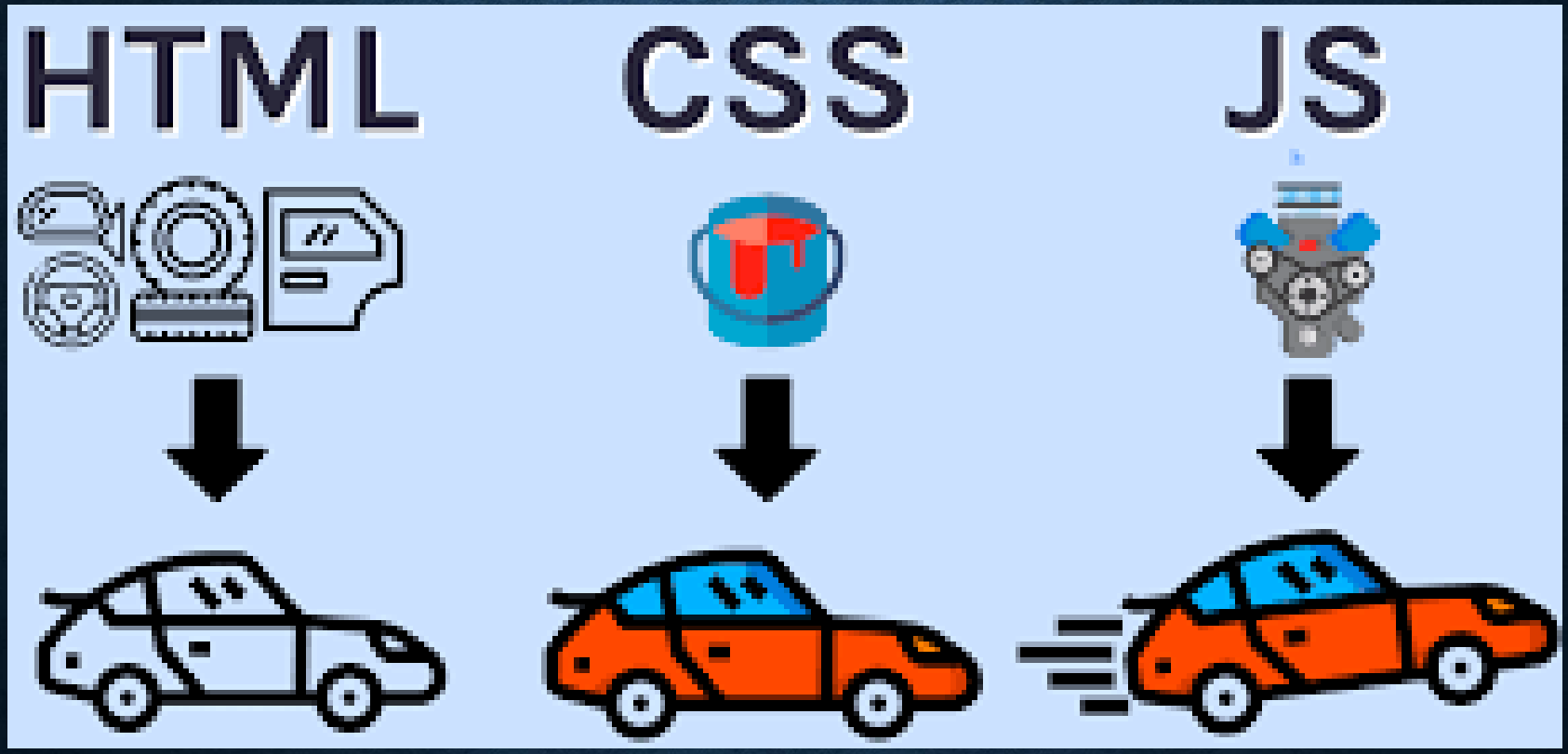
WEB TECHNOLOGY
LECTURE – 06
(JAVASCRIPT)

WHAT IS JAVASCRIPT?



INTRO TO JAVASCRIPT

- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- JavaScript was designed to add interactivity to HTML pages

- Where to put JS?

