Դաս 08: Ժառանգում (Inheritance) և Պոլիմորֆիզմ (Polymorphism)

Ժառանգումը թույլ է տալիս մեկ կլասին ժառանգել մյուսի հատկությունները և մեթոդները:

```
class Person {
  constructor(name) {
      this.name = name;
  }
}

class Employee extends Person {
  constructor(name, position) {
          super(name);  // Կանչում ենք ծնողական կլասի կոնստրուկտորը
      this.position = position;
  }
}

let emp = new Employee("Ani", "Developer");
console.log(emp.name); // Ani
console.log(emp.position); // Developer
```

Պոլիմորֆիզմը (Polymorphism) թույլ է տալիս տարբեր օբյեկտներ գործել նույն կերպ՝ հիմնվելով ընդհանուր կլասի վրա: Ջավասկրիպտում դա հաճախ իրականացվում է մեթոդների վերասահմանմամբ (method overriding):

```
class Animal {
  speak() {
      console.log("Animal speaks");
  }
}
```

```
class Dog extends Animal {
  speak() {
      console.log("Dog barks");
  }
}

class Cat extends Animal {
  speak() {
      console.log("Cat meows");
  }
}

let animals = [new Animal(), new Dog(), new Cat()];
animals.forEach((animal) => animal.speak());
// Output:
// Animal speaks
// Dog barks
// Cat meows
```

Կլասի օբյեկտի կառուցման և նրա սկզբնավորման ժամանակ կարող ենք ստուգել, թե որ կլասին է պատկանում տվյալ օբյեկտը՝ օգտագործելով **constructor** հատկությունը:

```
class Employee extends Person { /*...*/ }

let emp = new Employee("Ani", "Developer");
console.log(emp.constructor.name); // Employee

let arr = [1, 2, 3];
console.log(arr.constructor.name); // Array
```

Prototype մեթոդների ավելացում

JavaScript-ում հնարավոր է նոր մեթոդներ ավելացնել ցանկացած օբյեկտի պրոտոտիպում (prototype):
Օրինակ՝ ավելացնենք զանգվածի համար maximum մեթոդ՝ մեծագույն արժեքը վերադարձնելու համար:

```
Array.prototype.maximum = function () {
  let max = this[0];
  for (let i in this) {
      if (this[i] > max) max = this[i];
  }
  return max;
};

let x = [1, 2, 3];
console.log(x.maximum()); // 3
```

Ստատիկ դաշտեր և մեթոդներ

Ստատիկ դաշտերը և մեթոդները պատկանում են հենց կլասին և ոչ թե դրա օբյեկտներին: Դրանք կոչելու համար օգտագործվում է հենց կլասի անունը:

```
class MyClass {
  static name = 'Ani';
  static greet() {
      console.log('Hello from MyClass!');
  }
}

console.log(MyClass.name);   // Ani
MyClass.greet();             // Hello from MyClass!
```

Ստատիկ դաշտերը և մեթոդները հաճախ օգտագործվում են ֆունկցիոնալ հնարավորություններ ապահովելու համար, որոնք կախված չեն կոնկրետ օբյեկտից:

Աբստրակտ կլասը կլաս է, որը նախատեսված է միայն ժառանգման համար և նրանից հնարավոր չէ օբյեկտ ստեղծել: JavaScript-ում այս մեխանիզմը իրականացնելու համար հաճախ օգտագործվում է կոնստրուկտորում սխալ նետելը (throw error):

```
class AbstractClass {
  constructor() {
      if (new.target === AbstractClass) {
              throw new Error('Cannot instantiate an
abstract class');
      }
  }
}

class ConcreteClass extends AbstractClass {
        // Իրականացնում եմ աբստրակտ կլասի
ֆունկցիոնալությունը
}

let obj = new ConcreteClass(); // Աշխատում է
let abstractObj = new AbstractClass(); // Error: Cannot
instantiate an abstract class
```

Այս օրինակում **AbstractClass** չի կարող օգտագործվել օբյեկտ ստեղծելու համար, սակայն կարելի է օգտագործել որպես հիմք այլ կլասների համար: