying the data to a user. Whenever the view has changed or some event clicked by a user, angular JS tried to handle this at a controller and change the data in model and displays to a user.

AngularJS मुख्य रूप से मॉडल व्यू कंट्रोलर यानी MVC कॉन्सेप्ट पर काम करता है, जो डेटा, लॉजिक और व्यू लेयर के व्यवहार को परिभाषित करता है। ये परतें डेटा को लाने, डेटा को संशोधित करने और उपयोगकर्ता को डेटा प्रदर्शित करने में मदद करती हैं। जब भी दृश्य बदला है या किसी उपयोगकर्ता द्वारा क्लिक की गई कोई घटना है, तो एंग्यूलर जेएस ने एक कंट्रोलर पर इसे संभालने और उपयोगकर्ता के लिए मॉडल और डिस्प्ले में डेटा बदलने की कोशिश की।

6.1 Introduction to Client Side Scripting Language

क्लाइंट साइड स्क्रिप्टिंग भाषा का परिचय

For a Web page, HTML supplies document content and structure while CSS provides presentation styling. In addition, client-side scripts can control browser actions associated with a Web page. Scripts are programs written in a simple and easy-to-use language to specify control of other programs. Client-side scripts are almost always written in the JavaScript language to control browsers actions.

एक वेब पेज के लिए, HTML डॉक्यूमेंट सामग्री और संरचना की आपूर्ति करता है जबकि CSS प्रजेंटेशन स्टाइल प्रदान करता है। इसके अलावा, क्लाइंट-साइड स्क्रिप्ट वेब पेज से जुड़े ब्राउजर एक्शन को नियंत्रित कर सकते हैं। अन्य प्रोग्रामों के नियंत्रण को निर्दिष्ट करने के लिए स्क्रिप्ट को एक सरल और आसान भाषा में लिखा गया प्रोग्राम है। ब्राउजर क्रियाओं को नियंत्रित करने के लिए क्लाइंट-साइड स्क्रिप्ट लगभग हमेशा जावास्क्रिप्ट भाषा में लिखी जाती हैं।

Layer- परत	Content- सामग्री	Format - स्वरूप
Data Layer डेटा लेयर	Text/images to display. प्रदर्शित करने के लिए टेक्स्ट/इमेज।	HTML एचटीएमएल
Presentation Layer प्रजेंटेशन लेयर	Style rules to overwrite browser defaults. स्टाइल रूल्स डिफॉल्ट ब्राउजर को ओवरराइट करते हैं।	CSS सीएसएस
Activity Layer एक्टिविटी लेयर	Scripting to provide interactivity to the site. साइट से इंटरैक्टिविटी प्रदान करने के लिए स्क्रिप्टिंग।	JavaScript, TypeScript etc., जावास्क्रिप्ट, टाइपस्क्रिप्ट आदि।

Tasks performed with client-side scripts include:

क्लाइंट-साइड स्क्रिप्ट के साथ किए गए कार्य में शामिल हैं:

- Asking the browser to display information - ब्राउजर से जानकारी प्रदर्शित करने के लिए पूछना।
- Making the Web page different depending on the browser and browser features.

ब्राउजर और ब्राउजर फीचर के आधार पर वेब पेज को अलग बनाना।

- **Monitoring user events and specifying reactions.** - उपयोगकर्ता की घटनाओं की निगरानी करना और प्रतिक्रियाओं को निर्दिष्ट करना।
- **Generating HTML code for parts of the page.** - पेज के कुछ हिस्सों के लिए HTML कोड बनाना।
- **Modifying a page in response to events.** - घटनाओं के जवाब में एक पृष्ठ को संशोधित करना।
- **Checking correctness of user input.** - उपयोगकर्ता इनपुट की शुद्धता की जाँच करना।
- **Replacing and updating parts of a page.** - पृष्ठ के कुछ हिस्सों को बदलना और अपडेट करना।
- **Changing the style and position of displayed elements dynamically.** - प्रदर्शित तत्वों की शैली और स्थिति को गतिशील रूप से बदलना।

Client-side scripting can make Web pages more dynamic and more responsive. We will see how JavaScript is used in combination with other Web constructs for user interface purposes. The importance is in applying JavaScript in practice rather than studying it as a programming language.

क्लाइंट-साइड स्क्रिप्टिंग वेब पृष्ठों को अधिक डायनामिक और अधिक रिस्पॉन्सिव बना सकती है। हम देखेंगे कि उपयोगकर्ता इंटरफेस प्रयोजनों के लिए अन्य वेब निर्माणों के संयोजन में जावास्क्रिप्ट का उपयोग कैसे किया जाता है। महत्वपूर्ण यह है कि जावास्क्रिप्ट को प्रोग्रामिंग भाषा के रूप में अध्ययन करने के बजाय प्रैक्टिस में लागू किया जाए।

JavaScript - जावास्क्रिप्ट

JavaScript is a widely used scripting language originally developed by Netscape for both client-side and server-side scripting. The language is becoming an international standard, approved by the European standards body ECMA (ECMA-262) in 1997 and later by the ISO in 1998. Client-side JavaScript is used widely and supported well by major browsers including NN, IE, AOL, MOZILLA, and Opera. We shall present client-side JavaScript for adding dynamism and interactivity to Web pages and will refer to it simply as **JavaScript**.

जावास्क्रिप्ट एक व्यापक रूप से इस्तेमाल की जाने वाली स्क्रिप्टिंग भाषा है जो मूल रूप से क्लाइंट-साइड और सर्वर-साइड स्क्रिप्टिंग दोनों के लिए नेटस्केप द्वारा विकसित की गई है। यह भाषा एक अंतर्राष्ट्रीय मानक बन रही है, इसको 1997 में यूरोपीय मानक निकाय ECMA (ECMA-262) के द्वारा अनुमोदित कर दिया गया था, बाद में 1998 में ISO द्वारा अनुमोदित एक अंतर्राष्ट्रीय मानक बन गया है। क्लाइंट-साइड जावास्क्रिप्ट का व्यापक रूप से उपयोग किया जाता है और NN, IE, AOL, MOZILLA, और ओपेरा सहित प्रमुख ब्राउजरों द्वारा इसका अच्छी तरह से समर्थन किया जाता है। हम वेब पेजों में गतिशीलता और अन्तर्क्रियाशीलता जोड़ने के लिए क्लाइंट-साइड जावास्क्रिप्ट प्रस्तुत करेंगे और इसे केवल जावास्क्रिप्ट के रूप में संदर्भित करेंगे।

Javascript programs are not written in Java which actually itself is another programming language. Javascript programs are embedded in web pages and are executed by browsers that provide the host environment or execution context. HTML code can be generated by Javascript code to get included in the page and perform tasks as a reaction to certain events. The host environments supplies document objects, and javascript built-in objects for the script. These objects represents the whole document and well-defined parts in it, such as ; windows, buttons, menus, pop-ups, dialog boxes, text areas, anchors, frames, history, cookies, and input/output. For various purposes objects provide useful methods and functions contained in it.

नाम का सुझाव देने के विपरीत, जावा में जावास्क्रिप्ट प्रोग्राम नहीं लिखे जाते हैं जो वास्तव में एक अन्य प्रोग्रामिंग भाषा है। जावास्क्रिप्ट प्रोग्रामों को वेब पृष्ठों में एम्बेड किया जाता है और उन ब्राउजरों द्वारा निष्पादित किया जाता है जो होस्ट वातावरण या निष्पादन संदर्भ प्रदान करते हैं। जावास्क्रिप्ट कोड पेज में शामिल किए जाने और कुछ घटनाओं की प्रतिक्रिया में कार्यों को करने के लिए HTML कोड उत्पन्न कर सकता है। होस्ट इनवॉयरमेंट स्क्रिप्ट हेरफेर (manipulation) के लिए

डॉक्यूमेंट ऑब्जेक्ट्स, ब्राउजर ऑब्जेक्ट्स और जावास्क्रिप्ट बिल्ट-इन ऑब्जेक्ट्स की आपूर्ति करता है। ये ऑब्जेक्ट पूरे डॉक्यूमेंट और इसमें अच्छी तरह से परिभाषित भागों का प्रतिनिधित्व कर सकते हैं, उदाहरण के लिए, विंडोज, बटन, मैन्यू, पॉप-अप, डायलॉग बॉक्स, टेक्स्ट क्षेत्र, एंकर, फ्रेम, हिस्ट्री, कुकीज और इनपुट/आउटपुट। ऑब्जेक्ट्स विभिन्न उद्देश्यों के लिए ऑब्जेक्ट में निहित उपयोगी तरीके, फंक्शन प्रदान करते हैं।

To connect scripting code the host environment also provides a means with events such as focus and mouse actions, page and image loading, form input and submission, error and abort. While the page is loading; the Scripting code can also perform computation. Therefore, the displayed page is the result of a combination of HTML, CSS, and JavaScript actions.

होस्ट इनवॉयरमेंट फोकसिंग कोड को फोकस और माउस एक्शन, पेज और इमेज लोडिंग, फॉर्म इनपुट और सबमिशन, एरर और एबॉर्ट जैसी घटनाओं से जोड़ने का साधन भी प्रदान करता है। स्क्रिप्टिंग कोड भी गणना कर सकता है क्योंकि पेज लोड हो रहा है। इस प्रकार, प्रदर्शित पृष्ठ HTML, CSS और जावास्क्रिप्ट एक्शन के संयोजन का परिणाम है।

A first JavaScript program - एक पहला जावास्क्रिप्ट प्रोग्राम

You type your JavaScript code inside `<script>` and `</script>` tags. The script tags can go inside header or body of the HTML. Take a look at below example.

आप अपने जावास्क्रिप्ट कोड को `<script>.....</script>` टैग के अंदर रखें। स्क्रिप्ट टैग एचटीएमएल के हेडर या बॉडी के अंदर जा सकते हैं। नीचे दिए गए उदाहरण पर एक नजर डालें।

h1 h2 h3 Tip b i u span link img ul li	PREVIEW
<pre> 1 <!DOCTYPE html> 2 <html> 3 <body> 4 <h1>JavaScript Exmple</h1> 5 <script> 6 document.write("<h4>Hello world !!!</h4>"); 7 </script> 8 </body> 9 </html> </pre>	<p>JavaScript Exmple Hello world !!!</p>

When the above page is rendered by a browser, it will understand that all content inside the script tag is JavaScript. The optional `type="text/javascript"` indicates the browser that the code is in JavaScript language. The browser will now invoke the JavaScript engine (interpreter) and executes the code line by line. The outcome of the above program will be a line of text written to document element.

जब उपर्युक्त पेज एक ब्राउजर द्वारा प्रस्तुत किया जाता है, तो यह समझा जाएगा कि स्क्रिप्ट टैग के अंदर सभी सामग्री जावास्क्रिप्ट है। वैकल्पिक प्रकार= "टेक्स्ट/जावास्क्रिप्ट" ब्राउजर को इंगित करता है कि कोड जावास्क्रिप्ट भाषा में है। ब्राउजर जावास्क्रिप्ट इंजन (इंटरप्रिटर) को काल कर कोड को लाइन बाई लाइन एक्जिक्यूट करेगा। उपरोक्त प्रोग्राम का परिणाम डॉक्यूमेंट एलिमेंट में लिखे गए टेक्स्ट की एक पंक्ति होगी।

The **document** element is the root of the DOM (document object model) available in all browsers. DOM provides a structure to the document and provisions a way for the browser and its JavaScript engine to access/change the style and content of the elements of the document structure.

डॉक्यूमेंट एलिमेंट सभी ब्राउजरों में उपलब्ध DOM (डॉक्यूमेंट ऑब्जेक्ट मॉडल) की जड़ है। DOM डॉक्यूमेंट को एक संरचना प्रदान करता है और डॉक्यूमेंट संरचना के तत्वों की शैली और सामग्री को एक्सेस/बदलने के लिए ब्राउजर और उसके जावास्क्रिप्ट इंजन के लिए एक तरीका प्रदान करता है।

Before adding javascript to HTML we have a brief knowledge of javascript function. Let us see here:

HTML में जावास्क्रिप्ट जोड़ने से पहले हमें जावास्क्रिप्ट फंक्शन का एक संक्षिप्त ज्ञान होना जरूरी है। आइये यहाँ देखते हैं:

Short Note on JavaScript Function

Function: It is the fundamental building blocks in JavaScript. It is the set of statements that performs a task or calculates a value. To use a function, you must define it somewhere in the scope from which you wish to call it. In javascript if you want to define a function then use function keyword before the function name as given below:

फंक्शन: यह जावास्क्रिप्ट में एक फंक्शन ब्लॉक है। यह उन स्टेटमेंट्स का एक समूह है जो किसी कार्य या किसी मान की गणना करता है। किसी फंक्शन का उपयोग करने के लिए, आपको इसे उस स्कोप के अंदर कहीं डिफाइन (परिभाषित) करना होगा, जहाँ से आप इसे कॉल करना चाहते हैं। जावास्क्रिप्ट में अगर आप किसी फंक्शन को परिभाषित करना चाहते हैं तो नीचे दिए गए सिनटैक्स उपयोग करें जहाँ फंक्शन के नाम से पहले फंक्शन कीवर्ड का होना जरूरी होता है।

Syntax: `function function_name(parameters)`
`{ statements; }`

Adding JavaScript to HTML - जावास्क्रिप्ट को HTML में जोड़ना

JavaScript code can be placed inside `<script>.....</script>` tags. The script tags can go anywhere in `<body>` or `<head>` section of HTML page. Remember, JavaScript code must be inside `<script>` tags to translate. Anything outside of `<script>` tag is taken as normal HTML code and not execute as a JavaScript code. You can place unlimited number of scripts in an HTML page. Scripts code can be placed anywhere between `<head>` tag or `<body>` tag or both.

जावास्क्रिप्ट कोड को `<script>.....</script>` टैग के अंदर रखा जा सकता है। स्क्रिप्ट टैग HTML पेज के `<body>` या `<head>` सेक्शन में कहीं भी जा सकते हैं। याद रखें, अनुवाद करने के लिए जावास्क्रिप्ट कोड `<script>` टैग के अंदर होना चाहिए। `<script>` टैग के बाहर कुछ भी सामान्य HTML कोड के रूप में लिया जाता है और जावास्क्रिप्ट कोड के रूप में एक्जिक्यूट नहीं किया जाता है। आप HTML पेज में असीमित संख्या में स्क्रिप्ट रख सकते हैं। स्क्रिप्ट कोड को `<head>` टैग या `<body>` टैग या दोनों के बीच कहीं भी रखा जा सकता है।

Java Script inside `<head>` tag - जावा स्क्रिप्ट `<head>` टैग के अंदर

JavaScript code is written within `<head>` section of page. Take a look at below example. जावास्क्रिप्ट कोड किसी पेज के `<head>` सेक्शन में लिखा जाता है। नीचे दिए गए उदाहरण पर एक नजर डालें।

```


1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <title>Example of Java Script inside head tag:</title>
5 <script>
6     function myFunction()
7     {
8         document.getElementById("id2").innerHTML="CCC is the first book of Balaji publication";
9     }
10 </script>
11 </head>
12 <body>
13 <center>Balaji Publication</center>
14 <button type="button" onclick="myFunction()">Click Me</button>
15 <h1 id="id2">Welcome to Balaji Publicaton Books</h1>
16 </body>
17 </html>


