UI WEB DEVELOPMENT

(MR23-1CS0102)

MALLA REDDY UNIVERSITY

UNIT-III

JavaScript-1

A Brief History of JavaScript

- JavaScript is a object-based scripting language and it is light weighted. It is first implemented by **Netscape** (with help from Sun Microsystems). JavaScript was created by **Brendan Eich** at Netscape in **1995** for the purpose of allowing code in web-pages (performing logical operation on client side).
- Using HTML we can only design a web page but you can not run any logic on web browser like addition of two numbers, check any condition, looping statements(for, while), decision making statement(if-else) etc.
- All these are not possible using HTML so for perform all these task we use JavaScript.
- Using HTML we can only design a web page if we want to run any programs like c programming we use JavaScript. Suppose we want to print sum of two number then we use JavaScript for coding.

Introduction to JavaScript

❑**JavaScript** is a object-based scripting language. It is light weighted. Using HTML we can only design a web page but you can not run any logic on web browser like addition of two numbers, check any condition, looping statements (for, while), decision making statement (if-else) at client side.

❑All these are not possible using HTML So for perform all these task at client side you need to use JavaScript.

JavaScript Comments –

❑JavaScript comments can be used to **explain** JavaScript code, and to make it more **readable**. JavaScript comments can also be used to **prevent execution**, when **testing** alternative code.

Single Line Comments –

- Single line comments start with **//**.

Any text between **//** and the end of the line will be **ignored** by JavaScript (will not be executed).

Example –

- In the example below, the first line – ***"Hello World 1"*** will not be executed.

```
//document.write("<h1>Hello World 1</h1>")
document.write("<h1>Hello World 2</h1>")
```

Multi-line Comments –

- Multi-line comments start with /* and end with */.

Any text between /* and */ will be ignored by JavaScript.

Example –

- In the example below, both the lines will not be executed.

```
/* document.write("<h1>Line 1</h1>"); document.write("Line 2") */
```

Statements in JavaScript –

- A computer program is a list of **"instructions"** to be **"executed"** by a computer. In a programming language, these programming instructions are called **statements**. JavaScript program is a list of programming statements.
- Most JavaScript programs contain many JavaScript statements. The statements are executed, one by one, in the same order as they are written.
- **Semicolons (;)**
- Semicolons separate JavaScript statements. On the web, you might see examples without semicolons. **Ending statements with semicolon is not required, but highly recommended.**

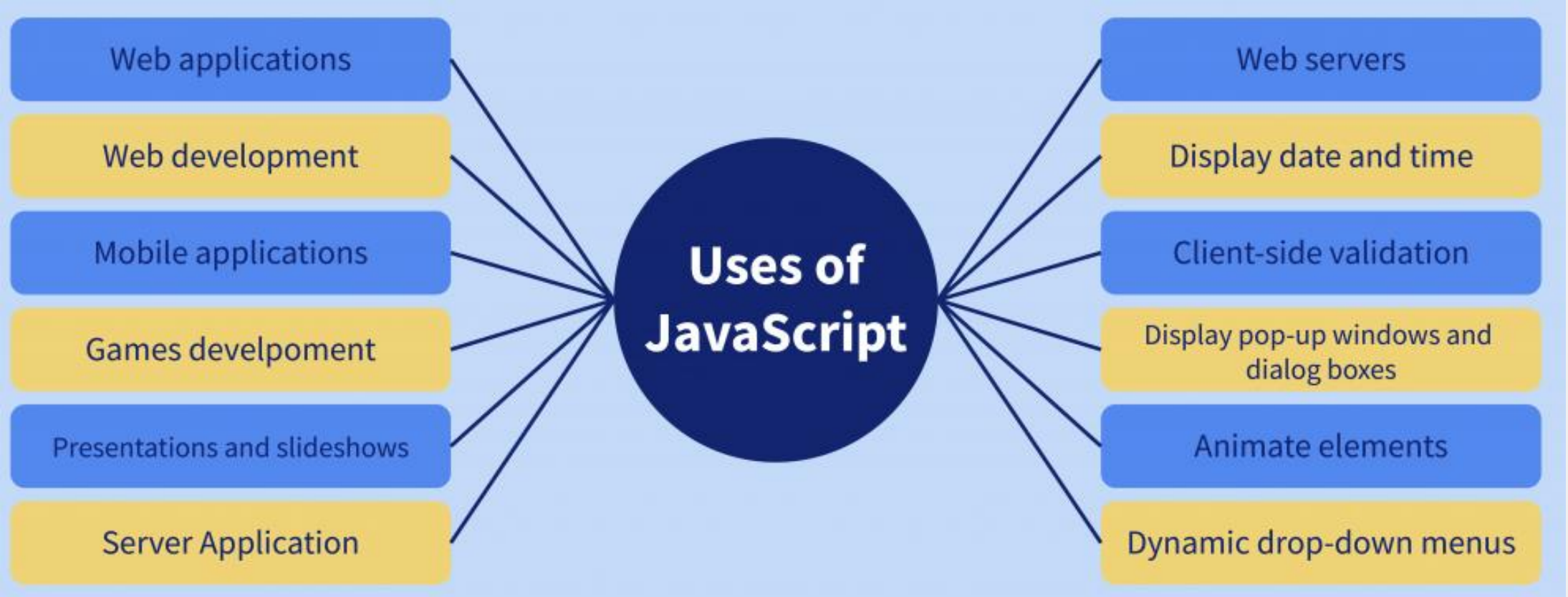
Statements in JavaScript –

```
var a, b, c;      // Declare 3 variables
a = 5;            // Assign the value 5 to a
b = 6;            // Assign the value 6 to b
c = a + b;        // Assign the sum of a and b to c
```

