axess academy

JavaScript

standard chartered

27/05/2024

Agenda

axess academy

S/W Requirements
IDE : Visual Studio
Browser: Google Chrome

Module 1: JavaScript Fundamentals

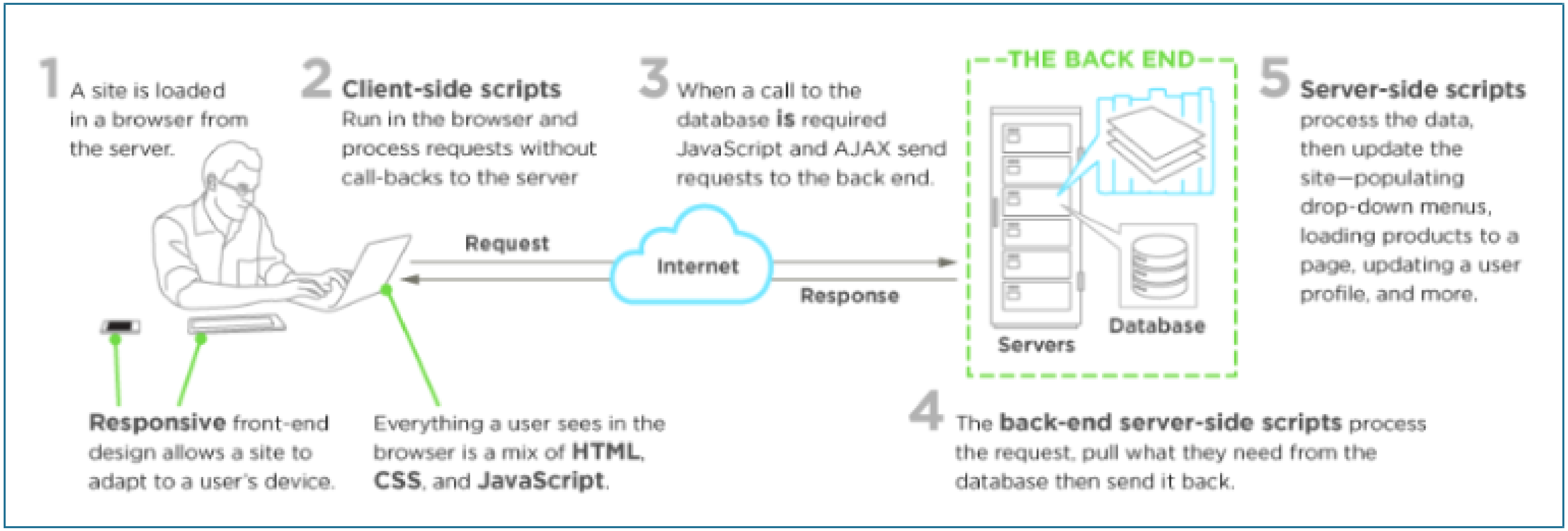
What is JavaScript

axess academy

- JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive.
- When a JavaScript is placed inside a web page, the browser loads the page & built-in interpreter reads the JavaScript code & execute
- JavaScript is case sensitive and dynamic typing, loosely typed i.e. variable data types are not declared
- JavaScript is used in Web pages for
  - Validating data.
  - Putting dynamic content into an HTML page.
  - To make them interactive

Client and Server-side scripting

axess academy

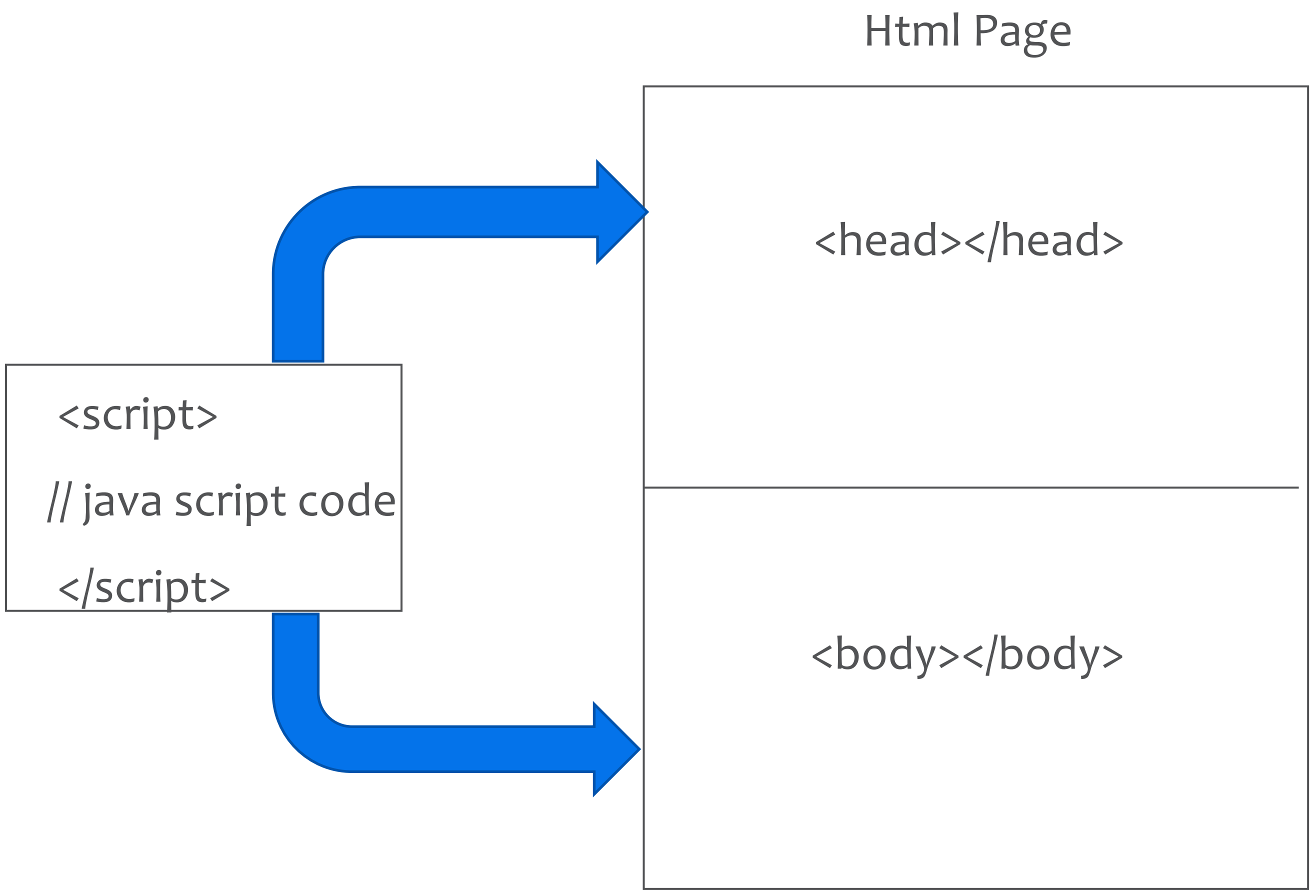

1 A site is loaded in a browser from the server.
2 **Client-side scripts** Run in the browser and process requests without call-backs to the server
3 When a call to the database is required JavaScript and AJAX send requests to the back end.
THE BACK END
5 **Server-side scripts** process the data, then update the site—populating drop-down menus, loading products to a page, updating a user profile, and more.
Request
Internet
Response
Servers
Database
Responsive front-end design allows a site to adapt to a user's device.
Everything a user sees in the browser is a mix of **HTML**, **CSS**, and **JavaScript**.
4 The **back-end server-side scripts** process the request, pull what they need from the database then send it back.

Let us start with JS

Where to write java script

axess academy

- Inline Scripting : <script></script>
- External Scripting: <script src="myjs.js"> </script>

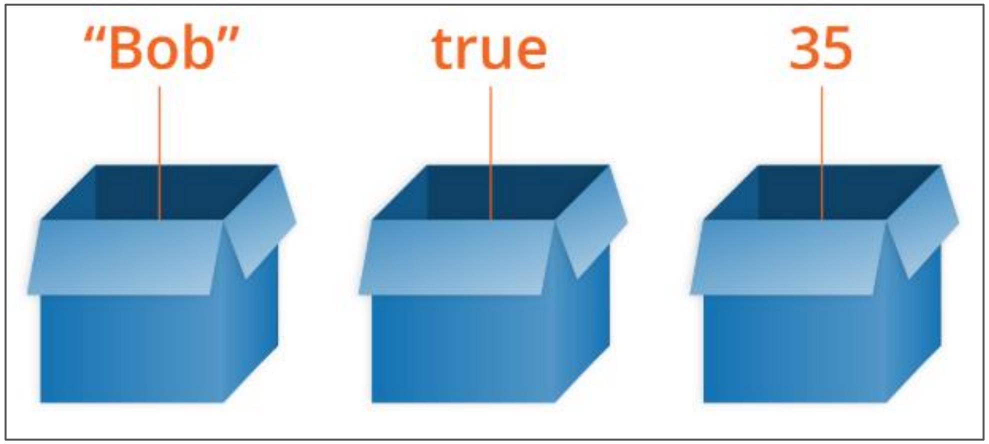


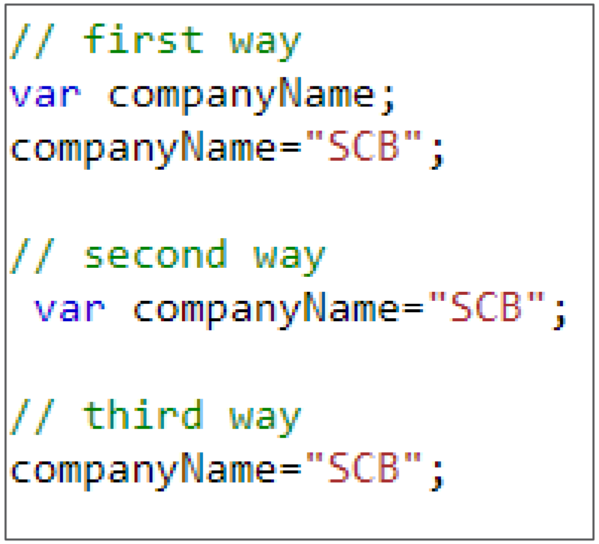
Module 2: Java Script Essentials

Variable and variable declarations

- A variable is a container for a value, like a number we might use in a sum, or a string that we might use as part of a sentence.

- Declares a variable, optionally initializing it to a value.

