ARUN COMPUTER & IT

PDF Study Material

इंटरनेट एंड वेब डिजाइन

Unit- 3

विषय सूची:

- Cascading Style Sheet (CSS)
- JAVA SCRIPT
- Operators, Objects,
- Control Statements, Functions, Array
- Document Object Model(DOM)
- Events, Dialog Box

हमारी वेबसाइट से डाउनलोड करें

Latest version available only at: www.hamararewa.in

Internet And Web Designing – Unit-III

Cascading Style Sheet (CSS)

CSS का पूरा नाम **Cascading Style Sheet** है। Web Page बनाने में HTML और CSS दोनों की महत्वपूर्ण भूमिका होती है। HTML वेब पेज की संरचना (Structure) तैयार करता है, जबकि CSS वेब पेज को आकर्षक रूप और डिजाइन प्रदान करता है। सामान्यतः HTML और CSS का उपयोग साथ-साथ किया जाता है।

CSS एक सरल Design Language है, जिसे Web Pages की प्रस्तुति (Presentation) को आसान बनाने के लिए विकसित किया गया है। इसके माध्यम से Web Page के Layout, Background, Text Color, Font Style, Spacing आदि को नियंत्रित किया जा सकता है।

CSS की सहायता से Web Pages को अधिक आकर्षक, व्यवस्थित और Professional बनाया जा सकता है।

CSS का इतिहास

CSS का विकास 10 अक्टूबर 1994 को Håkon Wium Lie द्वारा किया गया था। CSS को Maintain और Update करने के लिए World Wide Web Consortium के अंतर्गत **CSS Working Group** बनाया गया, जो CSS के Standards और Specifications तैयार करता है।

CSS के प्रकार

CSS को मुख्य रूप से तीन प्रकार से उपयोग किया जाता है —

1. Inline Style Sheet
2. Internal (Embedded) Style Sheet
3. External Style Sheet

1. Inline Style Sheet

Inline CSS का उपयोग किसी विशेष HTML Element पर Style लागू करने के लिए किया जाता है। इसमें style Attribute सीधे HTML Tag के अंदर लिखा जाता है।

इसका प्रभाव केवल उसी Element पर पड़ता है।

उदाहरण

```
<h1 style="color:blue;">A Blue Heading</h1>
<p style="color:red;">A red paragraph.</p>
```

2. Internal (Embedded) Style Sheet

Internal CSS को HTML Document के <head> Section के अंदर <style> Tag में लिखा जाता है।

इस प्रकार की Style Sheet का प्रभाव पूरे Web Page पर पड़ता है।

उदाहरण

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-color: powderblue;
    }
h1 {
    color: blue;
}
p {
    color: red;
}
</style>
</head>
<body>
        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>
</body>
</html>
```

3. External Style Sheet

External CSS में CSS Code को अलग .css File में लिखा जाता है। बाद में उस File को HTML Document से <link> Tag द्वारा जोड़ा जाता है।

इसका उपयोग विशेष रूप से बड़ी Websites में किया जाता है, जहाँ एक ही CSS File से कई Web Pages का Design नियंत्रित किया जा सकता है।

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

```
        <link rel="stylesheet" href="styles.css">
    </head>
    <body>
            <h1>This is a heading</h1>
            <p>This is a paragraph.</p>
    </body>
</html>

CSS File (styles.css)
body {
       background-color: powderblue;
      }
h1 {
    color: blue;
   }
p {
    color: red;
   }
```

CSS के फायदे (Benefits of CSS)

1. समय की बचत

CSS में एक बार Style लिखने के बाद उसे कई HTML Elements तथा अनेक Web Pages पर पुनः उपयोग किया जा सकता है। इससे समय की बचत होती है।

2. Page Load Time कम करता है

CSS के उपयोग से Web Page में कम Code लिखना पड़ता है, जिससे Page तेजी से Load होता है।

3. Maintain करना आसान

CSS की सहायता से किसी Website के कई Pages का Design एक ही स्थान से नियंत्रित और Update किया जा सकता है।

4. Bandwidth की बचत

Table आधारित Layout की तुलना में CSS आधारित Design हल्का होता है, जिससे File Size कम होती है और Bandwidth की बचत होती है।

5. Multiple Devices के लिए उपयुक्त

CSS की सहायता से एक ही Web Page को विभिन्न Devices जैसे —

- Computer
- Mobile Phone
- Tablet
- Printer

आदि के अनुसार अनुकूल (Responsive) बनाया जा सकता है।

6. Search Engine Ranking में सुधार

CSS के उपयोग से Website का Structure अधिक व्यवस्थित हो जाता है, जिससे Search Engines के लिए Website को Crawl और Index करना आसान हो जाता है। इससे Search Engine Ranking बेहतर हो सकती है।

Tag

 Tag का उपयोग HTML और CSS में Web Page को अलग-अलग भागों (Sections) में विभाजित करने के लिए किया जाता है। यह एक Block-level Element होता है, जो उपलब्ध पूरी चौड़ाई (Width) घेरता है।

 Tag का उपयोग Web Page की संरचना (Structure) और Layout तैयार करने में किया जाता है। इसके अंदर CSS की सहायता से Color, Border, Margin, Padding, Width, Height आदि निर्धारित किए जा सकते हैं।

Web Designing में Header, Footer, Sidebar तथा Content Section बनाने के लिए Tag का व्यापक उपयोग होता है। यह स्वयं कोई विशेष Design प्रदर्शित नहीं करता, लेकिन CSS और JavaScript के साथ मिलकर आकर्षक तथा Responsive Web Pages बनाने में सहायता करता है।

JavaScript की सहायता से Element में Runtime पर परिवर्तन (Modification) भी किया जा सकता है। एक Web Page में अनेक Tags का उपयोग किया जा सकता है।

CSS Syntax

CSS Selectors

CSS Selectors का उपयोग HTML Elements को Select करने तथा उन पर Style लागू करने के लिए किया जाता है।

डिक्लेरेशन ब्लॉक (Declaration Block) में एक या अधिक Declarations हो सकते हैं, जिन्हें Semicolon (;) द्वारा अलग किया जाता है।

प्रत्येक Declaration में CSS Property तथा उसकी Value होती है, जिन्हें Colon (:) द्वारा अलग किया जाता है।

पूरा Declaration Block Curly Braces { } के अंदर लिखा जाता है।

In this example all <p> elements will be center-aligned, with a red text color:

```
p {
  color: red;
 text-align: center;
}
```

CSS Selectors के प्रकार

CSS Selectors को मुख्य रूप से पाँच श्रेणियों में बाँटा जाता है —

1. Simple Selectors
2. Combinator Selectors
3. Pseudo-class Selectors
4. Pseudo-element Selectors
5. Attribute Selectors

1. Simple Selectors

यह Element के Name, ID, Class आदि के आधार पर Element को Select करता है।

उदाहरण

निम्न उदाहरण में Page के सभी <p> Elements Center-Aligned होंगे तथा उनका Text Color लाल होगा।

```
p {
  text-align: center;
  color: red;
}
```

2. Combinator Selectors

यह Elements के बीच विशेष संबंध (Relationship) के आधार पर Elements को Select करता है।

3. Pseudo-class Selectors

यह किसी Element की विशेष अवस्था (State) के आधार पर Element को Select करता है।

उदाहरण: :hover, :focus आदि।

4. Pseudo-element Selectors

यह किसी Element के विशेष भाग (Part) को Select तथा Style करता है।

उदाहरण: ::first-letter, ::before आदि।

5. Attribute Selectors

यह Attribute अथवा Attribute Value के आधार पर Elements को Select करता है।

CSS Element Selector

Element Selector किसी विशेष HTML Element को उसके Tag Name के आधार पर Select करता है, जैसे —

```
<p>
<h1>
<div>
```

जब किसी Element को उसके नाम से Select किया जाता है, तब निर्धारित Style उस प्रकार के सभी Elements पर लागू हो जाती है।

CSS ID Selector

ID Selector किसी विशेष Element को Select करने के लिए उसके id Attribute का उपयोग करता है।

किसी Web Page में id Unique होती है, इसलिए इसका उपयोग केवल एक विशेष Element को Select करने के लिए किया जाता है।

ID Selector में Hash (#) चिन्ह का उपयोग किया जाता है।

Example

The CSS rule below will be applied to the HTML element with id="para1":

```
#para1 {
  text-align: center;
  color: red;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
text-align: center;
color: red;
}
</style>
</head>
<body>
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
</body>
</html>
```

नोट: id का नाम संख्या (Number) से प्रारंभ नहीं होना चाहिए।

CSS Class Selector

Class Selector किसी विशेष class Attribute वाले HTML Elements को Select करता है।

Class Selector में Period (.) चिन्ह का उपयोग किया जाता है।

```
.center {
 text-align: center;
color: red;
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
text-align: center;
```

```
            color: red;
            }
            </style>
        </head>
        <body>
    <h1 class="center">Red and center-aligned heading</h1>
    <p class="center">Red and center-aligned paragraph.</p>
      </body>
</html>
```

CSS Universal Selector

Universal Selector (*) Web Page के सभी HTML Elements को Select करता है।

Example

The CSS rule below will affect every HTML element on the page:

```
* {
 text-align: center;
 color: blue;
 }
<!DOCTYPE html>
<html>
<head>
<style>
     * {
    text-align: center;
    color: blue;
     }
</style>
</head>
<body>
      <h1>Hello world!</h1>
      <p>Every element on the page will be affected by the style.</p>
      <p id="para1">Me too!</p>
      <p>And me!</p>
      </body>
</html>
```

CSS Comments

Browser CSS Comments को Ignore कर देता है।

CSS Comment /* से प्रारंभ होता है तथा */ पर समाप्त होता है। A CSS comment is placed inside the <style> element

```
/* This is a single-line comment */
p {
  color: red;
}
```

CSS Background Color

CSS की सहायता से HTML Elements का Background Color निर्धारित किया जा सकता है।

Example-

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

```
<!DOCTYPE html>
<html>
    <body>
        <h1 style="background-color:DodgerBlue;">Hello World</h1>
        <p style="background-color:Tomato;">
            Lorem ipsum dolor sit ametV, consectetuer adipiscing elit, sed diam nonummy nibh euismod  tincidunt ut laoreet dolore magna aliquam erat volutpat.  enim ad minim veniam, quis
        </p>
  </body>
</html>
```

CSS Text Color

CSS की सहायता से Text का Color निर्धारित किया जा सकता है।

Example

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

CSS Border Color

CSS द्वारा Border का Color निर्धारित किया जा सकता है।

Example

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

Example 1 – with CSS

```
<html>
  <head>
   <style>
    .myDiv {
        border: 5px outset red;
        background-color: lightblue;
        text-align: center;
        }
   </style>
  </head>
  <body>
    <div class="myDiv">
      <h2>This is a heading in a div element</h2>
      <p>This is some text in a div element.</p>
    </div>
  </body>
</html>
```

Example 2 – Square and Circle Div

```
<html>
  <head>
    <title>Style Example</title>
    <style>
```

```
.square {
  background-color: #2ecc71;
  width: 200px;
  height: 200px;
   }
#circle {
  background-color: #2ecc71;
  width: 200px;
  height: 200px;
  border-radius: 50%;
   }
</style>
</head>
<body>
 This is div example <br><br><br>
<div id="circle">
  This is first division
</div>
<br>
<div class="square">
  This is second division
</div>
</body>
</html>
```

Example 3 – Styled Div

```
<html>
  <head>
  <title>Style Example</title>
  <style>
     .square {
          background-color: #2ecc71;
          width: 200px;
          height: 200px;
          font-family: cursive, sans-serif;
          font-size: 1.3rem;
          font-weight: bold;
          font-style: italic;
          }
```

