Zed-King

GROUP OF INSTITUTIONS

ISO 9001 : 2015 Certified

**M2-R5.1**

**CHAPTER-6**

# JavaScript and Angular JS

## WEB DESIGNING & PUBLISHING

" Data • Logic • Output "

**Topics Covered**

- ⚙ Introduction to JavaScript
- □ Features & Uses of JS
- ✍ Variables, Data Types & Constants
- ➗ Operators in JavaScript
- 🔄 Control Statements (if, else, switch)
- 👆 Events & Event Handling
- 🖼 JavaScript in Web Pages (Client-side Scripting)



Web Designing Instructor

**Ms. Doli**

Presented By

Zed-King Group of Institutions, Kaithal

*"JS — Code smart. Code clean."*

ZED-KING INSTITUTE

# INDEX


| Section | Topic Name | Page No. |
|---|---|---|
| **6.0** | JavaScript and Angular JS | - |
| **6.1** | Introduction to Client Side Scripting Language | - |
| **6.1.1** | Advantage and Disadvantage of Client Side Scripting | - |
| **6.1.2** | JavaScript | - |
| **6.1.2.1** | Adding JavaScript with HTML File | - |
| **6.2** | Variables in JavaScript | - |
| **6.2.1** | Declaring JavaScript Variables | - |
| **6.2.2** | Scope of the Variable | - |
| **6.2.3** | Datatypes in JavaScript | - |
| **6.3** | Operators in JavaScript | - |
| **6.4** | Condition Statements in JavaScript | - |
| **6.5** | JavaScript Popup Boxes | - |
| **6.6** | JavaScript Events | - |
| **6.7** | Basic Form Validations in JavaScript | - |
| **6.8** | Angular JS | - |
| **6.8.1** | Advantage of Angular JS | - |
| **6.8.2** | Disadvantage of Angular JS | - |
| **6.8.3** | How to Write First Angular JS Program | - |
| **6.8.4** | Concepts of Angular JS (Modules, Directives, Expression, Controller, Scope, Validations, Filters, Services, Testing) | - |


ZED-KING INSTITUTE

# Chapter 6 – JavaScript and Angular JS

## 6.0 JavaScript and Angular JS




**JavaScript** – It is a *client-side scripting language* used to make webpages interactive. It is executed inside the browser (like Chrome, Edge, Firefox).



Invented by **Brendan Eich**.

Invented in **May 1995**.

Created at **Netscape Communications Corporation**.

First name **Mocha**, then **LiveScript**, finally **JavaScript**.

Made to make **web pages interactive and dynamic**.

- Helps to **validate forms, handle events, and add animations**.
- Developed in **only 10 days**.
- Standardized by **ECMA International** as **ECMAScript** (in 1997).
- Runs on **client-side (browser)** but now also on **server-side (Node.js)**.
- **Brendan Eich** later helped to create **Mozilla Firefox**.

ZED-KING INSTITUTE

**AngularJS** – It is a *JavaScript framework* developed by Google to make web applications dynamic and easy to manage.

# 6.1 Introduction to Client-Side Scripting Language

## Scripting Language

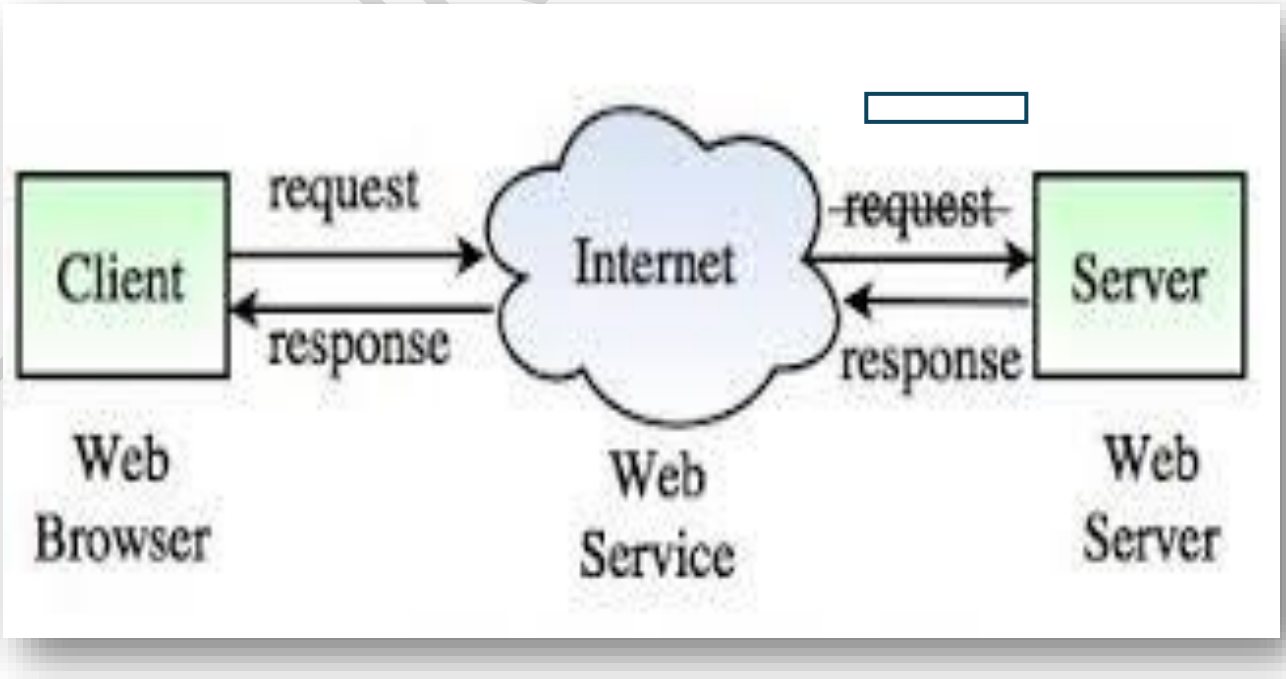
- A scripting language is a lightweight programming language used to control another software application.
- Client-side scripting runs on the user's browser (not server).

## Examples:

- JavaScript
- VBScript
- AngularJS

## Uses:

- Validating user input before sending data to server
- Showing popup messages
- Changing HTML/CSS dynamically




NG INSTITUTE

## 6.1.1 Advantages and Disadvantages of Client-Side Scripting

### Advantages:

1. Fast execution – runs on browser.
2. Reduces server load.
3. Better user interaction.
4. Can change webpage content without reloading.

### Disadvantages:

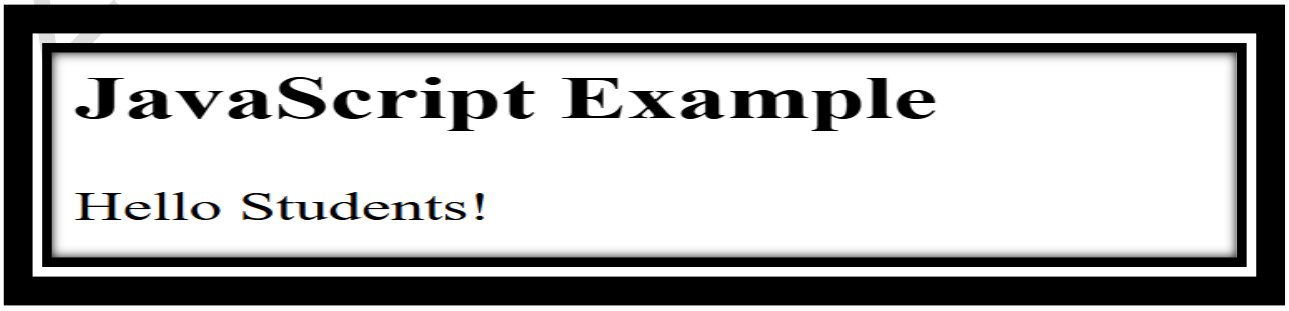
1. Less secure – code is visible to user.
2. May not run if JavaScript is disabled.
3. Browser dependent.

## 6.1.2 JavaScript

### Definition:

JavaScript is an interpreted, object-based scripting language used to create dynamic and interactive webpages.

```
<html>
<body>
  <h2>JavaScript Example</h2>
  <script>
    document.write("Hello Students!");
  </script>
</body>
</html>
```




INSTITUTE

# Extra Exam Points

- JavaScript code is written using the **<script> tag** in HTML.
- The <script> tag can be placed in **<head>** or **<body>** section.
- External JS is linked using:

  **<script src="filename.js"></script>**
- JavaScript code always ends with a **semicolon ( ; )** (recommended).
- A JavaScript file must be saved with extension **.js**
- JavaScript is case-sensitive (e.g., alert is different from Alert).
- JavaScript runs **line by line** (interpreted language).
- Browser executes JavaScript using its **JavaScript Engine**.
- Internal and External JS should be placed **before closing </body> tag** for fast loading.
- JavaScript can change:
  - HTML content
  - HTML style (CSS)
  - HTML attributes
  - Page behavior
- JavaScript is used to add **interactivity** to a webpage.
- Inline JS is not recommended because it makes code **hard to maintain**.
- External JS is the **best practice** because it supports reusability.
- JavaScript can also be written in browser **Developer Console** (Shortcut: F12).
- JavaScript can control **events** like click, mouseover, keypress, etc.
- Script tag must be properly **opened and closed** to run JS correctly.
- External JS files load faster due to **caching**.

## Features:

- Case sensitive
- Lightweight
- Event-driven
- Platform independent
- Can be embedded in HTML

ZED-KING INSTITUTE




# 6.1.2.1 Adding JavaScript with HTML File

## 1. Inline JavaScript

Written directly inside an HTML tag.

```
<button onclick="alert('Hello!')">Click Me</button>
```

```
<html>
<head>
  <title>Inline JavaScript Example</title>
</head>
<body>

<h2>Inline JavaScript Demo</h2>

<button onclick="alert('Welcome to JavaScript!')">
  Click Here
</button>

</body>
</html>
```



## 2. Internal JavaScript

Written inside the <script> tag in the same HTML file.

```
<script>
  document.write("Welcome to JS!");
</script>
```

```
<html>
<head>
  <title>Internal JS Easy Example</title>
  <script>
    function display() {
      document.write("Hello Students!");
    }
  </script>
</head>
<body>
    <h2>Internal JavaScript Demo</h2>
        <button onclick="display()">Show Message</button>
</body>
</html>
```




- JavaScript is written **inside** **<script> tag** → Internal JS
- display() function prints **Hello Students!** on the page
- When you click the button → function runs → message appears

## 3. External JavaScript

Saved in a separate .js file and linked using src attribute.

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

## 1st file:-

- **Index.html**

```
<html>
<head>
  <title>External JavaScript Example</title>
</head>
<body>
    <h2>External JavaScript Demo</h2>
        <button onclick="showAlert()">Click Me</button>
        <!-- Linking External JS File -->
        <script src="script.js"></script>
</body>
</html>
```

## 2nd File:-

- **Script.js**

```
function showAlert() {
  alert("Hello from External JavaScript!");
}
```



- JavaScript code is written in **script.js** (external file).
- HTML connects it using:

  **<script src="script.js"></script>**
- Button click runs the JS function from the external file.
- Alert popup appears on screen.

## Comments in JavaScript:-

- Comments are notes written inside the code **that JavaScript does NOT execute**.
- They are used to explain code, make it readable, and for debugging.
- JavaScript supports **two types of comments**:

## 1). Single-line Comment

Used to comment one line only.

**Syntax:**

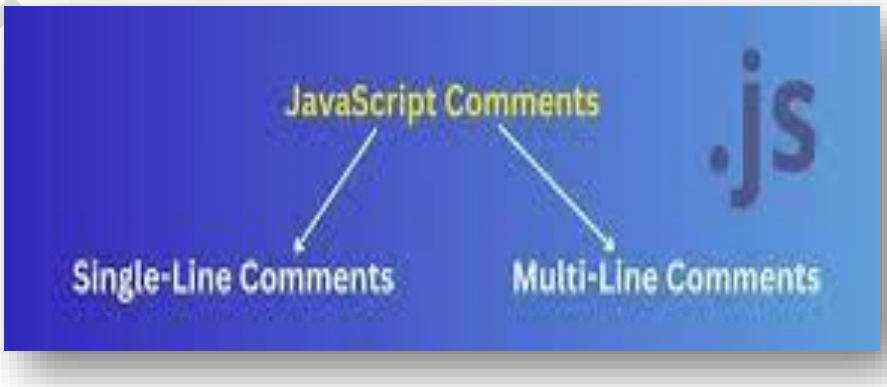
```
// This is a single-line comment
```

## 2). Multi-line Comment

Used for long explanations or multiple lines.

**Syntax:**

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




ZED-KING INSTITUTE

## Example of Single-line & Multi-line comment in JS:-

```html
<html>
<head>
  <title>JS Comments Example</title>
  <script>
    // Single-line comment
    function hello() {
      /* This is a multi-line comment.
         It explains the function. */
      alert("Hello Everyone!");
    }
  </script>
</head>
<body>
    <button onclick="hello()">Click Me</button>
</body>
</html>
```

# 6.2 Variables in JavaScript

A **variable** is a container used to store data values.

## Declaration Methods:

| Keyword | Description |
|---|---|
| var | Old method (function scoped) |
| let | Block scoped (used in modern JS) |
| const | For fixed values (cannot change) |




ZED-KING

**Example:**

var name = "Doli";

let age = 20;

const pi = 3.14;

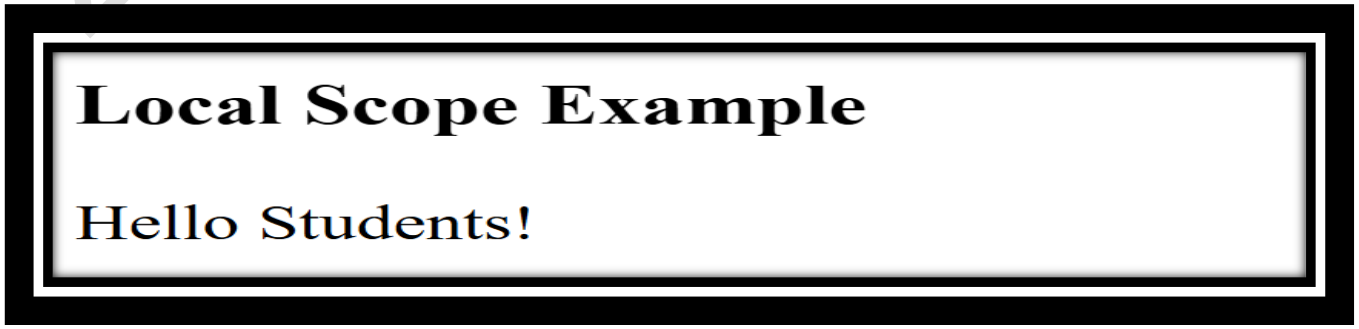
## 6.2.2 Scope of the Variable

**Local Scope:**

- Declared inside a function — can be used only inside that function.
- A **local variable** is a variable declared **inside a function**.
- It can be **used only inside that function**, not outside.

**Example:**

```
<html>
<body>
<h3>Local Scope Example</h3>

<script>
function showMessage() {
  let message = "Hello Students!"; // local variable
  document.write(message);
}

showMessage();  // ? Works fine

document.write(message); // ? Error! (cannot access outside function)
</script>

</body>
</html>
```



## Explanation:

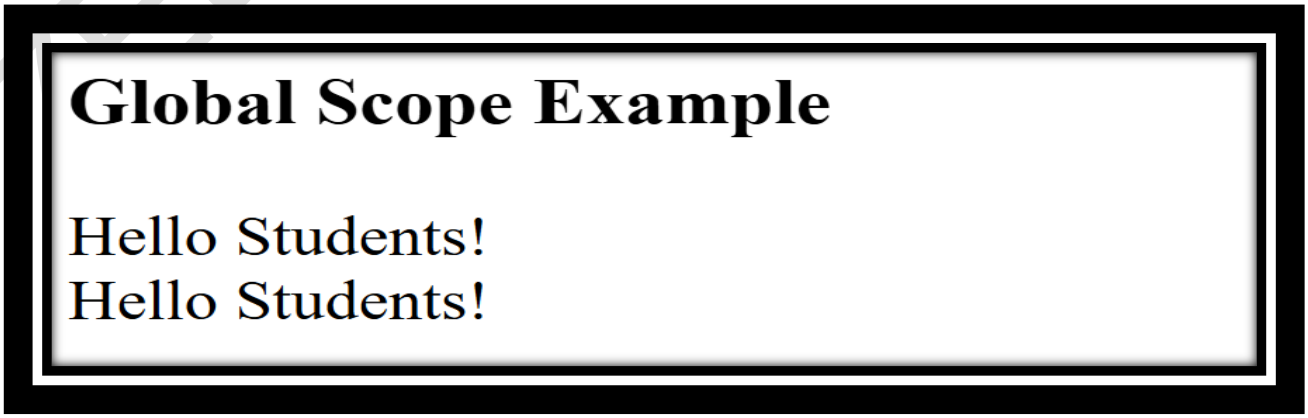
- message is declared **inside** the function showMessage().
- So it has **local scope** — it exists only **inside** that function.
- Trying to use it **outside** gives an **error**.

## Global Scope:

- A **global variable** is declared **outside any function**.
- It can be **used anywhere** in the program — inside or outside functions.

## Example:

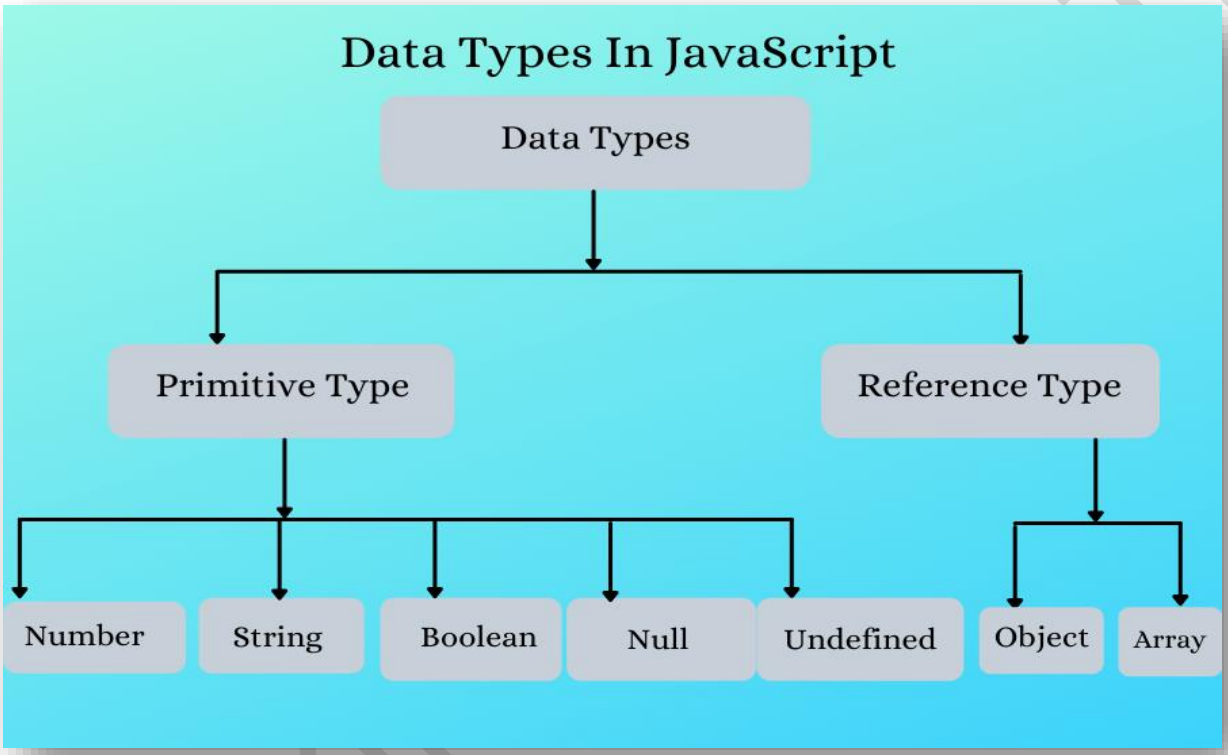
```
<html>
<body>
<h3>Global Scope Example</h3>

<script>
let message = "Hello Students!";  // global variable

function showMessage() {
  document.write(message + "<br>"); // can use inside function
}

showMessage();   // ? Works
document.write(message); // ? Works outside function too
</script>

</body>
</html>
```



**Explanation:**

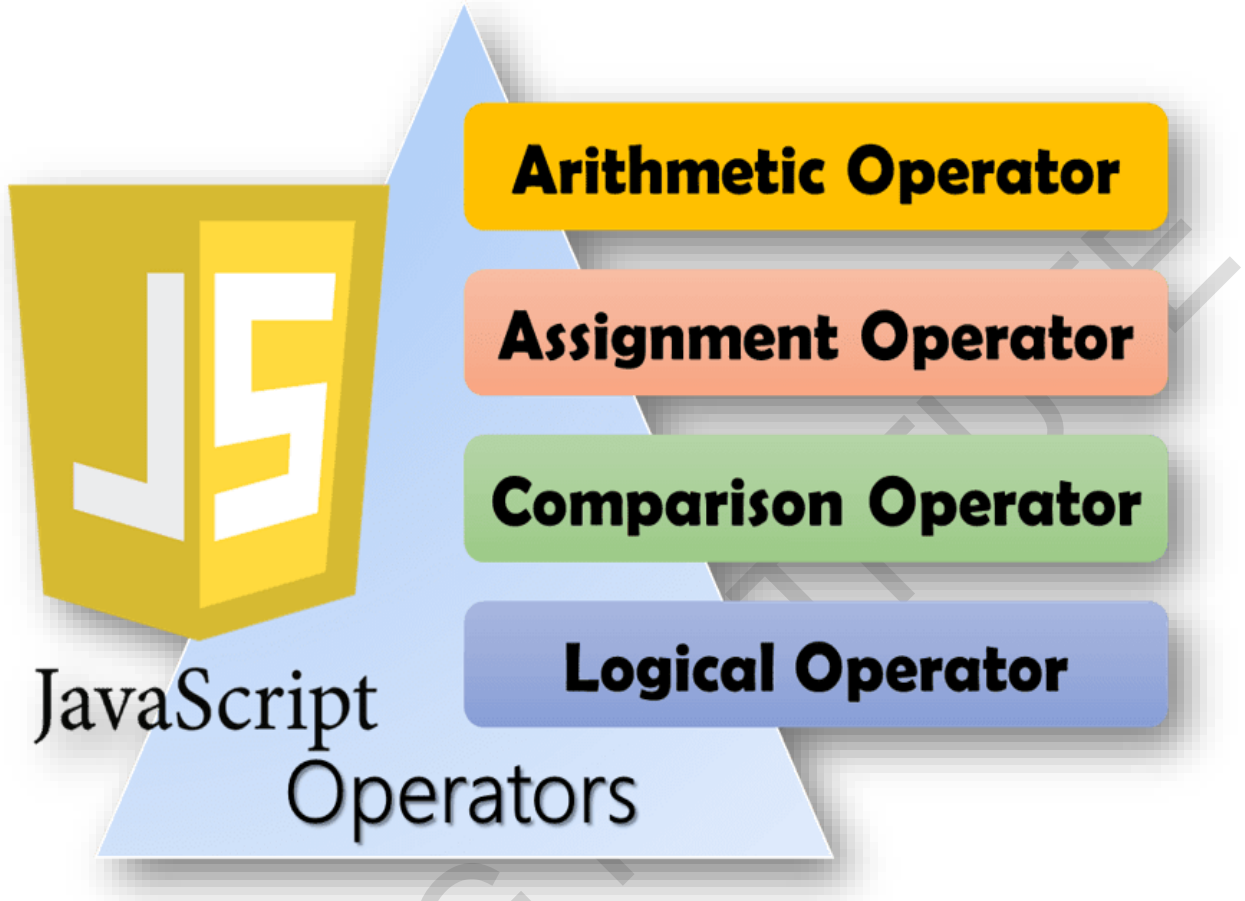
- message is declared **outside** any function.
- So it has **global scope** — accessible **everywhere** in the script.
- You can use it **inside** or **outside** functions.

## 6.2.3 Data Types in JavaScript




| Data Type | Example |
|---|---|
| **String** | "Hello" |
| **Number** | 25, 3.14 |
| **Boolean** | true, false |
| **Array** | [10,20,30] |
| **Object** | {name:"Doli", age:20} |
| **Null** | let x = null; |
| **Undefined** | let x; |

# 6.3 Operators in JavaScript




**1). Arithmetic Operators**

**2). Assignment Operators**

**3). Comparison Operators**

**4). Logical Operators**

**5). Bitwise Operators**

**6). Conditional OR Ternary Operators**

**7). Typeof Operators**

**8). Special Operators**

ZED KING

## 1. Arithmetic Operators

### ❖+, -, *, /, %, ++, --

- **++a (Pre-Increment):-**

  In which Firstly increase the value of two times then display.)

- **a++ (Post-Increment):-**

  In which Firstly display the value and then increase of two times .)

- **--a (Pre-Decrement):-**

  In which Firstly decrease the value of two times then display.)

