Prepared by Muthulakshmi V

Outline

- What is JavaScript?
- Latest version of JavaScript
- Programming with JavaScript
  - Declaring variables
  - Displaying messages using JavaScript
    - document.write
    - console.log
  - Data Types in JavaScript
  - The alert box
  - The prompt box – getting input from users
  - Adding Comments
  - Operators

Outline

- Conditionals
  - If
  - If else
  - switch
- Looping
- Arrays
- Functions
- Exception handling
- Document Object Model
  - The DOM Hierarchy
  - DOM Methods
  - DOM Properties
- AJAX

What is JavaScript?

- Is a programming language
- Adds interactivity to the web page
- Examples are responses when buttons are pressed or data entered in forms, dynamic styling, animations etc.
- Was created in 10 days in May 1995 by Brendan Eich, then working at Netscape and now of Mozilla.
- JavaScript was not always known as JavaScript: the original name was Mocha, a name chosen by Marc Andreessen, founder of Netscape

Latest version of JavaScript

- Became an ECMA standard in 1997.
- ECMA-262 is the official name of the standard.
- ECMAScript is the official name of the language.

Latest version of JavaScript

Year	Version	Description
1997	ECMAScript 1	First Edition
1998	ECMAScript 2	Editorial Changes
1999	ECMAScript 3	Regular Expressions , Exception Handling
	ECMAScript 4	Never Released
2009	ECMAScript 5	Adding “strict mode”, JSON
2011	ECMAScript 5.1	Editorial Changes
2015	ECMAScript 6	Classes, Modules, additional array operators, collection objects
2016	ECMAScript 7	Added more array operators

ECMAScript 6 is also called ECMAScript 2015.

ECMAScript 7 is also called ECMAScript 2016.

Browser Support

- ECMAScript 3 is fully supported in all browsers.
- ECMAScript 5 is fully supported in all modern browsers*.
- ECMAScript 6 is partially supported in all modern browsers.
- ECMAScript 7 is poorly supported in all browsers.

Including Script in HTML Pages

```
<head>
    <script>
        //In line scripts
    </script>
</head>

<head>
    <script src = 'xyz.js'>
        //External scripts
    </script>
</head>
```

Displaying Messages

- To display messages in the browser we have the following 2 basic methods.
- **document.write(**message);
  - This outputs the message in the browser window.
- **console.log**(message);
  - This outputs the message in the console window. Usually used for debugging purposes.

Variables

- Variables are containers that you can store values in.
- You start by declaring a variable with the var keyword, followed by any name you want to call it
- **var** firstname;

Note:

A semicolon at the end of a line indicates where a statement ends;

it is only absolutely required when you need to separate statements on a single line.

However, some people believe that it is a good practice to put them in at the end of each statement

Variables

Note:

You can name a variable nearly anything, but there are some name restrictions

Note:

JavaScript is case sensitive – firstName is a different variable to firstname. If you are getting problems in your code, check the casing!

Variables

- After declaring a variable, you can give it a value.
- firstname = ‘bob’;
- You can do both these operations on the same line if you wish
  - **var** firstname = ‘bob’;
- You can retrieve the value by just calling the variable by name
  - firstname;
- After giving a variable a value, you can later choose to change it
  - **var** firstname = 'Bob';
  - firstname = 'Steve';

Variables – Data Types

Type	Description	Example
String	A sequence of text known as string. Enclosed with single or double quotes.	var title = ‘JavaScript’
Number	A number. Should not enclose within single or double quotes.	var score = 92;
Boolean	A true or false value.	var isValid = true;
Array	A structure that allows to store multiple values in one reference	var evenNumbers = [2,4,6,8];
Object	Basically anything. Everything in JS is an object.	
Undefined	The type of the value in an unitialized variable	var myVariable;

Variables – Data Types

- The **typeof** operator is used to identify the type of the variable at any point of time.

var myVariable;

typeof myVariable will return undefined now.

myVariable = 5;

typeof myVariable will return Number.