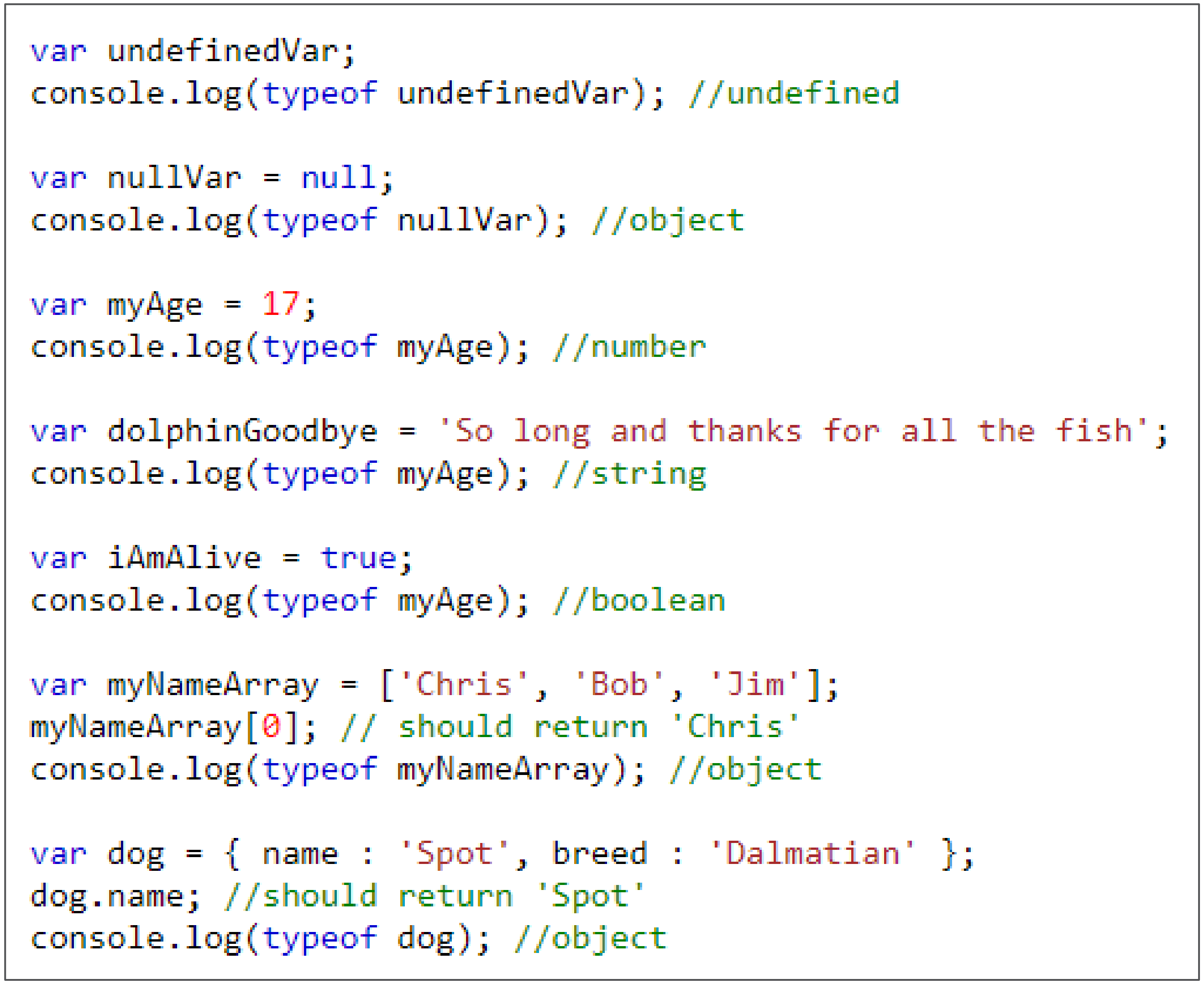




```
// first way
var companyName;
companyName="SCB";

// second way
 var companyName="SCB";

// third way
companyName="SCB";
```

Different ways to declare a variable

- undefined
- null
- number
- string
- boolean
- array
- object



```
var undefinedVar;
console.log(typeof undefinedVar); //undefined

var nullVar = null;
console.log(typeof nullVar); //object

var myAge = 17;
console.log(typeof myAge); //number

var dolphinGoodbye = 'So long and thanks for all the fish';
console.log(typeof myAge); //string

var iAmAlive = true;
console.log(typeof myAge); //boolean

var myNameArray = ['Chris', 'Bob', 'Jim'];
myNameArray[0]; // should return 'Chris'
console.log(typeof myNameArray); //object

var dog = { name : 'Spot', breed : 'Dalmatian' };
dog.name; //should return 'Spot'
console.log(typeof dog); //object
```

Manipulating Strings in JavaScript

axess academy

- Create a string by enclosing a sequence of characters in quotes.

1. Single Quotes (' ')
2. Double Quotes (" ")
3. Backticks (‘ ‘)

```
let sone =  'Hello SCB';
//output: Hello SCB

let stwo = "Welcome to the Bootcamp";
//output: Welcome to the Bootcamp

let text = `It's secured "Bank"`;
//output:  It's secured "Bank"
```

String Methods

axess academy

- JavaScript provides several methods that you can use to manipulate strings. Let's look at some of the most used methods:

Methods	Description
String.charAt()	Returns a string representing the character at the given index.
String.charCodeAt()	Returns a number representing the UTF-16 code unit value of the character at the given index.
String.concat()	Returns a new string containing the concatenation of the given strings.
String.endsWith()	Returns true if the string ends with the given string, otherwise false.
String.includes()	Returns true if the string contains the given string, otherwise false.
String.indexOf()	Returns the index within the string of the first occurrence of the specified value, or -1 if not found.
String.startsWith()	Returns true if the string starts with the given string, otherwise false.
String.toLowerCase()	Returns a new string with all the uppercase characters converted to lowercase.
String.toUpperCase()	Returns a new string with all the lowercase characters converted to uppercase.
String.toString()	Returns the string representation of the specified object.

String Methods cont..

axess academy

Methods	Description
String.lastIndexOf()	Returns the index within the string of the last occurrence of the specified value, or -1 if not found.
String.match()	Returns a list of matches of a regular expression against a string.
String.matchAll()	Returns a list of matches of a regular expression against a string.
String.padEnd()	Returns a new string with some content padded to the end of the string.
String.repeat()	Returns a new string which contains the specified number of copies of the string.
String.replace()	Returns a new string with some or all matches of a regular expression replaced by a replacement string.
String.search()	Returns the index within the string of the first occurrence of the specified value, or -1 if not found
String.substring()	Returns a new string containing the characters of the string from the given index to the end of the string.
String.trim()	Returns a new string with the leading and trailing whitespace removed.
String.trimEnd()	Returns a new string with the trailing whitespace removed.
String.trimStart()	Returns a new string with the leading whitespace removed.

Understanding JavaScript Numbers

axess academy

- JavaScript Numbers are primitive data types.
- There are two types of JavaScript numbers: integer and floating-point. An integer is a whole number, whereas a floating-point number includes a decimal point. For example, 5 is an integer, and 3.14 is a floating-point number.
- JavaScript numbers are always stored in double-precision 64-bit binary format IEEE 754.

Property	Description
MAX_VALUE	The largest number possible in JavaScript
MIN_VALUE	The smallest number possible in JavaScript
POSITIVE_INFINITY	Infinity (returned on overflow)
NEGATIVE_INFINITY	Negative infinity (returned on overflow)
NaN	A "Not-a-Number" value
EPSILON	The difference between 1 and the smallest number > 1.

Understanding JavaScript Numbers Methods

axess academy

These **number methods** can be used on all JavaScript numbers:

Method	Description
toString()	Returns a number as a string
toExponential()	Returns a number written in exponential notation
toFixed()	Returns a number written with a number of decimals
toPrecision()	Returns a number written with a specified length
valueOf()	Returns a number as a number

- In addition to the above methods, JavaScript provides various math methods such as Math.round(), Math.ceil(), Math.floor(), Math.abs(), and many more that can be used to perform mathematical operations on numbers.

JavaScript Type Conversion

- While JavaScript provides numerous ways to convert data from one type to another there are two most common data conversions:

  1. Converting Number to Strings
  2. Converting a String to Numbers

Module 3: Functions

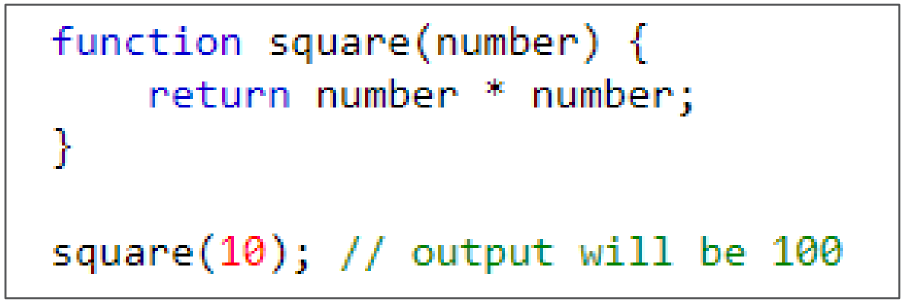
What is a Function

- Fundamental building blocks in JavaScript.
- A set of statements that performs a task or calculates a value.
- To use a function, you must define it somewhere in the scope from which you wish to call it.
- Function can be created in many ways:

    Function Declaration

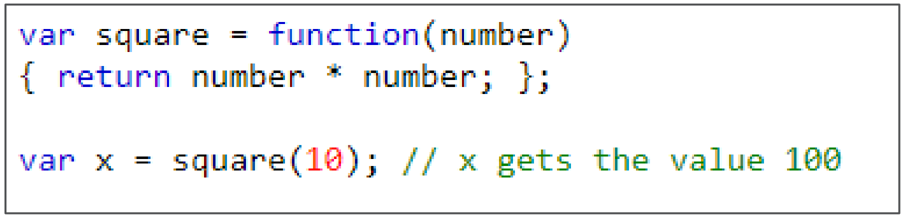
    Function Expression

Function Declaration



```
function square(number) {
    return number * number;
}

square(10); // output will be 100
```

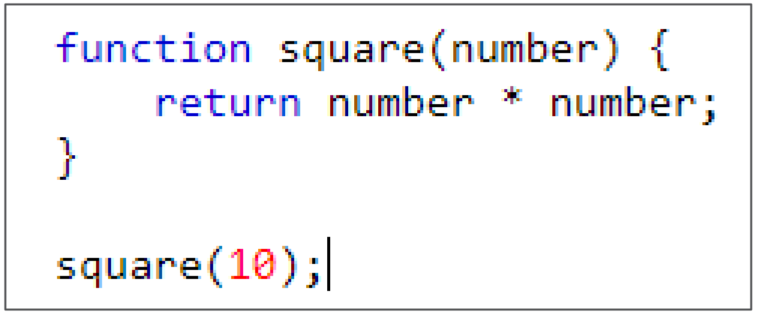
Function Expression



```
var square = function(number)
{ return number * number; };

var x = square(10); // x gets the value 100
```

Function Declaration

- A function declaration consists of the function keyword, followed by:
  - The name of the function.
  - A list of parameters to the function, enclosed in parentheses and separated by commas.
  - The JavaScript statements that define the function, enclosed in curly brackets, { }.



```
function square(number) {
    return number * number;
}

square(10);
```

Function Expression

axess academy

Anonymous Function expression

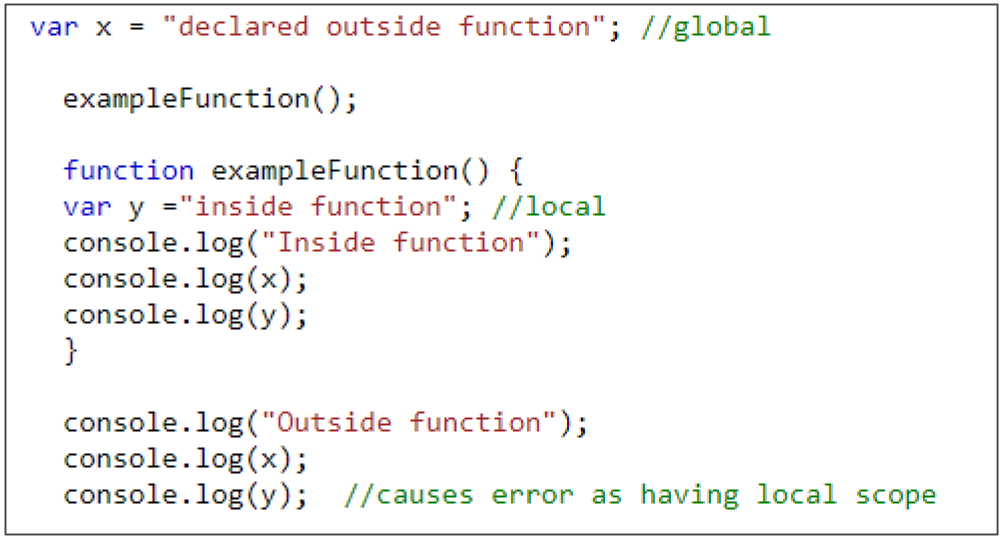
```
var square = function(number)
{ return number * number; };

var x = square(10); // x gets the value 100
```

Named Function Expression

```
var factorial = function fac(n) {
return n < 2 ? 1 : n * fac(n - 1);   };

console.log(factorial(3)); //output will be 6
```

Scoping

- When you declare a variable outside of any function, it is called a *global* variable, because it is available to any other code in the current document.
- When you declare a variable within a function, it is called a *local* variable, because it is available only within that function.



```
var x = "declared outside function"; //global

  exampleFunction();

  function exampleFunction() {
  var y ="inside function"; //local
  console.log("Inside function");
  console.log(x);
  console.log(y);
  }

  console.log("Outside function");
  console.log(x);
  console.log(y);  //causes error as having local scope
```

Hands on variable and functions

axess academy

- Create a function which displays the Fibonacci series for 10 numbers: 0,1,1,2,3,5,8,13,21,25
- Create a function which display the table of 8.

