Module 32-(Advance) ES6, Class, Inheritance

32-1 Module Introduction and Basic ES6 Recap

```
// 1. let and const
const hubby = "Omor Sani";
let phone = "iphone 15";
phone = "Samsung Galaxy S17";

// 2. default
// 5. spread or three dots (...)
function maxNumber(array = []) {
  const max = Math.max(...array);
  return max;
}

const biggest = maxNumber();
console.log(biggest);

// 3. template string
const myNotes = `I am mojnu of ${hubby}. I have a ${phone}.`;
console.log(myNotes);

// 4. arrow function
// function square(x) {
//     return x * x;
// }

const square = (x) => x * x;
console.log(square(9));
```

32-2 Destructuring Object to extract values to variables

অবজেক্টে একাধিক প্রপার্টি এর ভ্যালু একসাথে বের করে ভেরিয়েবল এ রাখাকে destructuring বলে।

```
const fish = {
  id: 58,
  name: "King Hilsha",
  price: 9000,
  phone: "01717555555555",
  address: "Chandpur",
  dress: "silver",
```

```
};

// const phone = fish.phone;
// const price = fish.price;
// const dress = fish.dress;
// const id = fish.id;

const { phone, price, dress, id } = fish;

// console.log(phone, price);
// console.log(phone, id);
// console.log(phone, dress);
// console.log(phone, price);
// console.log(phone);

const company = {
  name: "GP",
  ceo: { id: 1, name: "ajmol", food: "fuchka", work: "website
development" },
  web: {
    work: "website development",
    employee: 22,
    framework: "react",
    tech: {
      first: "html",
      second: "css",
      third: "js",
    },
  },
};

// const work=company.web.work;
// console.log(work);

// const fremework=company.web.framework;
// console.log(fremework);

const { work, framework } = company.web;
const { food } = company.ceo;
const { second, third } = company.web.tech;
console.log(work, framework, food);
```

32-3 (advanced) Array Destructuring, nested object Optional chaining

```
// declare variable based on the name of an object property
const myObject = { x: 2, y: 50, z: 600, a: 25, b: 68 };
const {x,b}=myObject;
// console.log(x,b);

console.log("myObject.p", myObject.p);
console.log("myObject.p", myObject?.p?.q);

// destructuring array
const [p, q,r] = [45, 37, 91, 86];
// console.log(p,q,r);

const [best, faltu] = ["momotaj", "poroshi"];
// console.log(best,faltu);

const { sky, color, money } = {
  sky: "blue",
  soil: "matti",
  color: "red",
  money: 500,
};

//chaining
const company = {
  name: "GP",
  ceo: { id: 1, name: "ajmol", food: "fuchka" },
  web: {
    work: "website development",
    employee: 22,
    framework: "react",
    tech: {
      first: "html",
      second: "css",
      third: "js",
    },
  },
};

console.log(company.web.tech.third);

console.log(company?.web?.tech?.third);
console.log(company?.backend?.tech.third);
```

32-4 Array map to do one line loop magic

```
const numbers = [4, 6, 8, 10];
const output2 = [];

// function doubleOld(number)
// {
//     return number*2;
// }

const doubleIt=number=>number*2;

for(const number of numbers)
{
    // const result=number*2;
    const result=doubleIt(number);
    output2.push(result);
}
// console.log(output2);

// 1. loop through each element
// 2. for each element call the provided function
// 3. result for each element will be stored in an array

// const output=numbers.map(doubleIt);
// const output = numbers.map((number) => number * 2);
const output=numbers.map(x=>x*2);
// console.log(output);

const squares=numbers.map(x=>x*x);
console.log(squares);
```

