HAZZA INSTITUTE OF TECHNOLOGY

iqracity

MERN – ES6 + React

Module 11: ES6 - I

Outline

Module 11: ES6 - I

HAZZA INSTITUTE OF TECHNOLOGY

- ES6 Basics
- Understand concepts of Babel
- Understand concepts of Let and const
- Understand concepts of Creating variables
- Understand concepts of Template and Strings
- Understand concepts of Arrow Functions
- Understand concepts of Rest and Spread Operators

New Features in ES6

HAZZA INSTITUTE OF TECHNOLOGY

JavaScript ES6

- The let keyword
- The const keyword
- Arrow Functions
- For/of
- Map Objects
- Set Objects
- Classes
- Promises
- Symbol
- Default Parameters
- Function Rest Parameter
- String.includes()
- String.startsWith()
- String.endsWith()
- Array.from()
- Array keys()
- Array find()
- Array findIndex()
- New Math Methods
- New Number Properties
- New Number Methods
- New Global Methods
- Object entries
- JavaScript Modules

JavaScript Maps

HAZZA INSTITUTE OF TECHNOLOGY

A Map holds key-value pairs where the keys can be any datatype.

How to Create a Map

You can create a JavaScript Map by:

- Passing an Array to `new Map()`
- Create a Map and use `Map.set()`

```
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);
```

```
// Create a Map
const fruits = new Map();

// Set Map Values
fruits.set("apples", 500);
fruits.set("bananas", 300);
fruits.set("oranges", 200);
```

Essential Map Methods

Method	Description
new Map()	Creates a new Map
set()	Sets the value for a key in a Map
get()	Gets the value for a key in a Map
delete()	Removes a Map element specified by the key
has()	Returns true if a key exists in a Map
forEach()	Calls a function for each key/value pair in a Map
entries()	Returns an iterator with the [key, value] pairs in a Map
Property	**Description**
size	Returns the number of elements in a Map

Tryit Editor v3.7 (w3schools.com)

JavaScript Sets

HAZZA INSTITUTE OF TECHNOLOGY

A JavaScript Set is a collection of unique values.

How to Create a Set

You can create a JavaScript Set by:

- Passing an Array to `new Set()`
- Create a new Set and use `add()` to add values
- Create a new Set and use `add()` to add variables

```
// Create a Set
const letters = new Set(["a","b","c"]);
```

```
// Create a Set
const letters = new Set();

// Add Values to the Set
letters.add("a");
letters.add("b");
letters.add("c");
```

Essential Set Methods

Method	Description
new Set()	Creates a new Set
add()	Adds a new element to the Set
delete()	Removes an element from a Set
has()	Returns true if a value exists in the Set
forEach()	Invokes a callback for each element in the Set
values()	Returns an iterator with all the values in a Set
Property	**Description**
size	Returns the number of elements in a Set

Tryit Editor v3.7 (w3schools.com)

JavaScript Classes

JavaScript ES6

HAZZA INSTITUTE OF TECHNOLOGY

JavaScript Classes are templates for JavaScript Objects.

Use the keyword `class` to create a class.

Always add a method named `constructor()`:

Syntax

```
class ClassName {
  constructor() { ... }
}
```

Example

```
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
}
```

```
const myCar1 = new Car("Ford", 2014);
const myCar2 = new Car("Audi", 2019);
```

JavaScript Class Inheritance

HAZZA INSTITUTE OF TECHNOLOGY

JavaScript ES6

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return 'I have a ' + this.carname;
  }
  get cnam() {
    return this.carname;
  }
  set cnam(x) {
    this.carname = x;
  }
}
```

```
class Model extends Car {
  constructor(brand, mod) {
    super(brand);
    this.model = mod;
  }
  show() {
    return this.present() + ', it is a ' + this.model;
  }
}

let myCar = new Model("Ford", "Mustang");
document.getElementById("demo").innerHTML = myCar.show();
```

JavaScript Promises

JavaScript ES6

HAZZA INSTITUTE OF TECHNOLOGY

A Promise is a JavaScript object that links "Producing Code" and "Consuming Code".

"Producing Code" can take some time and "Consuming Code" must wait for the result.

Promise Syntax

```
const myPromise = new Promise(function(myResolve, myReject) {
// "Producing Code" (May take some time)

  myResolve(); // when successful
  myReject();  // when error
});

// "Consuming Code" (Must wait for a fulfilled Promise).
myPromise.then(
  function(value) { /* code if successful */ },
  function(error) { /* code if some error */ }
);
```

Tryit Editor v3.7 (w3schools.com)

JavaScript Async

HAZZA INSTITUTE OF TECHNOLOGY

"async and await make promises easier to write"

Async Syntax

The keyword `async` before a function makes the function return a promise:

Example

```
async function myFunction() {
  return "Hello";
}
```

Await Syntax

The keyword `await` before a function makes the function wait for a promise:

```
let value = await promise;
```

The `await` keyword can only be used inside an `async` function.

Tryit Editor v3.7 (w3schools.com)

Default Parameter Values

JavaScript ES6

HAZZA INSTITUTE OF TECHNOLOGY

ES6 allows function parameters to have default values.

Example

```
function myFunction(x, y = 10) {
  // y is 10 if not passed or undefined
  return x + y;
}
myFunction(5); // will return 15
```

Function Rest Parameter

HAZZA INSTITUTE OF TECHNOLOGY

JavaScript ES6

The rest parameter (...) allows a function to treat an indefinite number of arguments as an array:

Example

```
function sum(...args) {
  let sum = 0;
  for (let arg of args) sum += arg;
  return sum;
}

let x = sum(4, 9, 16, 25, 29, 100, 66, 77);
```

Modules

HAZZA INSTITUTE OF TECHNOLOGY

JavaScript modules allow you to break up your code into separate files.

- JavaScript modules rely on the import and export statements.
- **Export**
  - You can export a **function** or **variable** from any file.
  - There are two types of exports: **Named** and **Default**.
- Named Exports
  - You can create named exports two ways. In-line **individually**, or all **at once** at the bottom.

person.js

```
export const name = "Jesse";
export const age = 40;
```

person.js

```
const name = "Jesse";
const age = 40;

export {name, age};
```

message.js

```
const message = () => {
const name = "Jesse";
const age = 40;
return name + ' is ' + age + 'years old.';
};

export default message;
```

Modules

HAZZA INSTITUTE OF TECHNOLOGY

JavaScript modules allow you to break up your code into separate files.

- JavaScript modules rely on the import and export statements.
- Import
  - You can import modules into a file in two ways, based on if they are **named** exports or **default** exports.
  - Named exports are constructed using curly braces. Default exports are not.

Import named exports from the file person.js:

```
import { name, age } from "./person.js";
```

Import a default export from the file message.js:

```
import message from "./message.js";
```

JavaScript in DOM and BOM

JavaScript HTML DOM

HAZZA INSTITUTE OF TECHNOLOGY

*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.

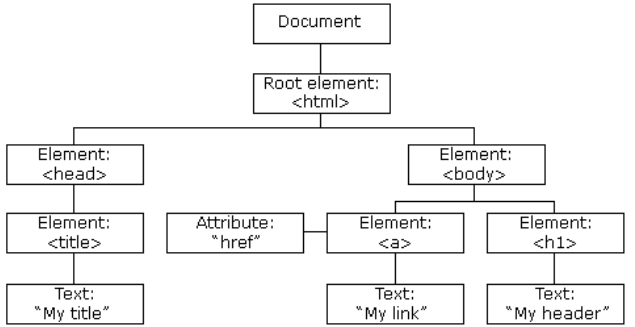
The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- 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.**

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- 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



The DOM Programming Interface

HAZZA INSTITUTE OF TECHNOLOGY

JavaScript HTML DOM

```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

The getElementById Method

The most common way to access an HTML element is to use the `id` of the element.

In the example above the `getElementById` method used `id="demo"` to find the element.

The innerHTML Property

The easiest way to get the content of an element is by using the `innerHTML` property.

The `innerHTML` property is useful for getting or replacing the content of HTML elements.

The DOM Programming Interface

HAZZA INSTITUTE OF TECHNOLOGY

*The HTML DOM document object is the **owner** of all other objects in your web page.*

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

Changing HTML Elements

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

The DOM Programming Interface

HAZZA INSTITUTE OF TECHNOLOGY

*The HTML DOM document object is the **owner** of all other objects in your web page.*

Adding and Deleting 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(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Adding Events Handlers

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

JavaScript Form Validation

HAZZA INSTITUTE OF TECHNOLOGY

JavaScript HTML DOM

Data Validation

Data validation is the process of ensuring that user input is clean, correct, and useful.

Typical validation tasks are:

- has the user filled in all required fields?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Most often, the purpose of data validation is to ensure correct user input.

Validation can be defined by many different methods, and deployed in many different ways.

Server side validation is performed by a web server, after input has been sent to the server.

Client side validation is performed by a web browser, before input is sent to a web server.

Example 1

Example 2

Example 3

Changing HTML Style

JavaScript HTML DOM

HAZZA INSTITUTE OF TECHNOLOGY

- **Changing HTML Style**
  - The HTML DOM allows JavaScript to change the style of HTML elements.

```
document.getElementById(id).style.property = new style
```

Example 1

Example 2

Example 3

JavaScript Events

HAZZA INSTITUTE OF TECHNOLOGY

JavaScript basics

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

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

```
<element event='some JavaScript'>
<element event="some JavaScript">
```

Example

JavaScript Events

JavaScript basics

HAZZA INSTITUTE OF TECHNOLOGY

Common HTML Events

Here is a list of some common HTML events:

Event	Description
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

Example 1

Example 2

Example 3

Module 11: ES6 - I

Module 11

HAZZA INSTITUTE OF TECHNOLOGY

- ES6 Basics
- Understand concepts of Babel
- Understand concepts of Let and const
- Understand concepts of Creating variables
- Understand concepts of Template and Strings
- Understand concepts of Arrow Functions
- Understand concepts of Rest and Spread Operators