Module 4: JavaScript Multiple Values:

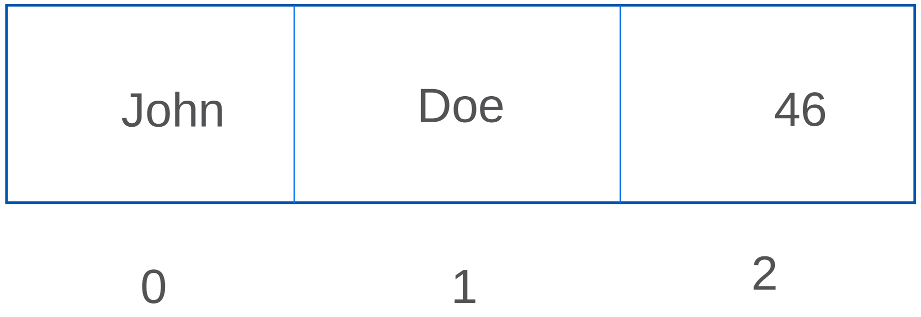
Array, String and Objects

Array

axess academy

- An array can hold many values under a single variable name, and you can access the values by referring to an index number.
- Array can consist homogenous as well as heterogenous elements.
- Iteration of the array can be done using loops.

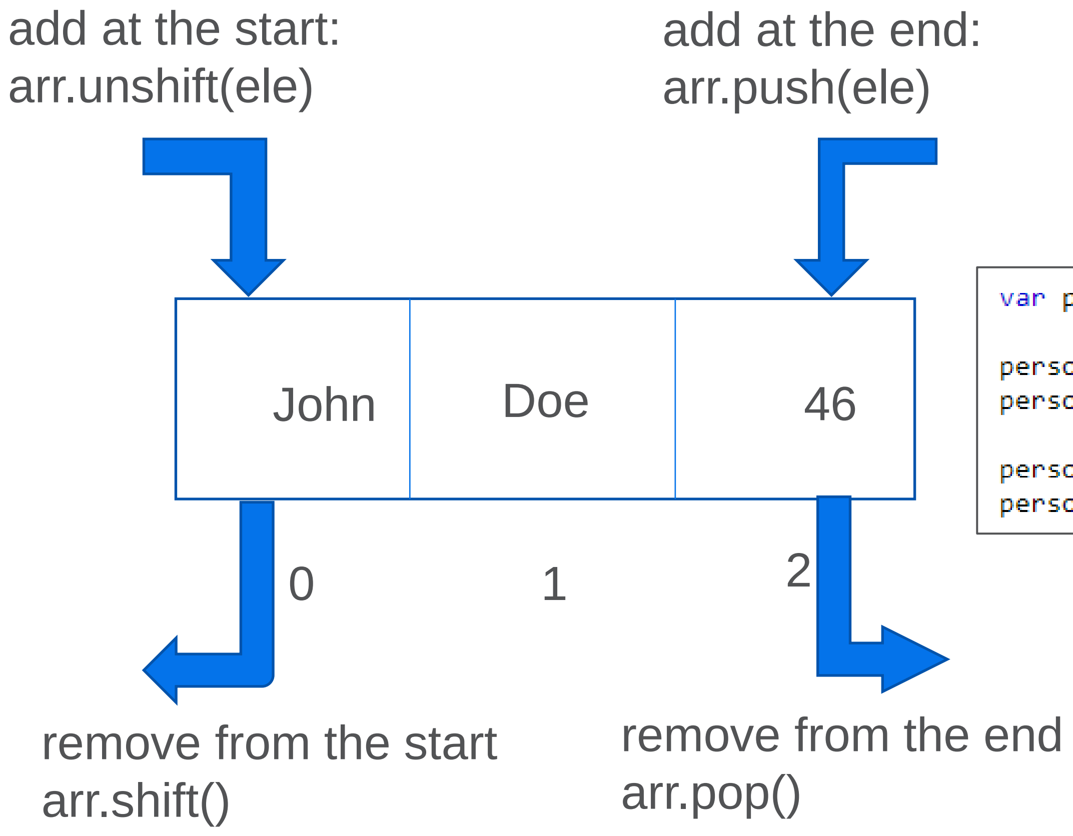
```
var person = ["John", "Doe", 46];

for(var i=0;i<person.length;i++){
  console.log(person[i]); // displays "John"", "Doe" ,46
}
```



Array Methods

axess academy

- Below are the methods to add or delete the elements from the array (starting , ending or from some index number also)



```
var person = ["John", "Doe", 46];

person.push("Developer"); //returns 4 (the new array length)
person.pop(); // returns Developer

person.unshift(101); //returns 4 (the new array length)
person.shift(); //returns 101
```

Array Methods cont…

axess academy

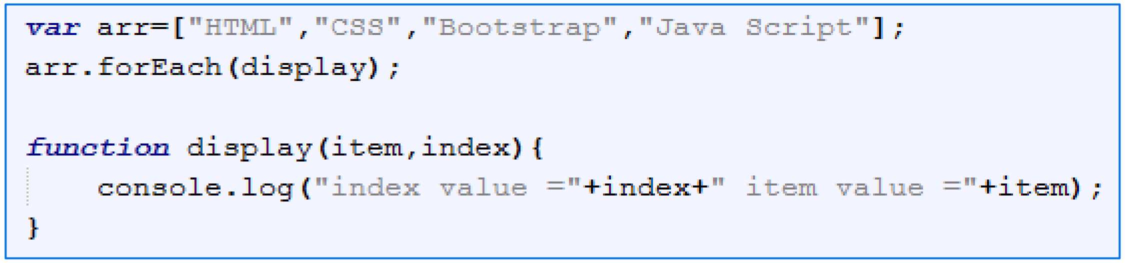
Scenarios	Code	Output

Reverse of an array	var arr= [24,27,20,12,28];
	arr.reverse();
		28,12,20,27,24

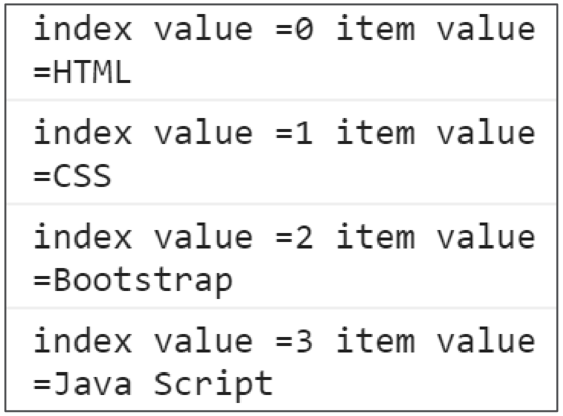
Sorting of an array	arr.sort()	12,20,24,27,28
Converting an array into a string	arr.toString()	“24,27,20,12,28”

Array Methods cont…

- forEach() : method calls a function once for each element in an array, in order. It is a better version of for loop.

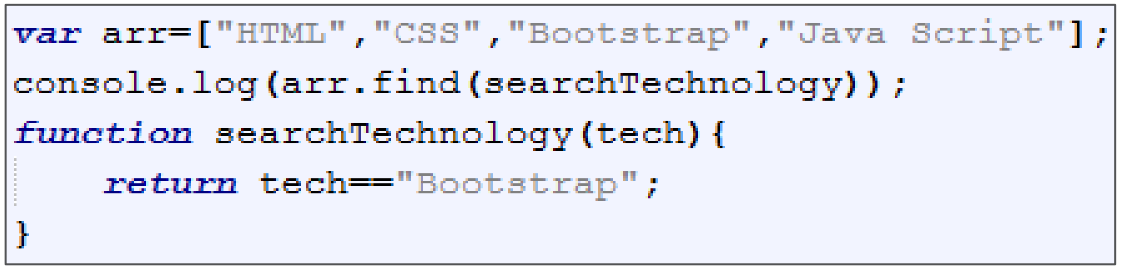


```
var arr=["HTML","CSS","Bootstrap","Java Script"];
arr.forEach(display);

function display(item,index){
    console.log("index value ="+index+" item value ="+item);
}
```

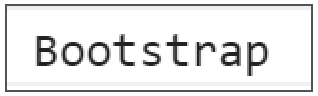


```
index value =0 item value
=HTML
index value =1 item value
=CSS
index value =2 item value
=Bootstrap
index value =3 item value
=Java Script
```

- The find() method executes the function once for each element present in the array:
  - If it finds an array element where the function returns a *true* value, find() returns the value of that array element (and does not check the remaining values) otherwise it returns undefined



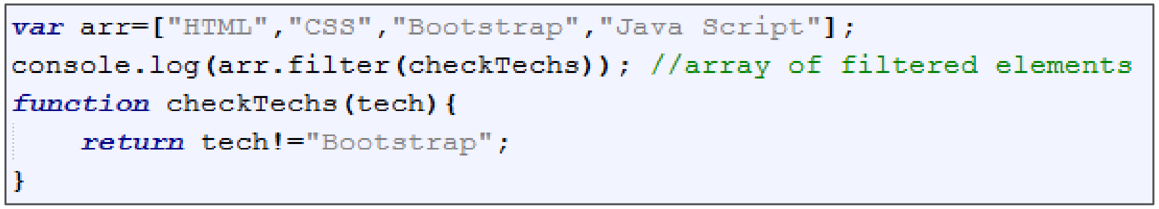
```
var arr=["HTML","CSS","Bootstrap","Java Script"];
console.log(arr.find(searchTechnology));
function searchTechnology(tech){
    return tech=="Bootstrap";
}
```



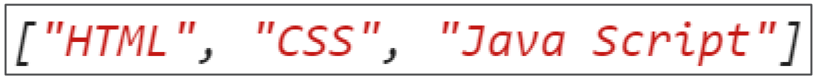
```
Bootstrap
```

Array Methods cont…

- The filter() method creates an array filled with all array elements that pass a test (provided as a function).