32-5 Map string array, array of objects map, foreach

map() অ্যারের প্রতিটি উপাদান এর জন্য ফাংশনকে কল করে এবং সব ফাংশনকে কল করে এবং সব ফাংশনের রেজাল্টকে একটি নতুন অ্যারে তে নিয়ে রিটার্ন করে।

forEach() একটি অ্যারের প্রতিটি উপাদানের জন্য একটি ফাংশন কল করে। কিন্তু কোন কিছু রিটার্ন করে না

```
const friends = ["Tom Hanks", "Tom Cruise", "Tom Brady", "Tom
Solaiman"];

const fLengths = friends.map((friend) => friend.length);
// console.log(fLengths);
```

```
const products = [
  { name: "water bottle", price: 50, color: "yellow" },
  { name: "mobile phone", price: 15000, color: "black" },
  { name: "smart watch", price: 3000, color: "black" },
  { name: "sticky note", price: 30, color: "pink" },
  { name: "water glass", price: 3, color: "white" },
];
const productNames = products.map((product) => product.name);
// console.log(...productNames);
// console.log(productNames);

const productPrices = products.map((product) => product.price);
// console.log(...productPrices);
// console.log(productPrices);

// products.map(product => console.log(product));
// products.map((product) => console.log(product.name.length));

products.forEach(product=>console.log(product));
```

32-6 (advanced) Implement filter, find on an array of objects

filter() অ্যারের প্রতিটি উপাদানের মধ্যে যেটা যেটা শর্ত ফুলফিল করবে তাদেরকে নতুন একটা অ্যারে তে রেখে সেটাকে রিটার্ন করবে।

find() অ্যারে এর প্রথম যে উপাদান শর্ত ফুলফিল করবে শুধু সেই উপাদানকে রিটার্ন করে।

```
const numbers = [5, 13, 7, 41, 30, 5, 2, 19];

const bigNumbers = numbers.filter(number => number > 20);
// console.log(bigNumbers);

const smallNumbers = numbers.filter(number => number < 10);
// console.log(smallNumbers);

const products = [
    { name: 'water bottle', price: 50, color: 'yellow' },
    { name: 'mobile phone', price: 15000, color: 'black' },
    { name: 'smart watch', price: 3000, color: 'black' },
    { name: 'sticky note', price: 30, color: 'pink' },
    { name: 'water glass', price: 3, color: 'white' }
];
```

```
const expensive = products.filter(produc => produc.price > 100);
// console.log(expensive);

const blacks = products.filter(product => product.color == 'pink');
// console.log(blacks);

const whiteItem = products.find(product => product.color == 'black');
console.log(whiteItem);
```

32-7 (advanced) Class, constructor, method, create object from class

একই ধরনের অনেকগুলো অবজেক্ট তৈরি করার জন্য class ব্যবহার করা হই।

```
class Support {
  name;
  designation = "Support Web Dev";
  address = "BD";
  constructor(name,address)
  {
      this.name=name;
      this.address=address;
  }
  startSession() {
    console.log(this.name,"start a support session");
  }
}

const aamir = new Support("Aamir Khan", "BD");
aamir.startSession();
console.log(aamir);

const salman = new Support("Solaiman Khan", "Dubai");
salman.startSession();
console.log(salman);

const sharuk = new Support("SRK Khan", "Dubai");
sharuk.startSession();
console.log(sharuk);

const akshay = new Support("Akshay Kumar", "Dubai");
akshay.startSession();
console.log(akshay);
```

32-8 (advanced) Inheritance, extends class, super, class method

```
class TeamMember {
  name;
  designation = "Support Web Dev";
  address = "BD";
  constructor(name, address) {
    this.name = name;
    this.address = address;
  }
}

class Support extends TeamMember {
  groupSupportTime;
  constructor(name, address, time) {
    super(name, address);
    this.groupSupportTime = time;
  }
  startSession() {
    console.log(this.name, "start a support session");
  }
}

class StudentCare extends TeamMember {
  buildARoutine(student) {
    console.log(this.name, "Build a routine for", student);
  }
}

class NeptuneDev extends TeamMember {
  codeEditor;
  constructor(name, address, editior) {
    super(name, address);
    this.codeEditor = editior;
  }
  releseApp(version) {
    console.log(this.name, "release app version", version);
  }
}

const aamir = new Support("Aamir Khan", "BD", 11);
// console.log(aamir);

const salman = new StudentCare("Solaiman Khan", "Dubai", 4);
```

```
// console.log(salman);

const sharuk = new Support("SRK Khan", "Dubai", 9);
// console.log(sharuk);

const akshay = new Support("Akshay Kumar", "Dubai", 11);
// console.log(akshay);

const ash = new NeptuneDev("Ash", "Mumbai", "Android studio");
ash.releseApp(1.4);
console.log(ash);
```

32-9 (advanced) Prototypical inheritance and module summary