When separated by semicolons, multiple statements on one line are allowed:

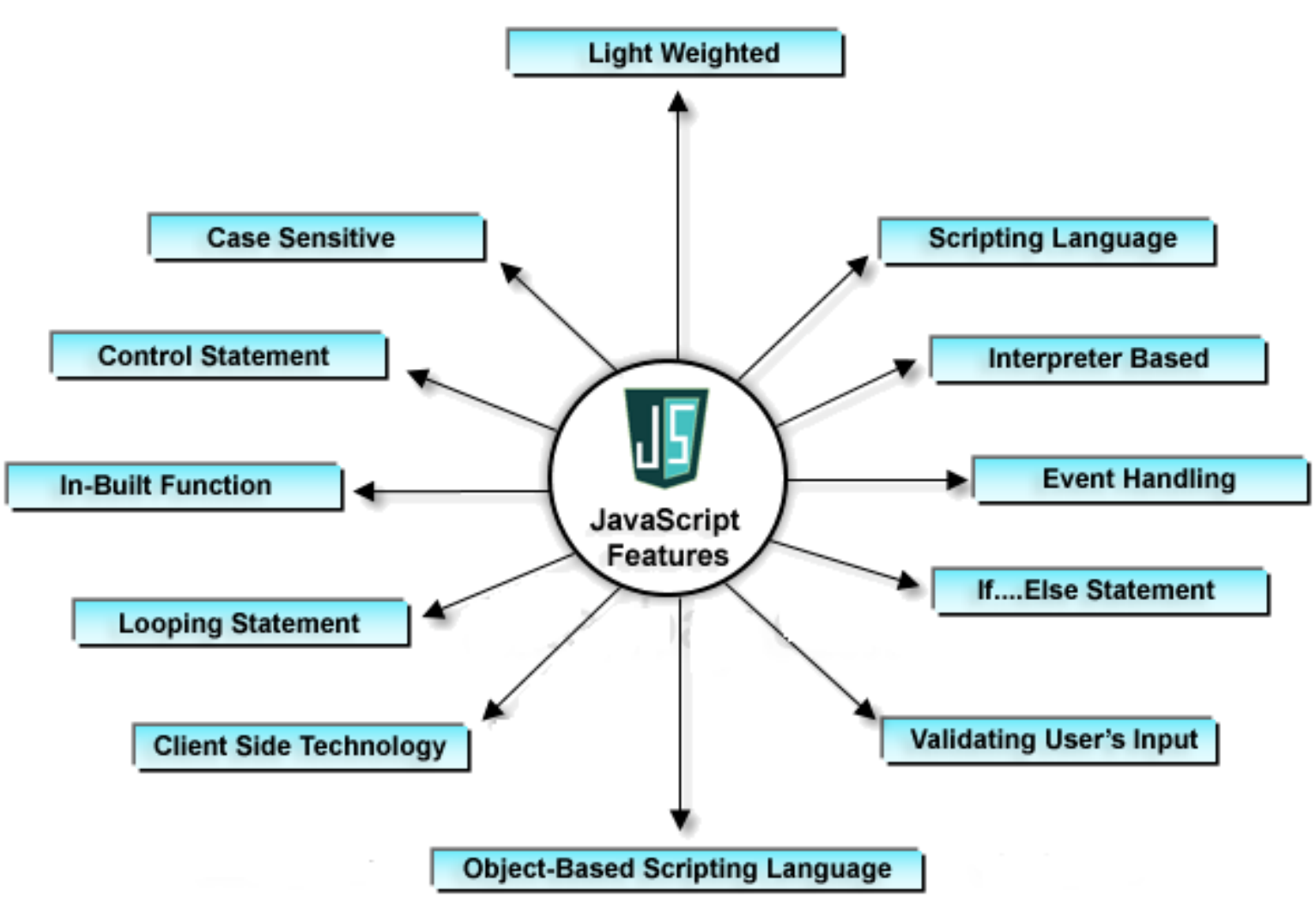
```
a = 5; b = 6; document.write("<h1>Hello World</h1>") ;
```

Applications of JavaScript



Features of JavaScript

❑JavaScript is a client side technology, it is mainly used for gives client side validation, but it have lot of features which are given below;



Evolution of JavaScript

- In 1996, Microsoft released JScript, which was based on LiveScript but had many fundamental differences.
- In November 1996, Netscape submitted JavaScript to ECMA(**European Computer Manufacturer's Association)** International to standardize JavaScript and get it adopted by developers.
- This brought about ECMAScript, which uses most of JavaScript's original syntax and has served as the standard for JavaScript ever since.
- In 1997, ECMA released ECMAScript 1.0, which is now referred to as JavaScript. ECMAScript 2.0 or ES2 was released the following year with minor changes to keep up with the ISO standards for the language.