Variables

- **Converting Variables to Numbers**

```
var x = 50.55;

Number(x)     -> returns 50.33

parseInt(x)     -> returns 50

parseFloat(x) -> 50.33
```

Variables

- **Converting Variables to Numbers**

There are 3 JavaScript methods that can be used to convert variables to numbers:

Method	Description
Number()	Returns a number, converted from its argument.
parseFloat()	Parses its argument and returns a floating point number
parseInt()	Parses its argument and returns an integer

Variables

- **Working with Dates**

The Date object lets you work with dates (years, months, days, hours, minutes, seconds, and milliseconds).

- A JavaScript date can be written as a string:

Tue Aug 01 2017 05:45:04 GMT+0530 (India Standard Time)

or as a number:

1501546504478

- Dates written as numbers, specifies the number of milliseconds since January 1, 1970, 00:00:00

Variables

- **Creating Date Objects**

  The Date constructor is used to create dates.

  new Date()

  new Date(milliseconds)

  new Date(dateString)

  new Date(year, month, day, hours, minutes, seconds, milliseconds)

Variables

- **Date Methods**

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

The alert box

- The alert box is used to display a **popup** message to the users.

```
alert("I am message displayed in alert box!");
```

(or)

```
window.alert("I am message displayed in alert box!");
```

The 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.
- window.prompt(“message","defaultValue");

```
var firstNumber = prompt(‘Please enter the first number’,’10’);
```

Adding Comments

- Comments are for the reference of the developers to understand the code being written. In JS we have 2 ways to add comments.
- Comments will not be executed by the browser.
- Single Line Comments

```
// This line is a comment
```

- Multi Line Comments

```
/*

    All these are comments.

    I am another line in the comment.

*/
```

Operators

- An operator is a valid symbol, which produces a new value based on a value (or) values.
- Arithmetic Operators

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Operators

- Assignment Operators

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Operators

- Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Operators

- Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

Operators

- JS Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

Conditionals

- In any programming language, code needs to make decisions and carry out actions accordingly depending on different inputs.
- **if ... else statements**

```
1 if (condition) {
2   code to run if condition is true
3 } else {
4   run some other code instead
5 }
```

Conditionals

- **if ... else statements - explanation**
- **The keyword if followed by some parentheses.**

A condition to test, placed inside the parentheses . This condition will make use of the comparison operators we discussed in the last module, and will return true or false.

A set of curly braces, inside which we have some code — this can be any code we like, and will only be run if the condition returns true.

- **The keyword else.**

Another set of curly braces, inside which we have some more code — this can be any code we like, and will only be run if the condition is not true.

Conditionals

- **if ... else statements - example**

```
var x = prompt('Enter a Number');
if(x % 2 == 0){
    document.write('Even Number!');
}else{
    document.write('Odd Number!');
}
```

Conditionals

- if ... else statements - example

```
<script>
    var x = prompt('Enter a Number');
    if(x < 0){
        document.write('Negative Number');
    }else if(x == 0){
        document.write('Zero');
    }else{
        document.write('Positive Number');
    }
</script>
```

Conditionals

- **switch statement**

They take a single expression/value as an input, and then look through a number of choices until they find one that matches that value, executing the corresponding code that goes along with it.

Conditionals

- **switch statement**

```
switch (expression) {
  case choice1:
    run this code
    break;

  case choice2:
    run this code instead
    break;

  // include as many cases as you like

  default:
    actually, just run this code
}
```

Conditionals

- **switch statement**

```
var langChoice = prompt('Enter a Language (en - English, fr - French, gr - German');
switch(langChoice){
    case 'en':{
        document.write('Good Monring!');
        break;
    }
    case 'fr':{
        document.write('Bonjour!');
        break;
    }
    case 'gr':{
        document.write('Guten Morgen!');
        break;
    }
    default:{
        document.write('Invalid Choice!');
    }
}
```

Looping

- Loops can execute a block of code a number of times.
- A **counter**, which is initialized with a certain value — this is the starting point of the loop.
- An **exit condition**, which is the criteria under which the loop stops — usually the counter reaching a certain value.
- An **iterator**, which generally increments the counter by a small amount on each successive loop, until it reaches the exit condition.