- **a-- (Post-Deccrement):-**

  In which Firstly display the value and then decrease of two times .)

| Arithmetic Operator | Name | Example |
|---|---|---|
| + | Addition | a + b |
| - | Subtraction | a – b |
| * | Multiplication | a * b |
| / | Division | a / b |
| % | Modulus | a % b |
| ++ | Increment Operator | a++ |
| -- | Decrement Operator | a-- |

INSTITUTE

# Example Code:-

```html
<html>
<head>
    <title>Arithmetic Operators + Pre/Post</title>
</head>
<body>
    <script>
        let a = 10;
        let b = 3;
        document.write("a = " + a + "<br>");
        document.write("b = " + b + "<br><br>");
        // Arithmetic operators
        document.write("a + b = " + (a + b) + "<br>");
        document.write("a - b = " + (a - b) + "<br>");
        document.write("a * b = " + (a * b) + "<br>");
        document.write("a / b = " + (a / b) + "<br>");
        document.write("a % b = " + (a % b) + "<br><br>");
        // Post Increment
        document.write("a++ (post increment) = " + (a++) + "<br>");
        document.write("Value of a after a++ = " + a + "<br><br>");
        // Pre Increment
        document.write("++a (pre increment) = " + (++a) + "<br>");
        document.write("Value of a after ++a = " + a + "<br><br>");
        // Post Decrement
        document.write("b-- (post decrement) = " + (b--) + "<br>");
        document.write("Value of b after b-- = " + b + "<br><br>");
        // Pre Decrement
        document.write("--b (pre decrement) = " + (--b) + "<br>");
        document.write("Value of b after --b = " + b + "<br>");
    </script>
</body>
</html>
```

```
a = 10
b = 3

a + b = 13
a - b = 7
a * b = 30
a / b = 3.3333333333333335
a % b = 1

a++ (post increment) = 10
Value of a after a++ = 11

++a (pre increment) = 12
Value of a after ++a = 12

b-- (post decrement) = 3
Value of b after b-- = 2

--b (pre decrement) = 1
Value of b after --b = 1
```

## 2. Assignment Operators

**=, +=, -=, *=, /=,%=**

| Operator | Meaning |
|---|---|
| = | Simple assignment |
| += | Add assignment |
| -= | Subtract assignment |
| *= | Multiply assignment |
| /= | Divide assignment |
| %= | Modulus assignment |

## Example Code:-

```
<html>
<head>
    <title>Assignment Operators in JS</title>
</head>
<body>
    <script>
        let x = 10;
        document.write("Initial Value: " + x + "<br><br>");

        x += 5;
        document.write("After x += 5 : " + x + "<br>");

        x -= 5;
        document.write("After x -= 5 : " + x + "<br>");

        x *= 5;
        document.write("After x *= 5 : " + x + "<br>");

        x /= 5;
        document.write("After x /= 5 : " + x + "<br>");

        x %= 5;
        document.write("After x %= 5 : " + x + "<br>");

        x **= 5;
        document.write("After x **= 5 : " + x + "<br>");
    </script>
</body>
</html>
```

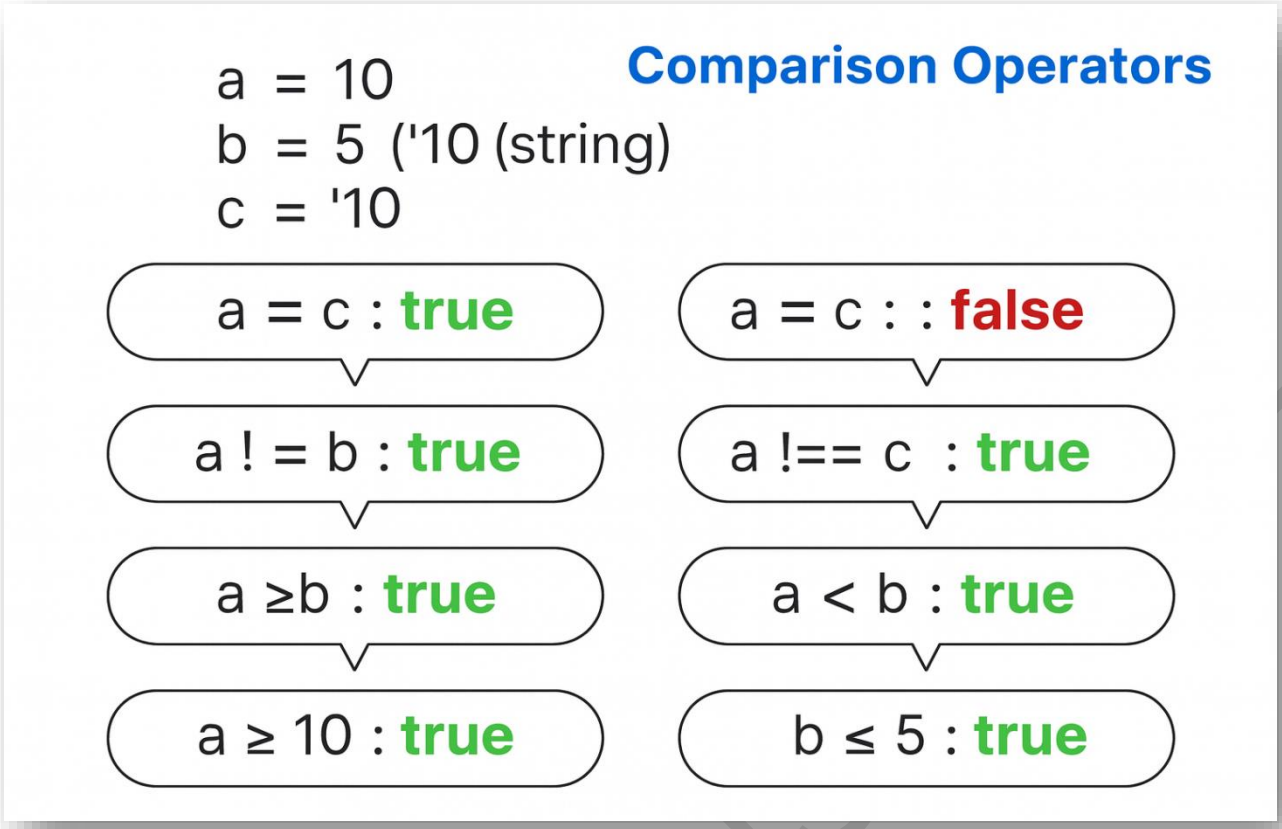
```
Initial Value: 10

After x += 5 : 15
After x -= 5 : 10
After x *= 5 : 50
After x /= 5 : 10
After x %= 5 : 0
After x **= 5 : 0
```

## 3. Comparison Operators

==, ===, !=,!===, >, <, >=, <=

| Operator | Name | Meaning |
|---|---|---|
| == | **Equal To** | Value same ho to true |
| === | **Strict Equal** | Value + Type dono same ho |
| != | **Not Equal** | Value different ho |
| !== | **Strict Not Equal** | Value ya type different ho |
| > | **Greater Than** | Left value bada ho |
| < | **Less Than** | Left value chhota ho |
| >= | **Greater Than or Equal** | Left value bada ya equal ho |
| <= | **Less Than or Equal** | Left value chhota ya equal ho |

ZED-KING INSTITUTE




```html
<html>
<head>
    <title>Comparison Operators</title>
</head>
<body>
    <script>
        let a = 10;
        let b = 5;
        let c = "10";
        document.write("a = " + a + "<br>");
        document.write("b = " + b + "<br>");
        document.write("c = '10' (string)" + "<br><br>");
        // Equal ==
        document.write("a == c : " + (a == c) + "<br>");
        // Strict Equal ===
        document.write("a === c : " + (a === c) + "<br><br>");
        // Not Equal !=
        document.write("a != b : " + (a != b) + "<br>");
        // Strict Not Equal !==
        document.write("a !== c : " + (a !== c) + "<br><br>");
        // Greater Than >
        document.write("a > b : " + (a > b) + "<br>");
        // Less Than <
        document.write("a < b : " + (a < b) + "<br><br>");
        // Greater Than or Equal >=
        document.write("a >= 10 : " + (a >= 10) + "<br>");
        // Less Than or Equal <=
        document.write("b <= 5 : " + (b <= 5) + "<br>");
    </script>
</body>
</html>
```

```
a = 10
b = 5
c = '10' (string)

a == c : true
a === c : false

a != b : true
a !== c : true

a > b : true
a < b : false

a >= 10 : true
b <= 5 : true
```

## 4. Logical Operators

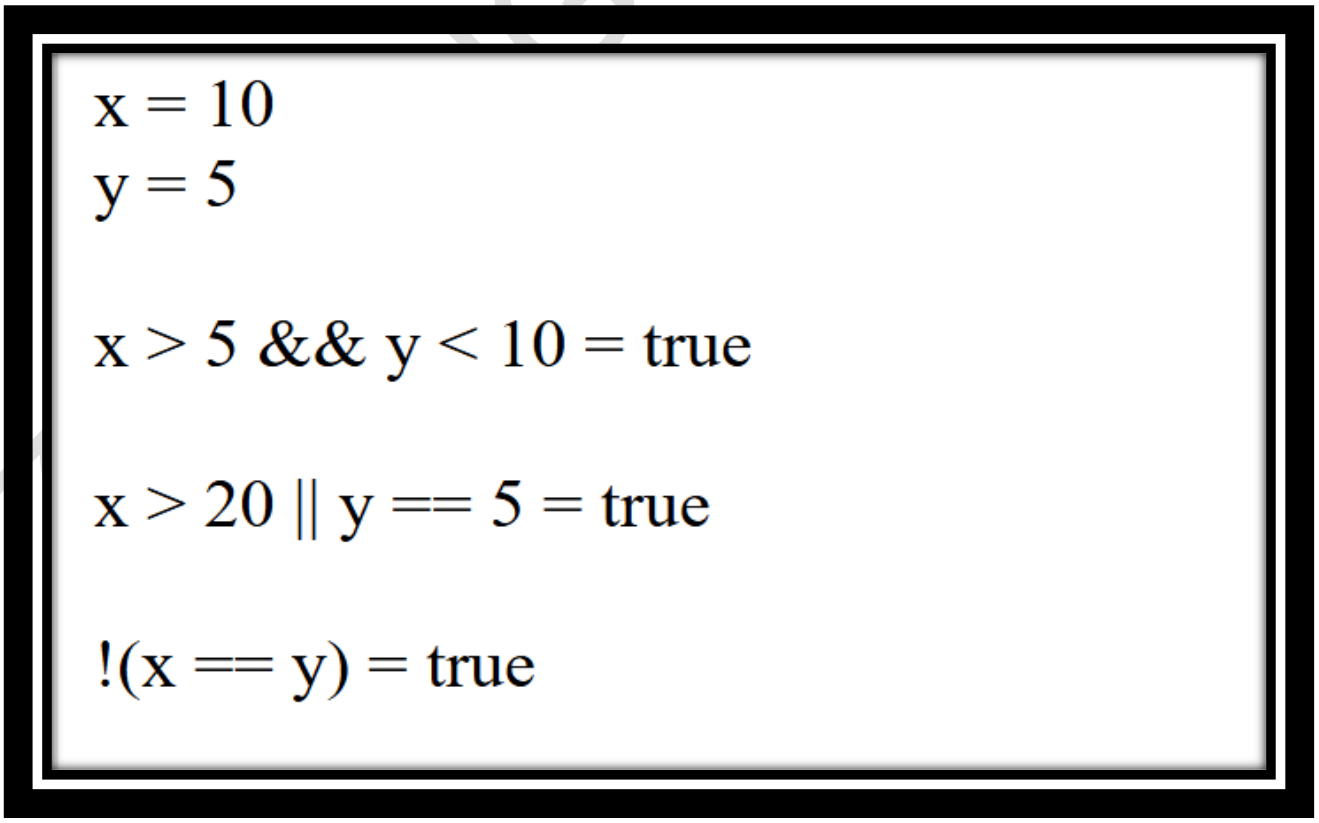
**&&, ||, !**

- **&& (Logical AND)** **Left to Right(Lowest Priority)**
- **|| (Logical OR)** **Left to Right**
- **! (Logical NOT)** **Right to Left(Highest Priority)**

| Operator | Meaning |
|---|---|
| && | Logical AND |
| \|\| | Logical OR |
| ! | Logical NOT |

**Example Code:-**

```
<html>
<head>
    <title>Logical Operators</title>
</head>
<body>
    <script>
        let x = 10;
        let y = 5;
        document.write("x = " + x + "<br>");
        document.write("y = " + y + "<br><br>");
        // AND (&&)
        document.write("x > 5 && y < 10 = " + (x > 5 && y < 10) + "<br><br>");
        // OR (||)
        document.write("x > 20 || y == 5 = " + (x > 20 || y == 5) + "<br><br>");
        // NOT (!)
        document.write("!(x == y) = " + !(x == y) + "<br>");
    </script>
</body>
</html>
```



## Output Explanation

| Expression | Result | Reason |
|---|---|---|
| **a && b** | **true** | both must be true |
| **a \|\| b** | **true** | at least one is true |
| **!a** | **false** | opposite of true |
| **!b** | **true** | opposite of false |

## 5). Bitwise Operators

| Operator | Meaning | Easy Explanation |
|---|---|---|
| **&** | **AND** | Dono 1 → result 1 |
| \| | **OR** | Koi ek 1 → result 1 |
| ^ | **XOR** | Alag bits → 1 |
| ~ | **NOT** | Bits ulte ho jaate |
| << | **Left Shift** | Number × 2 |
| >> | **Right Shift** | Number ÷ 2 |
| >>> | **Unsigned Right Shift** | Right shift without sign |

| Operator | Name | Description (Easy) | Example | Output |
|---|---|---|---|---|
| **&** | **Bitwise AND** | Dono bits 1 honge tab 1 milega | `5 & 3` | **1** |
| ** | ** | Bitwise OR | Agar koi ek bit 1 ho → 1 | `5 |
| ^ | **Bitwise XOR** | Different bits → 1, same → 0 | `5 ^ 3` | **6** |
| ~ | **Bitwise NOT** | Sare bits ulte ho jaate | `~5` | **-6** |
| << | **Left Shift** | Bits left shift → ×2 hota | `5 << 1` | **10** |

ZED-KING INSTITUTE

| Operator | Name | Description (Easy) | Example | Output |
|---|---|---|---|---|
| >> | **Right Shift** | Bits right shift → ÷2 hota | `5 >> 1` | **2** |
| >>> | **Unsigned Right Shift** | Right shift, sign ignore | `-5 >>> 1` | Large positive value |

| Operation | Binary Form | Result (Binary) | Result (Decimal) |
|---|---|---|---|
| **5 & 3** | 0101 & 0011 | **0001** | **1** |
| **5 | 3** | 0101 | 0011 | **0111** | **7** |
| **5 ^ 3** | 0101 ^ 0011 | **0110** | **6** |
| **~5** | ~0101 | **1010 (2's complement)** | **-6** |
| **5 << 1** | 0101 → 1010 | **1010** | **10** |
| **5 >> 1** | 0101 → 0010 | **0010** | **2** |
| **-5 >>> 1** | Sign removed | (Binary shifted) | **Large +ve** |

## Example code:-

```
<html>
<head>
    <title>Bitwise Operators in JS</title>
</head>
<body>
    <script>
        let a = 5; // 0101
        let b = 3; // 0011
        document.write("a = " + a + "<br>");
        document.write("b = " + b + "<br><br>");
        document.write("a & b = " + (a & b) + " (AND)<br>");
        document.write("a | b = " + (a | b) + " (OR)<br>");
        document.write("a ^ b = " + (a ^ b) + " (XOR)<br>");
        document.write("~a = " + (~a) + " (NOT)<br>");
        document.write("a << 1 = " + (a << 1) + " (Left Shift)<br>");
        document.write("a >> 1 = " + (a >> 1) + " (Right Shift)<br>");
        document.write("a >>> 1 = " + (a >>> 1) + " (Unsigned Right Shift)<br>");
    </script>
</body>
</html>
```

```
a = 5
b = 3

a & b = 1 (AND)
a | b = 7 (OR)
a ^ b = 6 (XOR)
~a = -6 (NOT)
a << 1 = 10 (Left Shift)
a >> 1 = 2 (Right Shift)
a >>> 1 = 2 (Unsigned Right Shift)
```

## Truth Table of AND ( && or & )

| A | B | A AND B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | **1** |

**➞ AND = Only when both are 1**

## Truth Table of OR ( || or | )

| A | B | A OR B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | **1** |

**➞ OR = If any one is 1 → result 1**

ZED-KING INSTITUTE

## Truth Table of NOT ( ! or ~ )

| A | NOT A |
|---|---|
| 0 | **1** |
| 1 | **0** |

**➞ NOT = Opposite value**

## Truth Table of XOR ( ^ )

| A | B | A XOR B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | 0 |

**➞ XOR = When A and B are different → 1**

## 6). Ternary Operator in JavaScript

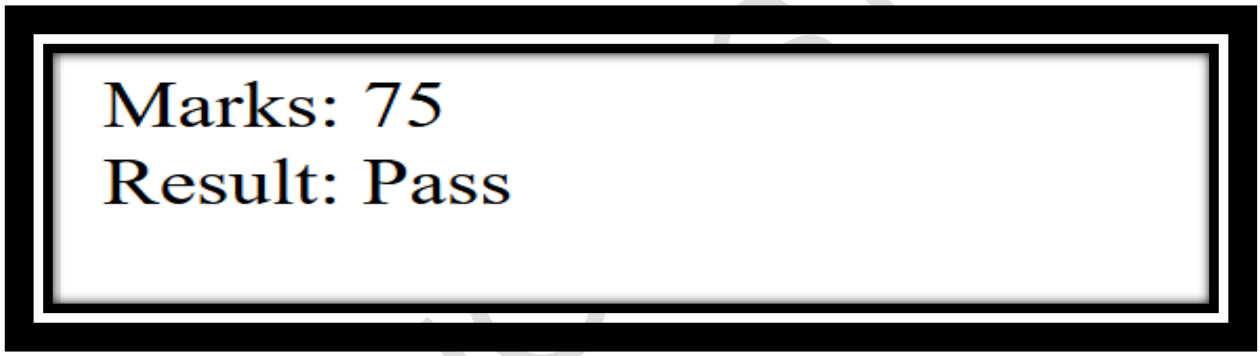
Ternary operator ek **shortcut if–else** hota hai.

| Part | Meaning |
|---|---|
| `condition` | check hota hai (true/false) |
| `?` | true ho to ye part chalega |
| `:` | false ho to ye part chalega |

ZED-KING INSTITUTE

## Example:-

```
<html>
    <head>
        <title>Ternary Operators in JS</title>
    </head>
<body>
    <script>
        let marks = 75;
        let grade = (marks >= 33) ? "Pass" : "Fail";
        document.write("Marks: " + marks + "<br>");
        document.write("Result: " + grade);
    </script>
</body>
</html>
```




## 7). Typeof Operator:-

```
<html>
<head>
    <title>Use of TypeOf Operator</title>
</head>
<body>
<pre>
    <script>
        var id = 101;
        var EmpName = "Manish";
        var salary = 25000;
        var experience = [2016, 2017, 2018, 2019, 2020];
        var address;
        document.write("EmpName : " + typeof(EmpName));
        document.write("<br><br>");
        document.write("id : " + typeof(id));
        document.write("<br><br>");
        document.write("salary : " + typeof(salary));
        document.write("<br><br>");
        document.write("Experience : " + typeof(experience));
        document.write("<br><br>");
        document.write("address : " + typeof(address));
    </script>
</pre>
</body>
</html>
```

```
        EmpName : string
id : number
salary : number
Experience : object
address : undefined
```

| Value / Type | Result of `typeof` |
|---|---|
| 10 | **"number"** |
| 10.5 | **"number"** |
| "Hello" | **"string"** |
| true / false | **"boolean"** |
| undefined | **"undefined"** |
| null | **"object"** ***(JS bug)*** |
| [1,2,3] (Array) | **"object"** |
| {name:"A"} (Object) | **"object"** |
| function(){} | **"function"** |
| NaN | **"number"** |

## 8). JavaScript Specail Operators:-

| Operator | Description |
|---|---|
| **,** | Comma Operator allows multiple expressions to be evaluated as single statement. |
| **delete** | Delete Operator deletes a property from the object. |
| **in** | In Operator checks if object has the given property. |
| **instanceof** | Checks if the object is an instance of given type. |
| **new** | Creates an instance (object). |
| **void** | It discards the expression's return value. |

ZED-KING INSTITUTE

| Operator | Description |
|---|---|
| **yield** | Checks what is returned in a generator by the generator's iterator. |

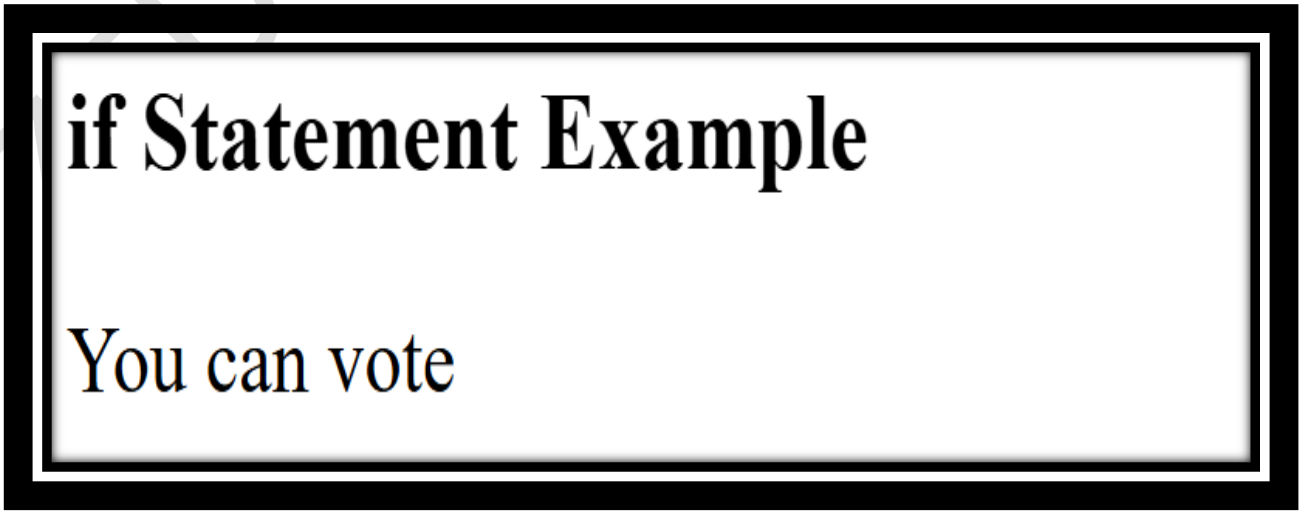
## 6.4 Conditional Statements in JavaScript

Used to perform actions based on conditions.

1. **if condition**
2. **if-else condition**
3. **if-else-if condition**
4. **switch case condition**

### 1). if statement

```
<html>
<head>
  <title>if Statement Example</title>
</head>
<body>
  <h3>if Statement Example</h3>

  <script>
    let age = 20;

    if (age >= 18) {
      document.write("You can vote");
    }
  </script>
</body>
</html>
```



2). **if...else**

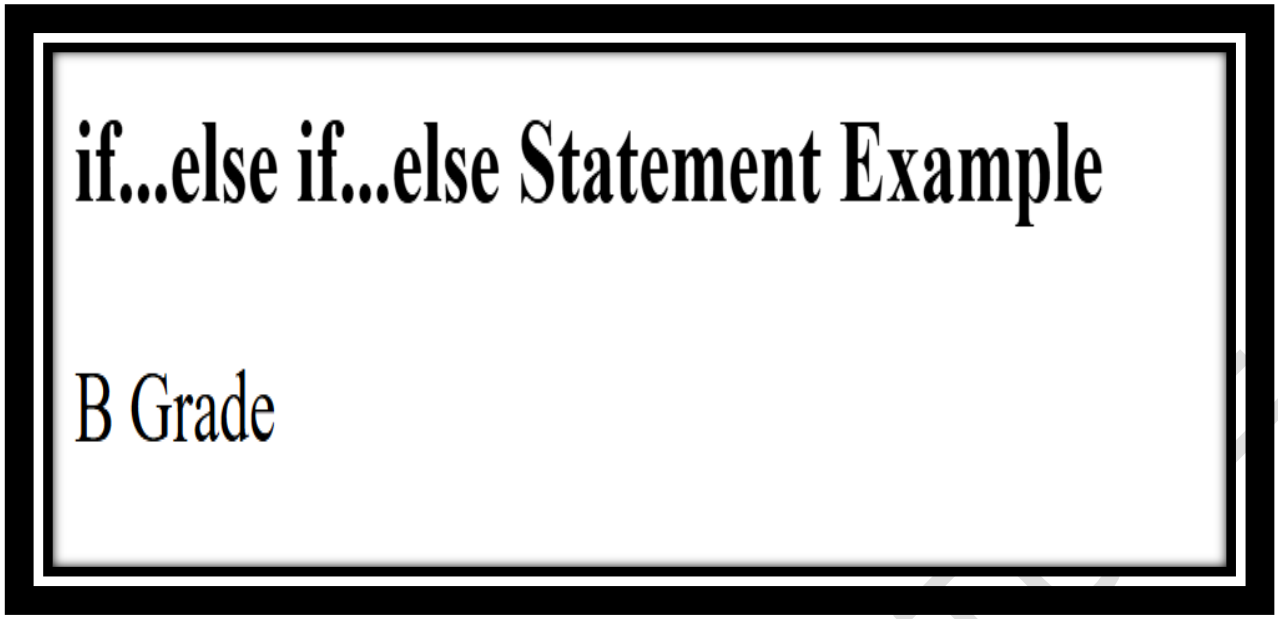
```
<html>
<head>
  <title>if else Statement Example</title>
</head>
<body>
  <h3>if...else Statement Example</h3>

  <script>
    let marks = 35;

    if (marks >= 40)
      document.write("Pass");
    else
      document.write("Fail");
  </script>
</body>
</html>
```

**if...else Statement Example**

Fail

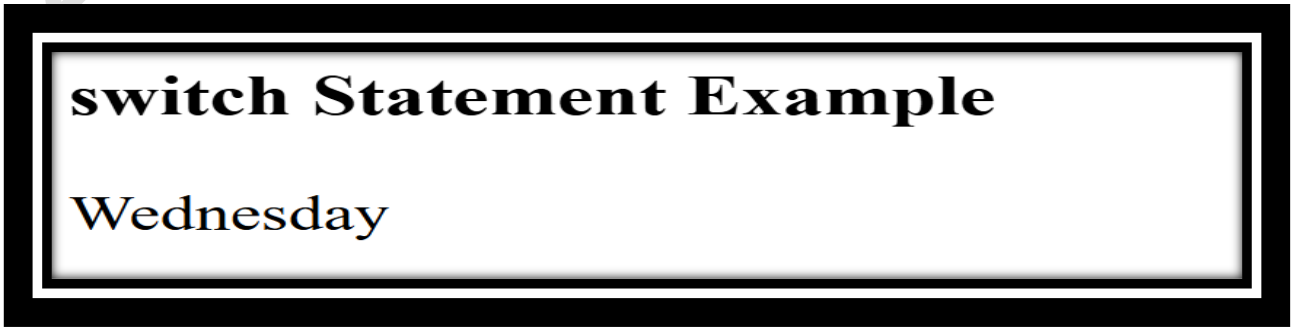
3). **if...else if...else**

```
<html>
<head>
  <title>if else if else Example</title>
</head>
<body>
  <h3>if...else if...else Statement Example</h3>

  <script>
    let marks = 75;

    if (marks >= 80)
      document.write("A Grade");
    else if (marks >= 60)
      document.write("B Grade");
    else
      document.write("C Grade");
  </script>
</body>
</html>
```




4). **switch statement**

```
<html>
<head>
  <title>switch Statement Example</title>
</head>
<body>
  <h3>switch Statement Example</h3>

  <script>
    let day = 3;

    switch(day){
      case 1: document.write("Monday"); break;
      case 2: document.write("Tuesday"); break;
      case 3: document.write("Wednesday"); break;
      case 4: document.write("Thursday"); break;
      case 5: document.write("Friday"); break;
      case 6: document.write("Saturday"); break;
      case 7: document.write("Sunday"); break;
      default: document.write("Invalid Day");
    }
  </script>
</body>
</html>
```



# 6.5 JavaScript Popup Boxes

Here are three types of boxes:-

**1. Alert Box**

**2. Confirm Box**

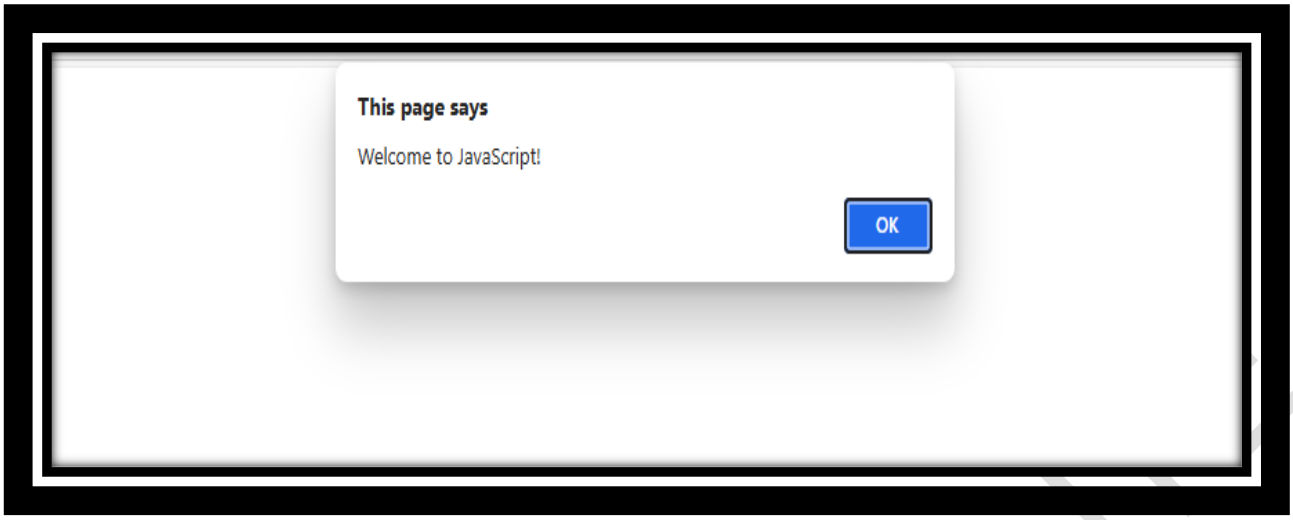
**3. Prompt Box**




| Box | Function | Example |
|---|---|---|
| **alert()** | Display message | alert("Hello"); |
| **confirm()** | OK / Cancel | confirm("Are you sure?"); |
| **prompt()** | Take input | name = prompt("Enter your name"); |

## 1). Alert Box Example

```
<html>
<head>
  <title>Alert Box Example</title>
</head>
<body>
  <h3>Alert Box Example</h3>

  <script>
    alert("Welcome to JavaScript!");
  </script>
</body>
</html>
```




## 2). Confirm Box Example

```
<html>
<head>
  <title>Confirm Box Example</title>
</head>
<body>
  <h3>Confirm Box Example</h3>

  <script>
    let choice = confirm("Do you want to continue?");
    if (choice)
      document.write("You pressed OK!");
    else
      document.write("You pressed Cancel!");
  </script>
</body>
</html>
```



**Explanation:**

confirm() ek message dikhata hai jisme **OK** aur **Cancel** button hote hain.

- **OK → true return karta hai**
- **Cancel → false return karta hai**

## 3). Prompt Box Example

```
<html>
<head>
  <title>Prompt Box Example</title>
</head>
<body>
  <h3>Prompt Box Example</h3>

  <script>
    let name = prompt("Enter your name:");
    document.write("Hello, " + name + "!");
  </script>
</body>
</html>
```



# 6.6 JavaScript Events

An **event** is an action like click, load, keypress, etc.

| Event | Description |
|---|---|
| **onclick** | When user clicks |
| **onmouseover** | When mouse moves over element |
| **onload** | When page is loaded |

- Events are **actions or occurrences** that happen in the browser.
- Example: When a user clicks a button, moves the mouse, or presses a key.
- JavaScript can "react" to these events.

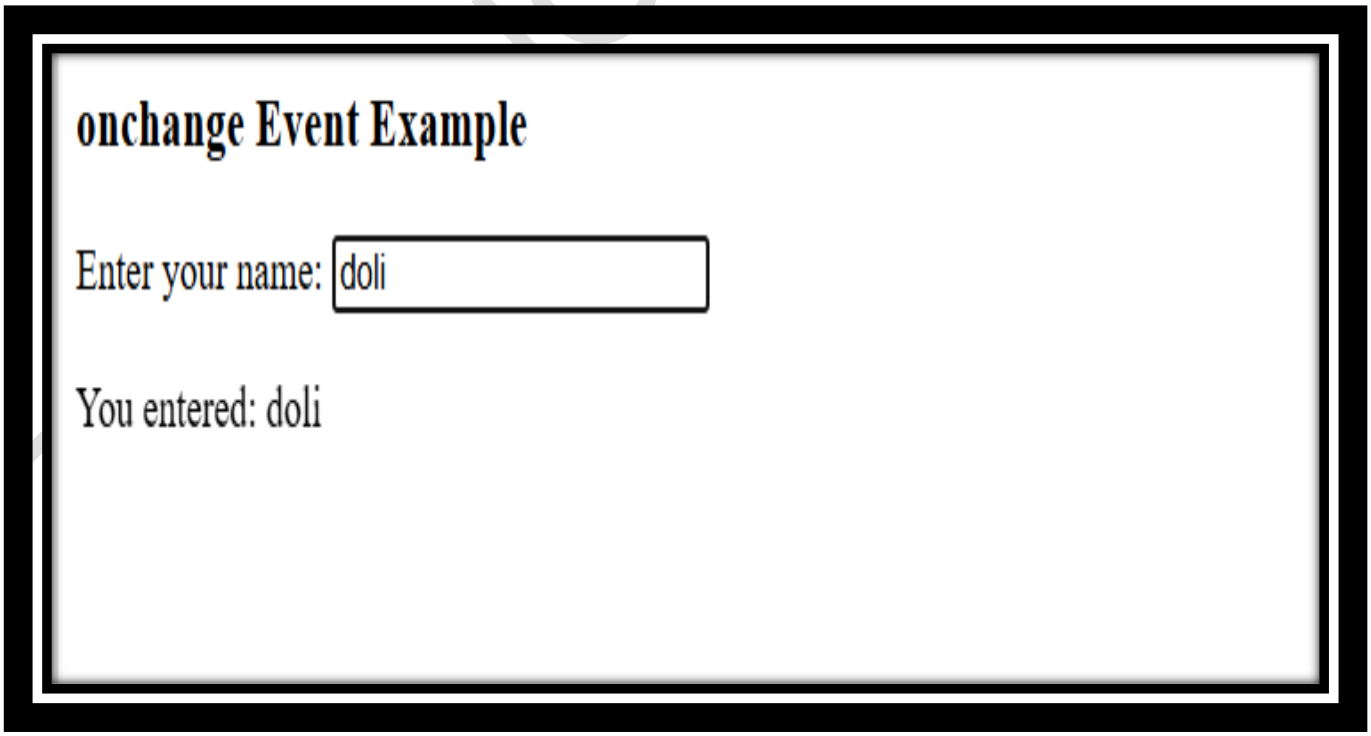
## Common JavaScript Events

| Event | Description |
|---|---|
| **onchange** | When user clicks an value is changed |
| **onclick** | When user clicks |
| **onmouseover** | When mouse moves over element |
| **onmouseout** | When mouse moves out |
| **onkeydown** | When a key is pressed |
| **onload** | When page loads |

ZED-KING INSTITUTE

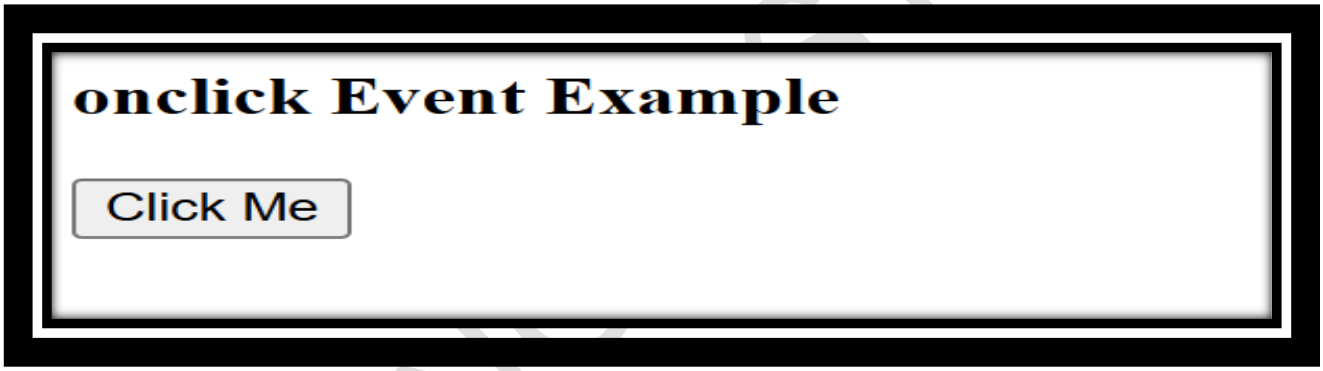
## 1). onchange Event

```
<html>
<body>
    <h3>onchange Event Example</h3>
        <!-- Input box -->
        Enter your name:
        <input type="text" id="nameInput" onchange="showName()">
        <p id="result"></p>
    <script>
        function showName() {
          let name = document.getElementById("nameInput").value;
          document.getElementById("result").innerHTML = "You entered: " + name;
        }
    </script>
</body>
</html>
```



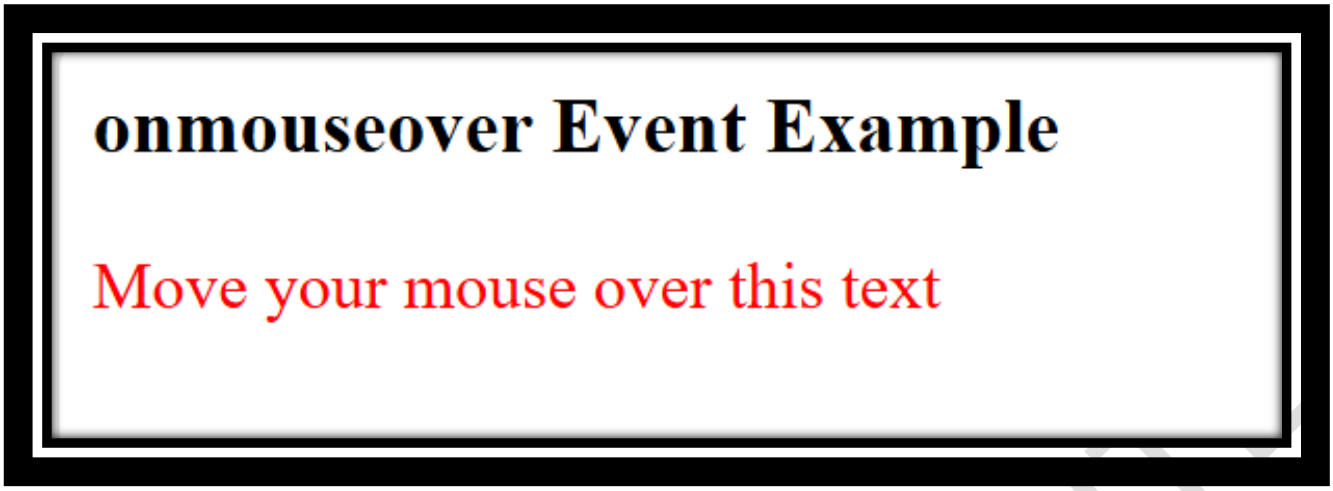
## 2). onclick Event

```
<html>
<head>
  <title>onclick Event Example</title>
</head>
<body>
  <h3>onclick Event Example</h3>
  <button onclick="document.write('Button Clicked!')">Click Me</button>
</body>
</html>
```



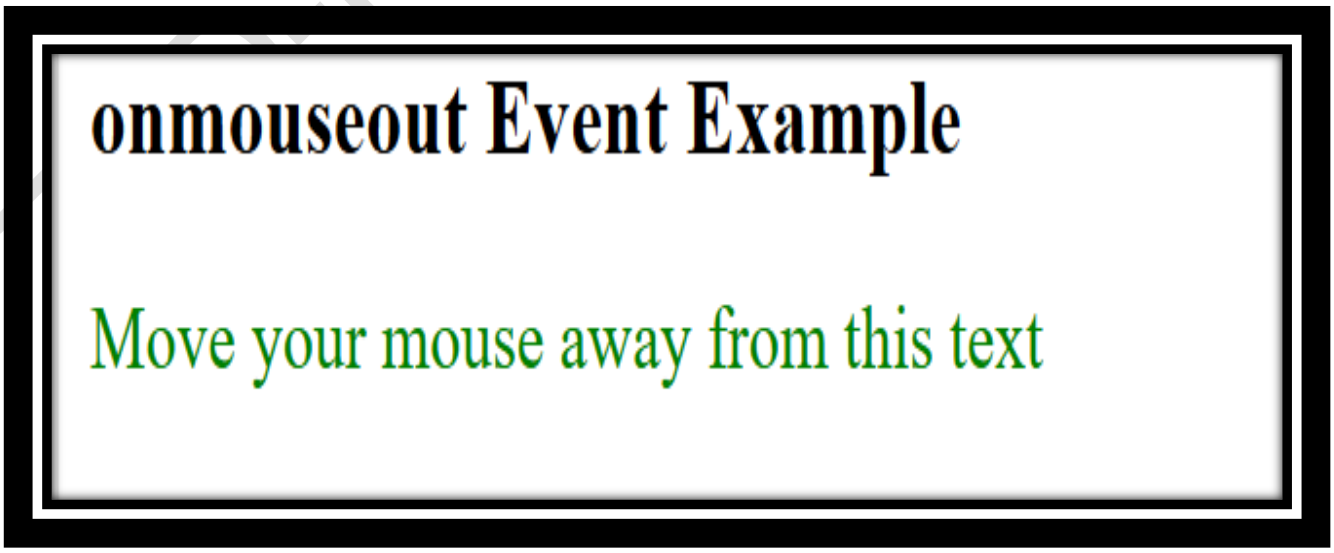

## 3). onmouseover Event

```
<html>
<head>
  <title>onmouseover Event Example</title>
</head>
<body>
  <h3>onmouseover Event Example</h3>
  <p onmouseover="this.style.color='red'">Move your mouse over this text</p>
</body>
</html>
```



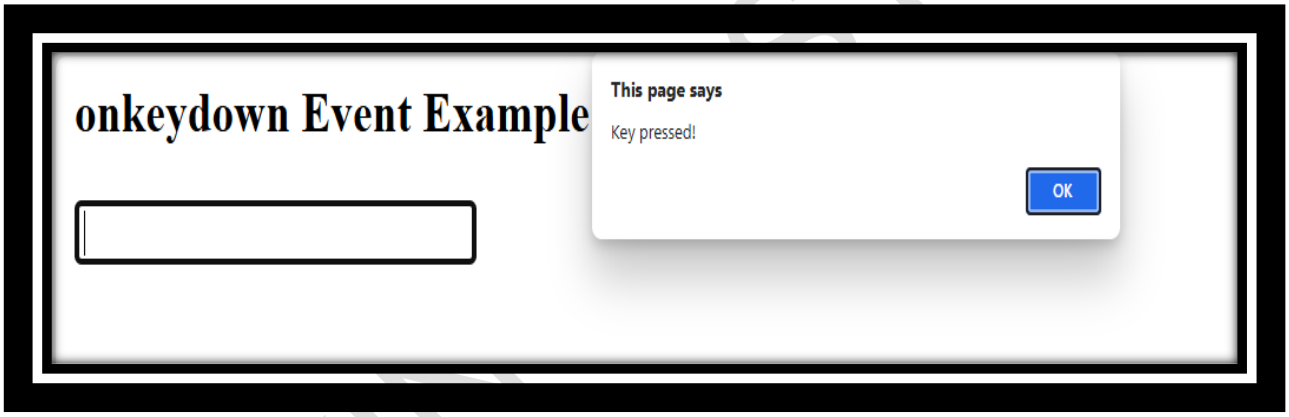

**4). onmouseout Event**

```
<html>
<head>
  <title>onmouseout Event Example</title>
</head>
<body>
  <h3>onmouseout Event Example</h3>
  <p onmouseout="this.style.color='green'">Move your mouse away from this text</p>
</body>
</html>
```



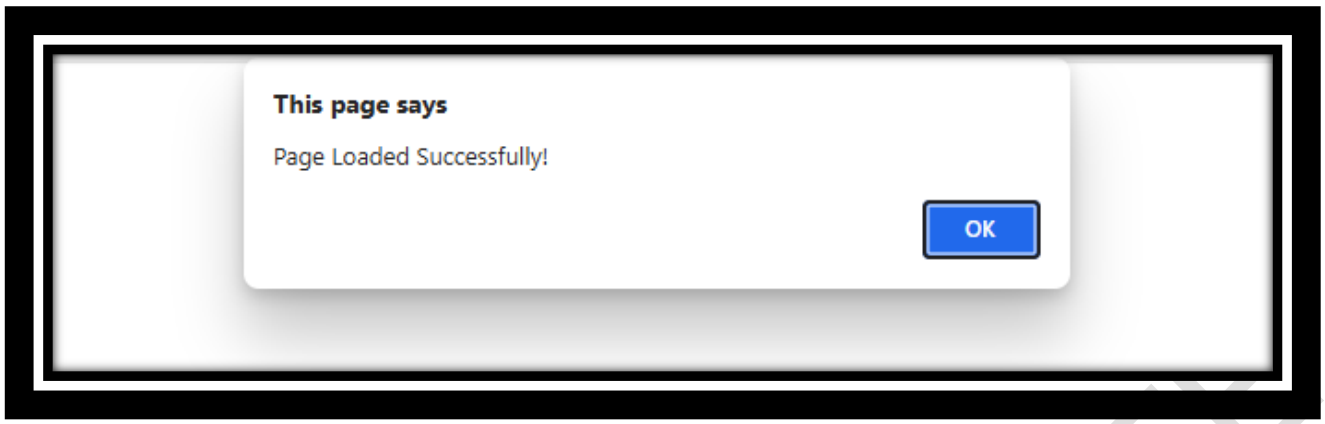
## 5). onkeydown Event

```
<html>
<head>
  <title>onkeydown Event Example</title>
</head>
<body>
  <h3>onkeydown Event Example</h3>
  <input type="text" onkeydown="alert('Key pressed!')">
</body>
</html>
```




## 6). onload Event

```
<html>
<head>
  <title>onload Event Example</title>
</head>
<body onload="alert('Page Loaded Successfully!')">
  <h3>onload Event Example</h3>
</body>
</html>
```




| Attribute गुण | Value | Description विवरण |
|---|---|---|
| offline ऑफलाइन | script | Triggers when the document goes offline. डॉक्यूमेंट ऑफलाइन होने पर ट्रिगर। |
| onabort ऑनअबॉर्ट | script | Triggers on an abort event. एक अबॉर्ट इवेंट पर ट्रिगर करता है। |
| onafterprint ऑनऑफ्टरप्रिंट | script | Triggers after the document is printed. डॉक्यूमेंट प्रिंट होने के बाद ट्रिगर करता है। |
| onbeforeonload ऑनविफोरऑनलोड | script | Triggers before the document loads. डॉक्यूमेंट लोड होने से पहले ट्रिगर करता है। |
| onbeforeprint ऑनविफोरप्रिंट | script | Triggers before the document is printed. डॉक्यूमेंट प्रिंट होने से पहले ट्रिगर। |
| onblur ऑनब्लर | script | Triggers when the window loses focus. जब विंडो फोकस को लॉस करता है तो ट्रिगर |
| oncanplay ऑनकैनप्ले | script | Triggers when media can start play, but might has to stop for buffering. ट्रिगर जब मीडिया प्ले शुरू कर सकता है, लेकिन बफरिंग के लिए रुकना पड़ सकता है। |
| oncanplaythrough ऑनकैनप्लेथ्रू | script | Triggers when media can be played to the end, without stopping for buffering. ट्रिगर जब मीडिया अंत तक प्ले किया जा सकता है, बफरिंग के लिए बिना रुके। |
| onchange ऑनचेंज | script | Triggers when an element changes. जब कोई एलीमेंट बदलता है तो ट्रिगर। |
| onclick ऑनक्लिक | script | Triggers on a mouse click.एक माउस क्लिक पर ट्रिगर। |
| oncontextmenu ऑनकंटेक्स्टमेन्यू | script | Triggers when a context menu is triggered. एक कंटेक्स्ट मेन्यू ट्रिगर होने पर ट्रिगर। |
| ondblclick ऑनडबलक्लिक | script | Triggers on a mouse double-click माउस डबल--क्लिक पर ट्रिगर। |
| ondrag ऑनड्रैग | script | Triggers when an element is dragged. किसी एलिमेंट को खींचने पर ट्रिगर। |
| ondragend ऑनड्रैगइंड | script | Triggers at the end of a drag operation. एक ड्रैग ऑपरेशन के अंत में ट्रिगर। |
| ondragenter ऑनड्रैगइंटर | script | Triggers when an element has been dragged to a valid drop target. जब कोई एलिमेंट किसी वैध ड्रॉप टार्गेट पर खींचा जाता है तो ट्रिगर। |
| ondragleave ऑनड्रैगलीव | script | Triggers when an element is being dragged over a valid drop target. जब कोई एलिमेंट वैध ड्रॉप टार्गेट पर खींचा जा रहा है तो ट्रिगर। |
| ondragover ऑनड्रैगओवर | script | Triggers at the start of a drag operation. ड्रैग ऑपरेशन की शुरुआत में ट्रिगर। |
| ondragstart ऑनड्रैगस्टार्ट | script | Triggers at the start of a drag operation. ड्रैग ऑपरेशन की शुरुआत में ट्रिगर। |
| ondrop ऑनड्रॉप | script | Triggers when dragged element is being dropped. ड्रैग एलिमेंट को जब ड्रॉप करेंगे तो ट्रिगर। |
| ondurationchange ऑनड्यूरेशनचेंज | script | Triggers when the length of the media is changed. मीडिया की लंबाई बदलने पर ट्रिगर। |
| onemptied ऑनइम्पटीड | script | Triggers when a media resource element suddenly becomes empty. ट्रिगर जब एक मीडिया रिसोर्स एलीमेंट अचानक खाली हो जाता है। |

| | | |
|---|---|---|
| **onended**<br>ऑनइंडेड | **script** | Triggers when media has reach the end.<br>ट्रिगर जब मीडिया अंत तक पहुँच गया है। |
| **onerror**<br>ऑनएरर | **script** | Triggers when an error occur.<br>एरर होने पर ट्रिगर। |
| **onfocus**<br>परफोकस | **script** | Triggers when the window gets focus.<br>विंडो फोकस होने पर ट्रिगर। |
| **onformchange**<br>ऑनफॉर्मचेंज | **script** | Triggers when a form changes<br>जब कोई फार्म बदलता है तो ट्रिगर। |
| **onforminput**<br>ऑनफॉर्मइनपुट | **script** | Triggers when a form gets user input.<br>ट्रिगर जब एक फार्म उपयोगकर्ता इनपुट लेता है। |
| **onhaschange**<br>ऑनहैसचेंज | **script** | Triggers when the document has change.<br>जब डॉक्यूमेंट बदल गया है तो ट्रिगर। |
| **oninput**<br>ऑनइनपुट | **script** | Triggers when an element gets user input.<br>जब किसी एलिमेंट को उपयोगकर्ता का इनपुट मिलता है तो ट्रिगर। |
| **oninvalid**<br>ऑनइनवैलिड | **script** | Triggers when an element is invalid.<br>किसी एलिमेंट के अमान्य होने पर ट्रिगर। |
| **onkeydown**<br>ऑनकीडाउन | **script** | Triggers when a key is pressed.<br>ट्रिगर जब एक key दबाया जाता है। |
| **onkeypress**<br>ऑनकीप्रेस | **script** | Triggers when a key is pressed and released.<br>ट्रिगर जब एक key दबाया और जारी किया जाता है। |
| **onkeyup**<br>ऑनकीअप | **script** | Triggers when a key is released.<br>एक key जारी होने पर ट्रिगर। |
| **onload**<br>ऑनलोड | **script** | Triggers when the document loads.<br>डॉक्यूमेंट लोड होने पर ट्रिगर। |
| **onloadeddata**<br>ऑनलोडेडडेटा | **script** | Triggers when media data is loaded.<br>मीडिया डेटा लोड होने पर ट्रिगर। |
| **onloadedmetadata**<br>ऑनलोडेडमेटाडेटा | **script** | Triggers when the duration and other media data of a media element is loaded.<br>ट्रिगर जब एक मीडिया एलिमेंट की अवधि और अन्य मीडिया डेटा लोड किया जाता है। |
| **onloadstart**<br>ऑनलोडस्टार्ट | **script** | Triggers when the browser starts to load the media data.<br>ट्रिगर जब ब्राउजर मीडिया डेटा लोड करना शुरू करता है। |
| **onmessage**<br>ऑनमैसेज | **script** | Triggers when the message is triggered.<br>मैसेज ट्रिगर होने पर ट्रिगर। |
| **onmousedown**<br>ऑनमाउसडाउन | **script** | Triggers when a mouse button is pressed.<br>ट्रिगर जब एक माउस बटन दबाया जाता है। |
| **onmousemove**<br>ऑनमाउसमूव | **script** | Triggers when the mouse pointer moves.<br>जब माउस पॉइंटर चलता है तो ट्रिगर। |
| **onmouseout**<br>ऑनमाउसआउट | **script** | Triggers when the mouse pointer moves out of an element.<br>ट्रिगर जब माउस पॉइंटर एक एलीमेंट से बाहर निकलता है। |
| **onmouseover**<br>ऑनमाउसओवर | **script** | Triggers when the mouse pointer moves over an element.<br>जब माउस पॉइंटर किसी एलीमेंट पर चलता है तो ट्रिगर। |
| **onmouseup**<br>ऑनमाउसअप | **script** | Triggers when a mouse button is released.<br>माउस बटन छोड़े जाने पर ट्रिगर। |
| **onmousewheel**<br>ऑनमाउसव्हील | **script** | Triggers when the mouse wheel is being rotated.<br>ट्रिगर जब माउस व्हील घुमाया जाता है। |
| **onoffline**<br>ऑनऑफलाईन | **script** | Triggers when the document goes offline.<br>डॉक्यूमेंट ऑफलाइन होने पर ट्रिगर। |
| **ononline**<br>ऑनऑनलाईन | **script** | Triggers when the document comes online.<br>डॉक्यूमेंट ऑनलाइन आने पर ट्रिगर। |

| | | |
|---|---|---|
| onpagehide<br>ऑनपेजहाईड | script | Triggers when the window is hidden.<br>जब विन्डो हिडेन हो तो ट्रिगर। |
| onpageshow<br>ऑनपेजशो | script | Triggers when the window becomes visible.<br>विंडो दिखाई देने पर ट्रिगर। |
| onpause<br>ऑनपॉज | script | Triggers when media data is paused.<br>मीडिया डेटा रोक दिए जाने पर ट्रिगर। |
| onplay<br>ऑनप्ले | script | Triggers when media data is going to start playing.<br>ट्रिगर जब मीडिया डेटा प्ले करना शुरू कर रहा है। |
| onplaying<br>ऑनप्लेईंग | script | Triggers when media data has start playing.<br>ट्रिगर जब मीडिया डेटा प्ले शुरू हो गया हो। |
| onpopstate<br>ऑनपॉपस्टेट | script | Triggers when the window's history changes.<br>जब विंडो का हिस्ट्री बदलता है तो ट्रिगर। |
| onprogress<br>ऑनप्रोग्रेस | script | Triggers when the browser is fetching the media data.<br>ट्रिगर जब ब्राउजर मीडिया डेटा प्राप्त कर रहा है। |
| onratechange<br>ऑनरेटचेंज | script | Triggers when the media data's playing rate has changed.<br>ट्रिगर जब मीडिया डेटा की प्लेइंग रेट बदल गई हो। |
| onreadystatechange<br>ऑनरेडीस्टेटचेंज | script | Triggers when the ready-state changes.<br>जब तैयार स्थिति बदल जाती है तो ट्रिगर। |
| onredo<br>ऑनरीडू | script | Triggers when the document performs a redo.<br>जब डॉक्यूमेंट redo होता है तो ट्रिगर। |
| onresize<br>ऑनरीसाइज | script | Triggers when the window is resized.<br>विंडो को रिसाईज करने पर ट्रिगर। |
| onscroll<br>ऑनस्क्रॉल | script | Triggers when an element's scrollbar is being scrolled.<br>जब किसी एलिमेंट के स्क्रॉलबार को स्क्रॉल किया जा रहा हो तो ट्रिगर। |
| onseeked<br>ऑनसीक्ड | script | Triggers when a media element's seeking attribute is no longer true, and the seeking has ended.<br>ट्रिगर जब एक मीडिया एलीमेंट का seeking ऐट्रीब्यूट लंबे समय के लिए सत्य नही होता और seeking का अन्त हो गया हो। |
| onseeking<br>ऑनसीकिंग | script | Triggers when a media element's seeking attribute is true, and the seeking has begun.<br>ट्रिगर जब एक मीडिया एलीमेंट का seeking ऐट्रीब्यूट सत्य होता है, और seeking (मांग) शुरू हो गई है। |
| onselect<br>ऑनसेलेक्ट | script | Triggers when an element is selected.<br>जब किसी एलीमेंट का चयन किया जाता है तो ट्रिगर। |
| onstalled<br>ऑनस्टाल्ड | script | Triggers when there is an error in fetching media data.<br>मीडिया डेटा प्राप्त करने में त्रुटि होने पर ट्रिगर। |
| onstorage<br>ऑनस्टोरेज | script | Triggers when a document loads.<br>डॉक्यूमेंट लोड होने पर ट्रिगर। |
| onsubmit<br>ऑनसबमिट | script | Triggers when a form is submitted.<br>फॉर्म सबमिट करने पर ट्रिगर। |
| onsuspend<br>ऑनसस्पेंड | script | Triggers when the browser has been fetching media data, but stopped before the entire media file was fetched.<br>ट्रिगर जब ब्राउजर मीडिया डेटा प्राप्त कर रहा है, लेकिन पूरी मीडिया फाइल को लाने से पहले रोक दिया गया था। |
| ontimeupdate<br>ऑनटाईपअपडेट | script | Triggers when media changes its playing position.<br>जब मीडिया अपनी प्लेइंग पोजीशन बदलता है तो ट्रिगर करता है। |
| onundo<br>ऑनअन्डू | script | Triggers when a document performs an undo.<br>जब कोई डॉक्यूमेंट अनडू (पूर्ववत) परफार्म करता है तो ट्रिगर। |
| onunload<br>ऑनअनलोड | script | Triggers when the user leaves the document.<br>उपयोगकर्ता द्वारा डॉक्यूमेंट छोड़ने पर ट्रिगर। |
| onvolumechange | script | Triggers when media changes the volume, also when volume is |

| | | |
|---|---|---|
| ऑनवाल्यूमचेंज | | set to "mute".<br>जब मीडिया वॉल्यूम बदलता है, तब भी जब वॉल्यूम "म्यूट" पर सेट होता है। |
| **onwaiting**<br>ऑनवेटिंग | **script** | Triggers when media has stopped playing, but is expected to resume. ट्रिगर जब मीडिया ने प्ले बंद कर दिया है, लेकिन फिर से शुरू होने की उम्मीद है। |

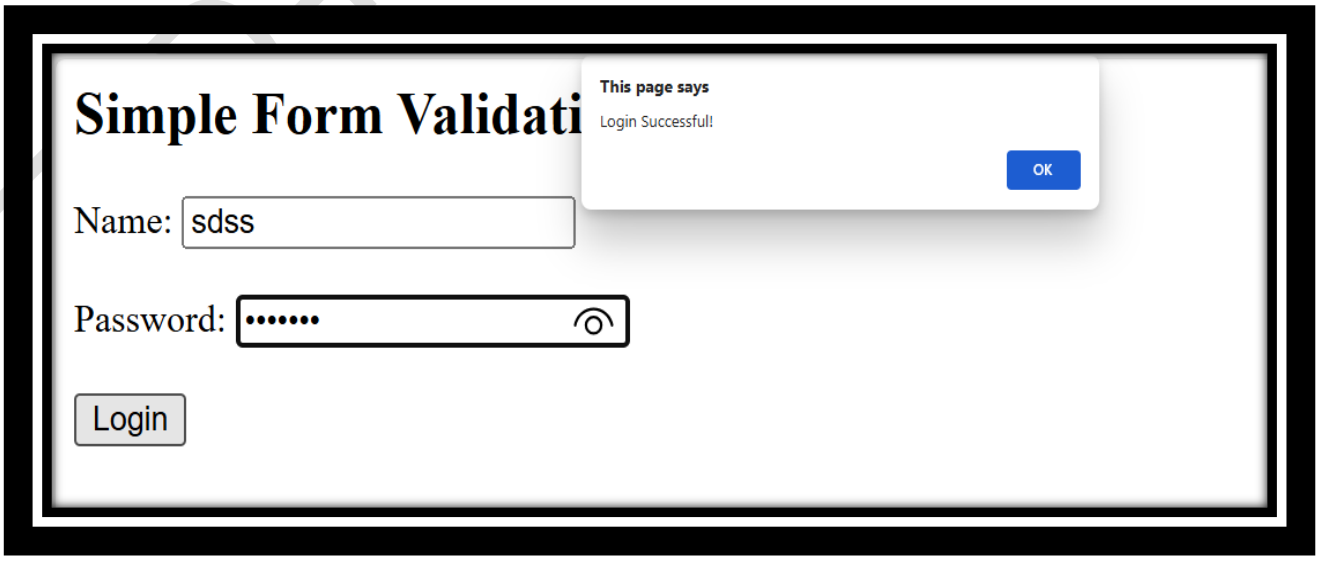
# 6.7 Basic Form Validation in JavaScript

Used to check form fields before submitting.