```
        #circle {
                background-color: #2ecc71;
                width: 200px;
                height: 200px;
                border-radius: 50%;
                }
    </style>
    </head>
    <body>
            This is div example <br><br><br>
        <div id="circle">
            This is first division
        </div>
         <br>
        <div class="square">
            This is second division
        </div>
     </body>
</html>
```

JAVA SCRIPT

JavaScript एक लोकप्रिय Scripting तथा Programming Language है, जिसके उपयोग से Web Pages को Dynamic (गतिशील) और Interactive बनाया जाता है।

यह एक **Interpreted Language** है, अर्थात इसका Code सीधे Browser द्वारा Run किया जाता है। JavaScript को सामान्यतः **Client-side Scripting Language** कहा जाता है, क्योंकि यह User के Browser पर Execute होती है।

JavaScript का विकास 1995 में Brendan Eich ने Netscape के लिए किया था। प्रारंभ में इसका नाम **LiveScript** था, बाद में इसका नाम JavaScript रखा गया।

JavaScript की विशेषताएँ

- Machine Independent Language
- Lightweight Language
- Interpreted Language

- Object-Oriented Language
- Case-Sensitive Language
- Cross-Platform Support
- Dynamic एवं Interactive Web Pages बनाने में उपयोगी
- Server से Communication करने में सक्षम

JavaScript का File Extension .js होता है।

Java और JavaScript दोनों अलग-अलग Programming Languages हैं।

- JavaScript एक Scripting Language होती है. इसलिए JavaScript कोड को HTML पेज के साथ ही Code किया जाता है.

  जावास्क्रिप्ट Web Designers को प्रोग्रामिंग की सुविधा प्रदान करती है।
- जब कोई यूजर इंटरनेट पर किसी ब्राउज़र में वेबपेज के लिए Request भेजता है तो कंप्यूटर सर्वर उस पेज के HTML Code के साथ-साथ जावास्क्रिप्ट कोड को भी अटैच कर वापस भेज देता है. उसके बाद वह ब्राउज़र आवश्यकता पड़ने पर कोड को Text के रूप में परिवर्तित कर हमें शो करता है.
- JavaScript एक ओपन तथा क्रॉस प्लेटफार्म है अर्थात इसका इस्तेमाल Windows, Mac ,आदि अनेक ऑपरेटिंग सिस्टम में किया जा सकता है.

Web Development में JavaScript का महत्व

प्रत्येक Web Developer को निम्न तीन भाषाओं का ज्ञान होना आवश्यक है –

1. HTML – Web Page की Structure बनाने के लिए
2. CSS – Web Page की Design एवं Layout के लिए
3. JavaScript – Web Page को Dynamic एवं Interactive बनाने के लिए

JavaScript का उपयोग

JavaScript की सहायता से Web Page में विभिन्न प्रकार के कार्य किए जा सकते हैं –

- Popup Window बनाना
- Animation Effect जोड़ना
- Image Slider बनाना
- Dropdown Menu बनाना

- Auto Complete Feature जोड़ना
- Browser Detection करना
- Cookies Handle करना
- Form Validation करना

HTML Forms में User द्वारा भरे गए Data को JavaScript की सहायता से Validate किया जा सकता है।

JavaScript और Events

Web Page में विभिन्न प्रकार के Events होते हैं, जैसे —

- Mouse Click
- Key Press
- Mouse Hover
- Page Load

JavaScript द्वारा यह निर्धारित किया जा सकता है कि किसी Event पर क्या प्रतिक्रिया (Response) होगी। स्क्रिप्टिंग में कोड को कम्पाइल करने की जरुरत नहीं होती यह रनटाइम में इंटरप्रेट होता है।

JavaScript के लाभ

- Web Page को Interactive बनाता है।
- Client-side Execution के कारण तेज कार्य करता है।
- Server पर Load कम करता है।
- Website की Speed बढ़ाता है।
- इसके जरिये हम Popup window, animation effect , image transition, image slider, dropdown menu, auto complete, browser detection, cookies जैसे विशेष फीचर जोड़ सकते है।
- HTML फार्म में यूजर द्वारा डाले गए इनपुट जावास्क्रिप्ट से वैलिडेट किये जा सकते है।

- क्लाइंट साइड स्क्रिप्टिंग होने के कारण ये जल्दी एक्सीक्यूट हो जाते है और सर्वर पर भी लोड नहीं पड़ता, इससे वेबसाइड की स्पीड बढ़ जाती है

JavaScript की सीमा

JavaScript का Code Web Page में उपस्थित होता है, इसलिए इसे Browser में देखा जा सकता है। इसी कारण यह पूरी तरह Secure नहीं माना जाता।

HTML में JavaScript जोड़ना

HTML Document में JavaScript जोड़ने के लिए <script> Tag का उपयोग किया जाता है। JavaScript Code को दो प्रकार से उपयोग किया जा सकता है —

1. Internal JavaScript
2. External JavaScript

Internal JavaScript

जब JavaScript Code सीधे HTML Page के अंदर लिखा जाता है, तब उसे Internal JavaScript कहते हैं।

<script> Tag को <head> या <body> Section में लिखा जा सकता है।

उदाहरण

```
<script>
    document.write("Hello World");
</script>
```

External JavaScript

जब JavaScript Code को अलग .js File में Save किया जाता है तथा HTML Document से जोड़ा जाता है, तब उसे External JavaScript कहते हैं।

उदाहरण

```
<script src="script.js"></script>
```

- Javascript का code implement करने के लिए script tag में साधारणतः 'type' नाम का attribute इस्तेमाल किया जाता है | यहाँ पर दिया हुआ content कौन से type का है वो दिया जाता है |
- Javascript के लिए सिर्फ <script> का भी इस्तेमाल किया जा सकता है

JavaScript में Comments

Single-line Comment

Single-line Comment के लिए // का उपयोग किया जाता है।

```
// This is a single-line comment
```

Multi-line Comment

Multi-line Comment के लिए /* */ का उपयोग किया जाता है।

```
/* This is a
multi-line comment */
```

Ex .2-

```
<!DOCTYPE html>
<html>
<head>
        <title> java script example 1 </title>
</head>
<body>
        <!-- this is html comment -->
        <script>
        // this is javascript this is single line comment
          document.write ("Hello World");
         window.alert("hello arun computer");
        </script>
</body>
</html>
```

Ex.3

```
<!DOCTYPE html>
<html>
<head>
        <title> java script  example 2</title>
</head>
<body>
        <script>
                document.write("1" + 2 + 3 + "<br />");
                document.write(2 + 3 + "3" + 4 + "<br />");
                document.write(2 + 3 + "3" + "<br />");
        </script>
</body>
</html>
```

Case-Sensitive Language

JavaScript एक Case-Sensitive Language है।

अर्थात a और A दोनों अलग-अलग Variables माने जाते हैं।

उदाहरण:

```
var age = 20;
var Age = 25;
```

Semicolon (;) का उपयोग

जब Statements अलग-अलग Lines में लिखे जाते हैं, तब Semicolon आवश्यक नहीं होता।

उदाहरण

```
<script>
    var a = 5
   document.write(a)
</script>
```

यदि एक ही Line में कई Statements लिखी जाएँ, तो उन्हें Semicolon (;) द्वारा अलग किया जाता है।

उदाहरण

