**Object Oriented Programming**

- javaScript is a scripting language, even type missing , Not a class based programming language.

- FROM ES 6+ we can use classes in javaScript for better programming experience and features.

- ES 6 + classes are not too powerful such as Java, PHP, or Python. It’s a syntactic sugar only.

- We should learn ES 6+ javaScript for accessing the future tech

- ES 6+ are modern JS but all modern code will convert as a vanilla code through babel JS, We should use it for browser capability.

**OOP Features**

- Vanilla JS ( ES 5 features )

- Modern JS ( ES 6+ features )

**- Design Pattern concept ( Advance ) with project**

**Class base OOP - ES6+**

**- Declare a class and init it**   
 class **Student** {

}

**- Create an instance of this class**  
 let **obj** = new **Student**();

**- Constructor method with property access**  
 class **Student** {  
 constructor( p1, p2, p3, . . . ){  
 this.p1 = p1;   
 this.p2 = p2;  
 }  
 **property\_name\_1** = val\_1,  
 **property\_name\_2**= val\_2,  
  
 }  
  
 **object**.propertyName;

**- object method methods - ( prototypes )**   
 class **Student** {  
 constructor( p1, p2, p3, . . . ){  
 this.p1 = p1;   
 this.p2 = p2;  
 }  
  
 ageCal(){  
  
 }  
  
 currencyConvert(){  
  
 }  
  
  
 }  
 **object**.methodName();

**- object method methods - ( Object Methods )**   
 class **Student** {  
 constructor( p1, p2, p3, . . . ){  
 this.p1 = p1;   
 this.p2 = p2;  
   
 this.methodName = function(){  
   
 }  
 }  
  
   
  
 }  
 **object**.methodName();

**- Static methods and its use**  
 class **Student** {  
 constructor( p1, p2, p3, . . . ){  
 this.p1 = p1;   
 this.p2 = p2;  
 }  
  
 static ageCal(){  
  
 }  
  
 static currencyConvert(){  
  
 }  
  
 }  
 **Student.**ageCal()**;   
 Student.**currencyConvert()**;**

**- Inheritance / Extends**   
 class **Student** {  
 constructor( p1, p2 ){  
 this.p1 = p1;   
 this.p2 = p2;  
 }  
  
 ageCal(){  
  
 }  
 }  
  
 class **Result** extends **Student**{  
 constructor( p1, p2, r1, r2 ){  
 super(p1,p2);

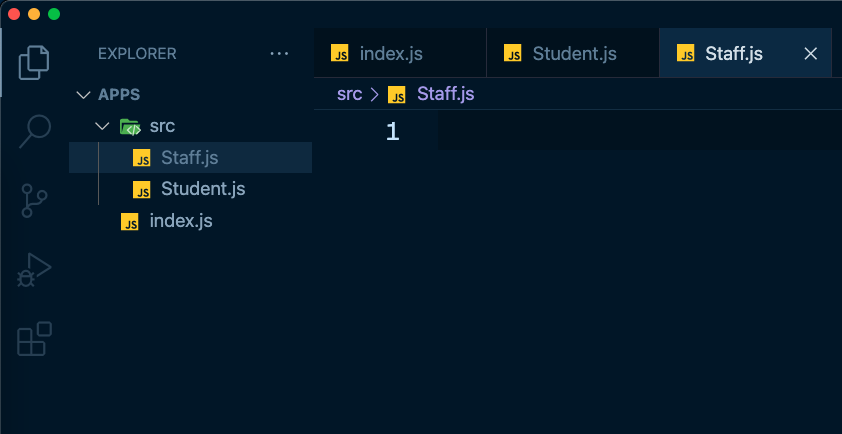
}

}

**- Get and set for exchange methods to property**   
 class **Student** {  
 constructor( p1, p2 ){  
 this.p1 = p1;   
 this.p2 = p2;  
 }

set age(){  
 return this.ageCal();  
 }  
  
 ageCal(){  
  
 }  
}  
**object**.age;

**Constructor Based OOP**

* **Declare a Constructor**   
  function Student(){  
   this.name = ‘Asraf’;  
   this.email = ‘[haq@gmail.com](mailto:haq@gmail.com)’;  
    
   this.ageCal = function(){  
    
   }  
  }
* **Create an instance for this constructor**   
  let student = new Student();
* **Set property and method in a constructor and call**  
  function Student(){  
   this.name = ‘Asraf’;  
   this.email = ‘[haq@gmail.com](mailto:haq@gmail.com)’;  
    
   this.ageCal = function(){  
    
   }  
  }  
    
  **Call property and method form a constructor**   
    
  let student = new Student();   
  student.name;  
  student.email;  
  student.ageCal();
* **Set prototypes methods**   
  **Student**.prototype.ageCal = function(){  
    
  }  
  **Student**.prototype.ageCal = () => {  
    
  }
* **Modular application structure**   
  -> Put all constructor / class in src/ directory   
  -> make class / Constructor name like this Student, Staff   
  -> set an entry point in js named app.js / index.js   
  
* **Export and import class / constructor**   
  -> set modular type in package.json file   
  {  
   “type” : “modular”  
  }
* **now export your any module from any module.js files**   
  export const data = {  
   name : ‘Asraful Haque’,   
   age : 10,   
   skill : ‘MERN Stack’  
  }  
    
  export const food = [‘alo’,’potol’,’lao’];  
    
  export function ageCal(){  
     
  }  
    
  export class Student {  
    
  }
* **Now import those module.js file or different files**import { data, food, ageCal, Student } from ‘./src/module.js’;  
  import { data } from ‘./src/data.js’;  
  import { food } from ‘./src/food.js’;  
  import { ageCal } from ‘./src/age.js’;  
  import { Student } from ‘./src/Student.js’;
* **Set alias when import**import { Student as Chatro } from ‘./src/Student.js’;  
  import { ageCal as boyos } from ‘./src/age.js’;
* **Export default and import**export default class Student {  
    
  }  
    
  class Student {  
    
  }  
  export default Student;

import Student, { data, food, ageCal } from ‘./src/module.js’;  
 import AnyName , { food, ageCal } from ‘./src/module.js’;

* **Module exports and requires**    
  const data = {  
   name : ‘Asraful Haque’,   
   age : 10,   
   skill : ‘MERN Stack’  
  }  
    
  module.exports = data;
* **Require a module**   
  const **data** = require(‘data.js’);
* **Get Form Data by FormData Object**-> set form fields name   
  -> set method   
  -> now fire submit event
* **Get Data by FormData Object**   
  let form\_data = new FormData(e.target);  
  let data = Object.formEntries(form\_data.entries());

**Call Back Function**

* I will Call you back later
* Passed argument one function to another
* A callback function run after finished another function
* Callback function can be called closer function
* Callback are asynchronous function
* **Syntax of a callback**

function info( **name, year, callback** ){  
   
 **callback();**  
}

**Promise function**

* Make a promise
* Promise return a resolve and reject data
* A nice asynchronous function
* Syntax   
    
  let marks = 30;  
  let result = new **Promise**(( resolve, reject ) => {  
   if( marks > 32 ){  
   resolve();   
   }else {  
   reject();   
   }  
    
  });

result.then( (**data**) => {  
 console.log(**data**);  
 }).catch( (**error**) => {  
 console.log(**error**);  
 } );

**Async Await**

* Async await is asynchronous function
* Return a promise for better response
* Syntax   
    
  function ageCal(){  
   let data = await info();  
  }

- Polymorphism

- Create a project like GPA, Result

- Prototype methods