Javascript

BCA III

About js

- Js is more popular because of flexibility
  - Javascript does not support structures like class and packages like other programming languages.
  - Javascript is complete object oriented language.
  - Everything in js including functions are objects.
  - Power of js object is that we can add and delete properties in it.
- We can create the objects using inbuilt types
- Js has two types
  - Primitive types (holds simple data)
  - Reference types (objects in memory, refer a object into a memory)

Primitive types

Js supports 5 different types

1. **Number**
2. **String**
3. **Boolean**
4. **Null**
5. **undefined**

Use of Primitive types

What js can do

1. Video game
2. Server application
3. Virtual reality
4. Web applications
5. Adding interactive behaviour in js
6. AI simulation

What js can’t do

1. Server applications
2. Database programming

Embedding javascript in web page

We can add JavaScript code in an HTML document by employing the dedicated HTML tag <script> that wraps around JavaScript code.

The <script> tag can be placed in the <head> section of your HTML or in the <body> section, depending on when you want the JavaScript to load.

Embedding javascript in web page

Embedding Javascript in footer is more better than embedding in header.

Embedding javascript in web page

Methods of embedding

- External
- Internal
- Inline

External Javascript

External js

External JavaScript is when the JavaScript code written in another file having an extension .js is linked to the HMTL with the src attribute of script tag.

Syntax:

<script src="url_of_js_file"> </script>

External js

Advantages

- HTML and JavaScript files become more readable and easy to maintain.
- Page loads speed up due to Cached JavaScript files.
- Reusable for all files

Disadvantages

- Coders can easily download your code using the url of the script(.js) file.
- An extra HTTP request is made by the browser to get this JavaScript code.

Example

```
<!DOCTYPE html>

<html>

<body>

  <h2 id="demo">Hello...</h2>

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

</body>

</html>
```

Internal Javascript

Internal js

Internal JavaScript refers to embedding JavaScript code directly within the HTML file using <script> tag, either inside the <head> or <body> tag. This method is useful for small scripts specific to a single page.

```
<script type=”text/javascript”>

  // JavaScript code here

</script>
```

Internal js example

```
<!DOCTYPE html>

<html>

<body>

  <script>

  /*Internal Javascript*/

  alert("Hi class, Welcome to BCA III");

  </script>

</body>

</html>
```

Internal js advantages and disadvantages

Advantages

- No need for extra HTTP requests to load scripts.
- Easy to use for small code snippets specific to a single HTML file.

Disadvantages

- Makes the HTML file less readable by mixing code and content.
- Difficult to maintain when the script grows large.
- Does not allow for JavaScript code caching.

Inline javascript

Inline js

Inline JavaScript refers to JavaScript code embedded directly within HTML elements using the onclick, onsubmit, onload or other event attributes.

This allows you to execute JavaScript code in response to user interactions or specific events without needing a separate JavaScript file or script block.

Examples:

<button type="button" onclick="showAlert();"> Submit <button>

<form onsubmit=”return validate();”></form>

Delete

Js Variables

Js Variable

- Variables are used to store data in JavaScript so that we can later access the data.
- JavaScript is a dynamically typed language so the type of variables is decided at runtime.
- Therefore there is no need to explicitly define the type of a variable.

We can declare variables in JavaScript in three ways:

1. JavaScript var keyword
2. JavaScript let keyword
3. JavaScript const keyword

```
var a = 10    // Old style

let b = 20;   // Preferred for non-const

const c = 30;  // Preferred for const (cannot be changed)
```

Js Data types

String

Number

Bigint

Boolean

Undefined

Null

Symbol

Object

Js Data types examples

```
// Numbers:
let length = 16;

// Strings:
let color = "Yellow";

// Booleans
let x = true;

// Object:
const person = {firstName:"John", lastName:"Doe"};

// Array object:
const cars = ["Saab", "Volvo", "BMW"];

let a; //undefined
```

Js Operators

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- String Operators
- Logical Operators
- Bitwise Operators
- Ternary Operators
- Type Operators

Js Expressions

Javascript expression is a combination of values, variables, and operators, which computes to a value.

5*10

x*20

Conditional statements

If

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

Syntax

```
if (condition) {
  //code block
}
```

If else

Use the if else statement to specify a block of JavaScript code to be executed if a condition is false.

Syntax

```
if (condition) {
  //  block of code to be executed if the condition is true
}else{
//  block of code to be executed if the condition is false
}
```

Else if

Use the else if statement to specify a new condition if the first condition is false.

Syntax

```
if (condition1) {

  //  block of code to be executed if condition1 is true

} else if (condition2) {
        //  block of code to be executed if the condition1 is false and condition2 is true

} else {
        //  block of code to be executed if the condition1 is false and condition2 is false
}
```

Js Switch

The JavaScript Switch Statement Use the switch statement to select one of many code blocks to be executed.