## Example:

## Form Validation Example

```html
<html>
<head>
    <title>Simple Form Validation</title>
</head>
<body>
    <h2>Simple Form Validation Example</h2>

    <form onsubmit="return checkData()">
        Name: <input type="text" id="name"><br><br>
        Password: <input type="password" id="pass"><br><br>
        <input type="submit">
    </form>

    <script>
        function checkData() {
            let name = document.getElementById("name").value;
            let pass = document.getElementById("pass").value;

            if (name == "" || pass == "") {
                alert("All fields are required!");
                return false;
            } else {
                alert("Login Successful!");
                return true;
            }
        }
    </script>

</body>
</html>
```



## Explanation (in short):

- onsubmit="return checkData()" → calls function when you click submit.
- document.getElementById() → gets input value.
- if (name == "" || pass == "") → checks if any field is empty.
- alert() → shows message to user.
- return false → stops form if empty.Top of Form Bottom of Form

# 6.8 Angular JS

## Definition

AngularJS is an *open-source JavaScript framework* developed by Google.
It helps to create dynamic single-page web applications (SPAs).

- **AngularJS** is a **JavaScript framework** developed by **Google**.
- It is used to **create dynamic, interactive, and single-page web applications (SPA)**.
- It allows developers to **extend HTML** with new attributes called **Directives**.
- It supports **two-way data binding**, which means —
  if you change the data in the program, it will automatically update in the webpage (and vice versa).

## Key Points:

- Developed by **Google** in **2010**
- It is a **client-side scripting framework**
- Based on **JavaScript and MVC pattern**
- Uses **Directives** (like ng-model, ng-app, ng-repeat)
- Provides **Two-way Data Binding**
- Helps build **Single Page Applications (SPAs)**

## Example (Simple AngularJS Program)

- **ng-app** → starts AngularJS application.

ZED-KING INSTITUTE

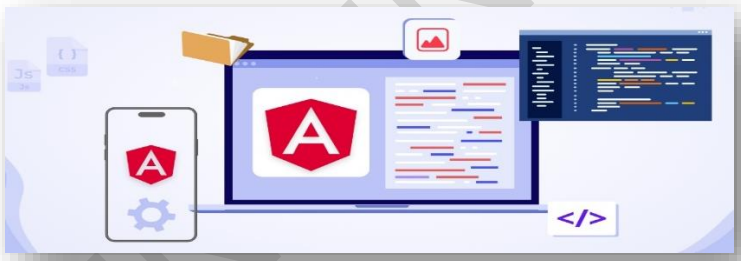
- **ng-model="name"** → binds input box with variable.
- **{{name}}** → displays the same text dynamically (data binding).

## 6.8.1 Advantages of Angular JS

1. Open source and free.
2. Two-way data binding.
3. Reusable components.
4. MVC (Model-View-Controller) support.
5. Easy testing and maintenance.

## 6.8.2 Disadvantages of Angular JS

1. Not suitable for very large applications.
2. Slower on older browsers.
3. Complex for beginners.



## 6.8.3 How to Write First Angular JS Program

### Step 1: Add AngularJS library

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
```

### Step 2: Create Angular app

```
<div ng-app="" ng-init="name='Doli'">
  <p>Hello {{name}}</p>
</div>
```

### Output:

ZED-KIDS INSTITUTE

# 6.8.3 How to Write First AngularJS Program

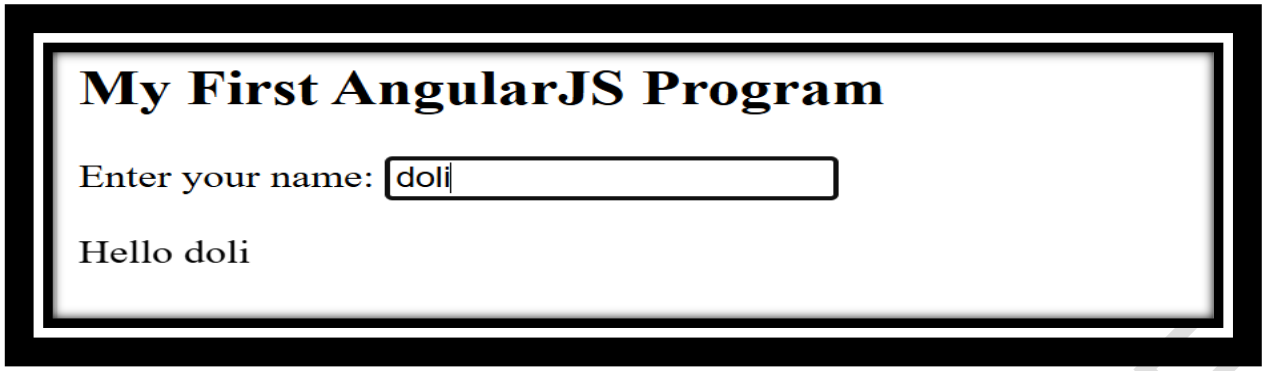
## Steps to Write First AngularJS Program:

- **Step 1** → Create a new HTML file (e.g., first.html)
- **Step 2** → Include AngularJS library using <script> tag
- **<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>**
- **Step 3** → Use ng-app directive to start AngularJS application
- **Step 4** → Use ng-model directive to bind data
- **Step 5** → Use {{ }} expression to display data
- **Step 6** → Save and open the file in a browser

## Example: First AngularJS Program

```
<html ng-app>
<head>
  <title>First AngularJS Program</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
  <h2>My First AngularJS Program</h2>

  <p>Enter your name: <input type="text" ng-model="name"></p>
  <p>Hello {{name}}</p>

</body>
</html>
```

KING INSTITUTE




## Explanation:

- **ng-app** → Defines the AngularJS application
- **ng-model** → Binds input data with variable name
- **{{name}}** → Displays the value entered by the user instantly (two-way data binding)

# 6.8.4 Concepts of Angular JS

## Main Concepts of AngularJS:

### 1. ng-app

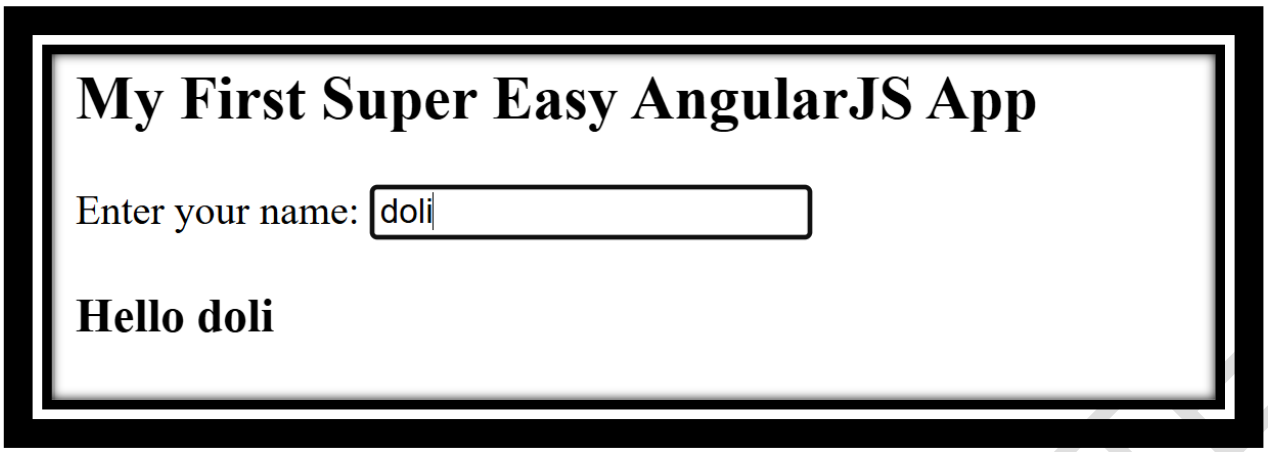
- It defines the **root element** of an AngularJS application.
- It tells the browser that this part of HTML will be controlled by AngularJS.

## Example:

### <html ng-app>

```
<html>
<head>
    <title>Easy AngularJS Example</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-app="">
    <h2>My First Super Easy AngularJS App</h2>
    Enter your name: <input type="text" ng-model="myName">
    <h3>Hello {{ myName }}</h3>
</body>
</html>
```

TED-KING INSTITUTE




## Explanation (Very Simple)

### 1. ng-app

This tells the browser:

"Start AngularJS on this page."

### 2. ng-model="myName"

This stores whatever the user types into a variable called **myName**.

### 3. {{ myName }}

This displays the value of **myName** on the screen.

## 2. ng-model

- It binds the **data** between HTML input fields and the application (two-way data binding).

**Example:**

```
<input type="text" ng-model="name">
```

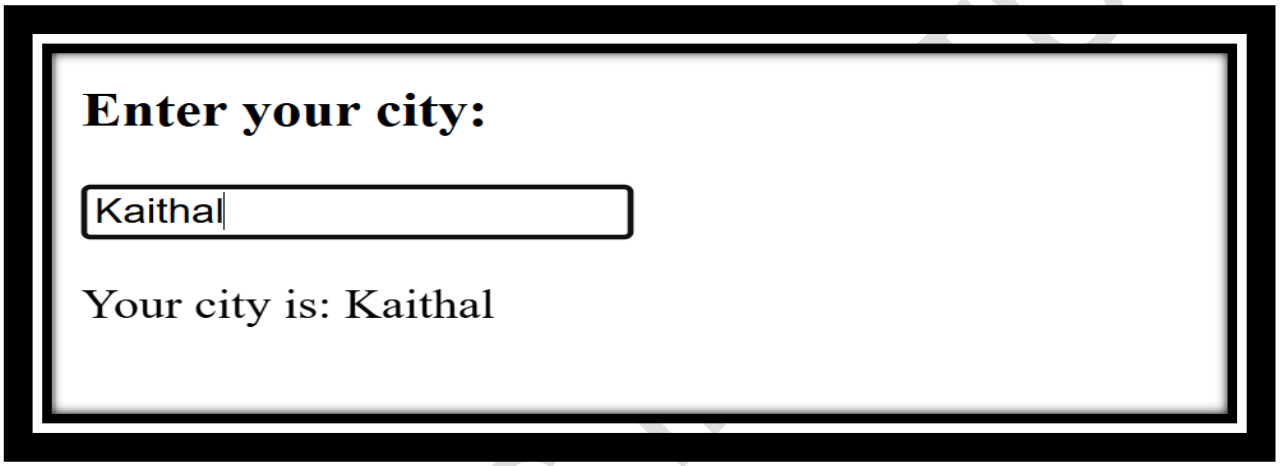
## 3. ng-bind

**Example:**

```
<p ng-bind="name"></p>
```

ZED-KING INSTITUTE

```html
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-app="">
    <h3>Enter your city:</h3>
    <input type="text" ng-model="city">
    <p>Your city is: <span ng-bind="city"></span></p>
</body>
</html>
```




## 4. ng-init

- It is used to **initialize variables** in AngularJS.

**Example:**

**<body ng-init="x=10; y=20">**

```html
<html ng-app="">
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-init="x=10; y=20">
  <h3>X value: {{x}}</h3>
  <h3>Y value: {{y}}</h3>
  <h3>Total: {{x + y}}</h3>
</body>
</html>
```

X value: 10

Y value: 20

Total: 30

## 5. ng-repeat

- It repeats HTML elements for **each item in a list or array**.

  **Example:**

**<li ng-repeat="item in fruits">{{item}}</li>**

```
<html ng-app="">
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-init="fruits=['Apple','Banana','Mango']">
  <h3>Fruits:</h3>
  <p ng-repeat="f in fruits">{{f}}</p>
</body>
</html>
```

**Fruits:**

Apple

Banana

Mango

## 6. Expressions ({{ }})

- Used to **display values or perform calculations** inside HTML.

  **Example:**

** Sum = {{5 + 5}} **

```
<html ng-app="">
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
  <p>Sum = {{5 + 5}}</p>
  <p>Multiply = {{4 * 3}}</p>
  <p>Subtraction = {{20 - 7}}</p>
</body>
</html>
```

Sum = 10

Multiply = 12

Subtraction = 13

## 7. Directives

- AngularJS uses directives (starting with ng-) to **extend HTML functionality**.

  **Examples:**

  - ng-app
  - ng-model
  - ng-bind

- ng-repeat

## 8. Two-Way Data Binding

- When the data in the model changes, the view (HTML) updates automatically and vice versa.

## 9. Controllers

- Controllers are **JavaScript functions** used to control data in AngularJS applications.

  **Example:**

```
<div ng-app="myApp" ng-controller="myCtrl">
 {{ message }}
</div>
```

## 10. Filters

- Filters are used to **format data** (like uppercase, lowercase, currency, etc.)

**Example:**

```
{{ name | uppercase }}
```

ZED-KING INSTITUTE

| Concept | Description | Example |
|---|---|---|
| **Module** | Main container | angular.module('myApp', []); |
| **Directive** | Add extra HTML power | ng-model, ng-bind |
| **Expression** | Show data | {{name}} |
| **Controller** | Handle logic | ng-controller="myCtrl" |
| **Scope** | Store app data | $scope.name="Doli"; |

**Summary:**

- JavaScript → scripting language
- AngularJS → JavaScript framework
- Both are used for creating **interactive and dynamic webpages**

ZED-KING INSTITUTE