Evolution of JavaScript (Continued)

- Even though developers were adopting JavaScript worldwide, building websites for Netscape and internet explorer was tedious. This was addressed by ES3, which came out in December 1999, 18 months after ES2.
- ECMAScript 3 features many changes and introduced the language's regular expression and exception handling features that we see today.
- While ECMAScript 4 was abandoned, the successor to ECMAScript 3 called ES5 was released in 2009 with many new features and the ability to pair with JSON files. ECMAScript 6 was released in 2015 but was renamed ECMAScript 2015, and this naming pattern has continued for the latest releases of the JavaScript standard.
- At the time of writing, ECMAScript 2022 is the latest version of JavaScript set to release later this year.

Java vs. JavaScript

JAVA	Java Script
• Compiled	• Interpreted
• Mainly used for back-end	• Mainly used for front-end
• Executed in JVM or in the browser	• Executed in the browser
• Allows better security	• Needs more effort to enhance security
• Static type checking	• Dynamic type checking
• The syntax is similar to C++	• The syntax is similar to C
• Requires Java Development Kit (JDK)	• Can be written in any text editor
• For various apps	• Mainly for web apps

Java vs. JavaScript (Continued)

JAVA	JAVASCRIPT
Java is a general-purpose, compiled language used for server-side programming.	JavaScript is an interpreted, scripting language used for client-side programming.
Java is an independent language executed by using a JVM (Java Virtual Machine).	JavaScript must be executed along with HTML in the web browser.
To code and debug in Java, you need JDK (Java Development Kit) and specific IDEs, such as NetBeans and Eclipse. The extension for Java files is `.java`.	JavaScript can be coded by using any text editor, such as Notepad and Sublime text. The extension for JavaScript files is `.js`.

JavaScript - Syntax

- JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.
- You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.
- The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.
- <script > JavaScript code </script>

The script tag takes two important attributes –

- **Language** – This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type** – This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

So your JavaScript segment will look like –

```
<script language = "javascript" type = "text/javascript"> JavaScript code
</script>
```

JavaScript First Program

❑JavaScript simple example to verify age of any person, if age is greater than 18 show message adult otherwise show under 18.

```
<html>
<head>
<script>
function verify()
{
var no;
no=Number(document.getElementById("age").value);
if(no<18)
{
alert("Under 18");
}
```

```
else
{
alert("You are Adult");
}
}
</script>
</head>
<body> Enter your age:<input id="age"><br />
<button onclick="verify()">Click me</button>
</body>
</html>
```

In JavaScript there are 3 types of popup boxes.

- Alert box
- Confirmation box
- Prompt box

Alert Box in JavaScript

- Alert box is a dialog box which displays a message in a small window with an OK button, it is used, and when we want to make sure the information is coming via user.
- User have to click to **OK** button to proceed.
- Alert box with text and [OK] button
- Just a message shown in a dialog box:

Syntax:

alert("text goes here");

Example to understand JavaScript Alert:

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
alert("Hit me to proceed!");
document.write("Have a Nice day!!!")
</script>
</body>
</html>
```

Output:

Confirmation box in JavaScript

- Contains text, [OK] button and [Cancel] button
- Confirm box is a dialog box which displays a message in a small window with two buttons. One OK and other Cancel.
- It is used when we want the user to confirm/verify something.
- When user clicks **OK** the box returns true else when user clicks **Cancel** the box returns false.

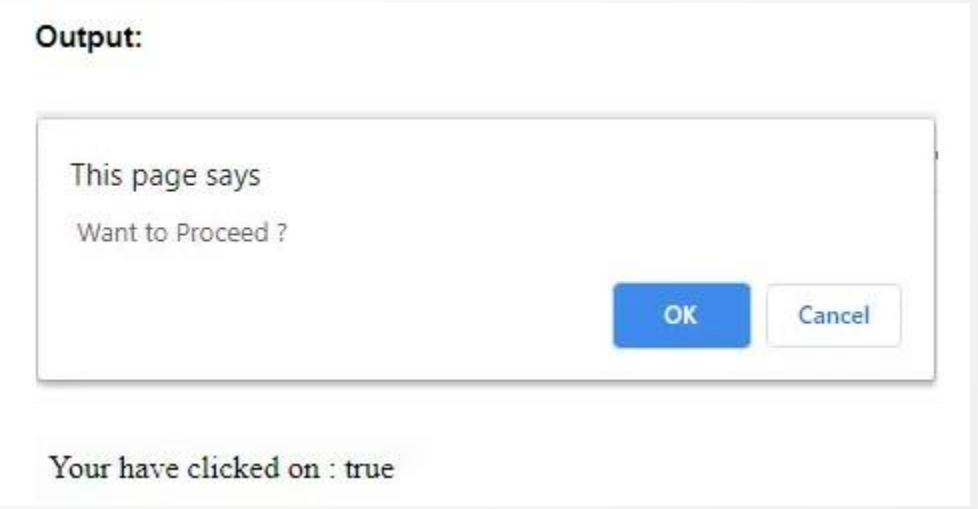
Syntax:

confirm("test goes here");

Example to understand JavaScript Confirmation Box:

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
a = confirm(" Want to Proceed ?");
document.write("Your  have clicked on : " + a);
</script>
</body>
</html>
```