Syntax

```
switch(expression) {

  case x:
        // code block
break;
  case y:
        // code block
break;
default:
// code block
}
```

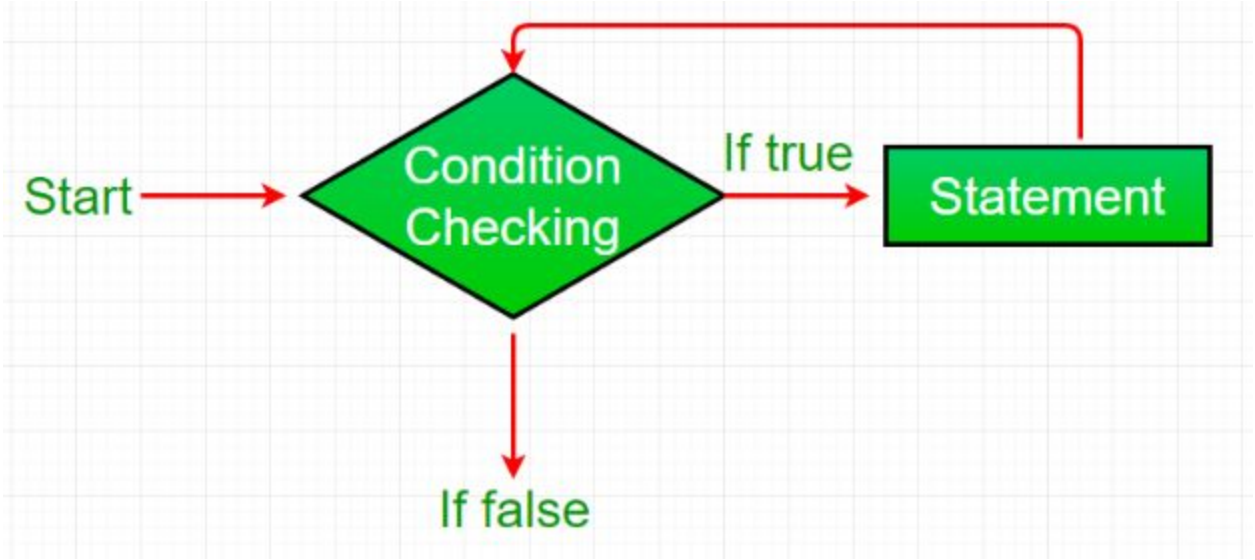
Loop

For loop

loops through a block of code a number of times

```
for (expression 1; expression 2; expression 3) {
  // code block to be executed
}
```

While loop



While loop

The while loop executes as long as the condition is true. It can be thought of as a repeating if statement.

```
while (condition) {
   // Code to execute
}
```

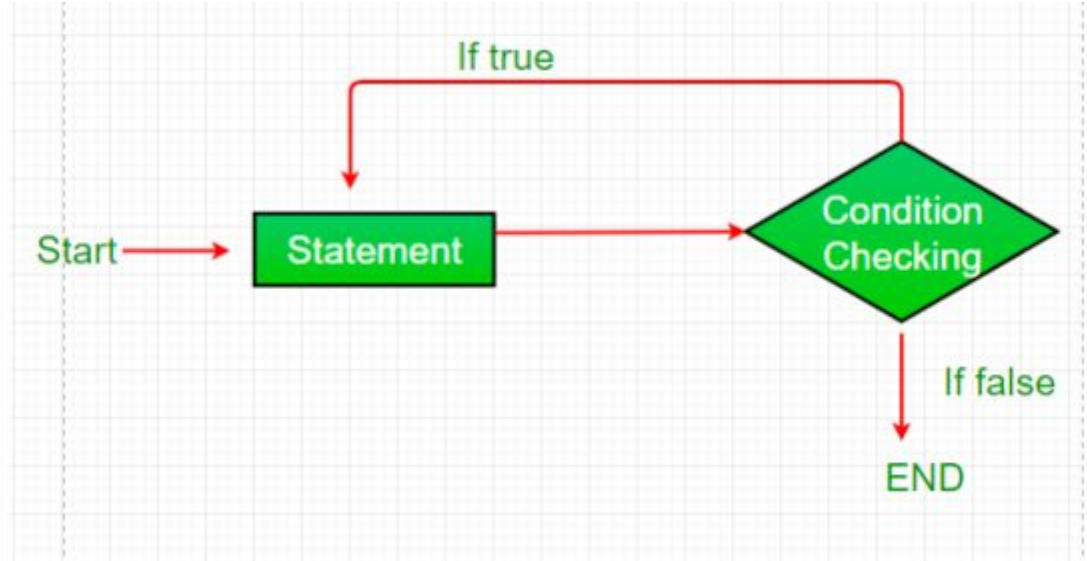
Do while loop

The do-while loop is similar to while loop except it executes the code block at least once before checking the condition.

```
do {

   // Code to execute

} while (condition);
```



for-in Loop

The for…in loop is used to iterate over the properties of an object. It is generally used in extract object items and array items.

```
for (let key in object) {

   // Code to execute

}
```

```
const obj = { name: "Ashish", age: 25 };

for (let key in obj) {

   console.log(key, ":", obj[key]);

}
```

Labels break and continue

To label JavaScript statements you precede the statements with a label name and a colon:

Label:
    statements

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch() and loop statement.

