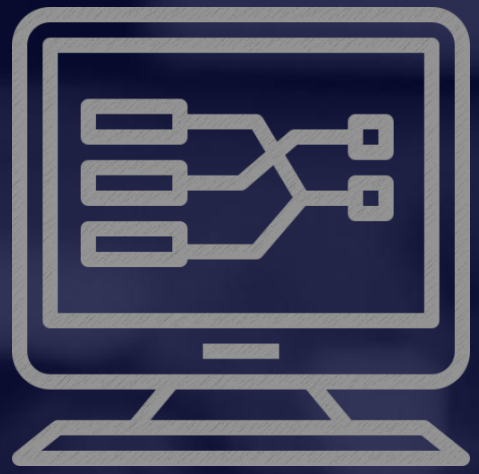

# JavaScript Classes



# Classes, Constructors, Properties

# Table of Contents

Definition, Declaration, Expression, Hoistin

# Classes in JS

# Class Definition

- **Structure** for objects
- Classes define:
  - **Data** (properties, attributes)
  - **Actions** (behavior)
- One class may have **many instances** (objects)
- The class syntax has two components:
  - **Class Expressions** and **Class Declarations**

# Class Declaration

- Use the **class keyword** with the name of the class
- The **constructor** defines class data

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
}
```

# Class Expression

- Another way to **define a class**
  - Class expressions can be **named** or **unnamed**

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
}
```

```
let Rectangle = class Rectangle2 {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
};
```

# Hoisting

- Function declarations **are hoisted** and class declarations **are not**
- You first need to declare your class and then access it, otherwise a **ReferenceError** will be thrown

```
const p = new Rectangle(); // ReferenceError
class Rectangle {}
```

- **Class expressions** are subject to the same **hoisting restrictions**

# Problem: Rectangle

- Write a **class** for a rectangle object
  - It needs to have the following properties:
    - **`width`**, **`height`** and **`color`**
  - And a **`calcArea()`** method

```
let rect = new Rectangle(4, 5, 'red');
console.log(rect.width);       // 4
console.log(rect.height);      // 5
console.log(rect.color);       // Red
console.log(rect.calcArea()); // 20
```

# Solution: Rectangle

```
class Rectangle {
    constructor(width, height, color) {
        this.width = width;
        this.height = height;
        this.color = color;
    }
    calcArea() {
        return this.width * this.height;
    }
}
```

# Class Body and Methods

## Definition, Constructor, Prototype, Fields

# Class Body

- The **constructor** is a special method for **creating** and **initializing** an object created with a class
- A **SyntaxError** will be thrown if a class contains **more than one** occurrence of a **constructor method**

```
class Rectangle {
  // Class Body
}
```

```
constructor() {
  // Class Body
}
```

```
class Rectangle() {
  // Syntax Error
}
```

# Prototype

- Objects **inherit properties** and **methods** from a **prototype**
- The **Prototype Property** allows you to add **new properties** to object **constructors**

```
function Person(first, last, age) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
}
Person.prototype.nationality = “American";
```

# Prototype Methods

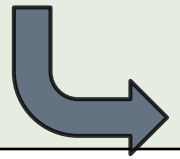
- Before ES2015 (ES6), **classes** were composed **manually**

```
function Rectangle(width, height) {
    this.width = width;
    this.height = height;
}

Rectangle.prototype.area = function () {
    return this.width * this.height;
}

let rect = new Rectangle(3, 5);
```

# Comparison with the New Syntax

```
class Rectangle {
    constructor(width, height) {
        this.width = width;
        this.height = height;
    }

    area() {
        return this.width * this.height;
    }
}
```

```
function Rectangle(width, height) {
    this.width = width;
    this.height = height;
}
```



```
Rectangle.prototype.area = function () {
    return this.width * this.height;
}
```

# Static Methods

- The **static** keyword defines a **static method** for a class

```
static staticMethod() { return 'Static method has been called';
}
```

- Called **without instantiating** their class and **cannot be called** through a class instance
- To call a **static** method of the same class, you can use the **this** keyword

```
static anotherStaticMethod() {
    return this.staticMethod() + ' from another method';
}
```

# Accessor Properties

Property getter

Property setter

Read-only property "area"

```
class Circle {
    constructor(radius) { this.radius = radius; }
    get diameter() { return 2 * this.radius; }
    set diameter(diameter) {
        this.radius = diameter / 2;
    }
    get area() {
        return Math.PI * this.radius * this.radius;
    }
}
```

# Accessor Properties in Action

```
let c = new Circle(2);
console.log(`Radius: ${c.radius}`); // 2
console.log(`Diameter: ${c.diameter}`); // 4
console.log(`Area: ${c.area}`); // 12.566370614359172
```

```
c.diameter = 1.6;
console.log(`Radius: ${c.radius}`); // 0.8
console.log(`Diameter: ${c.diameter}`); // 1.6
console.log(`Area: ${c.area}`); // 2.0106192982974678
```

# Private Properties

- Prefix each **private** property name with an **#**

```
function Point(x, y) {
    this.#x = x;
    this.#y = y;
}
```

- To make a **private** property **readable/writable** from any function, it's common to define **getters/setters**

# Accessing Private Properties

```
Point.prototype.getX = function () {
    return this.#x;
};
```

```
Point.prototype.setX = function (x) {
    this.#x = x;
};
```

```
Point.prototype.getY = function () {
    return this.#y;
};
```

```
Point.prototype.setY = function (y) {
    this.#y = y;
};
```

# Problem: Person

- Write a **class** that represent a personal record
- It needs to have the following properties:
  - **`firstName`**, **`lastName`**, **`age`** and **`email`**
- And a **`toString()`** method

```
let person = new Person('Anna', 'Simpson', 22, 'anna@yahoo.com');
console.log(person.toString());
// Anna Simpson (age: 22, email: anna@yahoo.com)
```

# Solution: Person

```
class Person {
    constructor(fName, lName, age, email) {
        this.firstName = fName;
        this.lastName = lName;
        this.age = age;
        this.email = email;
    }
    toString() {
        return `${this.firstName} ${this.lastName}
                (age: ${this.age}, email: ${this.email})`
    }
}
```

# Problem: Get People

- Write a **function** that returns an array of **Person** objects
  - Use the class from the previous task
  - There will be **no input**, the data is **static** and matches on this data

| First Name | Last Name | Age | Email |
|---|---|---|---|
| Anna | Simpson | 22 | anna@yahoo.com |
| Kingsland | | | |
| Stephan | Johnson | 25 | |
| Gabriel | Peterson | 24 | g.p@gmail.com |

# Solution: Get People

```
class Person {
    constructor(firstName, lastName, age, email) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
        this.email = email;
    }
    toString() {
        return `${this.firstName} ${this.lastName}
        (age: ${this.age}, email: ${this.email})`
    }
    static getPeople() {
        return [new Person('Anna', 'Simpson', 22, 'anna@yahoo.com'),
        ... //TODO for the rest of the people
   }
}
```

# Class Inheritance

## Inheriting Data and Methods

# Class Inheritance

- Classes can **inherit** (extend) other classes
- Child class **inherits** data + methods from its parent
- The **extends** keyword is used to create a class which is a **child of another class**
- **Child class** can:
  - Add **properties** (data)
  - Add / replace **methods**
  - Add / replace **accessor** properties

# Class Inheritance - Example

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

```
class Teacher extends Person {
    constructor(name, email, subject) {
        super(name, email);
        this.subject = subject;
    }
}
```

# Class Inheritance - Example

```
let p = new Person("Anna", "anna@gmail.com");
console.log(`Person: ${p.name} (${p.email})`);
// Person: Anna (anna@gmail.com)
```

```
let t = new Teacher("John", "joe@yahoo.com", "JavaScript");
console.log(
    `Teacher: ${t.name} (${t.email}), teaches ${t.subject}`
);
// Teacher: John (doe@yahoo.com), teaches JavaScript
```

# Live Exercises

## Summary

- Classes:
  - Provide structure for objects
  - May define methods
  - May define accessor properties
  - Can inherit other classes

# Questions?

# License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © Kingsland University – https://kingslanduniversity.com


KINGSLAND
UNIVERSITY
THANK YOU