```
<script>
    var a = 5;
    var b = 6;
    document.write(a);
    document.write(b);
</script>
```

Ex.-
```
<script>
    var a = 5;      b = 6;
    document.write(a); document.write(b)
    var c = 7, d = 8;
    document.write(c), document.write(d)
</script>
```

Ex Question -

1- जावा स्क्रिप्ट क्या है जावा स्क्रिप्ट के लूपिंग स्ट्रक्चर क्या है -DCA II,Dec 2019

2- जावा स्क्रिप्ट में Fibonacci series की गणना कैसे करेंगे -DCA II,Dec 2019

3-जावा स्क्रिप्ट की प्रॉपर्टीज को कैसे पढ़ सकते है? उदाहरण सहित समझाइयए. -DCA II,Dec 2019

4 -ऐट्रीब्यूट्स Attributes और प्रॉपर्टीज Properties में क्या अंतर है? -DCA II,Dec 2019

5- java script के object, Events तथा Document Object Model को उचित उदहारण द्वारा समझाइये -pgdca II june2019

6 - निम्नलिखित को समझाइये OnLoad , OnUnload, OnMouseOver, OnClick -pgdca II june2019

7- जावा स्क्रिप्ट का कोड लिखे जो यह ज्ञात करे के दिया गया नंबर प्राइम है या नहीं। DCA II, June 2019

8- जावा क्रिप्ट में कंडीशनल स्टेटमेंट को उदाहरण सहित समझाइये -DCA II, June 2019

9- जावा स्क्रिप्ट को उसकी विषेशताओं के साथ बताइये -DCA II, June 2019

10- जावा स्क्रिप्ट में इवेंट का क्या आशय है किन्ही दो इवेंट को उदाहरण के साथ समझाइये -DCA II, June 2019

Data Types in JavaScript

JavaScript में मुख्यतः निम्न Data Types होते हैं —

1. Number
2. Boolean
3. String
4. Array
5. Undefined
6. Null

1. Number

Number Data Type में Integer तथा Floating Point Numbers आते हैं।

उदाहरण:

```
var a = 125;
var b = 125.52;
```

2. Boolean

Boolean Data Type के केवल दो मान होते हैं —

- true
- false

```
var result = true;
```

3. String

Single (' ') या Double (" ") Quotes में लिखे गए Characters को String कहते हैं।

```
var name = "Ashok";
```

4. Array

Array का उपयोग Multiple Values को Store करने के लिए किया जाता है।

```
var cars = ["Saab", "Volvo", "BMW"];
```

5. Undefined

जब किसी Variable को Value Assign नहीं की जाती, तब उसका मान Undefined होता है।

```
var a;
document.write(typeof(a));
```

6. Null

Null का अर्थ है कि Variable में कोई Value नहीं है।

var a = null;

Variable -

मेमोरी में एक निश्चित जगह (contaners for storing data) जिसमें हम अपनी सुविधानुसार बैल्यू स्टोर कर सकते है, जिसे प्रोग्राम execution के समय कभी भी बदला जा सकता है.

- Variables को declare करने के लिए 'var' keyword का इस्तेमाल किया जाता है |

उदाहरण

```
var a = 10;
```

- एक ही 'var' keyword के साथ multiple variables ,(comma) से declare या initialize किये जा सकते है |

```
var a = 5, b = 10;
```

- डिक्लेयरेशन के साथ वैल्यू भी एसाइन कर सकते है। वैल्यू एसाइन करना वेरिएबल को इनिशियलाइज करना कहते है
- किसी भी वैरियेबल को उपयोग करने के पहले डिक्लेयर करना होगा
- वैरियेबल डिक्लेयरेशन में सेव किये जाने वाले डाटा की टाइप बताना आवश्यक नहीं है इसलिये इन्हें loosely type कहते है।

Variables Naming Rules

- Variables के नाम की शुरुआत किसी भी letter(A-Z a-z) से, underscore(_) से या dollar($) sign से की जाती है |
- Variables की शुरुआत Numeric से नहीं हो सकती | लेकिन Variable alphanumeric हो सकता है | For eg. A1 = 5
- Variables Javascript का कोई keyword नहीं हो सकता |
- Variables case-sensitive होते है | 'A' और 'a' ये दोनों अलग-अलग variables है |
- Creating a variable in javascript is called declaring a variable.
- Name of variable is called identifires.
- सभी Variable Names Unique होने चाहिए।

Variable Ex.1-

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Variables</h2>
<p id="demo"></p>
<script>
var price1 = 5;
var price2 = 6;
var total = price1 + price2;
document.getElementById("demo").innerHTML =
"The total is: " + total;
var a = 5;
document.write("Value of a is " + a);
</script>
</body>
</html>
```

Variable Ex.2-

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Variables</h2>
<p id="demo"></p>
<script>
var a; a = 5; var b = 10;
document.write("Value of a is " + a + " ");
document.write("Value of b is " + b + " ");
</script>
</body>
</html>
```

Operators (ऑपरेटर्स)

Operators ऐसे symbols होते हैं जिनका उपयोग values को assign करने, दो values की तुलना करने तथा विभिन्न mathematical और logical operations करने के लिए किया जाता है। JavaScript में operators की सहायता से calculations और logical कार्य किए जाते हैं।

The Assignment Operator

JavaScript में equal sign (=) को Assignment Operator कहा जाता है। इसका उपयोग किसी variable में value assign करने के लिए किया जाता है।

यह गणित (Algebra) के बराबर (=) चिन्ह से अलग होता है।

उदाहरण:

```
x = x + 5;
```

यह statement Algebra में सही नहीं माना जाता, लेकिन JavaScript में इसका अर्थ है कि variable x की पुरानी value में 5 जोड़कर नई value फिर से x में store की जा रही है।

JavaScript में "equal to" की तुलना करने के लिए == operator का उपयोग किया जाता है।

Overwrite Variable Value

JavaScript में किसी variable की value को overwrite किया जा सकता है। अर्थात यदि किसी variable को नई value दी जाती है, तो पुरानी value हटकर नई value store हो जाती है।

Example

```
<script>
    var a = 5;
    document.write("Value of a is " + a + "<br>");
    a = 10;
    document.write("Value of a is " + a);
</script>
```

Output

Value of a is 5

Value of a is 10

Types of Operators in JavaScript

1. Arithmetic Operators
2. Assignment Operators

1. Arithmetic Operators

Arithmetic Operators का उपयोग mathematical calculations करने के लिए किया जाता है।

Operator	कार्य
+	Addition (जोड़)
–	Subtraction (घटाव)
*	Multiplication (गुणा)
/	Division (भाग)
%	Modulus (शेषफल निकालना)
++	Increment Operator (value को 1 बढ़ाना)
—	Decrement Operator (value को 1 घटाना)

Important Note

+ Operator का उपयोग Addition तथा Concatenation (strings को जोड़ने) दोनों कार्यों के लिए किया जाता है।

उदाहरण:

```
document.write("Arun " + "Computer");
```

Prefix Increment

++a में पहले variable की value 1 बढ़ती है, फिर उसे assign किया जाता है।

Example

```
<script>
    var a = 6;
    var b = ++a;   // Prefix Increment
    document.write("Value of b is " + b + "<br>");
    document.write("Value of a is " + a);
</script>
```

Output

Value of b is 7

Value of a is 7

Postfix Increment

a++ में पहले पुरानी value assign होती है, फिर value 1 बढ़ती है।

Example

```
<script>
    var a = 6;
    var b = a++;   // Postfix Increment
    document.write("Value of b is " + b + "<br>");
    document.write("Value of a is " + a);
</script>
```

Output

Value of b is 6

Value of a is 7

Arithmetic Operator Example

```
<script>
  // Multiply
    var a = 6, b = 3, c;
    c = a * b;
    document.write("Multiplication = " + c + "<br>");
  // Divide
    c = a / b;
    document.write("Division = " + c);
</script>
```

2. Assignment Operators

Assignment Operators का उपयोग variables में values assign करने के लिए किया जाता है।

Operator	कार्य
=	Value assign करता है
+=	x += y का अर्थ x = x + y
-=	x -= y का अर्थ x = x – y

Operator	कार्य
*=	x *= y का अर्थ x = x * y
/=	x /= y का अर्थ x = x / y
%=	x %= y का अर्थ x = x % y

Example of Assignment Operator

```
<!DOCTYPE html>
<html>
<body>
<h2>The -= Operator</h2>
<p id="demo"></p>
<script>
var x = 10;
x -= 5;
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

Objects

Object एक विशेष प्रकार का variable होता है, जिसमें एक से अधिक values को store किया जा सकता है।

उदाहरण के लिए, यदि car नाम का object बनाया जाए, तो उसमें car का color, model, number तथा methods आदि store किए जा सकते हैं।

JavaScript में Objects को curly braces { } के अंदर लिखा जाता है।

Object की properties name : value pairs के रूप में लिखी जाती हैं तथा इन्हें comma (,) द्वारा अलग किया जाता है।

Object की properties को access करने के लिए dot (.) operator का उपयोग किया जाता है।

Object Example

```
var person = {
    firstName : "John",
```

```
    lastName : "Doe",
    age : 50,
    eyeColor : "blue"
};
```

Complete Object Program

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p id="demo"></p>
<script>
var person = {
firstName : "John",
lastName : "Doe",
age : 50,
eyeColor : "blue"
};
document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>
</body>
</html>
```

Important Points

Object properties को dot (.) operator द्वारा access किया जाता है।

Operators का उपयोग calculations तथा comparisons के लिए किया जाता है।

= Assignment Operator है, जबकि == comparison के लिए उपयोग होता है।

JavaScript में variable values overwrite की जा सकती हैं।

Arithmetic Operators mathematical operations करते हैं।

Assignment Operators variables की values update करने के लिए उपयोग होते हैं।

Object में multiple properties और values store की जा सकती हैं।

Control Statements

प्रोग्रामिंग भाषा में Program के execution flow को नियंत्रित करने के लिए जिन statements या structures का उपयोग किया जाता है, उन्हें Control Statements कहा जाता है। Control Statements की सहायता से program में निर्णय लेना (decision making), conditions की जाँच करना तथा statements को बार-बार execute करना संभव होता है।

Control Statements मुख्य रूप से दो प्रकार के होते हैं –

1. Conditional Statements
2. Loop Statements

---

1. Conditional Statements

Conditional Statements का उपयोग किसी condition के आधार पर निर्णय लेने के लिए किया जाता है।

If Statement

If Statement में जब condition true होती है, तब if block के अंदर लिखे गए statements execute होते हैं।
यदि condition false होती है, तो statements execute नहीं होते।

Syntax

```
if(condition) {
  // statements
}
```

Example – If Statement

```
<!DOCTYPE html>
<html>
  <body>
    <script>
      var a = 10;
      var b = 15;
```

```
 if(a < b) {
 document.write("a is less than b");
 }
 </script>
 </body>
</html>
```

Output

a is less than b

2. Loop Statements

Loop का उपयोग किसी statement या code block को बार-बार execute करने के लिए किया जाता है।

For Loop

For Loop का उपयोग किसी statement या code block को निश्चित संख्या तक बार-बार execute करने के लिए किया जाता है।
इसमें एक variable की value को प्रत्येक iteration में बढ़ाया या घटाया जाता है। जब दी गई condition false हो जाती है, तब loop समाप्त हो जाता है।

Syntax

```
for(statement1; statement2; statement3) {
  // code block to be executed
}
```

Statement 1

यहाँ variable को initialize किया जाता है।
Loop इसी initial value से प्रारंभ होता है।

उदाहरण:

```
var i = 0;
```

नोट: JavaScript में statement1 optional होता है।

Statement 2

यहाँ condition लिखी जाती है।

जब तक condition true रहती है, loop execute होता रहता है।

Condition false होने पर loop समाप्त हो जाता है।

Statement 3

इसमें variable की value को increase (i++) या decrease (i--) किया जाता है।

यह प्रत्येक iteration के बाद execute होता है।

Example – For Loop

```
<!DOCTYPE html>
    <html>
       <body>
          <script>
            for(var i = 0; i < 10; i++) {
            document.write("Value of i is " + i + "<br>");
            }
          </script>
        </body>
</html>
```

While Loop

While Loop का उपयोग किसी statement या code block को तब तक बार-बार execute करने के लिए किया जाता है, जब तक दी गई condition true रहती है।

जब condition false हो जाती है, तब loop समाप्त हो जाता है।

Syntax

```
while(condition) {
  // code block to be executed
}
```

यदि condition में उपयोग किए गए variable की value को increment या decrement करना भूल जाएँ, तो loop कभी समाप्त नहीं होगा। इसे Infinite Loop कहते हैं।

Infinite Loop के कारण browser या system hang/crash भी हो सकता है।

Example – While Loop

```
<!DOCTYPE html>
<html>
  <body>
    <script>
      var i = 0;
      while(i < 10) {
       document.write("Table of 2 = " + i * 2 + "<br>");
       i++;
      }
    </script>
  </body>
</html>
```

Do While Loop

Do While Loop का उपयोग किसी statement या code block को बार-बार execute करने के लिए किया जाता है, जब तक दी गई condition true रहती है।

इस loop की विशेषता यह है कि यह condition की जाँच बाद में करता है। इसलिए यदि condition प्रारंभ में false हो, तब भी loop कम से कम एक बार execute अवश्य होता है।

Syntax

```
do {
  // statements
}
while(condition);
```

Example – Do While Loop

```
<!DOCTYPE html>
  <html>
    <body>
      <script>
        var i = 0;
        do {
        document.write("Table of 2 = " + i * 2 + "<br>");
        i++;
       }
        while(i < 10);
```

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

Continue and Break Statements

Continue तथा Break statements का उपयोग मुख्य रूप से loops के साथ किया जाता है। ये program के execution flow को नियंत्रित करने में सहायता करते हैं।

Continue Statement

Continue Statement का उपयोग loop की किसी विशेष iteration को skip करने के लिए किया जाता है।

जब program में continue execute होता है, तब उस iteration के बाद का code execute नहीं होता और loop अगली iteration पर चला जाता है।

Example – Continue Statement

```
<!DOCTYPE html>
<html>
  <body>
    <script>
      for(var i = 0; i < 10; i++) {
        if(i == 7) {
          document.write("Number " + i + " is skipped.<br>");
          continue;
        }
        document.write(i + "<br>");
      }
    </script>
  </body>
</html>
```

उपरोक्त उदाहरण में i == 7 होने पर loop की वह iteration skip हो जाती है।

Break Statement

Break Statement का उपयोग loop अथवा switch case के execution को किसी condition पर तुरंत रोकने के लिए किया जाता है।

जब break execute होता है, तब control loop या switch block से बाहर आ जाता है।

Example – Break Statement

```
<!DOCTYPE html>
<html>
    <body>
        <script>
        for(var i = 0; i < 10; i++) {
          if(i == 5) {
            break;
          }
          document.write(i + "<br>");
        }
        </script>
    </body>
</html>
```

उपरोक्त उदाहरण में i == 5 होने पर loop तुरंत समाप्त हो जाता है।

Important Notes

Infinite Loop से browser hang हो सकता है।

Control Statements program के execution flow को नियंत्रित करते हैं।

if statement decision making के लिए उपयोग किया जाता है।

Loops का उपयोग repeated execution के लिए किया जाता है।

continue current iteration को skip करता है।

break loop को तुरंत समाप्त कर देता है।

Function

Function code का एक समूह (block of code) होता है, जिसे किसी विशेष कार्य (task) को पूरा करने के लिए बनाया जाता है।

Function का उपयोग करने से एक ही code को बार-बार लिखने की आवश्यकता नहीं होती। आवश्यकता पड़ने पर function को call किया जाता है।

Functions सामान्यतः कोई value return कर सकते हैं।

JavaScript में मुख्य रूप से दो प्रकार के Functions होते हैं –

Built-in Functions

1- User-defined Functions

2- Built-in Functions

जो functions JavaScript में पहले से उपलब्ध होते हैं, उन्हें Built-in Functions कहा जाता है।

उदाहरण–

eval(), parseInt(), parseFloat() आदि।

User-defined Functions

जो functions programmer अपनी आवश्यकता के अनुसार स्वयं बनाता है, उन्हें User-defined Functions कहा जाता है।

User-defined Function का उपयोग करने से पहले उसकी definition लिखी जाती है।

Function Defining

JavaScript में Function बनाने के लिए मुख्य रूप से तीन चीजों की आवश्यकता होती है –

function keyword

Function का नाम

Parentheses () के अंदर parameter list

Function Name

Function का नाम कोई JavaScript keyword नहीं होना चाहिए।

Function के नाम लिखने के नियम variable के नाम जैसे ही होते हैं। इसमें letters, digits, underscore _ तथा dollar sign $ का उपयोग किया जा सकता है।

Function का नाम उसके कार्य (task) से संबंधित होना चाहिए, ताकि program को समझना आसान हो।

Parameters List

Function बिना parameters/arguments के भी हो सकता है।

यदि आवश्यक हो, तो comma (,) लगाकर एक से अधिक parameters भी दिए जा सकते हैं।

Parameters को function name के बाद parentheses () के अंदर लिखा जाता है।

Syntax

```
function function_name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Statements

जो statements बार-बार उपयोग किए जाते हैं, उन्हें function के अंदर लिखा जाता है।

General Syntax

```
function function_name(parameters_list) {
  statement(s);
}
```

Function Calling

प्रोग्राम में किसी Function का उपयोग करने के लिए उसे call करना आवश्यक होता है।
जब किसी function को call किया जाता है, तब function की body में लिखे गए statements execute हो जाते हैं।

Function को निम्न प्रकार से call किया जा सकता है –

किसी event जैसे mouse click द्वारा

JavaScript code के द्वारा

Automatically (स्वतः) program execution के दौरान

Example

```
<script>
function arun() {   // Function Definition
  document.write("Hello Arun Computer");
```

```
}
arun();  // Function Calling
</script>
```

उपरोक्त उदाहरण में arun() एक user-defined function है।

जब arun() function को call किया जाता है, तब "Hello Arun Computer" output के रूप में प्रदर्शित होता है।

Function Example

```
<!DOCTYPE html>
  <html>
    <head>
      <script>
        function func() {  // Function Definition
        document.write("Hello Arun Computer");
        }
      </script>
    </head>
    <body>
        <form>
            <input type="button" onclick="func()" value="Click Me">
        </form>
    </body>
 </html>
```

Return Statement

return statement Function के लिए optional होता है।

यह सामान्यतः function के अंतिम भाग में लिखा जाता है।

जब JavaScript return statement पर पहुँचती है, तब function का execution बंद हो जाता है।

यदि return keyword के साथ कोई value नहीं दी जाती, तो function undefined return करता है।

Function द्वारा return की गई value, function को call करने वाले स्थान पर वापस भेज दी जाती है।

Example

```
<!DOCTYPE html>
  <html>
    <body>
      <h2>JavaScript Functions</h2>
```

```
        <p>
          This example calls a function which performs a calculation and returns the
result:
        </p>
        <script>
            var x = myFunction(4, 4);
            document.write(x);
            function myFunction(a, b) {
            return a * b;
            }
        </script>
      </body>
</html>
```

उपरोक्त उदाहरण में myFunction() दो numbers का multiplication करता है तथा उसका result return statement द्वारा वापस भेजता है।

Array

JavaScript में किसी variable में एक से अधिक values को store करने के लिए Array का उपयोग किया जाता है।

Array एक special variable होता है, जिसमें कई values को एक साथ रखा जा सकता है।

Example

```
<script>
    var arr = [1, 2, 3, 4];
    document.write(arr);
</script>
```

कई programming languages में Array के सभी elements समान data type के होते हैं, लेकिन JavaScript में Array के elements विभिन्न data types के भी हो सकते हैं।

Example

```
<script>
var arr = ["Maruti", "Fiat", "Ambassador", "Honda", 4, 5, 6];
document.write(arr);
</script>
```

Creating an Array

JavaScript में Array create करने के लिए square brackets [ ] का उपयोग किया जाता है, जिनके अंदर multiple values लिखी जाती हैं।

Syntax

```
var arr_name = [element1, element2, ..., elementN];
```

Accessing Array by Index

Array के प्रत्येक element का एक index number होता है।

पहले element का index 0 होता है।

अंतिम element का index array length - 1 होता है।

Array के elements को उनके index number की सहायता से access किया जाता है।

Example

```
<script>
    var names = ["Rakesh", "Mukesh", "Ramesh"];
    document.write(names[0] + " ");
    document.write(names[1] + " ");
    document.write(names[2] + " ");
</script>
```

उपरोक्त उदाहरण में names[0], names[1] तथा names[2] द्वारा Array के elements को access किया गया है।

Associative Array

कई programming languages ऐसे arrays को support करती हैं, जिनमें named indexes का उपयोग किया जाता है। ऐसे arrays को Associative Arrays या Hashes कहा जाता है।

JavaScript में वास्तविक रूप से Associative Arrays का support नहीं होता।

JavaScript Arrays हमेशा numbered indexes का उपयोग करते हैं, जैसे – 0, 1, 2 आदि।

Important Points

JavaScript Arrays में अलग-अलग data types के elements रखे जा सकते हैं।
JavaScript Arrays हमेशा numbered indexes का उपयोग करते हैं, जैसे 0, 1, 2 आदि।

Function code reuse के लिए उपयोग किया जाता है।

JavaScript में Functions Built-in तथा User-defined दोनों प्रकार के होते हैं।

return statement function से value वापस भेजता है।

Array में multiple values store की जा सकती हैं।

Document Object Model (DOM)

DOM (Document Object Model) के अनुसार HTML Document के सभी tags (elements) को objects माना जाता है।

जो tags किसी अन्य tag के अंदर उपस्थित होते हैं, वे उस tag के child objects कहलाते हैं।

Tag के अंदर उपस्थित text भी object माना जाता है।

JavaScript की सहायता से इन सभी objects को access तथा modify किया जा सकता है।

HTML DOM Model एक tree structure के रूप में कार्य करता है।

उदाहरण

document.body.style.background = "red";

उपरोक्त उदाहरण में document.body एक object है, जो <body> tag को represent करता है।

यह code webpage का background color लाल (red) कर देता है।

HTML DOM Properties and Methods

HTML DOM Properties

HTML DOM Properties, HTML elements की values होती हैं, जिन्हें JavaScript की सहायता से बदला जा सकता है।

HTML DOM Methods

HTML DOM Methods वे actions होते हैं, जिन्हें HTML elements पर perform किया जाता है, जैसे —

- Element जोड़ना (Add)
- Element हटाना (Delete)
- Element बदलना (Modify)

Example – DOM Method and Property

```
<!DOCTYPE html>
    <html>
        <body>
            <h2>My First Page</h2>
            <p id="demo"></p>
            <script>
            document.getElementById("demo").innerHTML = "Hello World!";
            </script>
        </body>
</html>
```

इस उदाहरण में:

- getElementById() एक method है, जो document में दी गई ID वाले element को return करता है।
- innerHTML एक property है, जो किसी HTML element के content को access या modify करने के लिए उपयोग की जाती है।

यहाँ <p> element जिसकी ID "demo" है, उसका text "Hello World!" में बदल दिया गया है।

Important Points

- document object पूरे webpage को represent करता है।
- innerHTML property किसी भी HTML element का content बदल सकती है।
- JavaScript की सहायता से webpage को बिना reload किए modify किया जा सकता है।
- DOM webpage को interactive और dynamic बनाता है।

Events (इवेंट्स)

JavaScript में Events वे क्रियाएँ (actions) होती हैं, जो user या browser द्वारा webpage पर की जाती हैं।

उदाहरण:

- Mouse click करना
- Keyboard key दबाना
- Mouse move करना
- Page load होना

जब कोई event होता है, तब JavaScript किसी विशेष function को execute कर सकती है।
Events की सहायता से webpage को अधिक interactive तथा dynamic बनाया जाता है।

Common HTML Events

Event	Description
onclick	जब user किसी HTML element पर click करता है

Event	Description
onmouseover	जब mouse pointer element के ऊपर जाता है
onmouseout	जब mouse pointer element के ऊपर से हटता है
onkeydown	जब keyboard की कोई key दबाई जाती है
onfocus	जब element focus में आता है
onchange	जब element की value बदलती है
onblur	जब element से focus हट जाता है
onload	जब webpage पूरी तरह load हो जाता है
onunload	जब webpage बंद या unload होता है
alert()	Alert dialog box प्रदर्शित करता है
prompt()	User से input लेने के लिए dialog box प्रदर्शित करता है
confirm()	OK और Cancel वाला confirmation box प्रदर्शित करता है

onfocus Event

जब किसी input element पर focus किया जाता है (कर्सर लाया जाता है या क्लिक किया जाता है), तब onfocus event होता है।

यह event सामान्यतः <input>, <select> तथा <a> elements के साथ उपयोग किया जाता है।

Example – onfocus Event

```
<!DOCTYPE html>
    <html>
      <head>
          <title>JavaScript onfocus Event</title>
      </head>
      <body style="text-align:center;">
          <h1>Arun Computer</h1>
          <h2 style="color:red;">Example of JavaScript onfocus Event</h2>
```

```
        <p>This example demonstrates the onfocus event.</p>
        Enter your name:
        <input type="text" id="fname" onfocus="myFunction()">
    <script>
        function myFunction() {
          document.getElementById("fname").style.backgroundColor = "red";
        }
    </script>
  </body>
</html>
```

onchange Event

onchange event तब होता है जब किसी element की value बदल जाती है।

यह event सामान्यतः:

- Textbox
- Radio Button
- Checkbox
- Select Box

आदि के साथ उपयोग किया जाता है।

Example – onchange Event

```
<!DOCTYPE html>
<html>
  <head>
      <title>JavaScript onchange Event</title>
  </head>
  <body style="text-align:center;">
        <h1>Arun Computer</h1>
        <h2 style="color:red;">Example of JavaScript onchange Event</h2>
        <p>This example demonstrates the onchange event.</p>
        Enter your name:
        <input type="text" id="fname" onchange="myFunction()">
        <p>
        When you leave the input field, the entered text will change to
        uppercase.
        </p>
    <script>
    function myFunction() {
      var x = document.getElementById("fname");
      x.value = x.value.toUpperCase();
```

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

onblur Event

जब user किसी input field से बाहर क्लिक करता है या focus हट जाता है, तब onblur event होता है।

यह onfocus event का विपरीत (opposite) होता है।

इसका उपयोग सामान्यतः form validation में किया जाता है।

Example – onblur Event

```
<!DOCTYPE html>
    <html>
      <head>
            <title>JavaScript onblur Event</title>
       </head>
      <body style="text-align:center;">
            <h1>Arun Computer</h1>
            <h2 style="color:red;">Example of JavaScript onblur Event</h2>
            <p>
            Write something in the input field and then click outside the field.
            </p>
            <input type="text" onblur="myFunction()">
      <script>
      function myFunction() {
        alert("Input field lost focus.");
      }
      </script>
    </body>
</html>
```

onload Event

जब webpage पूरी तरह load हो जाता है, तब onload event होता है।

Page load होने के बाद इस event से संबंधित code execute होता है।

Example – onload Event

```
<!DOCTYPE html>
    <html>
        <head>
            <title>JavaScript onload Event</title>
```

```
 </head>
<body style="text-align:center;" onload="myFunction()">
    <h1>Arun Computer</h1>
    <h2 style="color:red;">Example of JavaScript onload Event</h2>
    <h3>Hello World!</h3>
<script>
function myFunction() {
  alert("Page is loaded");
}
</script>
</body>
</html>
```

onunload Event

जब webpage बंद किया जाता है, refresh किया जाता है, या user किसी दूसरे page पर जाता है, तब onunload event होता है।

इस event के होने पर उससे संबंधित code execute हो जाता है।

उदाहरण:

- Browser window बंद करना
- किसी link पर click करके दूसरे page पर जाना
- Page refresh करना

Important Notes

getElementById() सबसे अधिक उपयोग होने वाला DOM method है।

DOM webpage के सभी HTML elements को objects के रूप में represent करता है।

JavaScript DOM की सहायता से webpage को dynamically बदल सकती है।

Events webpage को interactive बनाते हैं।

innerHTML content बदलने के लिए उपयोग किया जाता है।

JavaScript Dialog Boxes

JavaScript में user से interaction करने के लिए कुछ built-in dialog box methods उपलब्ध होते हैं। ये methods browser window पर popup box प्रदर्शित करते हैं।

मुख्य Dialog Box Methods निम्न हैं –

1. alert()

2. prompt()
3. confirm()

1. alert() Method

alert() method web page पर एक alert dialog box प्रदर्शित करती है। इसे popup box भी कहा जाता है।

इस dialog box में कोई सूचना (message) user को दिखाई जाती है। इसमें केवल **OK** button होता है।

जब तक user OK button पर click नहीं करता, तब तक आगे की प्रक्रिया (execution) रुकी रहती है।

Syntax

```
alert("Message");
```

Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript Alert Method</title>
    </head>
    <body style="text-align:center;">
        <h1>Arun Computer</h1>
        <h2 style="color:red;">Example of JavaScript alert() Method</h2>
        <p>Click the button to display an alert box.</p>
        <button onclick="myFunction()">Try it</button>
        <script>
        function myFunction() {
        alert("Hello! I am an alert box!");
        }
        </script>
    </body>
</html>
```

2. prompt() Method

prompt() method का उपयोग user से input प्राप्त करने के लिए किया जाता है। इसे input dialog box भी कहा जाता है।

इस dialog box में एक text field तथा दो buttons — **OK** और **Cancel** होते हैं।

- यदि user OK button दबाता है, तो input value return होती है।
- यदि user Cancel button दबाता है, तो null return होता है।

Syntax

```
prompt("Message", "Default Value");
```

Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript Prompt Method</title>
    </head>
    <body style="text-align:center;">
        <h1>Arun Computer</h1>
        <h2 style="color:red;">Example of JavaScript prompt() Method</h2>
        <button onclick="myFunction()">Try it</button>
        <p id="demo"></p>
        <script>
        function myFunction() {
        var txt;
    var person = prompt("Please enter your name:", "Arun Computer");
    if (person == null || person == "") {
    txt = "User cancelled the prompt.";
    }
    else {
    txt = "Hello " + person + "! How are you today?";
    }
    document.getElementById("demo").innerHTML = txt;
    }
    </script>
  </body>
</html>
```

3. confirm() Method

confirm() method एक confirmation dialog box प्रदर्शित करती है, जिसमें कोई संदेश (message) दिखाया जाता है।

इस dialog box में **OK** तथा **Cancel** button होते हैं।

- यदि user OK button दबाता है, तो method true return करती है।
- यदि user Cancel button दबाता है, तो method false return करती है।

Syntax

```
confirm("Message");
```

Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript Confirm Method</title>
    </head>
    <body style="text-align:center;">
```

```
<h1>Arun Computer</h1>
<h2 style="color:red;">Example of JavaScript confirm() Method</h2>
<p>Click the button to display a confirm box.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
var txt;
var r = confirm("Press a button!");
if (r == true) {
txt = "You pressed OK!";
}
else {
txt = "You pressed Cancel!";
}
document.getElementById("demo").innerHTML = txt;
}
</script>
    </body>
</html>
```

Important Points

- alert() केवल message दिखाता है।
- prompt() user से input लेता है।
- confirm() user से confirmation प्राप्त करता है।
- ये सभी methods JavaScript के built-in window object का भाग हैं।
- इन methods को window.alert(), window.prompt() तथा window.confirm() भी लिखा जा सकता है।

Exam Questions – MCU 2019

JavaScript Dialog Boxes क्या हैं? इनके प्रकार लिखिए।

निम्नलिखित Events को समझाइए –

OnLoad
OnUnload
OnMouseOver
OnClick

JavaScript में alert(), prompt() तथा confirm() methods को उदाहरण सहित समझाइए।

ARUN COMPUTER
EDUCATION