Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

JavaScript Classes:

Introduction to Classes

In JavaScript, classes are a syntactic sugar over the existing prototype-based inheritance. They provide a clearer and more concise syntax for creating and managing objects and inheritance. Classes were introduced in ECMAScript 2015 (ES6) and are a part of modern JavaScript.

Key Features of JavaScript Classes

1. **Class Declaration**
2. **Class Expression**
3. **Constructor Method**
4. **Instance Methods**
5. **Static Methods**
6. **Inheritance with extends and super**

Class Declaration

A class declaration is a way to define a class using the class keyword

```
class Parent {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }

    greet() {
        return `Hello, my name is ${this.name} and I am ${this.age} years old.`;
    }
}

const parent1 = new Parent('Hema', 20);
console.log(parent1.greet()); // Output: Hello, my name is Hema and I am 20 years old.
```

Class Expression

A class expression is another way to define a class. It can be named or unnamed.

Example:

```
let Parent = class {
    constructor(name, age) {
```

```
        this.name = name;
        this.age = age;
    }

    greet() {
        return `Hello, my name is ${this.name} and I am ${this.age} years old.`;
    }
}

const parent1 = new Parent('Hema', 20);
console.log(parent1.greet()); // Output: Hello, my name is Hema and I am 20 years old.
```

Constructor Method

The constructor method is a special method for creating and initializing an object created with a class. There can be only one constructor method in a class.

```
class Parent {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
}
```

Instance Methods

Instance methods are methods that are defined in the class and are available to instances of the class.

Example:

```
let Parent = class {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }

    greet() {
        return `Hello, my name is ${this.name} and I am ${this.age} years old.`;
    }
}

const parent1 = new Parent('Hema', 20);
console.log(parent1.greet()); // Output: Hello, my name is Hema and I am 20 years old.
```

Static Methods

Static methods are defined on the class itself and are not available to instances of the class. They are called on the class itself.

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

    static species() {
        return 'Homo sapiens';
    }
}

console.log(Person.species()); // Output: Homo sapiens
```

Inheritance with extends and super

Classes can inherit from other classes using the extends keyword. The super keyword is used to call the constructor of the parent class and to access its methods.

```
class Parent {
  constructor(firstName,lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }
  fullName() {
    return `My Name is ${this.firstName}`
  }
  static sayHi(){
    return "Hello everyone..."
  }
}

class Child extends Parent{
    constructor(firstName,lastName,role){
        super(firstName,lastName)
        this.role = role;
    }
    getRole(){
        return `My role is ${this.role}`
    }
}
let parent1 = new Parent("Hema");
console.log(parent1);
let child1 = new Child("Mahesh","Coding","Student")
console.log(child1)
console.log(child1.role)
```

```
console.log(child1.getRole())
```

Additional Concepts

Private Fields

Private fields are properties that are not accessible outside the class. They are defined using the # prefix.

```
class Person {
    #ssn;

    constructor(name, age, ssn) {
        this.name = name;
        this.age = age;
        this.#ssn = ssn;
    }

    getSsn() {
        return this.#ssn;
    }
}

const person1 = new Person('Hema', 20, '123-45-6789');
console.log(person1.getSsn()); // Output: 123-45-6789
// console.log(person1.#ssn); // SyntaxError: Private field '#ssn' must be declared in an enclosing class
```

Object assign with Class:

```
let emp = {
    name1:"Mahesh",
    getName:function(){
        return "This is my name";
    }
}

class EmpInfo{
    constructor(name){
        this.name = name
    }
}
Object.assign(EmpInfo.prototype, emp);
let emp1 = new EmpInfo("Hema");
```

```
console.log(emp1);
console.log(emp1.name1);
console.log(emp1.getName());
```

Interview Questions

Question 1: What is a class in JavaScript?

- **Answer:** A class in JavaScript is a blueprint for creating objects. It encapsulates data and functions that operate on that data. Classes were introduced in ECMAScript 2015 (ES6) and provide a syntactic sugar over the existing prototype-based inheritance.

Question 2: How do you define a class in JavaScript?

- **Answer:** You can define a class in JavaScript using the class keyword followed by the class name and a block containing the constructor method and other methods.

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

    greet() {
        return `Hello, my name is ${this.name} and I am ${this.age} years old.`;
    }
}
```

Question 3: What is the purpose of the constructor method in a class?

- **Answer:** The constructor method is a special method for creating and initializing an object created with a class. It is called when an instance of the class is created and can be used to set initial properties of the object.

Question 4: How do you create an instance of a class in JavaScript?

- **Answer:** You create an instance of a class using the new keyword followed by the class name and any necessary arguments for the constructor.

```
let parent1 = new Parent("Hema");
```

Question 5: What are instance methods in a class?

- **Answer:** Instance methods are methods that are defined inside a class and are available to instances of the class. They operate on data contained in the instance.

```
class Person {
    constructor(name, age) {
```

```
        this.name = name;
        this.age = age;
    }

    greet() {
        return `Hello, my name is ${this.name} and I am ${this.age} years old.`;
    }
}
```

Question 6: What are static methods in a class?

- **Answer:** Static methods are defined on the class itself and are not available to instances of the class. They are called on the class directly and typically perform operations related to the class itself rather than any particular instance.

```
class Person {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
    static species() {
        return 'Homo sapiens';
    }
}
console.log(Person.species()); // Output: Homo sapiens
```

Question 7: How do you achieve inheritance in JavaScript classes?

- **Answer:** Inheritance in JavaScript classes is achieved using the extends keyword. The super keyword is used to call the constructor of the parent class and to access its methods.

```
class Person {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
    greet() {
        return `Hello, my name is ${this.name} and I am ${this.age} yearold.`;
    }
}
class Student extends Person {
    constructor(name, age, grade) {
        super(name, age); // Call the parent class constructor
        this.grade = grade;
    }
    study() {
```

```
        return `${this.name} is studying in grade ${this.grade}.`;
    }
}

const student = new Student('Hema', 20, 'A');
console.log(student.greet()); // Output: Hello, my name is Hema and I am 20 years old.
console.log(student.study()); // Output: Hema is studying in grade A.
```

Question 8: What are private fields in JavaScript classes, and how do you define them?

- **Answer:** Private fields in JavaScript classes are properties that are not accessible outside the class. They are defined using the # prefix.

```
class Person {
    #name = "";
    constructor(name){
        this.#name = name
    }
    getName(){
        return `My Name is ${this.#name}`
    }
}
let person1 = new Person("Hema")
console.log(person1)
console.log(person1.getName())
```