Classes in ES6

- In this tutorial we'll learn how to create a class that can be used to create multiple objects of the same structure.
- A class uses the keyword class and contains a constructor method for initializing.

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

- A declared class can then be used to create multiple objects using the keyword new.

```
const person1 = new Person("John", 30);
const person2 = new Person("Sara", 25);
```

- Note : Class Declarations are not hoisted while Function Declarations are. If you try to access your class before declaring it, ReferenceError will be returned.
- You can also define a class with a class expression, where the class can be named or unnamed.
- A named class looks like this:

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

- In the unnamed class expression, a variable is simply assigned the class definition:

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

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

Quick Quiz

- Fill in the blanks to complete this Point class

```
____ Point {
  ________(x, y) {
    this.x = x;
    this.y = y;
  }
}
getX() {
  return ____.x;
}
getY() {
  return ____.y;
}
```

Class Methords in ES6

- ES6 introduced a shorthand that does not require the keyword function to create a method.
- The method name is the same as the property name.
- One type of methord is prototype methord, which is available to all instances of the class.

For example, the following class has a methord called `area` which returns the value of the `height * width` properties.

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
  get area() {
    return this.calcArea();
  }
  calcArea() {
    return this.height * this.width;
  }
}
const square = new Rectangle(5, 5);
console.log(square.area); // 25
```

- In the code above, area is a getter, calcArea is a method.
- Getter is a methord that returns the value of a property.

- Another type of method is the static method, which cannot be called through a class instance. Static methods are often used to create utility functions for an application

```
class Point {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }
  static distance(a, b) {
    const dx = a.x - b.x;
    const dy = a.y - b.y;
    return Math.hypot(dx, dy);
  }
}
const p1 = new Point(7, 2);
const p2 = new Point(3, 8);
console.log(Point.distance(p1, p2));
```

- As you can see, the static distance method is called directly using the class name, without an object.

Quiz Time

- Fill in the blanks to complete this class so that it outputs : `Jimmy says woof!`

```
class Dog {
  constructor(name) {
    ____.name = name;
  }
  bark() {
    console.log(this.____ + " says woof!");
  }
}
let d = new Dog("Jimmy");
d.____();
```

Inheritance in ES6

- The extends keyword is used in class declarations or class expressions to create a child of a class.
- The child inherits the properties and methods of the parent.

```
class Animal {
  constructor(name) {
    this.name = name;
  }
  speak() {
    console.log(this.name + " makes a noise.");
  }
```

```
}
class Dog extends Animal {
  speak() {
    console.log(this.name + " says woof!");
  }
}
let dog = new Dog("Jimmy");
dog.speak(); // Jimmy says woof!
```

- In the code above, the Dog class is a child of the Animal class, inheriting its properties and methods.
- If there is a constructor present in the subclass, it needs to first call super() before using this.
- Also, the super keyword is used to call parent's methods.

```
class Animal {
  constructor(name) {
    this.name = name;
  }
  speak() {
    console.log(this.name + " makes a noise.");
  }
}

class Dog extends Animal {
  speak() {
    super.speak(); // Super
    console.log(this.name + " says woof!");
  }
}

let dog = new Dog("Jimmy");
dog.speak();
// Jimmy makes a noise.
// Jimmy says woof!
```

- In the code above, the parent's speak() method is called using the super keyword.

Quiz Time

- Fill in the blanks to declare a class `Student` which inherits from the `Human` class.

```
class Human {
  constructor(name) {
    this.name = name;
  }
}
class ____ extends ____ {
  constructor(name, age) {
    ____(name, age);
    this.age = age;
```

```
    }
  }
```