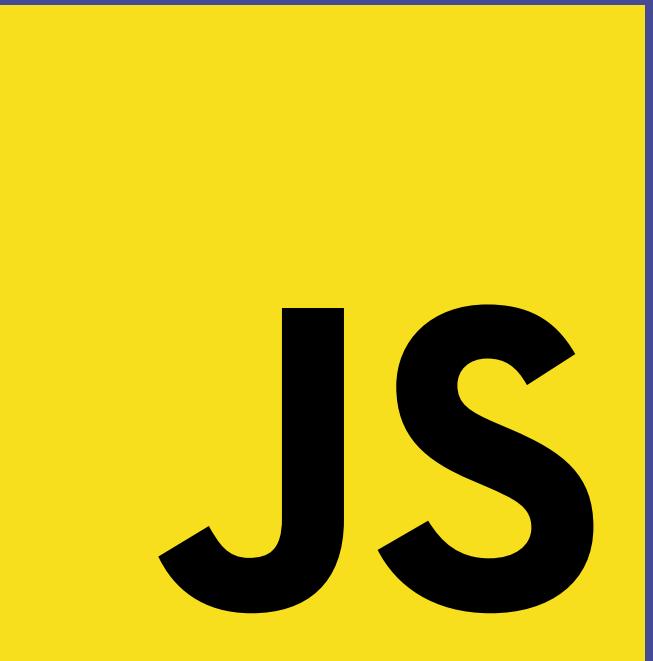


# JavaScript Fundamental

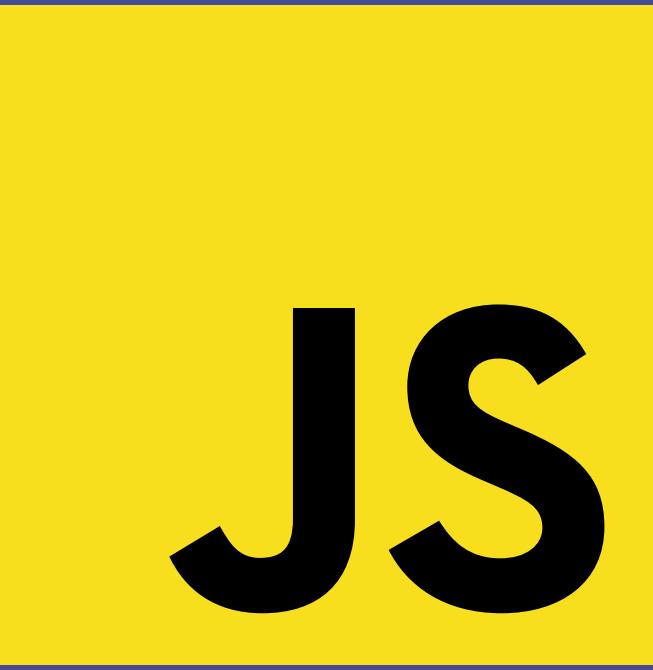
JavaScript (js) is a light-weight programming language which is used for web



JS

# Tools You Need

- Your Favorite Code Editor
- And Your Favorite Browser



JS

# MY FIRST JS

---

- Javascript example is easy to code

---

- JavaScript provides 3 places to put the JavaScript code: within body tag, within head tag and external JavaScript file

---

- document.write() function is used to display dynamic content through JavaScript.

---

 index.html

```
10 document.write("Hello JS");
```

# JS COMMENT

- The JavaScript comments are meaningful way to deliver message
- It is used to add information about the code, warnings or suggestions
- The JavaScript comment is ignored by the JavaScript engine

● ● ● index.html

```
10 // It is single line comment
11 document.write("Hello JS");
12 /* It is multi line comment.
13 It will not be displayed */
```

# JS VARIABLES

- A JavaScript variable is simply a name of storage location

- Name must start with a letter (a to z or A to Z), underscore( \_ ), or dollar( \$ ) sign

- After first letter we can use digits (0 to 9), for example value1.

- JavaScript variables are case sensitive, for example x and X are different variables.

 p1.html

```
8      var x = 10;  
9      var y = 20;  
10     var z=x+y;  
11     document.write(z);
```

# JAVASCRIPT DATA TYPES

JavaScript provides different data types to hold different types of values. There are two types of data types in JavaScript.

- Primitive data type
- Non-primitive (reference) data type



# PRIMITIVE DATA TYPE

There are five types of primitive data types in JavaScript.

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

# NON-PRIMITIVE DATA TYPE

The non-primitive data types are as follows

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

# JAVASCRIPT OPERATORS

JavaScript operators are symbols that are used to perform operations on operands

- Arithmetic Operators
- Comparison (Relational) Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Special Operators



# JAVASCRIPT ARITHMETIC OPERATORS

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	$10+20 = 30$
-	Subtraction	$20-10 = 10$
*	Multiplication	$10*20 = 200$
/	Division	$20/10 = 2$
%	Modulus (Remainder)	$20\%10 = 0$
++	Increment	<code>var a=10; a++; Now a = 11</code>
--	Decrement	<code>var a=10; a--; Now a = 9</code>

# JAVASCRIPT COMPARISON OPERATORS

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

Operator	Description	Example
<code>==</code>	Is equal to	<code>10==20 = false</code>
<code>===</code>	Identical (equal and of same type)	<code>10==20 = false</code>
<code>!=</code>	Not equal to	<code>10!=20 = true</code>
<code>!==</code>	Not Identical	<code>20!==20 = false</code>
<code>&gt;</code>	Greater than	<code>20&gt;10 = true</code>
<code>&gt;=</code>	Greater than or equal to	<code>20&gt;=10 = true</code>
<code>&lt;</code>	Less than	<code>20&lt;10 = false</code>
<code>&lt;=</code>	Less than or equal to	<code>20&lt;=10 = false</code>

# JAVASCRIPT BITWISE OPERATORS

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	$(10 == 20 \& 20 == 33) = \text{false}$
	Bitwise OR	$(10 == 20   20 == 33) = \text{false}$
^	Bitwise XOR	$(10 == 20 ^ 20 == 33) = \text{false}$
~	Bitwise NOT	$(\sim 10) = -10$
<<	Bitwise Left Shift	$(10 << 2) = 40$
>>	Bitwise Right Shift	$(10 >> 2) = 2$
>>>	Bitwise Right Shift with Zero	$(10 >>> 2) = 2$

# JAVASCRIPT LOGICAL OPERATORS

The following operators are known as JavaScript logical operators

Operator	Description	Example
<code>&amp;&amp;</code>	Logical AND	<code>(10==20 &amp;&amp; 20==33) = false</code>
<code>  </code>	Logical OR	<code>(10==20    20==33) = true</code>
<code>!</code>	Logical Not	<code>!(10==20) = true</code>

# JAVASCRIPT ASSIGNMENT OPERATORS

The following operators are known as JavaScript assignment operators

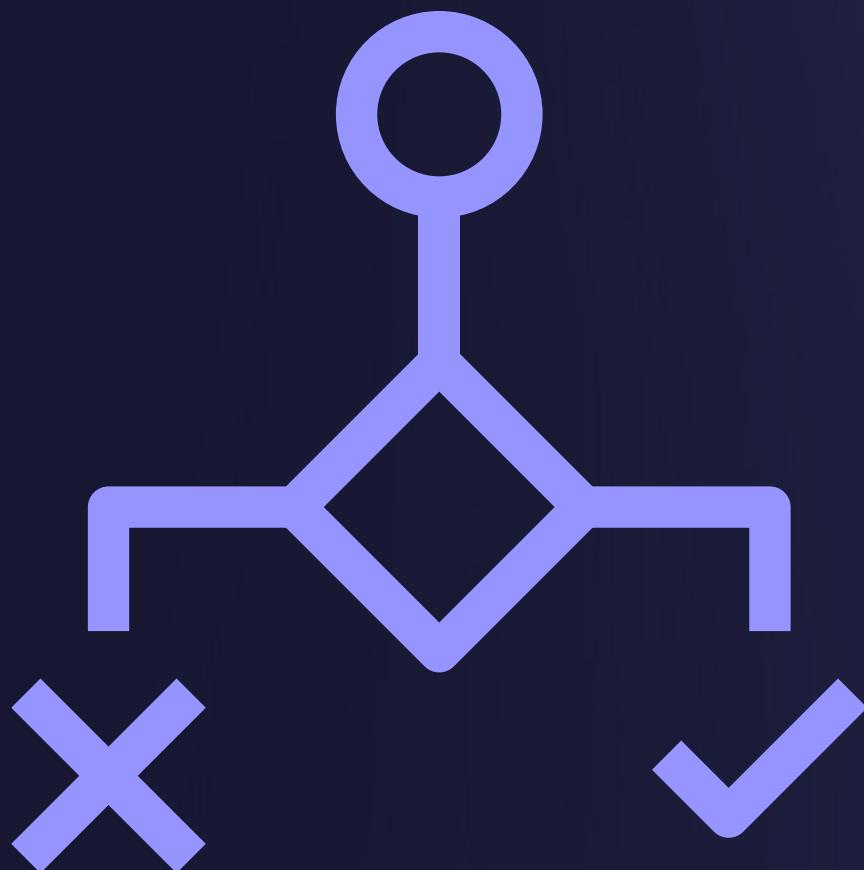
Operator	Description	Example
=	Assign	$10+10 = 20$
+=	Add and assign	<code>var a=10; a+=20; Now a = 30</code>
-=	Subtract and assign	<code>var a=20; a-=10; Now a = 10</code>
*=	Multiply and assign	<code>var a=10; a*=20; Now a = 200</code>
/=	Divide and assign	<code>var a=10; a/=2; Now a = 5</code>
%=	Modulus and assign	<code>var a=10; a%=2; Now a = 0</code>

# JAVASCRIPT IF-ELSE

---

The JavaScript if-else statement is used to execute the code whether condition is true or false.

1. If Statement
2. If else statement
3. if else if statement



# JAVASCRIPT IF STATEMENT

---

It evaluates the content only if expression is true.

 index.html

```
10 var age=20;  
11 if(age>18){  
12 document.write("You are adult");  
13 }
```

# IF...ELSE STATEMENT

---

It evaluates the content whether condition is true or false.

● ● ● index.html

```
10 var age=20;  
11 if(age>18){  
12     document.write("You are adult");  
13 }  
14 else{  
15     document.write("You are child");  
16 }
```

# IF...ELSE IF STATEMENT

---

It evaluates the content only if expression is true from several expressions.

● ● ● index.html

```
10 var a=20;
11 if(a==10){
12     document.write("a is equal to 10");
13 }
14 else if(a==15){
15     document.write("a is equal to 15");
16 }
17 else if(a==20){
18     document.write("a is equal to 20");
19 }
20 else{
21     document.write("a is not equal to 10, 15 or 20");
22 }
```

# JAVASCRIPT SWITCH

The JavaScript switch statement is used to execute one code from multiple expressions. It is just like else if statement

● ● ● index.html

```
12  switch(grade){  
13      case 'A':  
14          result="A Grade";  
15          break;  
16      case 'B':  
17          result="B Grade";  
18          break;  
19      case 'C':  
20          result="C Grade";  
21          break;  
22      default:  
23          result="No Grade";  
24  }
```

# JS LOOP

The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

---



**FOR LOOP**



**FOR IN LOOP**



**WHILE LOOP**



**DO WHILE LOOP**

# JAVASCRIPT FOR LOOP

---

The JavaScript for loop iterates the elements for the fixed number of times. It should be used if number of iteration is known.

● ● ● index.html

```
10  for (i=1; i<=5; i++){
11      document.write(i + "<br/>")
12  }
```

# JAVASCRIPT WHILE LOOP

---

The JavaScript while loop iterates the elements for the infinite number of times. It should be used if number of iteration is not known.

● ● ● index.html

```
10 var i=11;  
11 while (i<=15) {  
12     document.write(i + "<br/>");  
13     i++;  
14 }
```

# JAVASCRIPT DO WHILE LOOP

---

The JavaScript do while loop iterates the elements for the infinite number of times like while loop. But, code is executed at least once whether condition is true or false.

● ● ● index.html

```
10 var i=21;  
11 do{  
12     document.write(i + "<br/>");  
13     i++;  
14 }  
15 while (i<=25);
```

# JavaScript Functions

01

## Code Reusability

Functions are defined only once and can be invoked many times, like in other programming languages.

02

## Less Code

It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.

03

## Easy to understand

Functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

# JAVASCRIPT FUNCTIONS

---

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

 index.html

```
10  function msg(){  
11      alert("hello! this is message");  
12  }
```

# FUNCTION ARGUMENTS

---

We can call function by passing arguments.  
Let's see the example of function that has one argument.

● ● ● index.html

```
10  function msg(text){  
11      alert(text);  
12  }
```

# FUNCTION RETURN

---

We can call function that returns a value and use it in our program.

● ● ● index.html

```
10  function addTwoNum(){
11      var x=10;
12      var y=20;
13      var z=x+y;
14      return z
15 }
```

# JAVASCRIPT OBJECTS

---

A javaScript object is an entity having state and behavior (properties and method)

● ● ● index.html

```
11 var Person={  
12   Name:"Rabbil Hasan",  
13   Age:30,  
14   City:"Dhaka",  
15   Country:"Bangladesh"  
16 }
```

# JAVASCRIPT ARRAY

JavaScript array is an object that represents a collection of similar type of elements

● ● ● index.html

```
11 var city=["Dhaka","Rajshahi","Rangpur"];
12
13 for (i=0;i<city.length;i++){
14
15 document.write(city[i] + "<br/>");
16
17 }
```

# JAVASCRIPT ARRAY CONCAT()

---

The JavaScript array concat() method combines two or more arrays and returns a new string.

 index.html

```
11 var arr1=["A","B","C"];
12 var arr2=["D","E","F"];
13 var result=arr1.concat(arr2);
14 document.writeln(result);
```

# JAVASCRIPT ARRAY FROM()

---

The `from()` method creates a new array that holds the shallow copy from an array or iterable object.

When applied to a string, each word gets converted to an array element in the new array

 index.html

```
11 var arr=Array.from("RABBIL HASAN");
12 document.write(arr);
13
```

# JAVASCRIPT ARRAY FILTER()

---

The JavaScript array filter() method filter and extract the element of an array that satisfying the provided condition.

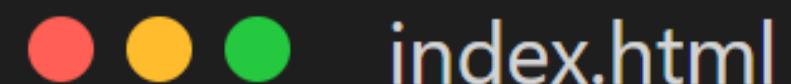
● ● ● index.html

```
11 var marks=[50,40,45,37,20];
12
13 function check(value)
14 {
15     return value>30;
16 }
17
18 document.writeln(marks.filter(check));
```

# JAVASCRIPT ARRAY FIND()

---

The JavaScript array `find()` method returns the first element of the given array that satisfies the provided function condition.



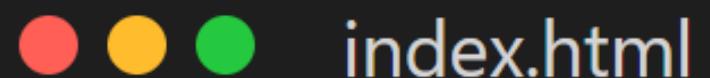
index.html

```
11 var arr=[5,22,19,25,34];
12 var result=arr.find(x=>x>20);
13 document.writeln(result)
```

# JAVASCRIPT ARRAY FINDINDEX()

---

The JavaScript array `findIndex()` method returns the index of first element of the given array that satisfies the provided function condition. It returns -1, if no element satisfies the condition.



index.html

```
11 var arr=[5,22,19,25,34];
12 var result=arr.findIndex(x=>x>20);
13 document.writeln(result)
```

# JAVASCRIPT ARRAY FOREACH()

---

The JavaScript array forEach() method is used to invoke the specified function once for each array element.



index.html

```
11  var arr=[5,22,19,25,34];
12  arr.forEach(function(fetch) {
13      document.writeln(fetch);
14  });
15
```

# JAVASCRIPT ARRAY INCLUDES()

---

The JavaScript array includes() method checks whether the given array contains the specified element. It returns true if an array contains the element, otherwise false.

● ● ● index.html

```
11 var arr=[5,22,19,25,34];
12 var result=arr.includes(22);
13 document.writeln(result);
```

# JAVASCRIPT ARRAY

## INDEXOF()

---

The JavaScript array `indexOf()` method is used to search the position of a particular element in a given array. This method is case-sensitive.

● ● ● index.html

```
11 var arr=[5,22,19,25,34];
12 var result= arr.indexOf(19);
13 document.writeln(result);
```

# JAVASCRIPT ARRAY

## POP()

---

The JavaScript array pop() method removes the last element from the given array and return that element.

 index.html

```
11 var arr=[5,22,19,25,34];
12 arr.pop();
13 document.writeln(arr);
```

# JAVASCRIPT ARRAY

## PUSH()

---

The JavaScript array push() method adds one or more elements to the end of the given array.

 index.html

```
11 var arr=[5,22,19,25,34];
12 arr.push(50);
13 document.writeln(arr);
```

# JAVASCRIPT ARRAY REVERSE()

---

The JavaScript array `reverse()` method changes the sequence of elements of the given array and returns the reverse sequence.

 index.html

```
11  var arr=[5,22,19,25,34];
12  array.reverse()
13  document.writeln(arr);
14
```

# JAVASCRIPT ARRAY

## SLICE()

---

The JavaScript array slice() method extracts the part of the given array and returns it. This method doesn't change the original array

 index.html

```
11 var arr=[5,22,19,25,34];
12 var result=arr.slice(1,3);
13 document.writeln(result);
```

# JAVASCRIPT ARRAY

## SORT()

---

The JavaScript array sort() method is used to arrange the array elements in some order. By default, sort() method follows the ascending order.

 index.html

```
11 var arr=[5,22,19,25,34];
12 var result=arr.sort();
13 document.writeln(result);
```

# JAVASCRIPT ARRAY

## SPLICE()

---

The JavaScript array splice() method is used to add/remove the elements to/from the existing array. It returns the removed elements from an array.

The splice() method also modifies the original array.

 index.html

```
11 var arr=["A","B","C","D","E"];
12 var result=arr.splice(2,0,"B")
13 document.writeln(arr);
```

# JAVASCRIPT STRING METHODS

The JavaScript string is an object that represents a sequence of characters

---

## charAt()

It provides the char value present at the specified index.

## concat()

It provides a combination of two or more strings.

## indexOf()

It provides the position of a char value present in the given string.

## lastIndexOf()

It provides the position of a char value present in the given string by searching a character from the last position.

## replace()

It replaces a given string with the specified replacement.

## substr()

It is used to fetch the part of the given string on the basis of the specified starting position and length.

# JAVASCRIPT STRING METHODS

The JavaScript string is an object that represents a sequence of characters

---

## substring()

It is used to fetch the part of the given string on the basis of the specified index.

## slice()

It is used to fetch the part of the given string. It allows us to assign positive as well negative index.

## toLowerCase()

It converts the given string into lowercase letter.

## toUpperCase()

It converts the given string into uppercase letter.

## trim()

It trims the white space from the left and right side of the string.

# JAVASCRIPT DATE OBJECT

The JavaScript date object can be used to get year, month and day.

---

## `getDate()`

It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.

## `getDay()`

It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.

## `getFullYear()`

It returns the integer value that represents the year on the basis of local time.

## `getHours()`

It returns the integer value between 0 and 23 that represents the hours on the basis of local time.

## `getMilliseconds()`

It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.

# JAVASCRIPT DATE OBJECT

The JavaScript date object can be used to get year, month and day.

---

## getMinutes()

It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.

## getMonth()

It returns the integer value between 0 and 11 that represents the month on the basis of local time.

## getSeconds()

It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.

# JAVASCRIPT DATE OBJECT

The JavaScript date object can be used to get year, month and day.

---

## getUTCDate()

It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of universal time.

## getUTCDay()

It returns the integer value between 0 and 6 that represents the day of the week on the basis of universal time.

## getUTCFullYear()

It returns the integer value that represents the year on the basis of universal time.

## getUTCHours()

It returns the integer value between 0 and 23 that represents the hours on the basis of universal time.

## getUTCMinutes()

It returns the integer value between 0 and 59 that represents the minutes on the basis of universal time.

## getUTCMonth()

It returns the integer value between 0 and 11 that represents the month on the basis of universal time.

# JAVASCRIPT MATH

The JavaScript math object provides several constants and methods to perform mathematical operation.

---

**abs()**

It returns the absolute value of the given number.

**ceil()**

It returns a smallest integer value, greater than or equal to the given number.

**floor()**

It returns largest integer value, lower than or equal to the given number.

**max()**

It returns maximum value of the given numbers.

**min()**

It returns minimum value of the given numbers.

**random()**

It returns random number between 0 (inclusive) and 1 (exclusive).

**round()**

It returns closest integer value of the given number.

# JAVASCRIPT NUMBER OBJECT

The JavaScript number object enables you to represent a numeric value. It may be integer or floating-point.

---

## `isFinite()`

It determines whether the given value is a finite number.

## `isInteger()`

It determines whether the given value is an integer.

## `parseFloat()`

It converts the given string into a floating point number.

## `parseInt()`

It converts the given string into an integer number.

## `toFixed()`

It returns the string that represents a number with exact digits after a decimal point.

## `toString()`

It returns the given number in the form of string.