Output:



JavaScript Prompt Box

- Contains text, input field with default value
- Prompt box is a dialog box which displays a message in a small window with a text box along with two buttons. One OK and other Cancel.
- It is used when we want the user to input a value before proceeding ahead.
- When a user clicks **OK** the box returns the input value else when user clicks **Cancel** the box return null

Syntax:

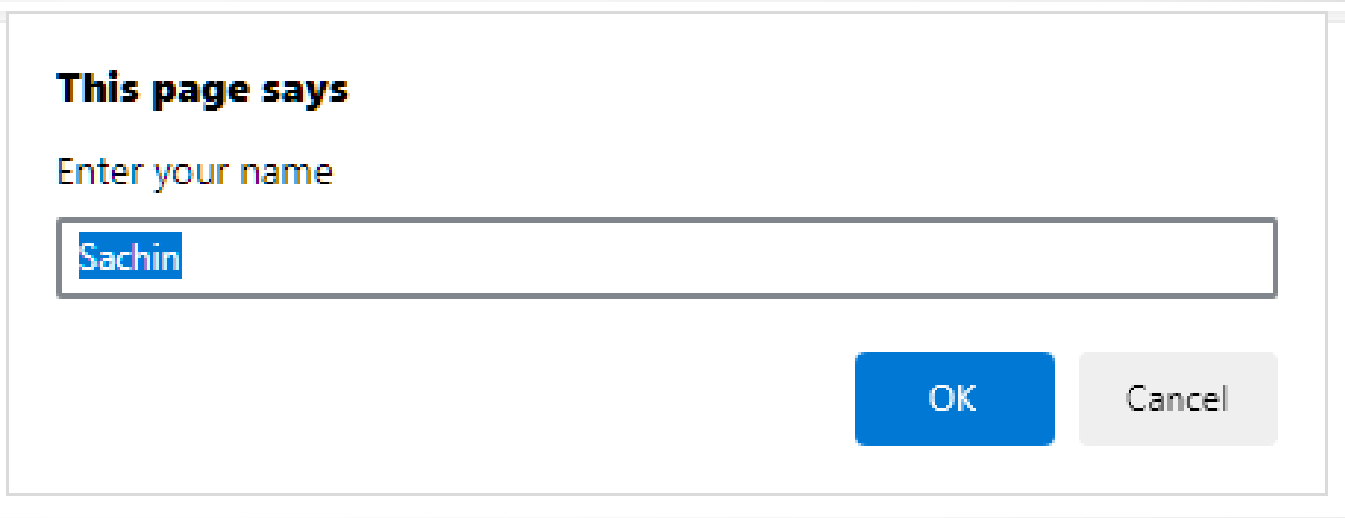
prompt(“text goes here”, “default value”);

Example to understand Prompt Box in JavaScript:

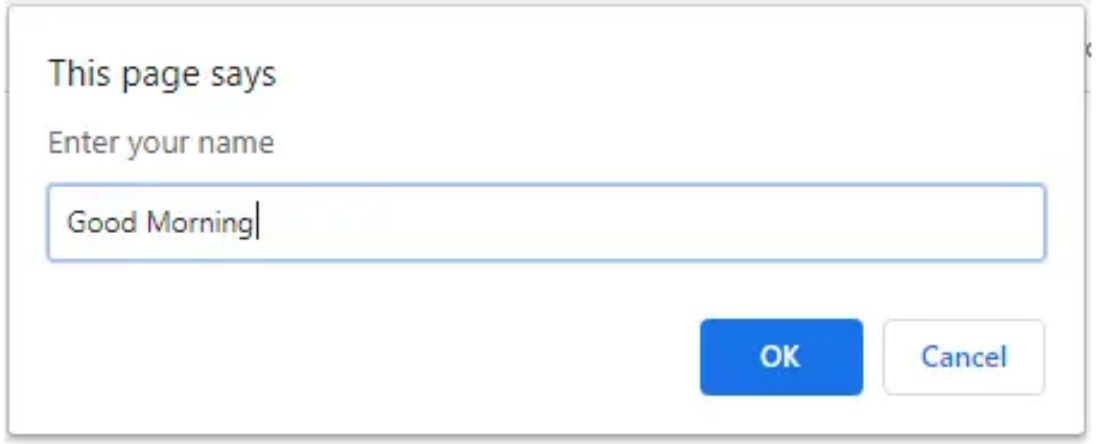
```
<html>
<head>
</head>
<body>
<script type="text/javascript">
a = prompt("Enter your name", “Sachin");//with default value assigned
a = prompt("Enter your name", "");//without default value
document.write("Your have entered : " + a);
</script>
</body>
</html>
```