```

Output of the above code is here: _____

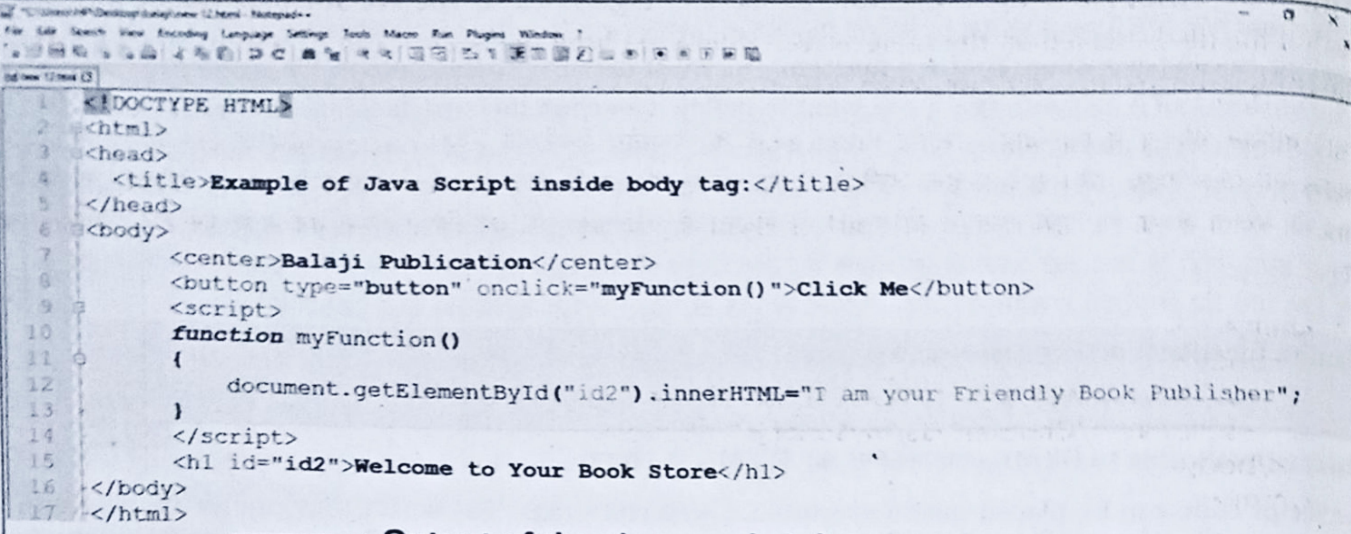
Before clicking Click Me button the output is:

After Clicking Click Me button the output is:

Balaji Publication

Welcome to Balaji Publicaton Books

Balaji Publication

CCC is the first book of Balaji publication

Java Script inside <body> tag - जावा स्क्रिप्ट <body> टैग के अंदर-

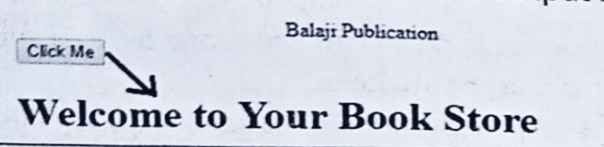
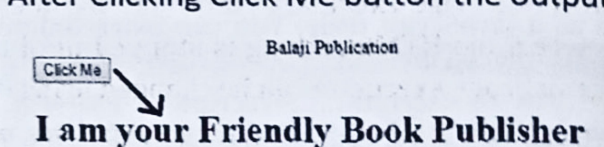


```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <title>Example of Java Script inside body tag:</title>
5 </head>
6 <body>
7   <center>Balaji Publication</center>
8   <button type="button" onclick="myFunction()" >Click Me</button>
9   <script>
10    function myFunction()
11    {
12      document.getElementById("id2").innerHTML="I am your Friendly Book Publisher";
13    }
14  </script>
15  <h1 id="id2">Welcome to Your Book Store</h1>
16 </body>
17 </html>

```

Output of the above code is here:

<p>Before clicking Click Me button the output is:</p> <div style="text-align: center;">  </div>	<p>After Clicking Click Me button the output is:</p> <div style="text-align: center;">  </div>
---	---

But it is a best practice to place scripts at the bottom of the page, just before closing of body tag i.e. before </body> tag. This will improve page load time. Otherwise JavaScript code execution will halt the page rendering, until it is fully done.

लेकिन </body> टैग से ठीक पहले पृष्ठ के निचले भाग पर स्क्रिप्ट रखना एक सर्वोत्तम अभ्यास है। इससे पेज लोड समय में सुधार होता है। अन्यथा जावास्क्रिप्ट कोड एक्जिक्यूशन पेज रेंडरिंग (प्रतिपादन) को रोक देगा, जब तक कि यह पूरी तरह से नहीं हो जाता।

JavaScript Outside of the HTML file - HTML फाइल के बाहर जावास्क्रिप्ट

If you want to place JavaScript code in an external file with .js extension; so you can place it by doing the step given below. The external scripts are useful for code reuse. When the same code is used in multiple sites or html pages, it makes sense to have the JavaScript code separated into its own js file. See the following example step by step.

यदि आप अपने जावास्क्रिप्ट कोड को .js एक्सटेंशन के साथ एक बाहरी फाइल में रखना चाहते हैं तो नीचे दिए गए स्टेप्स को फॉलो करें। स्क्रिप्ट कोड के पुनः उपयोग हेतु उपयोगी बनाने के लिए इसे बाहरी स्क्रिप्ट फाइल में रखना सही होगा। यदि सेम कोड कई साइट में या एचटीएमएल पेजेस में यूज किया गया हो तो यह अच्छा होगा की जावा स्क्रिप्ट्स कोड को इसके अपने जेएस फाइल में सेपरेट कर ले। स्टेप बाइ स्टेप आप नीचे दिए गए कोड को देखें।

Step 1: Write a program and save code in myScript.js file

एक प्रोग्राम लिखें और कोड को myScript.js फाइल नेम से सेव करें।

```

// write you own function here
function myFun() {
  document.getElementById("id1").innerHTML = "Really ... Awesome ...";
}
// end myScript.js file

```

Step 2: To make use of an external script file which is **myScript.js** in our current webpage, type the path of the script file in the **src** feature of **<script>** tag as below. The **src** attribute can be a relative path if the file is hosted on the same server where the html file is. Or it can be a fully qualified **url** if the **js** file is hosted on a different web server.

हमारे वर्तमान वेबपेज में एक बाहरी स्क्रिप्ट **myScript.js** का उपयोग करने के लिए, **<script>** टैग के **src** फीचर में स्क्रिप्ट फाइल का पाथ टाइप करें। यदि फाइल उसी सर्वर पर होस्ट की जाती है जहां **html** फाइल है, तो **src** फीचर एक सापेक्ष पाथ हो सकती है, या यह पूरी तरह से योग्य **url** हो सकता है यदि **js** फाइल को किसी भिन्न वेब सर्वर पर होस्ट किया गया हो।

```
<html>
  <body>
    <p id="id1">Welcome to Balaji Publications</p>
    <script src="/Chapter6/JS/myScript.js"></script>
  </body>
</html>
```

Script tag Attributes - स्क्रिप्ट टैग एट्रिब्यूट

Normal HTML page execution starts line by line. When an external JavaScript **<script>** element is encountered, the HTML parsing is stopped until a JavaScript is download and ready for execution. This normal page execution can be changed using **defer** and **async** attribute.

सामान्य HTML पेज एक्जीक्यूशन लाईन बाई लाईन शुरू होता है। जब एक एक्सटर्नल जावास्क्रिप्ट **<script>** एलिमेंट एन्काउंटर होता है, तो HTML पार्सिंग को तब तक रोक दिया जाता है जब तक कि जावास्क्रिप्ट डाउनलोड और क्रियान्वयन के लिए तैयार न हो। यह सामान्य पेज क्रियान्वयन **defer** और **async** एट्रिब्यूट का उपयोग करके बदला जा सकता है।

1. **HTML defer attribute:** JavaScript is downloaded parallelly with HTML parsing when a **defer** attribute is used but will be executed only after full HTML parsing is done.

HTML डिफर एट्रिब्यूट: जब डिफर एट्रिब्यूट का यूज किया जाता है, तो जावास्क्रिप्ट को HTML पार्सिंग के साथ समानांतर रूप से डाउनलोड किया जाता है, लेकिन पूर्ण HTML पार्सिंग होने के बाद ही इसे एक्सजिक्यूट किया जाता है।

```
<script src="/Chapter6/JS/myScript.js" defer></script>
```

2. **HTML async attribute:** When **async** attribute is used, the JavaScript will be downloaded as soon as the script is encountered and after the download, it will be executed parallelly along with HTML parsing.

HTML async एट्रिब्यूट: **async** एट्रिब्यूट को यूज करने पर: 'जावास्क्रिप्ट' स्क्रिप्ट के एन्काउंटर होते ही डाउनलोड होने लगता है और डाउनलोड होने के बाद यह एचटीएमएल पार्सिंग के साथ पैरलेली एक्सजिक्यूट होने लगता है।

```
<script src="/Chapter6/JS/myScript.js" async></script>
```

Note: Both **defer** and **async** attribute is only for external scripts and is only used if the **src** attribute is present.

When to use which attributes: कब कौन सी एट्रिब्यूटों का उपयोग करें:

1. If your script is independent of other scripts and is modular, use **async**. If you are loading **script1** and **script2** with **async**, both will run parallelly along with HTML parsing, as soon as they are downloaded and available

यदि आपकी स्क्रिप्ट अन्य स्क्रिप्ट से स्वतंत्र है और मॉड्यूलर हैं, तो `async` का उपयोग करें। यदि आप `async` के साथ `script1` और `script2` को लोड कर रहे हैं, तो दोनों ही HTML पार्सिंग के साथ समानांतर रूप से चलेंगे, जैसे ही वे डाउनलोड और उपलब्ध होंगे।

2. If your **script** depends on **another script** then use **defer** for both. When **script1** and **script2** are loaded in that order with **defer**, then **script1** is guaranteed to execute first. Then **script2** will execute after **script1** is fully executed. Must do this if `script2` depends on `script1`.

यदि आपकी स्क्रिप्ट किसी अन्य स्क्रिप्ट पर निर्भर करती है तो दोनों के लिए `defer` का उपयोग करें। जब `script1` और `script2` को `defer` के साथ उस क्रम में लोड किया जाता है, तो `script1` को पहले क्रियान्वित करने की गारंटी है तो `script2` पूरी तरह से क्रियान्वित होने के बाद `script2` क्रियान्वित करेगा। यदि `script2` `script1` पर निर्भर करता है तो यह अवश्य करना चाहिए।

3. If your **script** is small enough and is depended by another script of type **async** then use your script with no attributes and place it above all the **async** scripts.

यदि आपकी स्क्रिप्ट काफी छोटी है और किसी अन्य स्क्रिप्ट में टाइप की जाती है, तो अपनी स्क्रिप्ट का उपयोग बिना किसी एट्रिब्यूट के करें और इसे सभी `async` स्क्रिप्ट से ऊपर रखें।

6.2 Variables in Java Script - जावा स्क्रिप्ट में वैरियेबल

Variables are used for storing data values. JavaScript variables are containers (placeholders) for storing values. A variable can have different values at different phases of execution. Since the content of a variable varies throughout the execution of a program, they are called variables. In JavaScript `var` keyword is used to declare/define variables. `var` keyword is optionally followed by equal (=) sign to assign values to variables.

डेटा वैल्यू को संग्रहीत करने के लिए वैरियेबल का उपयोग किया जाता है। वैल्यू को संग्रहीत करने के लिए जावास्क्रिप्ट वैरियेबल एक कंटेनर (प्लेसहोल्डर) हैं। एक वैरियेबल के क्रियान्वयन के विभिन्न चरणों में अलग-अलग वैल्यू हो सकते हैं। चूंकि किसी प्रोग्राम के क्रियान्वयन के दौरान एक वैरिएबल की सामग्री बदलती रहती है, उन्हें वैरिएबल्स कहा जाता है। जावास्क्रिप्ट में कीवर्ड को वैरिएबल घोषित/परिभाषित करने के लिए उपयोग किया जाता है। `var` कीवर्ड वैकल्पिक रूप से इक्वैल (=) साइन के साथ वैरिएबल में वैल्यू असाइन करने के लिए होता है।

In below example, variable `var1` is defined with initial value of **100** whereas variable `var2` is defined without any initial value. Take a look at below program.

नीचे दिए गए उदाहरण में, वैरियेबल `var1` को 100 के प्रारंभिक मूल्य के साथ परिभाषित किया गया है जबकि वैरियेबल `var2` को बिना किसी प्रारंभिक मूल्य के परिभाषित किया गया है। नीचे प्रोग्राम पर एक नज़र डालें।

```

1 <!DOCTYPE html>
2 <html>
3 <body>
4 <h1>Variables in Java Script</h1>
5 <p>In this example, 'var1' is defined as a variable and assign value 100 but the 'var2' is
6   defined as variable which is not assigning any value.</p>
7 <p id="id1"></p>
8 <p id="id2"></p>
9 <script>
10 var var1, var2;
11 var1 = 100;
12   document.getElementById("id1").innerHTML = var1;
13   document.getElementById("id2").innerHTML = var2;
14 </script>
15 </body>
16 </html>

```

Generate preview file

Output of the code
is here

Take another example, a, b, and sum, are variables - एक और उदाहरण लें, a, b, और sum, वैरियेबल हैं-

