wantsome
The friendly IT Academy

Saptamana 12

Partea 2

Programare Front-End

1. OOP - Recap

wantsome
The friendly IT Academy

1.1 OOP - Recap - Concepte

Concepte de baza in OOP

Clasa

Obiect

Metode

Proprietati

De retinut

- o clasa este o implementare a unui tip de date abstract, definind atributele si metodele care implementeaza structura de date, respectiv operatiile tipului de date abstract

- un obiect este o instanta a unei clase fiind unic determinat de numele sau si are o stare reprezentata de valorile atributelor sale la un moment dat

Concepte de baza in OOP

Abstractizare

Incapsulare

Modularizare

Ierarhizare

1.2 OOP - Recap - Principii

Principii de baza in OOP

Abstractizare

Incapsulare

Modularizare

Ierarhizare - Mostenire, Agregare -

2. OOP in JS - Recap

wantsome
The friendly IT Academy

2.1 Constructor

Constructors

```
function Person(firstName, lastName, age, eyeColor) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
  this.eyeColor = eyeColor;
  this.changeName = function (name) {
    this.lastName = name;
  };
}

const student = new Person(“Elon”, “Musk”, 38, “bluy”);

//////////////////////////////////////////////////////////////

var x1 = new Object();    // A new Object object
var x2 = new String();    // A new String object
var x3 = new Number();    // A new Number object
var x4 = new Boolean();   // A new Boolean object
var x5 = new Array();     // A new Array object
var x6 = new RegExp();    // A new RegExp object
var x7 = new Function();  // A new Function object
var x8 = new Date();      // A new Date object
```

<-- custom

<-- built-in

2.2 Prototype

Prototype

```
function Person(first, last, age, eyecolor) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
}

Person.prototype.name = function() { // adding new property to the constructor of Person
  return this.firstName + " " + this.lastName;
};
```

2. OOP in JS - The new way

wantsome
The friendly IT Academy

2.1 ES6 Classes

ES6 Class - Syntax

```
class Rectangle {

      width = 0;

      height = 0;

      constructor(width, height) {

            this.width = width;

            this.height = height;

      }

            get area() { return this.computeArea(); } // Getter

      computeArea() { return this.width * this.height; } // Method

}

const square = new Rectangle(10, 10);
```

ES6 Class - Hoisting

```
const p = new Rectangle(); // ReferenceError

class Rectangle {}
```

Atentie: Pentru declaratiile de clase nu se face *hoisting* !

ES6 Class - Static methods

```
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(5, 5);

const p2 = new Point(10, 10);

console.log(Point.distance(p1, p2)); // 7.0710678118654755
```

wantsome
The friendly IT Academy

ES6 Class - Inheritance with ***extends*** keyword

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

    speak() {
        console.log(`${this.name} makes a noise.`);
    }
}

class Dog extends Animal {
    constructor(name) {
        super(name); // call the super class constructor and pass in the name parameter
    }

    speak() {
        console.log(`${this.name} barks.`);
    }
}

let d = new Dog('Mitzie');
d.speak(); // Mitzie barks.
```

wantsome
The friendly IT Academy

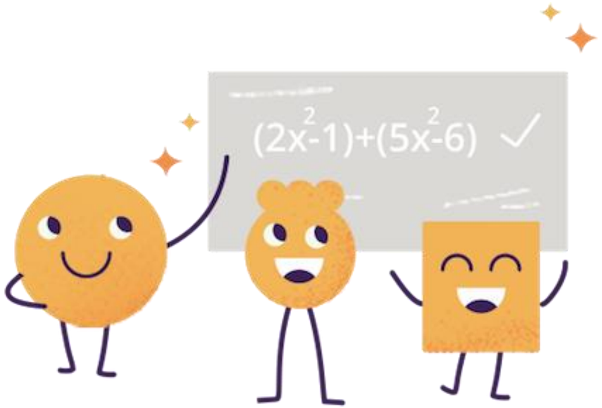
ES6 Class - ***super*** keyword

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

    speak() {
        console.log(`${this.name} makes a noise.`);
    }
}

class Lion extends Cat {
    speak() {
        super.speak();
        console.log(`${this.name} roars.`);
    }
}

let l = new Lion('Fuzzy');
l.speak();
// Fuzzy makes a noise.
// Fuzzy roars.
```

wantsome
The friendly IT Academy

PRACTICE: **ES6 Classes**

http://bit.do/ExClass1
http://bit.do/ExClassSuper
http://bit.do/exClassBonus



Project Guidelines:
Calling *Twitter API*

wantsome
The friendly IT Academy