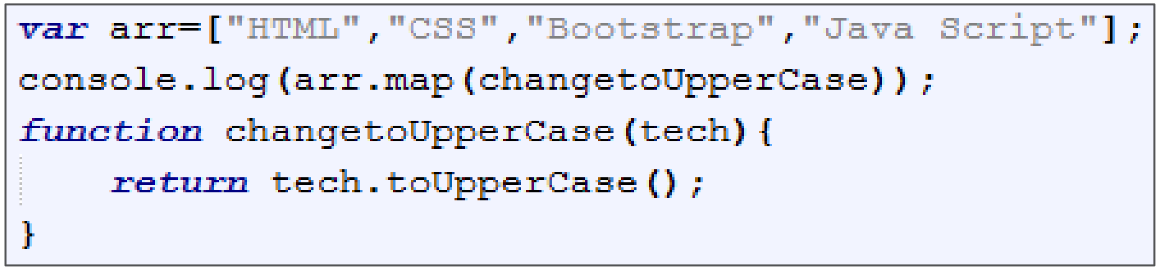


```
var arr=["HTML","CSS","Bootstrap","Java Script"];
console.log(arr.filter(checkTechs)); //array of filtered elements
function checkTechs(tech){
    return tech!="Bootstrap";
}
```

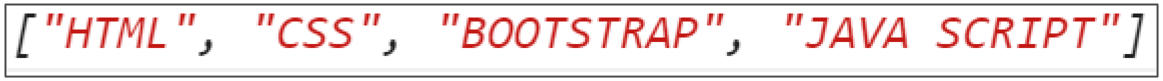


```
["HTML", "CSS", "Java Script"]
```

- The map() method creates a new array with the results of calling a function for every array element. It calls the provided function once for each element in an array, in order.



```
var arr=["HTML","CSS","Bootstrap","Java Script"];
console.log(arr.map(changetoUpperCase));
function changetoUpperCase(tech){
    return tech.toUpperCase();
}
```



```
["HTML", "CSS", "BOOTSTRAP", "JAVA SCRIPT"]
```

String and String methods

axess academy

JavaScript strings are used for storing and manipulating text. Below are various methods for string. Syntax:
var str=“company”;
Below are the list of methods supported in string :

Scenarios	**Code**	**Output**
Find the length of string	var str= “company”; str.length;	7

Change to upper or lower case	str.toLowerCase();
	str.toUpperCase();
		company
		COMPANY

To get a character at specified index	str.charAt[3];
	str[3];
		p
		p

Trims whitespace from both sides of a string	str.trim();	company

Finding a substring inside a string and extracting it	str.indexOf(“pan”);
	Str.slice(3,5);
		3 if not found then -1
		pan

Object

- An object is a collection of related data and/or functionality (which usually consists of several variables and functions — which are called properties and methods when they are inside objects.)

Declaration of object

```
var person = {
 name: ['Bob', 'Smith'],
 age: 32,
 gender: 'male',
 interests: ['music', 'skiing'],
 bio: function() {
     alert(this.name[0] + ' ' + this.name[1] + ' is ' + this.age + ' years old. He likes
   ' + this.interests[0] + ' and ' + this.interests[1] + '.');
    },
 greeting: function() {
     alert('Hi! I\'m ' + this.name[0] + '.');
}
};
```

Accessing the object properties

```
person.name[0];
person.age;
person.interests[1];
person.bio();
person.greeting();
```

Hands on Array, String and Object

axess academy

- Create an array of Strings which contains the values like [“sTandarD”,”CharTered”,”banK”] then replace the array values with corresponding Uppercase values only.
  [“STANDARD”,”CHARTERED”,”BANK”]

axess academy

Module 5: Event and Event Handler

Events

axess academy

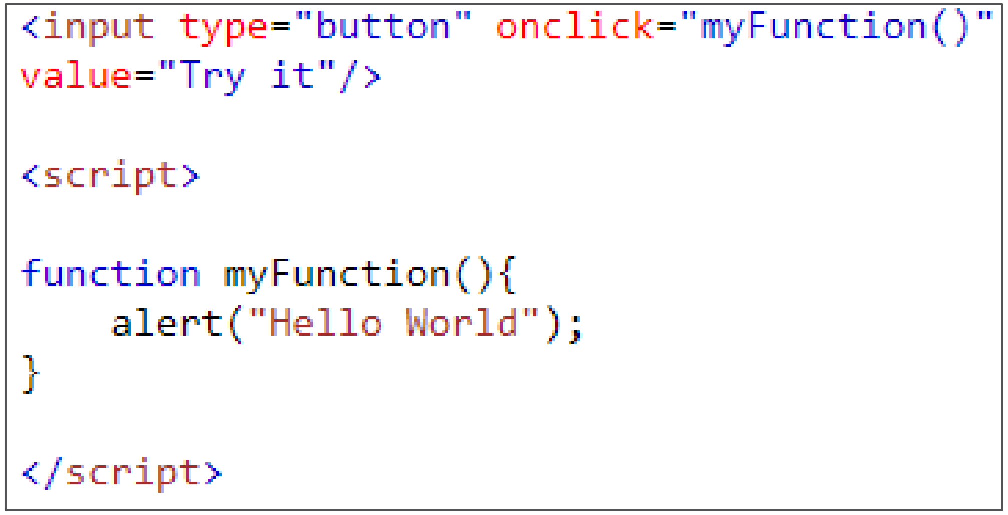
- Events are actions or occurrences that happen in the system you are programming, which the system tells you about so you can respond to them in some way if desired.
- For example, if the user clicks a button on a webpage, you might want to respond to that action by displaying an information box.
- Below table consists types of events-

Mouse Events	**Keyboard Events**	**Form Events**	**Document/Window Events**
onclick	onkeypress	onsubmit	onload
ondblclick	onkeydown	onchange	onunload
onmouseenter	onkeyup	onfocus	onscroll
onmouseleave		onblur	

Event Handler

axess academy

- Each available event has an event handler, which is a block of code (usually a user-defined JavaScript function) that will be run when the event fires.
- When such a block of code is defined to be run in response to an event firing, we say we are registering an event handler. Event handlers are sometimes called as event listeners



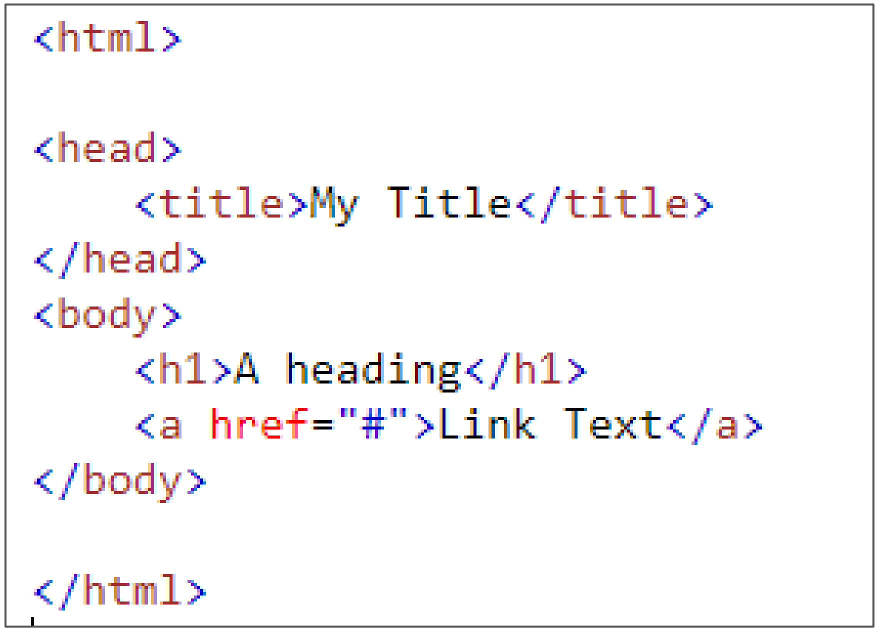
```
<input type="button" onclick="myFunction()"
value="Try it"/>

<script>

function myFunction(){
    alert("Hello World");
}

</script>
```

Module 6: DOM and Form Manipulation

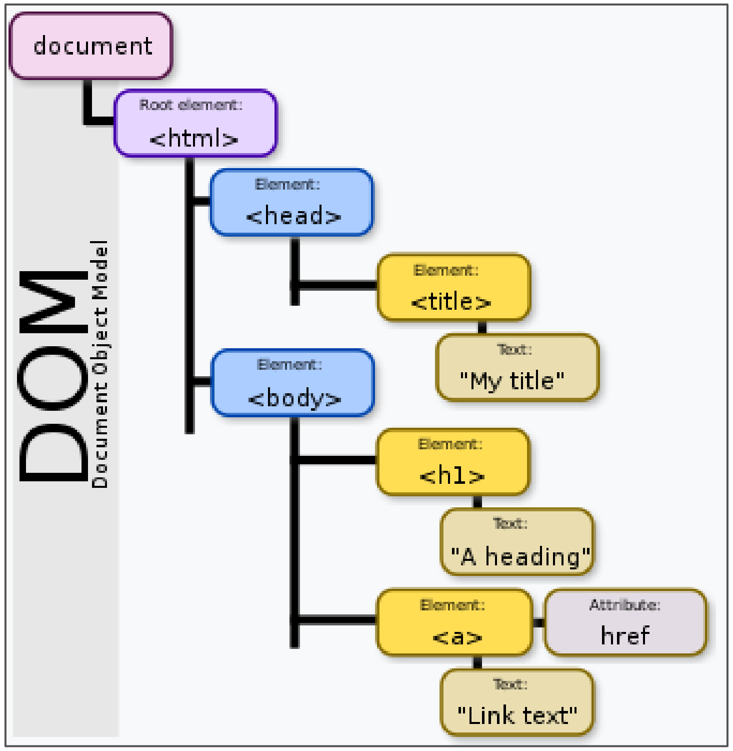
Document Object Model

axess academy

The **Document Object Model (DOM)** is a cross-platform and language-independent application programming interface that treats an HTML document as a tree structure wherein each node is an object representing a part of the document. The DOM represents a document with a logical tree.

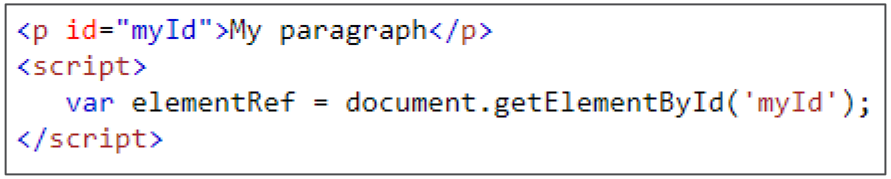


```
<html>

<head>
    <title>My Title</title>
</head>
<body>
    <h1>A heading</h1>
    <a href="#">Link Text</a>
</body>

</html>
```



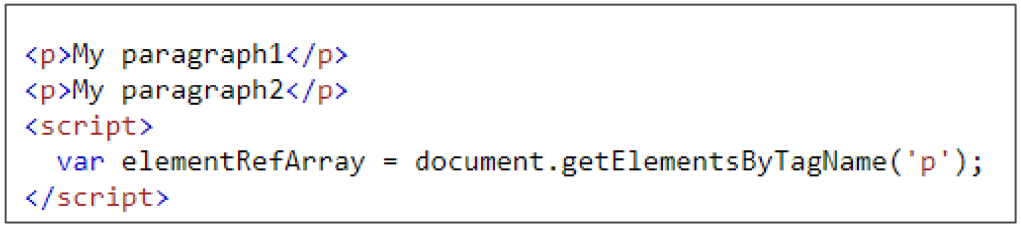
DOM Methods

- document.getElementById() : returns an element with a given id attribute value.



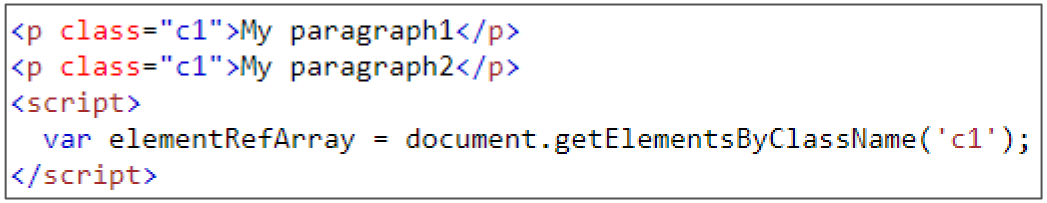
```
<p id="myId">My paragraph</p>
<script>
   var elementRef = document.getElementById('myId');
</script>
```

- document.getElementsByTagName() : returns an array containing all the elements on the page of a given type.



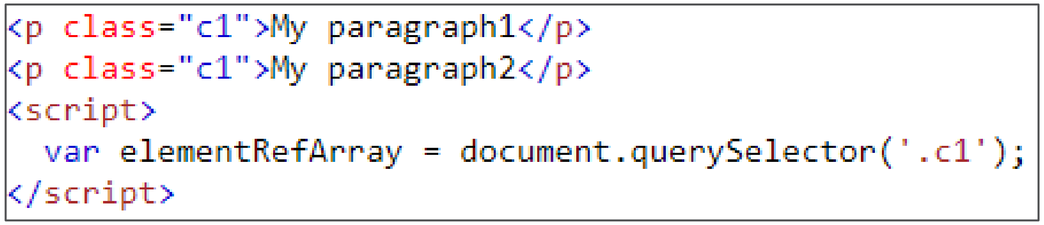
```
<p>My paragraph1</p>
<p>My paragraph2</p>
<script>
  var elementRefArray = document.getElementsByTagName('p');
</script>
```

- document.getElementsByClassName() : returns an array containing all the elements on the page of a given class.



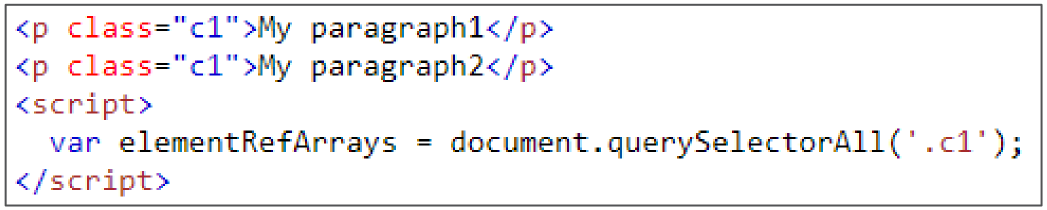
```
<p class="c1">My paragraph1</p>
<p class="c1">My paragraph2</p>
<script>
  var elementRefArray = document.getElementsByClassName('c1');
</script>
```

- document.querySelector() : returns the very first matching element as per the selector.



```
<p class="c1">My paragraph1</p>
<p class="c1">My paragraph2</p>
<script>
  var elementRefArray = document.querySelector('.c1');
</script>
```

- document.querySelectorAll() : returns the all matching elements as per the selector.



```
<p class="c1">My paragraph1</p>
<p class="c1">My paragraph2</p>
<script>
  var elementRefArrays = document.querySelectorAll('.c1');
</script>
```

DOM Manipulation

axess academy

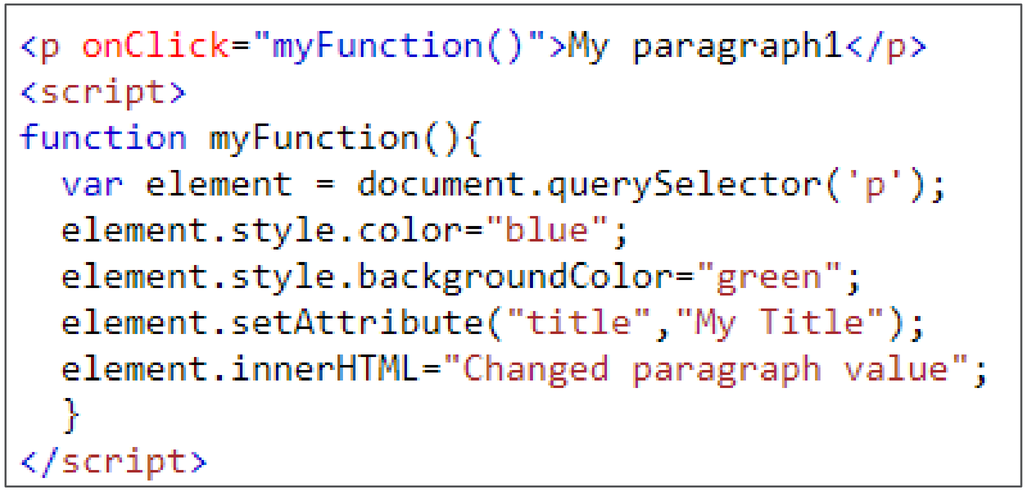
In DOM manipulation, we can modify below fields on existing DOM tree:

- Element CSS properties:

  ele.style.color=“blue”;
- Element Attribute values:

  ele.setAttribuite(“width”,”100px”);
- Element values

  ele.innerHTML=“Updated value”;

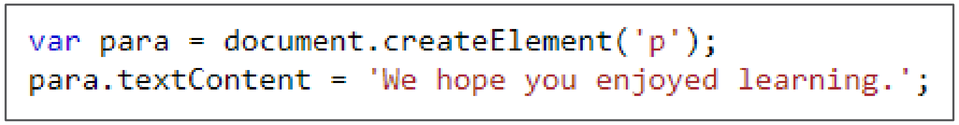


```
<p onClick="myFunction()">My paragraph1</p>
<script>
function myFunction(){
  var element = document.querySelector('p');
  element.style.color="blue";
  element.style.backgroundColor="green";
  element.setAttribute("title","My Title");
  element.innerHTML="Changed paragraph value";
  }
</script>
```

Creation or Deletion of nodes in DOM :

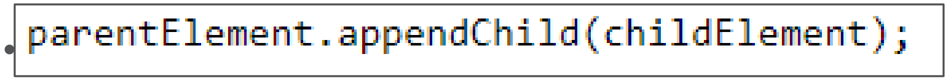
Let us learn how to create, add, replace or remove the node

- Element creation:

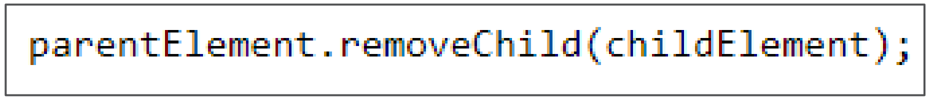


```
var para = document.createElement('p');
para.textContent = 'We hope you enjoyed learning.';
```

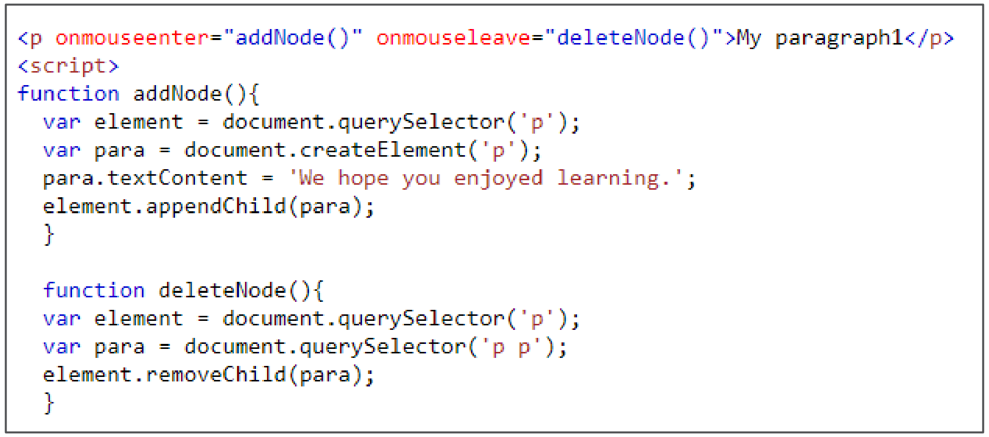
- Element addition in tree:



```
parentElement.appendChild(childElement);
```



```
parentElement.removeChild(childElement);
```



```
<p onmouseenter="addNode()" onmouseleave="deleteNode()">My paragraph1</p>
<script>
function addNode(){
  var element = document.querySelector('p');
  var para = document.createElement('p');
  para.textContent = 'We hope you enjoyed learning.';
  element.appendChild(para);
  }

  function deleteNode(){
  var element = document.querySelector('p');
  var para = document.querySelector('p p');
  element.removeChild(para);
  }
```

Regular Expression Pattern and Quantifiers

axess academy

Pattern	Description
[0-9]	Find any of the digits between the brackets
[A-Z]	Find any of the characters (capital letters) between the brackets
[0-9a-z]	Find any of the alphanumeric characters between the brackets
(x|y)	Find any of the alternatives separated with |

Quantifier	Description
[0-9]+	Matches any string that contains at least one *digit*
[0-9]*	Matches any string that contains zero or more occurrences of *n*
[0-9]?	Matches any string that contains zero or one occurrences of *n*
[0-9]{3}	Matches any string that contains exactly 3 occurrences of *n*
[0,9]{3,6}	Matches any string that contains minimum 3 occurrences of *n and max 6 occurrences*

- **Example 1**: To check name field can only have alphabets.
- The pattern can be :
  - /^[A-Za-z]+$/ or
  - “^[A-Za-z]+$”
  - /^[a-z]+$/i (case insensitive)
- **Example 2**: To check contact field can only have numbers of 10 digits.
- The pattern can be :
  - /^[7-9]{1}[0-9]{9}$/ or
  - “^[7-9]{1}[0-9]{9}$”

Form Validation

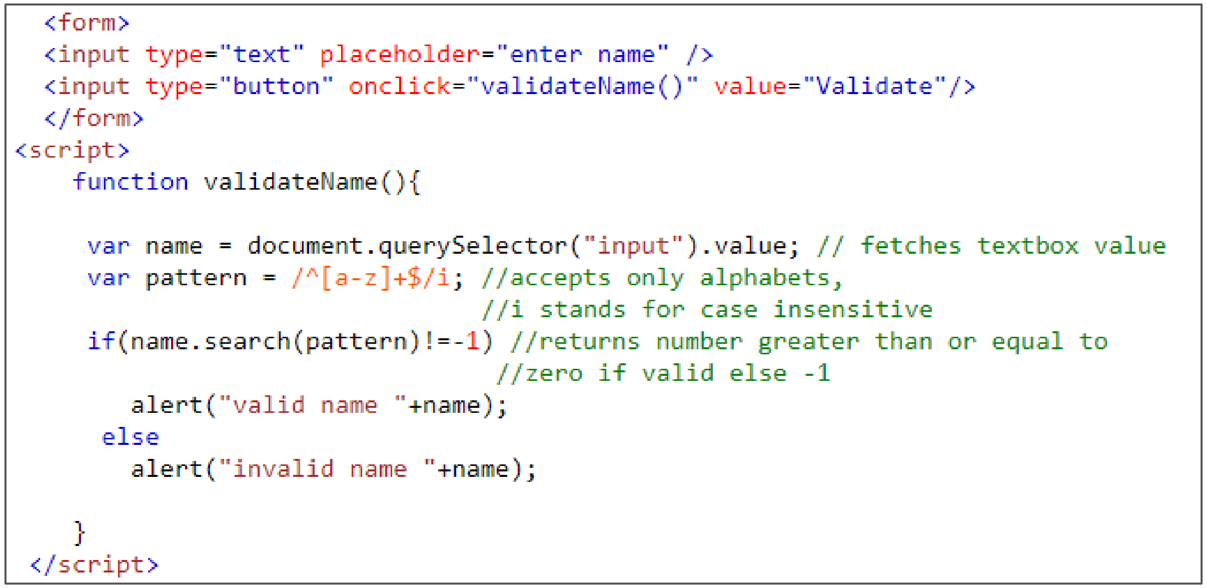
axess academy

- Fetch Form element values:

  Syntax: formelement.value

- The search() method that tests for a match in a string. It returns the index of the match, or -1 if the search fails.

  Syntax: name.search(pattern)



```
  <form>
  <input type="text" placeholder="enter name" />
  <input type="button" onclick="validateName()" value="Validate"/>
  </form>
<script>
    function validateName(){

     var name = document.querySelector("input").value; // fetches textbox value
     var pattern = /^[a-z]+$/i; //accepts only alphabets,
                                //i stands for case insensitive
     if(name.search(pattern)!=-1) //returns number greater than or equal to
                                  //zero if valid else -1
        alert("valid name "+name);
      else
        alert("invalid name "+name);

    }
 </script>
```

DOM and Form validation Hands-on

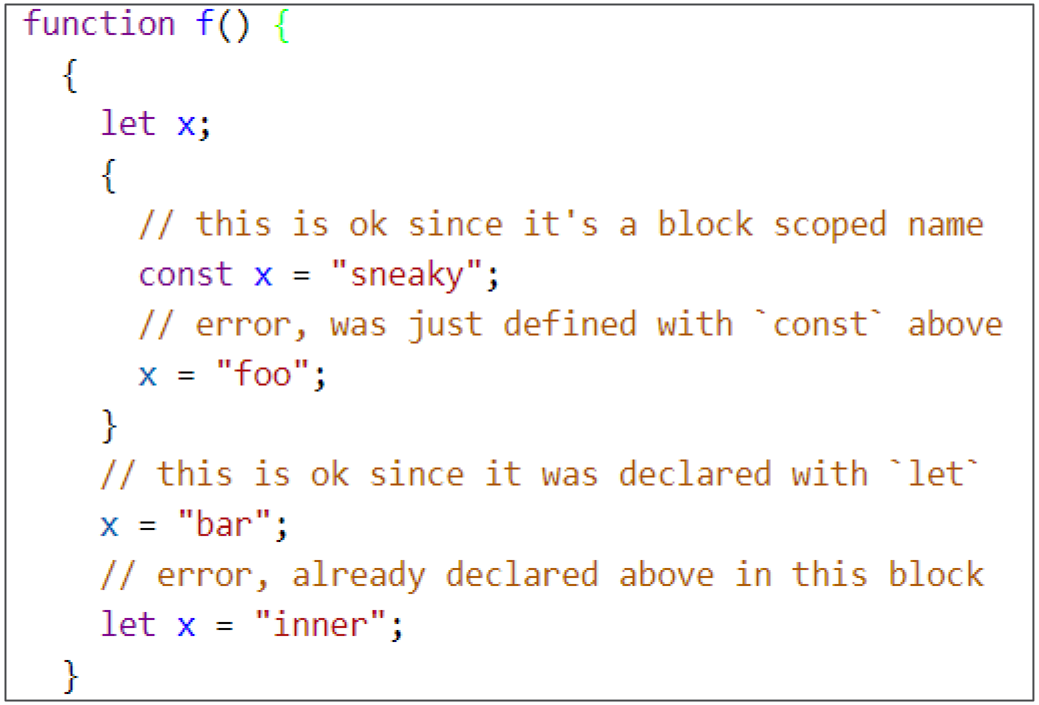
- Create an array of strings and display its values in list format in browser.
- Take input from the name text box and validate to check if it contains only alphabets or not. If it does not contain, display the error message next to the text box “please enter only alphabets”.

Module 7: Advanced JavaScript

- Let and const keywords
- Arrow functions
- class and object
- Destructuring
- The Spread Operator
- Promises
- The Fetch API
- Import and Export

let and const keyword

- Both are block scoped i.e. will be accessible only inside the block wherever they are declared.
- let is the new var and const is single-assignment.
- let declares a block-scoped, local variable, optionally initializing it to a value.
- const declares a block-scoped, read-only named constant.



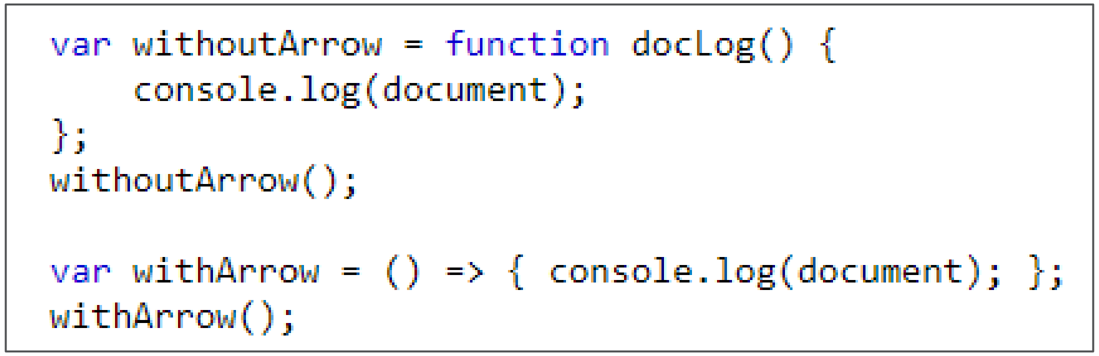
```
function f() {
  {
    let x;
    {
      // this is ok since it's a block scoped name
      const x = "sneaky";
      // error, was just defined with `const` above
      x = "foo";
    }
    // this is ok since it was declared with `let`
    x = "bar";
    // error, already declared above in this block
    let x = "inner";
  }
```

Arrow functions

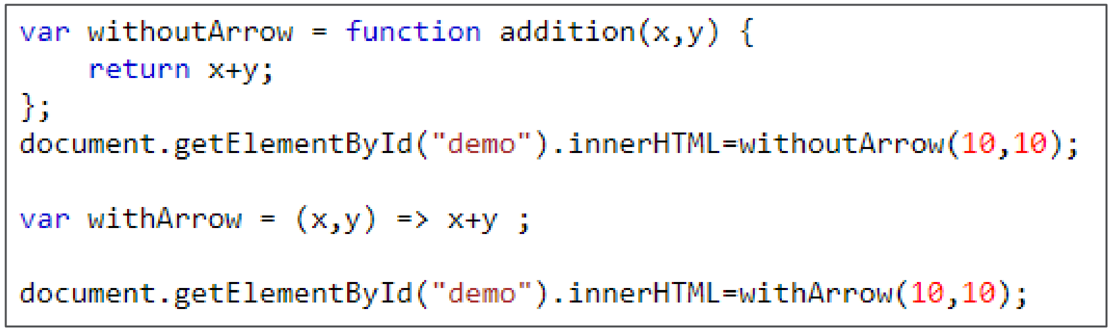
axess academy

- Arrow functions were introduced as a new syntax for writing JavaScript functions.
- They save developers time and simplify function scope.
- They use a token => that looks like a fat arrow
- Syntax:

  (param1,param2) => { //expression statements}



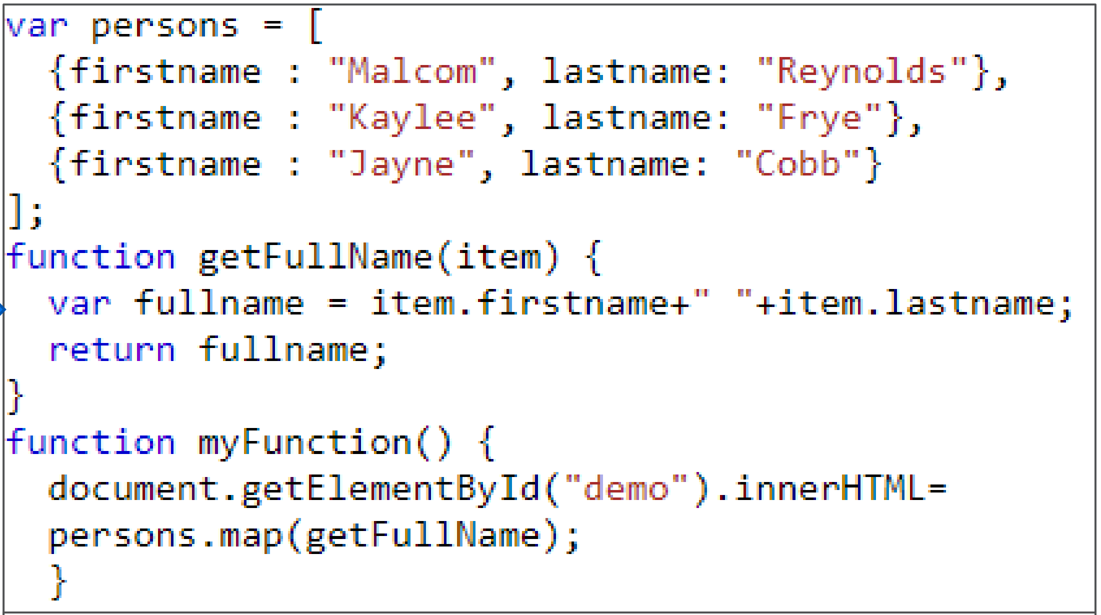
```
var withoutArrow = function docLog() {
    console.log(document);
};
withoutArrow();

var withArrow = () => { console.log(document); };
withArrow();
```



```
var withoutArrow = function addition(x,y) {
    return x+y;
};
document.getElementById("demo").innerHTML=withoutArrow(10,10);

var withArrow = (x,y) => x+y ;

document.getElementById("demo").innerHTML=withArrow(10,10);
```

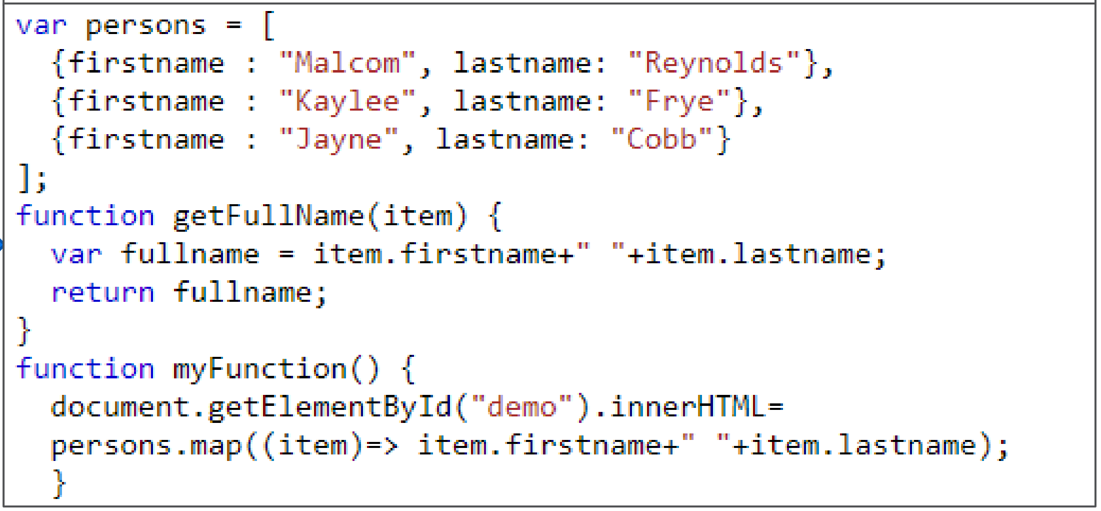
More Examples on Arrow Functions

Without Arrow Functions



```
var persons = [
  {firstname : "Malcom", lastname: "Reynolds"},
  {firstname : "Kaylee", lastname: "Frye"},
  {firstname : "Jayne", lastname: "Cobb"}
];
function getFullName(item) {
  var fullname = item.firstname+" "+item.lastname;
  return fullname;
}
function myFunction() {
  document.getElementById("demo").innerHTML=
  persons.map(getFullName);
  }
```

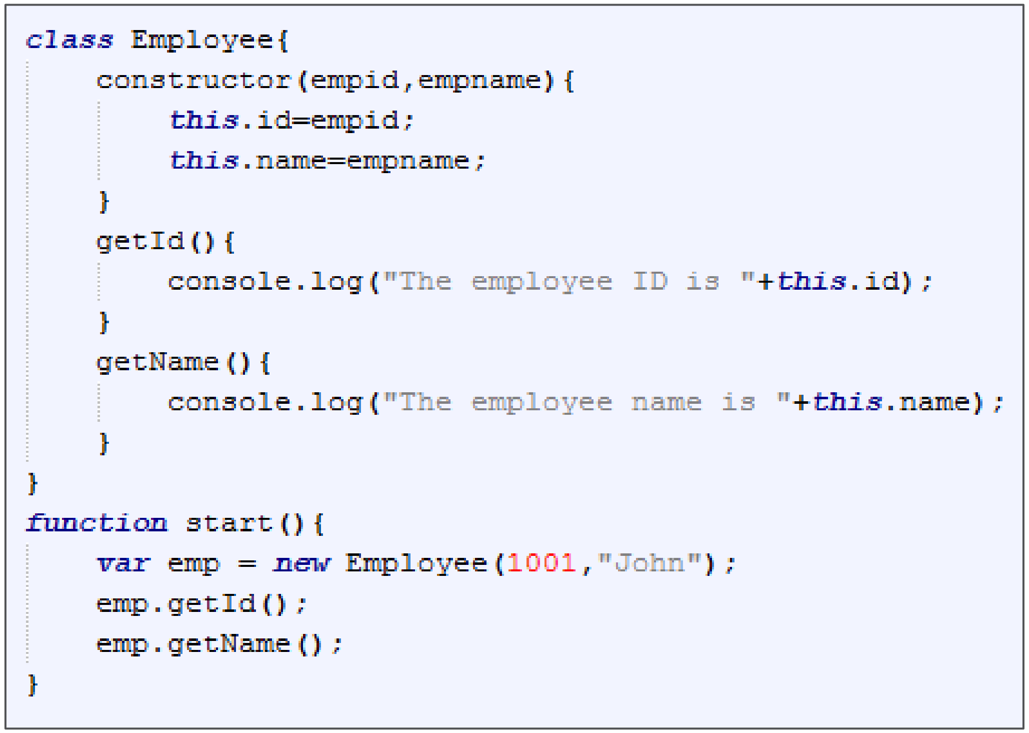
With Arrow Functions



```
var persons = [
  {firstname : "Malcom", lastname: "Reynolds"},
  {firstname : "Kaylee", lastname: "Frye"},
  {firstname : "Jayne", lastname: "Cobb"}
];
function getFullName(item) {
  var fullname = item.firstname+" "+item.lastname;
  return fullname;
}
function myFunction() {
  document.getElementById("demo").innerHTML=
  persons.map((item)=> item.firstname+" "+item.lastname);
  }
```

Class and Object

- A class is a blueprint that contains the variables and the methods.
- An object is an instance of class.
- A class contains:
  - Field / Data members
  - Constructors
  - Member functions



```
class Employee{
    constructor(empid,empname){
        this.id=empid;
        this.name=empname;
    }
    getId(){
        console.log("The employee ID is "+this.id);
    }
    getName(){
        console.log("The employee name is "+this.name);
    }
}
function start(){
    var emp = new Employee(1001,"John");
    emp.getId();
    emp.getName();
}
```

Destructuring

axess academy

- Destructuring helps to unpack values from arrays and objects.
- Assigns them to separate variable.

Without Destructuring

```
//Employee Object
    const employee = {
        name:"John",
        age:30,
        department: "Engineering"
    };

//Accessing properties without destructuring
    const name = employee.name;
    const age = employee.age;
    const department = employee.department;
```

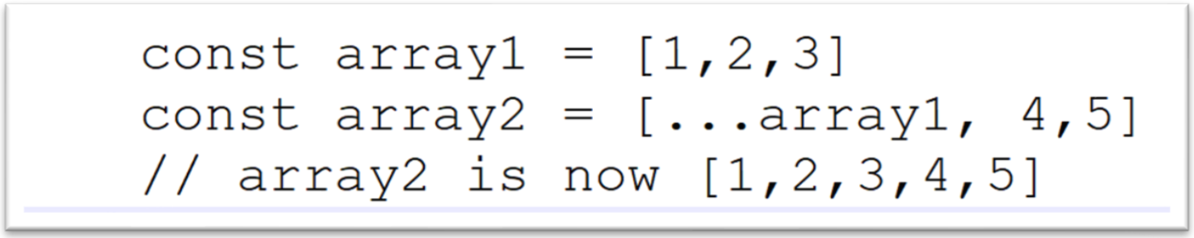
With Destructuring

```
//With Destructuring
    const employee = {
        name:"John",
        age:30,
        department: "Engineering"
    };

//Destructuring
    const { name, age, department } = employee;
```

The Spread Operator

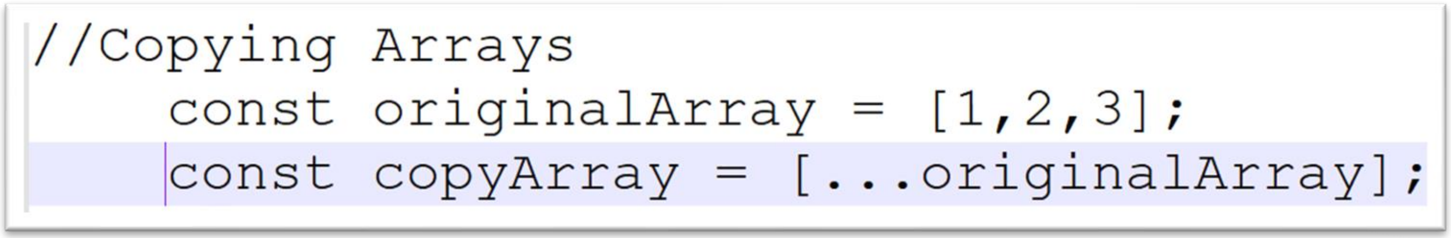
- It takes an iterable and expands it into individual elements
- The Spread operator is (‘**…**’)

Example 1



```
    const array1 = [1,2,3]
    const array2 = [...array1, 4,5]
    // array2 is now [1,2,3,4,5]
```

Example 2

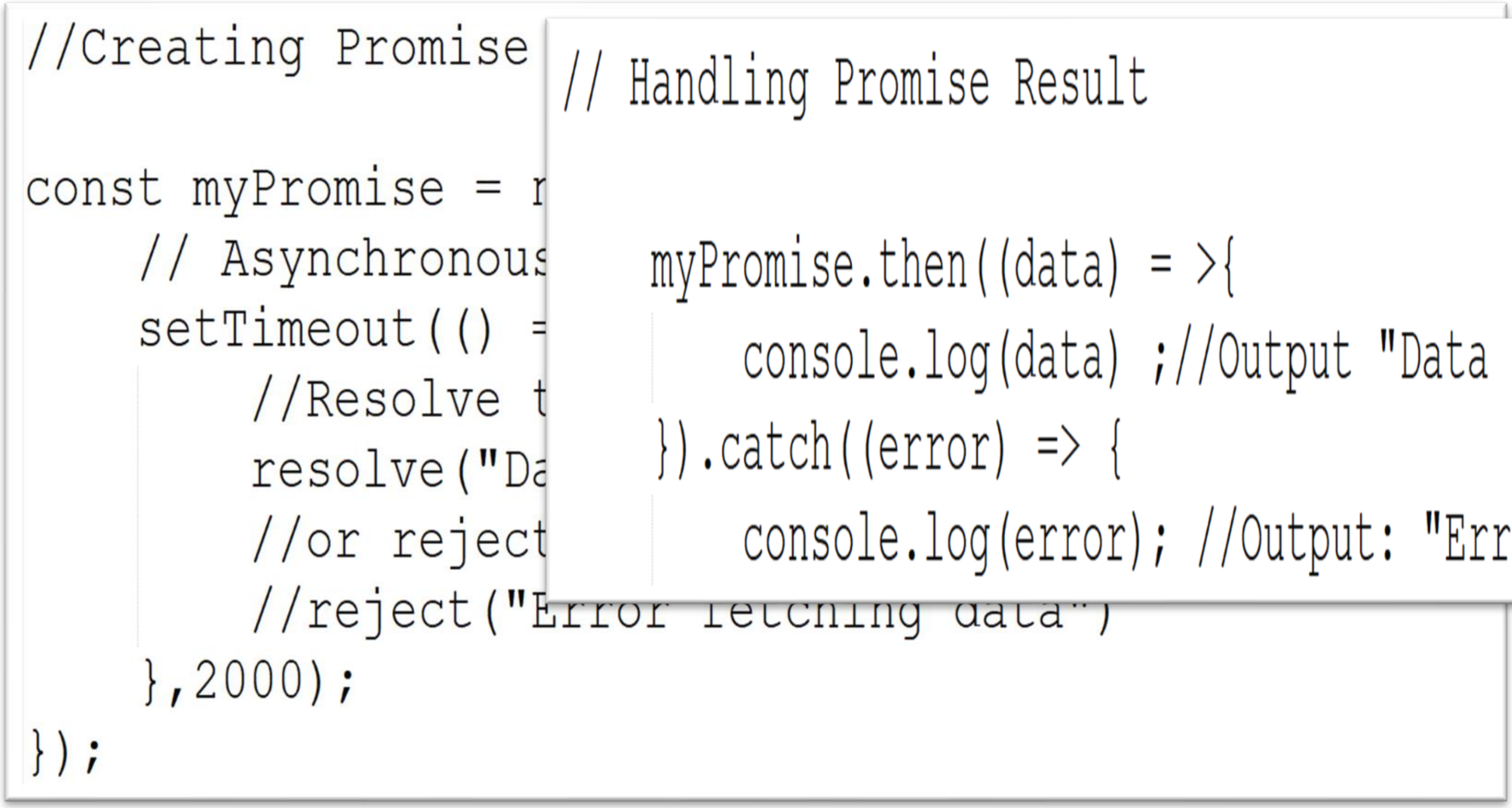


```
//Copying Arrays
    const originalArray = [1,2,3];
    const copyArray = [...originalArray];
```

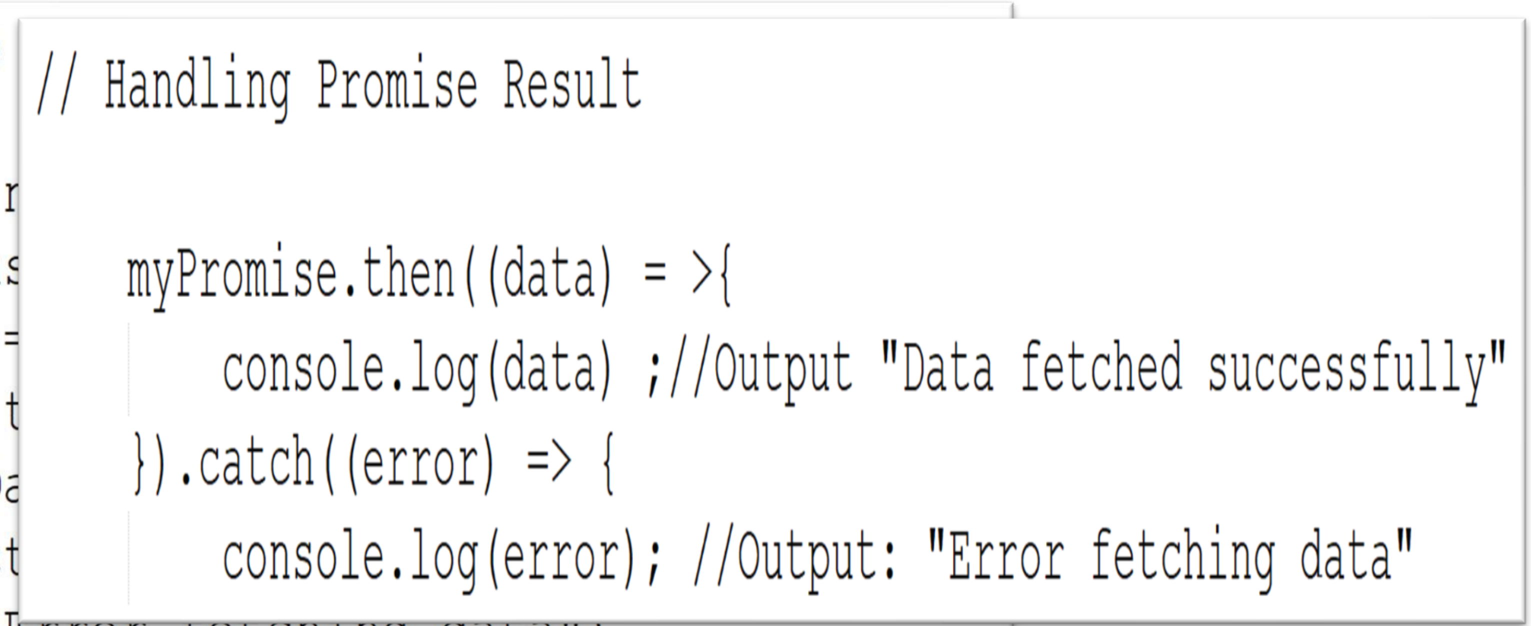
Promises

axess academy

- Promises in JavaScript are a way to handle asynchronous operations.
- They represent a value which may be available now, or in the future, or never.
- Promises are used to handle the results of asynchronous operations, such as fetching data from server, reading files.



```
//Creating Promise

const myPromise = r
    // Asynchronous
    setTimeout(() =
        //Resolve t
        resolve("Da
        //or reject
        //reject("Error fetching data")
    },2000);
});
```

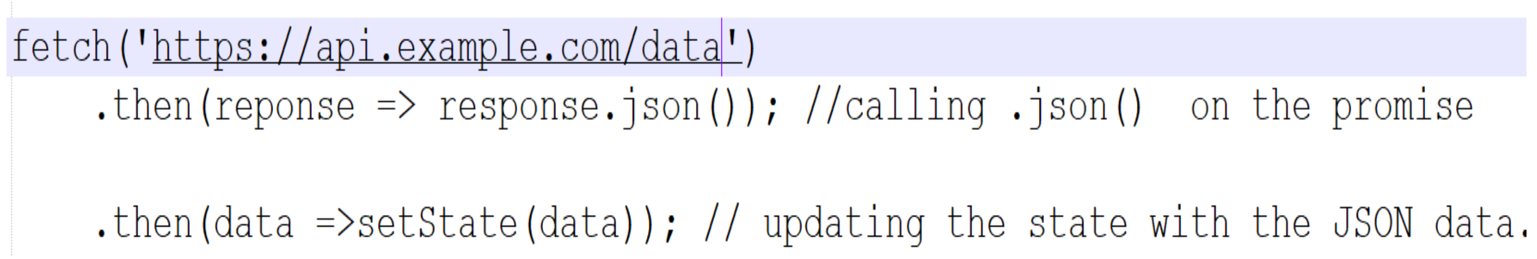


```
// Handling Promise Result


  myPromise.then((data) = >{
      console.log(data) ;//Output "Data fetched successfully"
  }).catch((error) => {
      console.log(error); //Output: "Error fetching data"
```

The Fetch API

- The Fetch API is a modern interface for fetching resources(like JSON, HTML, or images) across the network. It provides a more powerful and flexible features set than the older XMLHttpRequest (XHR) method.
- A basic fetch() takes one argument, the URL of the resource you want to fetch. It then returns another promise that resolves with a Response object.
- This **Response** object is the representation of the HTTP response.

Example:



```
fetch('https://api.example.com/data')
    .then(reponse => response.json()); //calling .json()  on the promise

    .then(data =>setState(data)); // updating the state with the JSON data.
```

Import and Export

- The export and import are the keywords used for exporting and importing one or more members in a module.
- Suppose you have a file named ‘math.js’ which contains some math-related functions that you want to use in another file.

```
// math.js
export const add = (a,b) => a + b;
export const subtract = (a,b) => a -b;
export const multiply = (a,b) => a * b;
export const divide = (a,b) => a / b;
```

- Now, let’s say you have another file named ‘main.js’ where you want to use these functions:

```
//main.js
import {add, subtract, multiply, divide } from './math.js'

console.log(add(5, 3)); //output: 8
console.log(subtract(10, 4)); //output: 6
console.log(multiply(2, 6)); //output: 12
console.log(divide(15, 3)); //output: 5
```

Import and Export contd..

- In the main.js file, we use the ‘import’ statement to bring in specific functionalites from the ‘math.js’ file. Then we can directly use those exported functions within the ‘main.js’ file.
- ‘export’ and ‘import’ can be used with various other syntaxes and configurations, such as default exports and named exports.

Async/await

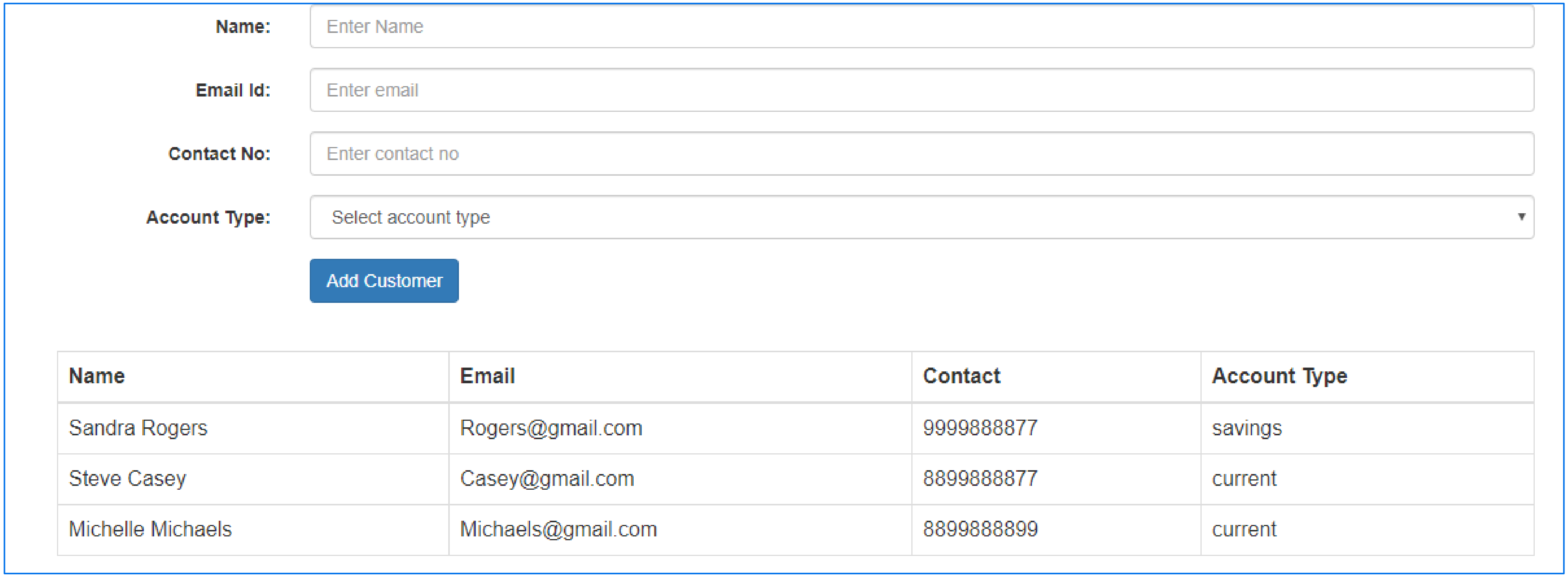
axess academy

- Async/Await functionality provides a better and cleaner way to deal with promises.
- JavaScript is synchronous in nature and async/await helps us write promise-based functions in such a way as of they were synchronous by stopping the execution of further code until the promise is resolved or rejected.

Use Case on Advanced JavaScript

Use Case on Advanced JS

axess academy



Name:	Enter Name
Email Id:	Enter email
Contact No:	Enter contact no
Account Type:	Select account type

Name	Email	Contact	Account Type
Sandra Rogers	Rogers@gmail.com	9999888877	savings
Steve Casey	Casey@gmail.com	8899888877	current
Michelle Michaels	Michaels@gmail.com	8899888899	current

Description on Use Case

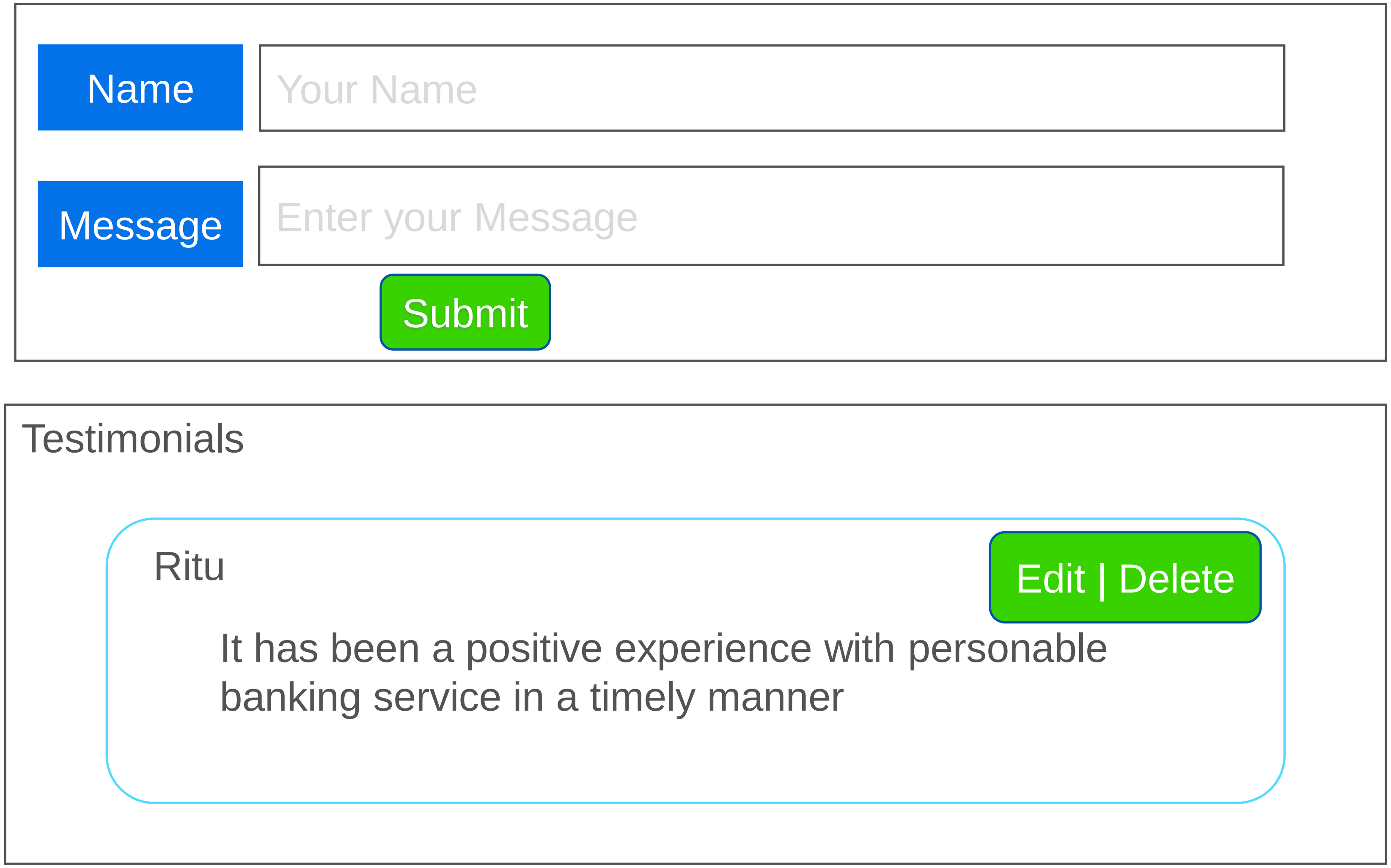
- Create the page using Bootstrap. After entering the valid values, once user clicks on Add Customer button, create the customer object and display in tabular format.
- Validate the fields name (alphabets allowed), email id (valid email id), contact no(only 10 digits and first digit should be 7-9) and account type should be selected.
- Whenever user inserts invalid value, show the error message next to text box for ex please enter only alphabets

2. Presenting the Use Case

Add testimonial page to sc website by collecting the data from the user and add it to the testimonials dynamically with edit and delete Operation using JavaScript.

Presenting the Use Case

axess academy



Ritu

It has been a positive experience with personable banking service in a timely manner

Cheat Sheet references

Cheat Sheet for JavaScript : JS Cheat Sheet.doc
Cheat Sheet for Advanced JS : Advanced JS Cheat Sheet

axess academy

Thank You