Code	PREVIEW
<pre> 1 <!DOCTYPE html> 2 <html> 3 <body> 4 <h1>sum of two variables</h1> 5 <p> a, b and sum are variables </p> 6 7 var a = 25;
 8 var b = 75;
 9 var sum = a + b;

 10 Sum : 11 12 <script> 13 var a = 25; 14 var b = 75; 15 var sum = a + b; 16 document.getElementById("sumID").innerHTML = sum; 17 </script> 18 </body> 19 </html> </pre>	<pre> sum of two variables a, b and sum are variables var a = 25; var b = 75; var sum = a + b; Sum : 100 </pre>

As per above example, 'a' stores the value 25, 'b' stores the value 75 and sum stores the value 100.
ऊपर दिए गए उदाहरण के अनुसार, 'a' में 25 स्टोर है, 'b' में 75 तथा sum में 100 स्टोर है।

Declaring JavaScript Variables - जावास्क्रिप्ट वैरिएबल की घोषणा

JavaScript variable can be declared with the **var** keyword. Creating a variable in any programming language is called "declaring" a variable.

जावास्क्रिप्ट वैरियेबल को **var** कीवर्ड द्वारा डिक्लेयर किया जा सकता है। किसी भी प्रोग्रामिंग लैंग्वेज में एक वैरिएबल बनाना एक वैरिएबल को "डिक्लेयर करना" कहा जाता है।

Syntax: var Name-of-variable;

Example:

```

var a;
var name;
var address;
var salary;

```

After declaration, the variable has no value. It has default value of undefined. You can use the equal sign to assign a value to the variable.

डिक्लेरेशन के बाद, वैरिएबल का कोई मूल्य नहीं होता है। इसमें डिफॉल्ट वैल्यू अपरिभाषित है। वैरिएबल के लिए एक मान निर्दिष्ट करने के लिए आप बराबर चिन्ह का उपयोग कर सकते हैं।

```
address = "prayagraj, UP";
```

It is also possible to assign value to a variable during declaration.

डिक्लेरेशन के दौरान भी वैरियेबल में वैल्यू को एसाइंग किया जा सकता है जैसा-

```
var defaultAdd = "GT Road, Prayagraj";
```

Declaring Multiple JavaScript Variables in single Statement - सिंगल स्टेटमेंट में मल्टीपल जावास्क्रिप्ट वैरियेबल का डिक्लेरेशन-

It is possible to declare many variables using single **var** keywords. Start with **var** keyword and then separate each variable with comma. In below example 3 variables are declared in single statement.

सिंगल **var** कीवर्ड का उपयोग करके कई वैरियेबल डिक्लेयर करना संभव है। **var** कीवर्ड से शुरू करें और फिर प्रत्येक वैरिएबल को कामा (अल्पविराम) से अलग करें। नीचे दिए गए उदाहरण में 3 वैरिएबल एक ही स्टेटमेंट में डिक्लेयर किए गए हैं।

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Variables</h1>
<p>You can declare many variables in one statement.</p>
var Name = "V K Misra", location = "Lodhi Road, Prayagraj", Sal = 20000;
<br />
value of Name of the location variable is : <span id="id2"></span>
<script>
var name = "V K Misra", locID = "Lodhi Road, Prayagraj", sal = 2000;
document.getElementById("id2").innerHTML = locID;
</script>
</body>
</html>
```

Output:

PREVIEW

JavaScript Variables

You can declare many variables in one statement.

```
var Name = "V K Misra", location = "Lodhi Road, Prayagraj", Sal = 20000;
value of Name of the location variable is : Lodhi Road, Prayagraj
```

Undefined Value - अपरिभाषित मूल्य

It is possible to declare variable without value. Variable declared without any value will have undefined value. Remember when a variable is declared without a value, it will have the value undefined. Take a look at below example.

वैल्यू के बिना वैरिएबल डिक्लेयर करना संभव है। बिना किसी वैल्यू के डिक्लेयर वैरिएबल का अनडिफाइंड वैल्यू (अपरिभाषित मूल्य) होगा। याद रखें जब एक वैरियेबल को बिना किसी वैल्यू के डिक्लेयर किया जाता है, तो इसका वैल्यू अनडिफाइंड (अपरिभाषित मूल्य) होगा। नीचे दिए गए उदाहरण पर एक नज़र डालें।