1. Internal: directly into HTML document using <script> tag.
   - can be placed in the <head> section.
   - can be placed in the <body> section.

2. External: an external (.js) file. (recommended)

```
<html>
  <body>
    <script src="app.js"></script>
  </body>
</html>
```

JS IN <HEAD>

```
<html>
  <head>
   <script type="text/javascript">
     alert('Hello World!!');
   </script>
   </head>
   <body>
   <h1>My Web Page</h1>
   <p id="demo">A Paragraph</p>
   </body>
</html>
```

JS IN <BODY>

```
<html>
  <body>
        <h1>My Web Page</h1>
        <p id="demo">A
  Paragraph</p>
        <script
  type="text/javascript">
          alert('Hello World!!');
        </script>
  </body>
</html>
```

DATA TYPES

- JavaScript has two categories' data types.

1. Primitive Data Types (Single value)
   I. String – Text. E.g, "Alice"
   II. Number – Numeric values. E.g: 123
   III. Boolean – True/False
   IV. Undefined – Variable declared but not assigned
   V. Null – Empty value
   VI. Symbol – Unique identifiers
   VII. BigInt – Large numbers beyond Number limits
2. Non-Primitive Data Types (Reference Types)
   I. Object – Key-value pairs
   II. Array – Ordered list of values
   III. Function – Code that can be reused

DISPLAY OUTPUT

You can display output in multiple ways.

1. console.log(): in developer console

Example: console.log("Hello, JavaScript!");

Open **Developer Tools (F12)** → **Console** in your browser to see the output.

2. alert(): Popup Message

Example: alert("Welcome to JavaScript!");

Use it (alert) for simple notifications but avoid excessive use, as it interrupts user experience.

DISPLAY OUTPUT (CONT.)

3. document.write(): directly writes to the whole page

Example: document.write("<h2>Hello, JavaScript!</h2>");

Warning: It overwrites the entire page content if used after page load.

4. innerHTML (Modify specific HTML element)

```
<p id="output"></p>

<script>

document.getElementById("output").innerHTML = "Hello, JavaScript!";

</script>
```

- There is another property called innerText which is used to set content as nonHTML element.

USER INPUT

1. Using prompt(): The prompt() function displays a popup box where users can enter a value.

```
let name = prompt("What is your name?");
console.log("Hello, " + name);
```

prompt() always returns a string, even if the user enters a number.

2. Using confirm() (For Yes/No Input): The confirm() function displays a popup with OK (true) / Cancel (false) options.

```
let isAdult = confirm("Are you 18 or older?");
```

3. Using HTML Input Fields (<input>)

```
<input type="text" id="userInput" placeholder="Enter something">
<button onclick="getInput()">Submit</button>
<script> function getInput() {
    let input = document.getElementById("userInput").value;
    console.log("User entered:", input); } </script>
```

DECLARING A VARIABLE

- JavaScript variables are used to hold values or expressions.

- You do not need to specify data type to declare a variable, you must use var, or let, or const instead.

1. var name = "John";
   - Can be re-declared and updated.
   - Has function scope (not block scope).
2. let age = 25;
   - Can be updated but not re-declared.
   - Has block scope.
   - E.g: { let x = 5; }

     console.log(x); // will cause error
3. const PI = 3.1416;
   - Cannot be changed or re-declared.
   - Must be initialized when declared.

TESTING DATA TYPE

you can check the data type of a variable using the typeof operator.

Example:

```
let name = "Alice";
let age = 25;
let isStudent = true;
let car;
let emptyValue = null;
```

- console.log(typeof name); // Output: "string"
- console.log(typeof age); // Output: "number"
- console.log(typeof isStudent); // Output: "boolean"
- console.log(typeof car); // Output: "undefined"
- console.log({name: "Alice"}); // Output: “object"
- console.log(typeof emptyValue); // Output: "object“ (js quirk)
- console.log(typeof Symbol(“any")); // Output: symbol

TESTING DATA TYPE (CONT.)