Array in js

An array is a special variable, which can hold more than one value:

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

Or,

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

Javascript scope

Javascript scope is accessibility of the variables

There are 3 types of scope in js

1) Block scope
2) Function scope
3) Global scope

Block Scope

- Before ES6 (2015), JavaScript variables had only Global Scope and Function Scope.
- ES6 introduced two important new JavaScript keywords: let and const.
- These two keywords provide Block Scope in JavaScript.
- Variables declared inside a { } block cannot be accessed from outside the block:

```
function validate() {

  let name = document.getElementById('name').value;

  // code here CAN use name

}
```

Function scope

- JavaScript has function scope: Each function creates a new scope.
- Variables defined inside a function are not accessible (visible) from outside the function.
- Variables declared with var, let and const are quite similar when declared inside a function.
- They all have Function Scope:

```
function validate() {

  const name = “Dhangadhi”;    // Function Scope

}
```

Global scope

A variable declared outside a function, becomes GLOBAL.

Var x=2;

Let x=2;

Const x=2;

functions

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

System functions

User defined functions

System functions

String inbuilt functions
String length

String charAt()

String charCodeAt()

String at()

String slice()

String substring()

String substr()

String toUpperCase()
String toLowerCase()
String concat()
String trim()
String trimStart()
String trimEnd()
String padStart()
String padEnd()
String repeat()
String replace()
String replaceAll()
String split()

Reverse string in js without use inbuilt function

Dialog box

1. Alert box
2. Prompt box
3. Confirm box

Alert box

To provide the information to user we use alert box.

alert(‘Information to user’);

Confirm box

- A confirm box is often used if you want the user to verify or accept something.
- A confirm box takes the focus away from the current window, and forces the user to read the message.
- Do not overuse this method. It prevents the user from accessing other parts of the page until the box is closed.

Confirm box

```
confirm("Press a button!\nEither OK or Cancel.");
```

Prompt box

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Prompt box

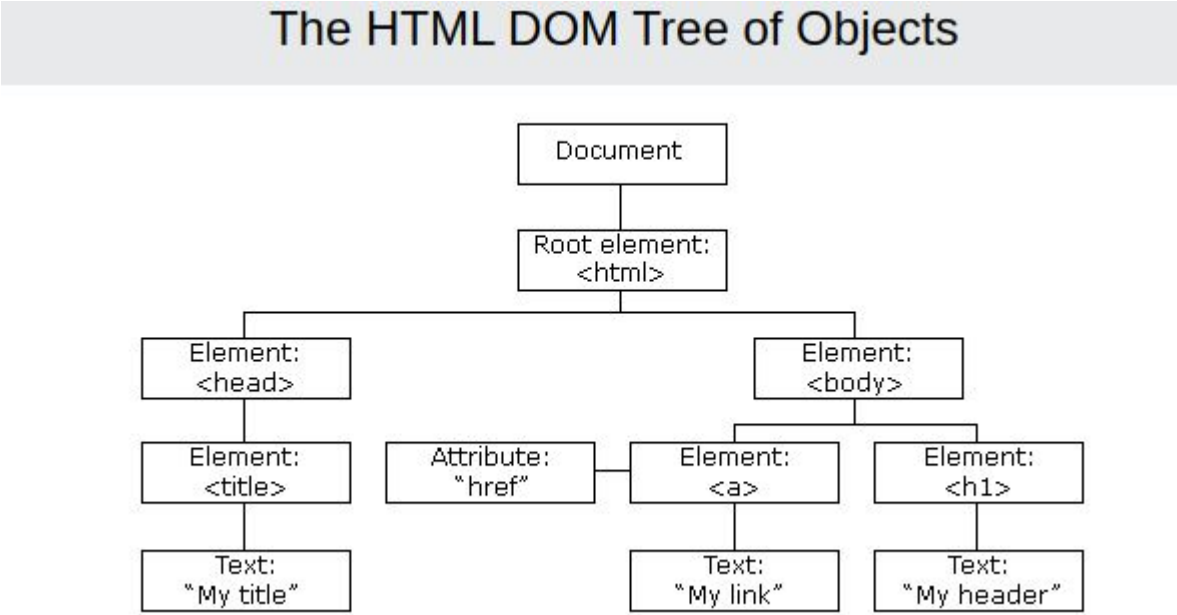
```
Let a = prompt("Please enter number");
```

DOM (Document object model)

DOM

When a web page is loaded, the browser creates a Document Object Model of the page.

The HTML DOM model is constructed as a tree of Objects:



DOM

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is HTML DOM ?

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

What is DOM ?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

Js Events

HTML EVENTS

onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

HTML EVENTS

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

JavaScript lets you execute code when events are detected.

EVENTS Handling

Event handlers can be used to handle and verify user input, user actions, and browser actions: