MERN CLASS-1 RESOURCES
=============

What is javaScript

- Popular web Programming Language
- Scripting language
- Lightweight
- Cross-platform
- Object-oriented syntax
- Run-on browser

History

In September 1995, a Netscape programmer named Brandan Eich developed a new scripting language in just 10 days. It was originally named Mocha, but quickly became known as LiveScript and, later, JavaScript.

Before javaScript

- HTML 5
- CSS 3
- jQuery
- Bootstrap 5
- Github

IDE

- VS code
- PHPStorm
- Sublime
- Atom

Extension for VS Code

- Auto Close Tag
- Auto Rename Tag
- Beautify
- HTML CSS Support
- HTML Snippets
- Intellisense for CSS class names in HTML
- Live Server
- Auto-Save on Window Change
- Lorem ipsum
- Live Sass Compiler(If you need)
- Path Intellisense
- PHP Intellisense
- Laravel Extra Intellisense(At this moment, no need)
- Prettier-Code formatter
- Vscode-icons/ Material Icon Theme
- Bracket Pair Colorize
- ESLint(For JS)
- JavaScript(ES6) code snippets

Dev fonts Download

Dev Fonts Download Link

Environment Setup

- Install VS Code and configure
- Install Node JS
- Install git

Start

- Script Tag
- Internal

- External
- Inline

Dev Tools

- Console log
- Use of console

Basic Function

- Alert
- Confirm
- Prompt

Statement & rules

- In a programming language, instructions (lines of code) are called statements.
- put a semicolon after a complete statement
- also, you can avoid semicolon
- two or more words are joined by using concatenation (+)

MERN CLASS-2 RESOURCES
=============

JavaScript Variables

- Var
- Let
- Const

- Rules for variables
  - Names can contain letters, digits, underscores, and dollar signs.
  - Names must begin with a letter
  - Names can also begin with $ and _
  - Names are case sensitive
  - Reserved words cannot be used as names

Template literal Syntax

- Template literals are literals delimited with backticks (`), allowing embedded expressions called *substitutions*.

- There are 2 type of template literal
  - untag template literal
  - Tag template literal
- Interpolation variables and expression - ${ var / ex }

Comments

- Make a comment by using //
- Line comment
- Multiline comments
- Doc block for documentation

Operators

- Arithmetic
  +, -, *, / , %, --, ++
- Assignment
  =
- Comparison
  == === < > <= >= != !== ?
- Logical
  && || !
- String
  + +=
- Bitwise
  & ~ ^ | << >> >>> <<<
- Special (type operator)
  (?:) , delete - in - instanceof - typeof - new - void - yield

Data Types

- **String**
- **Number**
- **Boolean**
- **Array**
- **Object**
- **Undefined**

Conditional Statement

- **if**
- **else**
- **else if**
- **switch case**

MERAN CLASS-3 RESOURCES

==============

Class 03 Overview

Conditional Statement

- **if**
- **else**
- **else if**
- **switch case**
- **Ternary operator (condition ? true return : false return)**
- **nullish coalescing operator (return the value ?? return something if the value null or undefined)**
- **undefined, null and empty value**
- **parseInt, Number, parseFloat for string to number conversion**
- **Truthy & Falsy Values (undefined, null, empty, 0, NaN)**

Loops

- **For loading some code a number of times**
- **loop structure**
  -> initial value
  -> condition / end step
  -> operators (++, --)

- **Loop Statement**
  -> for
  -> while / do while

MERAN CLASS-4 RESOURCES
==============

User-Defined Function

- **To avoid repeating the same code**
- **Create a complex functionality for use**
- **Declare one-time use many times**
- **The application will be scalable and clean**

Declare a function

to declare a function use function key and then put the name of the function

```
function functionName () {

          function output
}
```

Invoke a function

When we call a function then it is called **function invoke.** After declaring a function now it's time to use this function. Just call this function like this **functionName()**;

Arguments & Parameter

```
functionName(argument1, argument2, . . . . );
function functionName (parameter1, parameter2, . . . . ) {

          function output
}
```

function Expression

```
let functionName = function (parameter1, parameter2, . . . . ) {

          function output
}
```

Arrow function

```
let functionName = (parameter1, parameter2, . . . . ) =>  {
          function output
}
```

MERAN CLASS-5 RESOURCES
=============

Function Review

- function declaration
- function expression
- arrow function

Arrow function Details

- arrow function syntax
  ```
  let function_name = ( param1, param2, . . . ) => {
        output / functionality
  }
  ```

- Single line arrow function
  ```
  let func_name = ( p1, p2, . . .  ) => return output
  ```

- single parameter arrow function
  ```
  let func_name = ( p1 )   => output / functionality
  let func_name =   p1     => output / functionality
  ```

Constructor Function

- The leader of the function
- Many functions & variables live under the constructor function
- Just call the leader then you will call all the sub function

- syntax of constructor function
```
 function FunctionName(){
        output / functionality
 }
```

- constructor function expression
```
 let FunctionName = function(){
        output / functionality
 }
```

- Sub functions in constructor function
```
 let FunctionName = function(){
        this.variables1 = ‘ value1 ’;
        this.variables2 = ‘ value2 ’;
        this.variables3 = ‘ value 3’;

        this.fucntion1_name = function(){
                output / functionality
        }

        this.fucntion2_name = function(){
                output / functionality
        }
}
```

- Call the constructor function / instance
```
 let lead_name = new FunctionName;
```

- Call the sub functions and variables from constructor function after instance
```
let lead_name = new FunctionName;

 lead_name.variable1;
lead_name.variable2;

lead_name.fucntion1();
```

lead_name.fucntion2();

Project Work

- Create a Utility Constructor function
- Complete result system with

MERAN CLASS-6 RESOURCES
=============

Array (Data Structure)

- Used more than one value or can be more than one value
- The best way to decorate data for future
- Any type of data can be stored in an array
- declare an array
  const array_var = [v1, v2, v3 . . . .];

- Create some array data structure
  -> 10 flowers of Bangladesh
  -> 10 Rivers of Bangladesh
  -> 10 fish of Bangladesh
  -> 10 Birds of Bangladesh
  -> 10 vegetable of Bangladesh

- Get value from an array
  array_var[index_number]

- Get array length
  array_var.length

- Get all array value by for loop
  for(let i = 0; i < array_var.length; i++){

```
        return array_var[ i ];
}
```

- Get all array data by forEach & Map (iteration)

```
array_var.forEach( function(data){
        return data;
})

array_var.map((data) => {
        return data;
})
```

- Create an array by using Array Constructor

```
 new Array(item1, item2, . . . )
```

- Array to string conversion
  -> toString
  -> join

- Array Add & Remove
  -> push
  -> pop
  -> shift
  -> unshift
  -> slice
  -> splice

Array methods & Uses

-> concat()
-> reverse()
-> sort()
-> filter()
-> reduce()
-> every()
-> some()
-> indexOf()
-> find()

-> findIndex()
-> from()
-> keys()
-> includes()
-> isArray()
-> valueOf()

- **Multidimensional Array**

```
[
        [item1, item2, . . . . ],
        [item1, item2, . . . . ],
        [item1, item2, . . . . ]
```

MERAN CLASS-7 RESOURCES
==============

Object Data Structure

- A complete Data structure will be made by array and object
- In an array data structure we face some problems like data key
- But with array and object data, we can build a complete structure for the future.

- Declare a object data structure

```
 const obj_name = {
        name : 'Asraful Haque',
        age : 10,
        skill : 'Laravel Developer'
}
```

- Declare a object data structure with new Object

```
 const obj_name =  new Object({
```

```
          name : 'Asraful Haque',
          age : 10,
          skill : 'Laravel Developer'
      })
```

- Get data form an object data structure
  -> By dot notation obj_name.property_name;
  -> By array notation obj_name['property_name'];

Create a Complete Array and Object Data structure

```
const obj_name = [
      {
          name : 'Asraful Haque',

          age : 10,

          skill : 'Laravel Developer'

      },
      {
          name : 'Asraful Haque',

          age : 10,

          skill : 'Laravel Developer'

      },
      {
          name : 'Asraful Haque',

          age : 10,

          skill : 'Laravel Developer'

      }
]
```

==============

Date Object

- For date & time management
- Date object has a constructor
- Declare a Date object

  let date = new Date();

- Get date information

  -> date.getDate()
  -> date.getMonth()
  -> date.getFullYear()
  -> date.getHoures()
  -> date.getMinutes()
  -> date.getSeconds()
  -> date.getMilliseconds()
  -> date.getTime()

- Date formates

-> 2021-12-07	=> ISO
-> 07/12/2021	=> short
-> December 7, 2021	=> Long

- Find Today form Date Object
- Find current month name from JS object

Math Object

- Math object has no constructor
- It is static / you don't need to create any instance

- **Math Property**

  - Math.PI
  - Math.E

- Math.SQART2
- Math.SQUART1_2
- Math.LN2
- Math.LN10
- Math.LOG2E
- Math.LOG10E

- **Math methods**
  - Math.abs()
  - Math.ceil()
  - Math.floor()
  - Math.round()
  - Math.max()
  - Math.min()
  - Math.sqrt()
  - Math.pow()
  - Math.random()

String Object

- **The string can be an object**
  new String()
  but do not use this, please
- **String property**
  - constructor
  - length

- **String Methods**
  - concat
  - startWith
  - endsWith
  - includes
  - indexOf

- lastIndexOf
- repeat
- replace
- search
- slice
- split
- substr
- toUppercase / Local
- toLowercase / Local
- toString
- trim

Number Objects

- Developer can create a number object by using this
  new Number()
  but do not use this

JSON Data

- JSON stands for JavaScript Object Notation
- The lightweight data-interchange format
- Language independent
- Easy to understand and self-describing
- JSON is a text format for storing and transporting data
- JSON helps to convert array and object data to a string format for devs-friendly data use.

- **Declare a JSON**
  - It looks like an object
  - Data is named/Value pairs
  - Data is separated by a comma
  - Curly braces hold the object
  - Square brackets hold arrays

```
{
        "name1" : "value1",
        "name2": "value2"
}
```

- **JSON data types**
  - string
  - number
  - object
  - array
  - Boolean
  - null

- **JSON values cannot be one of the following types**
  - function
  - date
  - undefined

- JSON.parse()
  - **to convert JSON data string to object**
  - **to convert an array string to an array**
- **JSON.stringify**
  - to convert an **object** data to a **JSON string**
  - to convert an **array** to **JSON string**

- **JSON file**
  We have to create a JSON file by setting.json
  db.json / api.json

Errors handling

- **To Handle errors in a custom way**
- **Prevent apps crashing for an error**
- **Try Catch Finally**

  -> Try

  -> Catch

  -> Throw

  -> Finally

try {

Block of code to try

} catch(Err) {

Block of code to handle errors

}finally {

Block of code to be executed regardless of the

try/catch

}

Local Storage

- Browser storage for temporary data
- Send data to LS
  -> localStorage.setItem(‘key’, ‘value’);

- Get Data from LS
  -> localStorage.getItem(‘key’);

Session Storage

- Browser storage for temporary data
- Send data to SS
  -> sessionStorage.setItem(‘key’, ‘value’);

- Get Data from S
  -> sessionStorage.getItem(‘key’);

Cookie Storage

- Cookies are data, stored in small text files, on your computer
- It is used to remember a user from the browser
- Send data to a cookie

  document.cookie = “name = data ; expire ; path =/ ”;

- Get Data from cookie

  let cookie_data = document.cookie ;

Regular Expression

- **A regular expression is a sequence of characters that forms a search pattern**
- **A regular expression can be a single character or a more complicated pattern**
- **Syntax**

  / pattern / modifier

- **Modifier**

  - /i (case insensitive)
  - /g (global Search)
  - /m (multiline search for the match)
  - / (empty modifier is case sensitive)

- **Methods**

  - exec (check data is in an array or not)
  - test (return true or false for data check)
  - match (check the match is in or not)
  - search (search the index number of pattern)
  - replace (replace words of a string)

- **Literal character**

  - all regular character is a literal character

- **Meta Character**

- ^	(start with the character)
- $	(ends with character)
- .	(any character length will be one)
- *	(any character length one to more)
- ?	(set optional character by using this key)
- [abc]	(character group)
- [^abc]	(except those character)
- [A-Z][a-z][0-9]	(uppercase, lowercase and number)
- abc{2}	(quantifier for repeat character)
- ()	(for creating group)
- \w	(alphanumeric word character)
- \W	(non-word character)
- \d	(digit character)
- \D	(non-digit)
- \s	(white space)
- \S	(non-white space)
- \w	(word boundary)

MERAN CLASS-11-12-13 RESOURCES
=============

JavaScript DOM

- **DOM stands for Document Object Model**
- **The whole document of a website and its all elements are called DOM Elements**
- **We can add, remove, change and delete by DOM operation**
- **DOM operations are executed by DOM event**
- **DOM properties and methods help DOM manipulation**
- **DOM object is a document**

DOM Object

- Title
- Domain
- Images
- Doctype
- Body
- Head
- URL
- Links
- Scripts
- form
- Embeds
- Cookie

DOM Selectors

- getElementById()
- getElementsByClassName()
- getElementsByTagName()
- querySelector()
- querySelectorAll()

DOM Property & Methods

- innerHTML
- innerText
- textContent
- setAttribute
- getAttributes
- hasAttributes
- Style
- className
- classList.add / remove
- Id
- Children
- firstChild

- LastChild
- parentElement
- nextElementSiblings
- previoustElementSiblings
- Value

Add and Delete Elements

- createElement
- createTextNode
- Append
- appendChild
- removeChild / remove
- replaceChild
- insertBefore
- write

Event Listener

- Fire event in DOM
- addEventListener (eventName, callback)
- event with arrow

- Mouse Event
  - click
  - dblclick
  - mouseenter
  - mouseleave
  - mousedown
  - mouseup

- Keyboard
  - keydown
  - keyup
  - keypress

- Form Event
  - focus
  - blur
  - change
  - submit

CLASS RECOUSE: 18, 19

REST API - RESTful API

- A REST API also known as RESTful API is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding.

Roy Fielding - the founder

Roy Thomas Fielding is an American computer scientist, one of the principal authors of the HTTP specification and the originator of the Representational State Transfer architectural style. He is an authority on computer network architecture and co-founded the Apache HTTP Server project.

WHY API

- Multi Platform apps development
- Data sharing system
- Helps React, Angular, Vue, mobile apps developers for building their application

Fetch API

- Get get api data
- Make a complete data crud by using response and request

- Get api data by fetch api call

```
fetch('api.url').then( data => data.json() ).then(data => {
        // Data manage
 });
```

- Data fetch by axios

 -> install axios by npm/yarn

```
$ npm install axios
$ yarn add axios
```

-> get data get by axios

```
axios.get('api.url').then(data=> {
        // data.data
});
```

- Insert data by axios with POST request

```
 axios.post('api.url', {
        name : data,
        email : data,
        . . . . . .
 }).then(data=> {
        // data.data
});
```

- Delete data by axios with delete request

```
 axios.delete( 'api.url/' + id ).then(data=> {
        // data.data
});
```

- Edit data by axios with PUT/PATCH request

```
 axios.post('api.url', {
        name : data,
        email : data,
        . . . . . .
 }).then(data=> {
        // data.data
});
```

JSON Server

- This is a data store system server for react, angular, vue developers for entry
- For creating a REST API
- Complete no-SQL Database Standard
- Alternative of MongoDB, Firebase

- **Install JSON Server**
  -> First set a npm project
  -> Then install json server
  npm install json-server --save-dev

- **Setup Server Watch**
  -> open package.json file
  -> add a new script for json server watch
  Json-server --watch file.sjon

- **Create a JSON data structure for your project**
  -> Customer
  -> Products
  -> Users
  -> Role
  -> Blog Post
  -> Comments

- **Run JSON Server**
  npm run server-watch-code

OOP Resources

:https://docs.google.com/document/d/17Wb4e9fzas87Bz5PNOXzQ926REIyGquAP_FelxMJ8Sg/edit