- console.log(typeof [1, 2, 3]); // Output: "object" (Arrays are objects)

let fun = function myfun() {alert("Hello World")};

- Console.log(typeof fun) // function

let fun2 = function myfun() {return("Hello World")};

- Console.log(typeof fun2()) // string

THE "=" SIGN

1. Assignment Operator (=): is used to assign a value to a variable.

```
let x = 10; // Assigns 10 to x

let name = "Alice"; // Assigns "Alice" to name
```

2. Loose Equality (==): compares value only

```
console.log(5 == 5);     // true

console.log(5 == "5");  // true ("5" is converted to number)

console.log(true == 1);  // true (true is converted to 1)
```

3. Strict Equality (===): compares both the type and value

```
console.log(5 === "5");   // false

console.log(5 === 5);     // true

console.log(true === 1);   // false
```

SPECIAL CHECKING

1. Checking for Arrays

```
let colors = ["red", "green", "blue"];
console.log(Array.isArray(colors)); // Output: true
```

2. Checking null and undefined Properly

```
console.log(null == undefined);    // Output: true
```

Therefore, to differentiate them use strict equality check (===)

3. Object content equality check

```
let arr1 = [1,2,3];

let arr2 = [1,2,3];

console.log(arr1==arr2);   // Output: false

console.log(arr1===arr2); // Output: false
```

This happens as arrays and objects are reference type. Use following instead

```
console.log(JSON.stringify(value) === JSON.stringify(value2));
```

THE "+" OPERATOR

1. number + number = number
2. String + String = String (Concatenation):

txt1="What a very "; txt2="nice day"; txt3=txt1+txt2;	Result: What a very nice day

3. number + string = string + string = string

- **Examples:** (Evaluate from left to right)
  - 5 + 5 = 10
  - "5"+5 ="55"
  - 16 + 4 + "Volvo" = "20Volvo"
  - "Volvo" +16 + 4 = "Volvo16"+4 = 'Volvo164'

OPERATORS

Arithmetic Operators: Consider y=5

Operator	Description	Example	Result of x	Result of y
+	Addition	x=y+2	7	5
-	Subtraction	x=y-2	3	5
*	Multiplication	x=y*2	10	5
/	Division	x=y/2	2.5	5
%	Modulus	x=y%2	1	5
++	Increment	x=++y	6	6
		x=y++	5	6
--	Decrement	x=--y	4	4
		x=y--	5	4

Assignment Operators: Consider x=10 and y=5

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

OPERATORS (CONT.)

Comparison operators: Consider x=5

Operator	Description	Example
==	is equal to	x==8 is false x==5 is true
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

Logical operators: Consider x=6 and y=3

Operator	Description	Example
&&	And	(x < 10 && y > 1) is true
||	Or	(x==5 || y==5) is false
!	Not	!(x==y) is true

FOR…IN LOOP

- The JavaScript for in statement loops through the properties of an Object:

Example:

```
const person = {fname:"John", lname:"Doe", age:25};
for (let x in person) {
        console.log(x)
        console.log(person[x])
  }
```

- Output?

JAVASCRIPT FOR...OF LOOP

- Works on iterable data structures such as Arrays, Strings, etc.

Example:

```
const person = ["BMW", "Volvo", "Mini"];
let text = "";
for (let x of person) {
        text += x + " ";
}
Console.log(text);
```

- Output: BMW Volvo Mini

CONDITION

- **Conditional Statement Syntax:**

```
if (condition) {
        code to be executed if condition is true
} else {
        code to be executed if condition is not true
}
```

- The ternary/conditional operator syntax:

```
let res =(condition)?value1:value2
```

Explanation: res will be "value1" if the condition is true, value2 otherwise.

PLAIN FOR LOOP

```
for (var i=0; i<=5; i++) {
    console.log(i);
}
```

WHILE LOOP

```
var i=0;
while (i<=5) {
        document.write("The
   number is " + i);
        document.write("<br />");
        i++;
}
```

Also available:

- The break and continue statements.
- The Do while loop as you have used in C or java programming

END