JavaScript Classes - Complete Notes

Introduction to Classes

JavaScript classes are a template for creating objects. They were introduced in ES6 and provide a clearer syntax for creating objects and dealing with inheritance.

Defining a Class

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

  greet() {
    console.log(`Hello, my name is ${this.name}`);
  }
}

const p = new Person("Alice", 25);
p.greet(); // Hello, my name is Alice
```

Class Expressions

```
const MyClass = class {
  constructor(id) {
    this.id = id;
```

```
  }
  showId() {
    console.log(this.id);
  }
};

const obj = new MyClass(101);
obj.showId();
```

Adding Methods

- Methods defined in a class go into the prototype.
- No commas between methods.

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

Constructor Method

- The constructor() is a special method used for creating and initializing objects.
- Only one constructor per class is allowed.

```
class Car {
  constructor(brand) {
    this.brand = brand;
```

```
  }
}
```

Inheritance

Use `extends` to create a subclass.

```
class Vehicle {
  constructor(type) {
    this.type = type;
  }
  info() {
    console.log(`This is a ${this.type}`);
  }
}

class Bike extends Vehicle {
  constructor(type, brand) {
    super(type);
    this.brand = brand;
  }

  display() {
    console.log(`Brand: ${this.brand}`);
  }
}
```

```
const b = new Bike("Two-wheeler", "Hero");
b.info(); // from parent
b.display(); // from child
```

Super Keyword

- Used to call the constructor or methods of a parent class.

```
class Parent {
  sayHello() {
    console.log("Hello from Parent");
  }
}

class Child extends Parent {
  sayHello() {
    super.sayHello();
    console.log("Hello from Child");
  }
}
```

Getters and Setters

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

```
  get name() {
    return this._name.toUpperCase();
  }

  set name(newName) {
    this._name = newName;
  }
}

const u = new User("john");
console.log(u.name); // JOHN
u.name = "mike";
console.log(u.name); // MIKE
```

Static Methods

- Static methods belong to the class, not the instances.

```
class MathUtil {
  static add(x, y) {
    return x + y;
  }
}

console.log(MathUtil.add(5, 3)); // 8
```

Private Fields and Methods (ES2022)

- Use # to define private fields or methods.

```
class Sample {
  #secret = "hidden";

  getSecret() {
    return this.#secret;
  }
}

const s = new Sample();
console.log(s.getSecret()); // hidden
// console.log(s.#secret); // SyntaxError
```

Class vs Function Constructor

- Classes are syntactic sugar over function constructors.
- Classes are not hoisted.
- Classes use strict mode by default.

```
function PersonFn(name) {
  this.name = name;
}
PersonFn.prototype.greet = function() {
  console.log("Hi, I'm " + this.name);
};
```

```
const p = new PersonFn("Mark");

p.greet();
```