Looping

- For loop
- for (statement 1; statement 2; statement 3) {

code block to be executed

}

```
//Input any 3 numbers and find their sum
var sum = 0;
for(var i =1; i <= 3; i++){
    var x = prompt('Enter a number');
    sum = sum + Number(x);
}
document.write('The sum of given 3 numbers is ' + sum);
```

Looping

- while loop
- The while loop loops through a block of code as long as a specified condition is true.

```
var x = 1;
var sum = 0;
while(x <= 10){
    sum = sum + Number(x);
    x++;
}
document.write('The sum of first 10 numbers is ' + sum);
```

Arrays

- JavaScript arrays are used to store multiple values in a single variable.
- An array is a special variable, which can hold more than one value at a time.
- **Declaring arrays**

  var *array_name* = [*item1*, *item2*, ...];

- **Example**

  var fruits = [“apple", “orange", “grapes"];

  var fruit1 = fruits[0]; //Accessing the array element

  var evenNumbers = new Array (0,2,4,6,8);

Arrays

- **Identifying the number of elements in an array**

arrayname.length;

Example

```
var fruits = [“apple", “orange", “grapes"];

document.write(fruits.length); //will print 3
```

Arrays

- **Array Methods**
- **pop()**

  Removes the last element from the array.

  returns the value that was "popped out“

  Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();
// Removes & returns the last element ("Mango") from fruits
```

Arrays

- **Array Methods**
- **push()**

  Adds a new element to an array (at the end).

  Returns the new array length

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.push("Kiwi");    //  the value of x is 5
```

Arrays

- **Array Methods**
- **shift()**

  Shifting is equivalent to popping, working on the first element instead of the last.

  The **shift()** method removes the first array element and "shifts" all other elements to a lower index

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.shift();

// Removes and returns "Banana"
```

Arrays

- **Array Methods**
- **unshift()**

  The **unshift()** method adds a new element to an array (at the beginning), and "unshifts" older elements.

  ```
  var fruits = ["Banana", "Orange", "Apple", "Mango"];
  var x - fruits.unshift("Lemon");
  // Adds a new element "Lemon" to fruits, and returns 5
  ```

Arrays

- **Array Methods**
- **sort()**

  The **sort()** method sorts an array alphabetically

  ```
  var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.sort();
  ```

- **reverse()**

  The **reverse()** method reverses the elements in an array.

  To sort the array in the descending order

  ```
  fruits.sort();
  fruits.reverse();
  ```

Functions

- JavaScript functions are **defined** with the **function** keyword.
- You can use a function **declaration** or a function **expression**.
- **function** *functionName*(*parameters*) {

  code to be executed

  }

Example

```
function multiply(x,y){

    return x * y;

}
```

Functions

- **Function Expressions**

  A JavaScript function can also be defined using an **expression**.

  A function expression can be stored in a variable:

  var x = function (a, b) {return a * b};

  var z = x(4, 3);

  The function above is actually an **anonymous function** (a function

  without a name)

Exception Handling

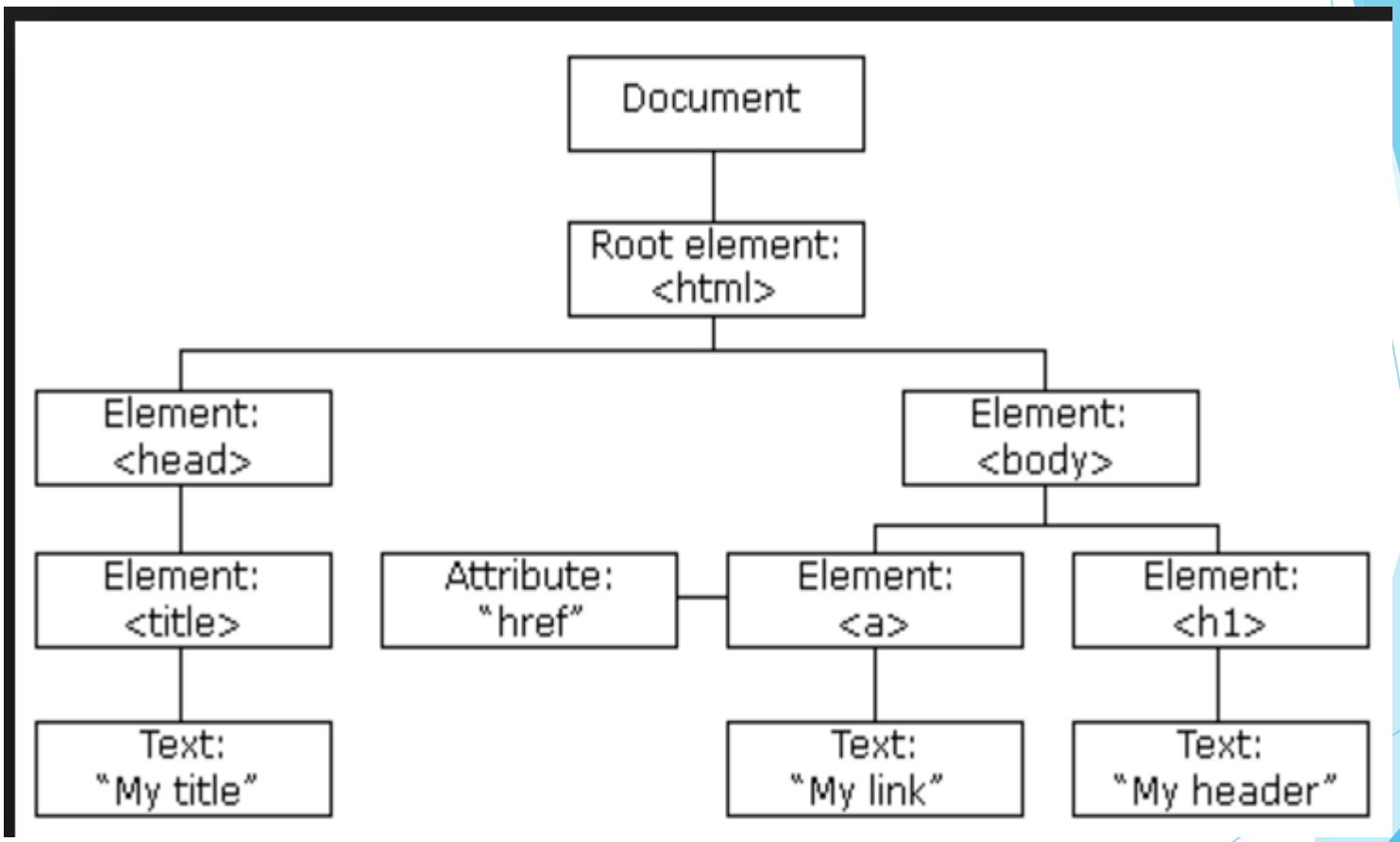
- The **try** statement lets you test a block of code for errors.
- The **catch** statement lets you handle the error.
- The **throw** statement lets you create custom errors.
- The **finally** statement lets you execute code, after try and catch, regardless of the result.

```
<script>
try {
    adddlert("Welcome guest!");
}
catch(err) {
    document.write(err.message);
}
</script>
```

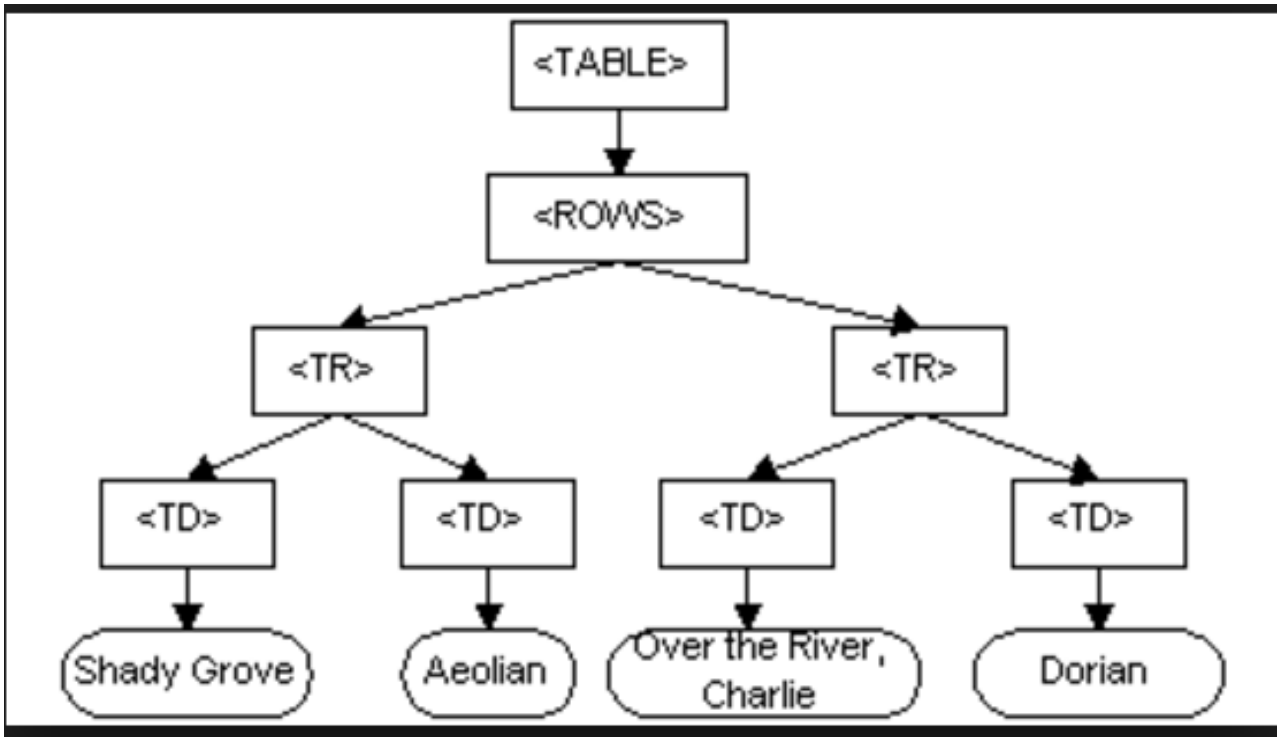
DOCUMENT OBJECT MODEL - DOM

- With the HTML DOM, JavaScript can access and change all the elements of an HTML document.
- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**

DOCUMENT OBJECT MODEL - DOM



DOCUMENT OBJECT MODEL - DOM



DOCUMENT OBJECT MODEL - DOM

- The document object represents your web page.
- If you want to access any element in an HTML page, you always start with accessing the document object.
- Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

DOCUMENT OBJECT MODEL - DOM

- **Changing HTML Elements**

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element

DOCUMENT OBJECT MODEL - DOM

- **Adding and Removing Elements**

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

DOCUMENT OBJECT MODEL - DOM

- **Adding Event Handlers**

Method	Description
document.getElementById(*id*).onclick = function(){*code*}	Adding event handler code to an onclick event

AJAX

- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background
- The core of AJAX is the **XMLHttpRequest** object.

```
var xhttp = new XMLHttpRequest();
```

AJAX

```
var xhttp;
xhttp=new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        console.log(this.responseText);
    }
};
var url = 'https://jsonplaceholder.typicode.com/users';
xhttp.open("GET", url, true);
xhttp.send();
```

AJAX - Request

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object.

```
xhttp.open("GET", “intro.txt", true);
xhttp.send();
```

Method	Description
open(*method*, *url*, *async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

AJAX - Response

The onreadystatechange Property

- The readyState property holds the status of the XMLHttpRequest.
- The onreadystatechange property defines a function to be executed when the readyState changes.
- The status property and the statusText property holds the status of the XMLHttpRequest object

AJAX - Response

The onreadystatechange Property

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Exercises

Exercise 1

Write a JavaScript that creates a different variable for each of the members of your group.

Use document.write to write a paragraph stating,

“Our group members are: “ and then each variable

(aka member name) being printed in the paragraph.

Exercise 2

Add to the JavaScript a prompt that asks the user their weight in pounds. It uses a second prompt and variable to ask the user to enter their height in inches.

It uses a third variable (not prompted) to hold the user's bmi, calculated as follows: (weight multiplied by 703), divided by (height times height).

Use document.write to write out the user's bmi.

Exercise 3

Find an image of a rock, an image of paper, and an image of scissors.

Prompt the user to enter 0 for rock, 1 for paper, and 2 for scissors.

i. If the user entered 0, use document.write to write out the html to display the image of arock.

ii. If the user entered 1, use document.write to write out the html to display the image of paper

iii. If the user entered 2, use document.write to write out the html to display the image of scissors.

Exercise 4

Loan Calculator

Enter Loan Data:

Amount of the loan (INR): [ ]

Annual interest (%): [ ]

Repayment period (years): [ ]

Approximate Payments: [Calculate]

Monthly payment: INR

Total payment: INR

Total interest: INR

Exercise 5

Contacts Search