Output:

With default value:



Without default value:

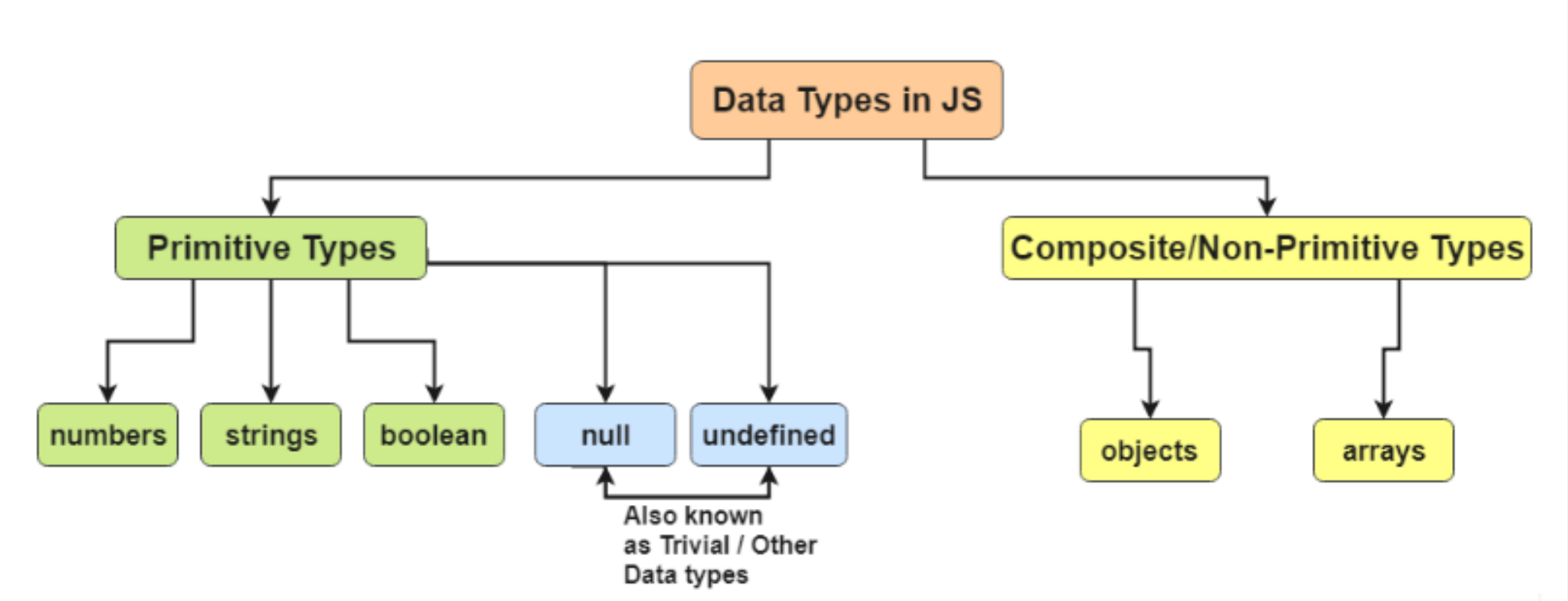


Your have entered : Good Morning

Data Types –

❑Data types are basically **type of data** that can be used and manipulated in a program.

Following are the different Data Types in JavaScript –



- JavaScript provides different data types to hold different types of values. There are two types of data types in JavaScript.(as shown in the diagram above)
- Primitive data type
- Non-primitive (reference) data type

JavaScript Primitive Data Types –

Data Type	Description
String	represents sequence of characters e.g. “hello”
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript Strings –

- A string (or a text string) is a series of characters like “Simple Snippets”. Strings are written with quotes.

```
var carName = "Mercedes";    // Using double quotes
var carName = 'BMW';    // Using single quotes
```

- You can use quotes inside a string, as long as they don’t match the quotes surrounding the string:

```
var answer = "It's alright";              // Single quote inside double quotes
var answer = "He is called 'Mark'";     // Single quotes inside double quotes
var answer = 'He is called "Mark"';     // Double quotes inside single quotes
```

JavaScript Numbers –

- JavaScript has only one type of numbers. Numbers can be written with, or without decimals:

```
var x1 = 34.00;      // Written with decimals
var x2 = 34;         // Written without decimals
```

- Extra large or extra small numbers can be written with scientific (exponential) notation:

```
var y = 123e5;       // 12300000
var z = 123e-5;      // 0.00123
```

JavaScript Booleans –

- Booleans can only have two values: true or false. Booleans are often used in conditional testing.

```
var flag1 = true;
var flag2 = false;
```

JavaScript Non-Primitive/Composite Data Types –

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values

JavaScript Objects

- A JavaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.
- JavaScript is an object-based language. Everything is an object in JavaScript.
- JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Creating Objects in JavaScript

❑There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

1) JavaScript Object by object literal

Syntax

object={property1:value1,property2:value2.....propertyN:valueN}

Example

```
<script>
emp={id:102,name:"Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

Output

102 Kumar 40000

2) By creating instance of Object

Syntax

var objectname=new Object();

Here, **new keyword** is used to create object.

Example

```
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

Output

101 Ravi 50000

3 By using an Object constructor

Syntax:

```
function ObjName(name) {
this.name = name;
}
//Object constructor syntax
var myobj = new ObjName("my object");
```

Example

```
<script>
function person(name, age)
{
this.name = name;
this.age = age;
}
var x=new person("Sachin",45);
document.write(x.name+" "+x.age);
</script>
```

Output

Sachin 45

Object Properties –

- The **name:values** pairs in JavaScript objects are called properties:

Property	Property Value
firstName	Ravi
lastName	Kumar
age	50
eyeColor	blue

Accessing Object Properties –

objectName.propertyName;

Example

person.lastName;

Object Methods –

❑Objects can also have methods. Methods are actions that can be performed on objects.

Property	Property Value
firstName	Ravi
lastName	Kumar
age	50
eyeColor	blue

Accessing Object Methods –

Syntax:

objectName.methodName()

name = person.fullName;

Arrays in JavaScript

❑JavaScript arrays are used to store multiple values in a single variable. **An array in JavaScript can hold different elements** We can store Numbers, Strings and Boolean in a single array. Also arrays in JavaScript are dynamic in nature, which means its size can increase or decrease at run time.

Syntax –

```
var fruits = [ "apple", "orange", "mango" ];
```

Access the Elements of an Array –

- You access an array element by referring to the **index number**.
- This statement accesses the value of the first element in cars:

```
var cars = ["Saab", "Volvo", "BMW"];
var name = cars[0];
```

Access the Full Array –

- With JavaScript, the full array can be accessed by referring to the array name:

```
var cars = ["Saab", "Volvo", "BMW"];
document.write("<h2>"+cars[i]+"</h2>");
```

Creating Variable in JavaScript –

```
var length = 5;    // Number
var lastName = "Simple Snippets";    // String
var flag = true;    // boolean
```

❑In the example above, we created 3 primitive variables and also instantiated them with their individual values.

❑JavaScript unlike C++ or Java Programming has two **unique characteristics** in context with **variables** and **data types** –

1. JavaScript is **LOOSELY/WEAKLY TYPED** programming language.
2. JavaScript is **DYNAMICALLY TYPED** programming language.

1. JavaScript is LOOSELY/WEAKLY TYPED:

❑JavaScript is known as **untyped/loosely/weakly typed** language. This means that we do not have to specify the data type in advance unlike other languages like C++, Java C# etc.

Example –

```
var x;
x = 5;
```

❑In the above example we can see that we did not have to specify any data type for the variable x in advance.

2. JavaScript is DYNAMICALLY TYPED :

❑JavaScript is known as **dynamically typed** language. This means, that once a variable is created in JavaScript using the keyword **var**, we can store any type of value in this variable supported by JavaScript.

Example –

```
// creating variable to store a number
var num = 5;

// store string in the variable num
num = "Simple Snippets";
```

❑In this example above, you can see that the data type of the variable num changes from number to string as we pass string data. This it is flexible in nature.

Variable Scope in JavaScript –

- Scope of a variable is the part of the program from where the variable may directly be accessible.
- In JavaScript, there are two types of scopes:
- **Global Scope** – Scope outside the outermost function.
- **Local Scope** – Inside the function being executed.

❑Let's look at the code below. We have a global variable defined in first line in global scope. Then we have a local variable defined inside the function fun().

Example

```
<script type = "text/javascript">
var globalVar = "This is a global variable";
 function fun()
{
var localVar = "This is a local variable";
console.log(globalVar);
console.log(localVar);
}
 fun();
console.log(localVar);
< /script>
```

Output:

This is a global variable
This is a local variable
Uncaught reference error: localVar is not defined

Type Conversion in JavaScript

- Type conversion in JavaScript is the process of converting data of one type to another.
- There are two types of type conversion in JavaScript.
- They are Implicit conversion and explicit conversion.
- Some examples of type conversion are converting from String data to Number, Number to String, etc.

Types of type conversions (Continued)

- Type conversion in JavaScript is of two types. They are implicit conversion and explicit conversion.
- Implicit type conversion - This means that the conversion happens automatically
- Explicit type conversion - This means that the conversion has to be done manually by the user.

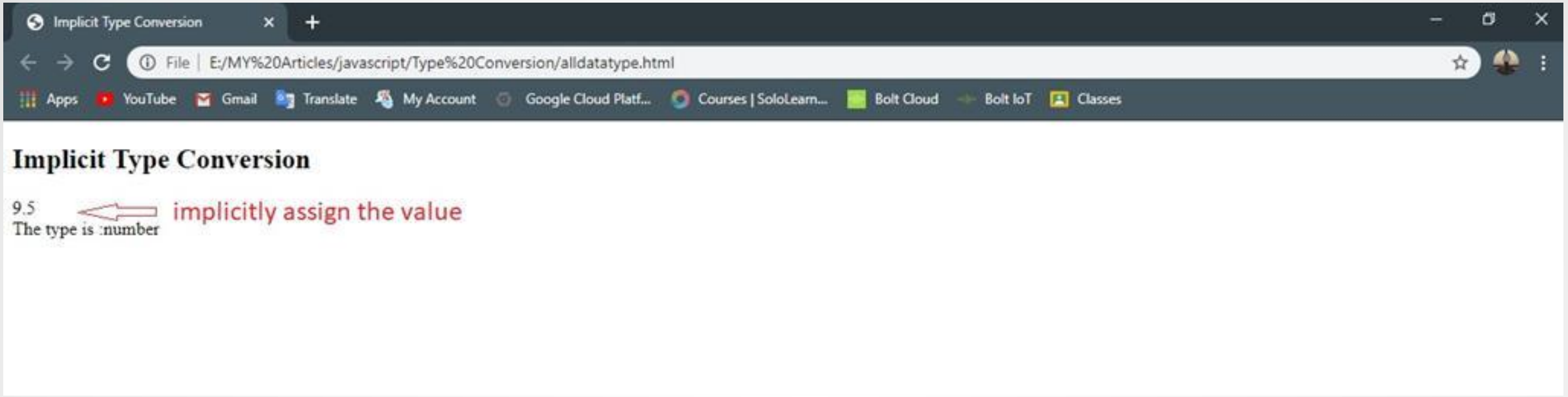
Implicit type Conversions

- Converting one type of data value to another type is called typecasting. If required JavaScript engine automatically converts one type of value into another type. It will try to convert it to the right type.

Example

```
<!DOCTYPE html>
<html>
<head>
   <title>Implicit Type Conversion</title>
</head>
<body>
<h2>Implicit Type Conversion</h2>
<script type="text/javascript">
document.write (4+5.5); //the output is 9.5
document.write("<br>");
document.write("The type is :"+typeof(4+4.5));
</script>
</body>
</html>
```

Output



❑In the above example, with four + five point three, we get the output nine point five. How the browser evaluates this expression, browser checks what operator you give to this expression. In this expression giving (‘+’) operator on the left side of the operator is an integer value and the right side of the operator real numbers. First, the browser converts the integer number to real number by putting (4.0+5.5) the add these values gives the output 9.5. Implicitly the browser converts integer numbers before evaluating the expression.

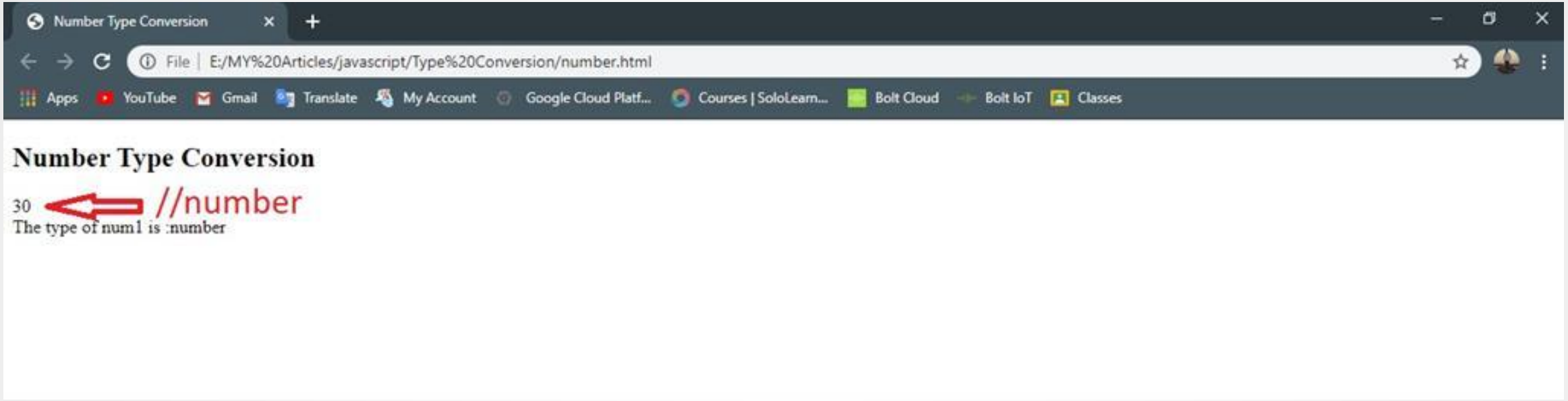
Explicit Type Conversions

- In explicit conversion to convert one type of value into another type. explicit type conversion is done by programmers using JavaScript functions. The JavaScript programmer can explicitly convert the data type. we can see the function using an explicit type conversion
- Number () – it converts a number to string and Boolean to string
- Boolean () – converts any type of given value into true or false (or) 0 and 1
- ParseInt () – the given value to converts an integer
- ParseFloat () – the given value to converts a float
- String () – it converts any type of given value into string type
- toString () – it converts the given number into the binary, octal, and hexadecimal

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Number Type Conversion</title>
</head>
<body>
<h2>Number Type Conversion</h2>
<script type="text/javascript">
var num1 = '30';
document.write(Number(num1));
document.write("<br>");
document.write("The type of num1 is :"+typeof(Number(num1)));
document.write("<br>");
</script>
</body>
</html>
```

Output


Number Type Conversion
File | E:/MY%20Articles/javascript/Type%20Conversion/number.html
Apps
YouTube
Gmail
Translate
My Account
Google Cloud Platf...
Courses | SoloLearn...
Bolt Cloud
Bolt IoT
Classes
Number Type Conversion
30
//number
The type of num1 is :number

JavaScript Operators

- An operator is used for manipulating a certain value or operand.

Operators are used to perform specific mathematical and logical computations on operands

What is an Operator?

- Operators are used for comparing values, perform arithmetic operations, etc. For example, if we take a simple expression, 4 + 5 is equal to 9. Here 4 and 5 are called operands and '+' is called the operator. JavaScript consists of different types of operators that are used to perform different operations.

Types of JavaScript Operators

❑There are different types of operators in JavaScript that are used for performing different operations. Some of the JavaScript Operators include:

- ➢Arithmetic Operators
- ➢Comparison Operators
- ➢Bitwise Operators
- ➢Logical Operators
- ➢Assignment Operators

Arithmetic Operators

❑Arithmetic operators are used to perform arithmetic operations on the operands. Here is a list of operators that are known as JavaScript arithmetic operators:

Operator	Description	Example
+	Adds two operands	10 + 20 = 30
-	Subtracts the second operand from the first	30 – 20 = 10
/	Divide the numerator by the denominator	20/10 = 2
*	Multiply two operands	5 * 5 = 25
%	Outputs the remainder of an integer division	20 % 10 = 0
++	Increases an integer value by one	var a=20; a++; Now a = 21
—	Decreases an integer value by one	var a=20; a–; Now a = 19

Comparison Operators

- The JavaScript comparison operator compares the two operands.
- The comparison operators are as follows:

Operator	Description	Example
==	Checks if two operands are equal or not. If yes, then the condition becomes true.	20==30 = false
===	Finds the identical (equal and of the same type)	10==20 = false
!=	Checks if two operands are equal or not. If the values are not equal, then the condition becomes true	20!=30 = true
!==	It implies that two values are not Identical	20!==20 = false
>	Checks if the value of the left operand is greater than the value of the right operand	30>10 = true
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand	20>=10 = true
<	This Checks if the value of the left operand is less than the value of the right operand	20<10 = false
<=	Checks if the value of the left operand is less than or equal to the value of the right operand	30<=10 = false

Bitwise Operators

❑The bitwise operators are used to perform bitwise operations on operands. Here is a list of bitwise operators:

Operator	Description	Example
&	Boolean AND operation on each bit of its integer arguments	(10==20 & 20==33) = false
|	It performs a Boolean OR operation on each bit of its integer arguments	(10==20 | 20==33) = false
^	This operator performs Bitwise XOR operation	(10==20 ^ 20==33) = false
~	It is a unary operator and operates by reversing all the bits in the operand	(~10) = -10
<<	Moves all the bits in its first operand to the left by the number of places specified in the second operand.	(10<<2) = 40
>>	The left operand's value is moved right by the number of bits specified by the right operand.	(10>>2) = 2
>>>	This operator is just like the >> operator, except that the bits shifted in on the left are always zero.	(10>>>2) = 2

Logical Operators

❑The list provides all the JavaScript logical operators:

Operator	Description	Example
&&	Logical **AND** – If both the operands are non-zero, then the condition becomes true	(10==20 && 20==33) = false
||	Logical **OR** – If any of the two operands are non-zero, then the condition becomes true.	(10==20 || 20==33) = false
!	Logical **NOT** – Reverses the logical state of its operand.	!(10==20) = true

Assignment Operators

❑The Assignment operators are used to assign values to the operand.

The following operators are known as JavaScript assignment operators:

Operator	Description	Example
=	Assigns values from the right side operand to the left side operand	20+10 = 30
+=	It adds the right operand to the left operand and assigns the result to the left operand	var a=20; a+=10; Now a = 30
-=	It subtracts the right operand from the left operand and assigns the result to the left operand	var a=30; a-=10; Now a = 20
*=	It multiplies the right operand with the left operand and assigns the result to the left operand	var a=10; a*=20; Now a = 200
/=	It divides the left operand with the right operand and assigns the result to the left operand	var a=10; a/=2; Now a = 5
%=	It takes modulus using two operands and assigns the result to the left operand	var a=10; a%=2; Now a = 0

MALLA REDDY UNIVERSITY

Thank You