ADVANCED JAVASCRIPT - ЧАСТ 1

ООП, ООП В JAVASCRIPT, THIS, НАСЛЕДЯВАНЕ, PROTOTYPE, PROTOTYPE CHAIN, CLOSURES

СЪДЪРЖАНИЕ

ДА ПРИПОМНИМ

- Каква е разликата между “ ‘ ` в JavaScript?

```
'Hello' === "Hello" //true
        'Hello' === `Hello` //true
```

- ` ___string___ ${_???_} ` ?

```
`Answer: ${6 * 7} !`
"Answer: 42 !"
```

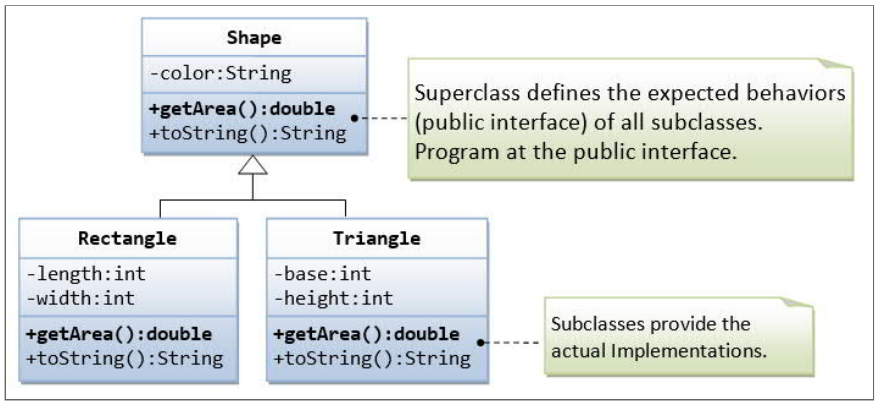
- Какво е spread operator?

```
[1, ...[2,  ...[3, 4]]] // [1, 2, 3, 4]
▶ (4) [1, 2, 3, 4]
```

ООП

- Позволява групиране на функции и променливи
- Парадигма в компютърното програмиране
- Програмата се моделира като набор от обекти от определен тип / шаблон / прототип клас

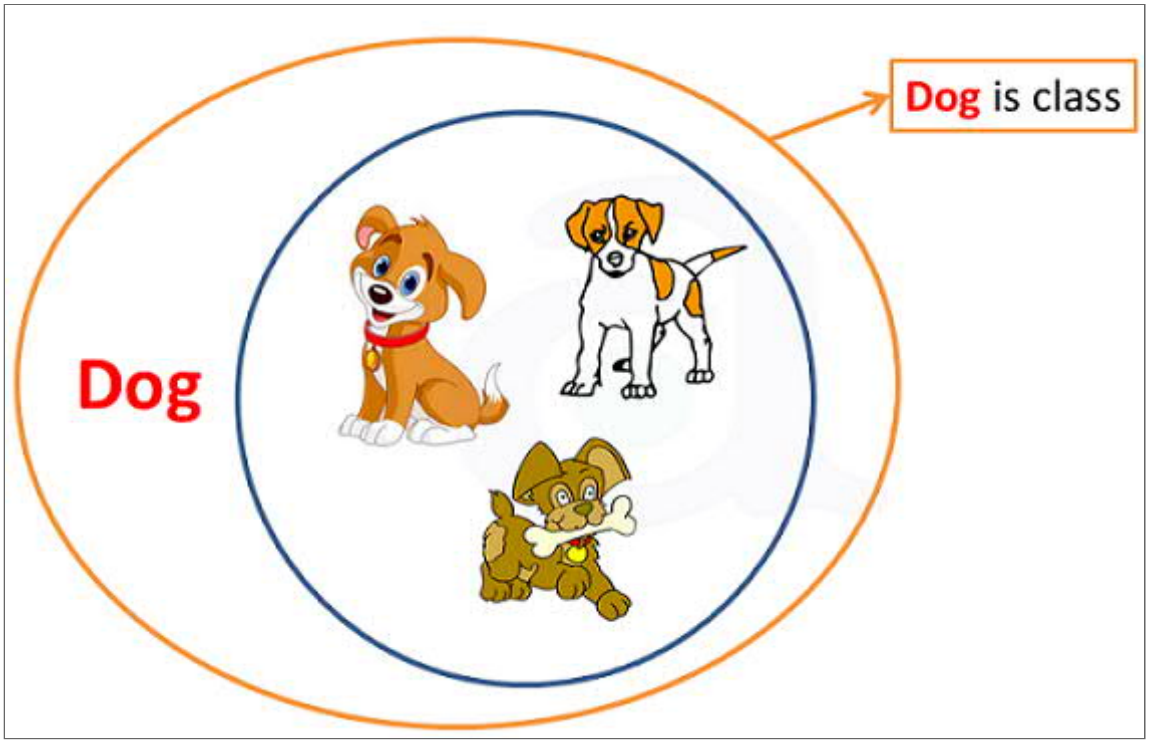
ООП ОСНОВНИ ПРИНЦИПИ



- Наследяване
  Квадрат наследява свойствата на **Правоъгълник**
- Полиморфизъм
  Квадрат е и **Правоъгълник**
- Енкапсулация
  Кръг не се интересува от това как се намира лице на **Правоъгълник**

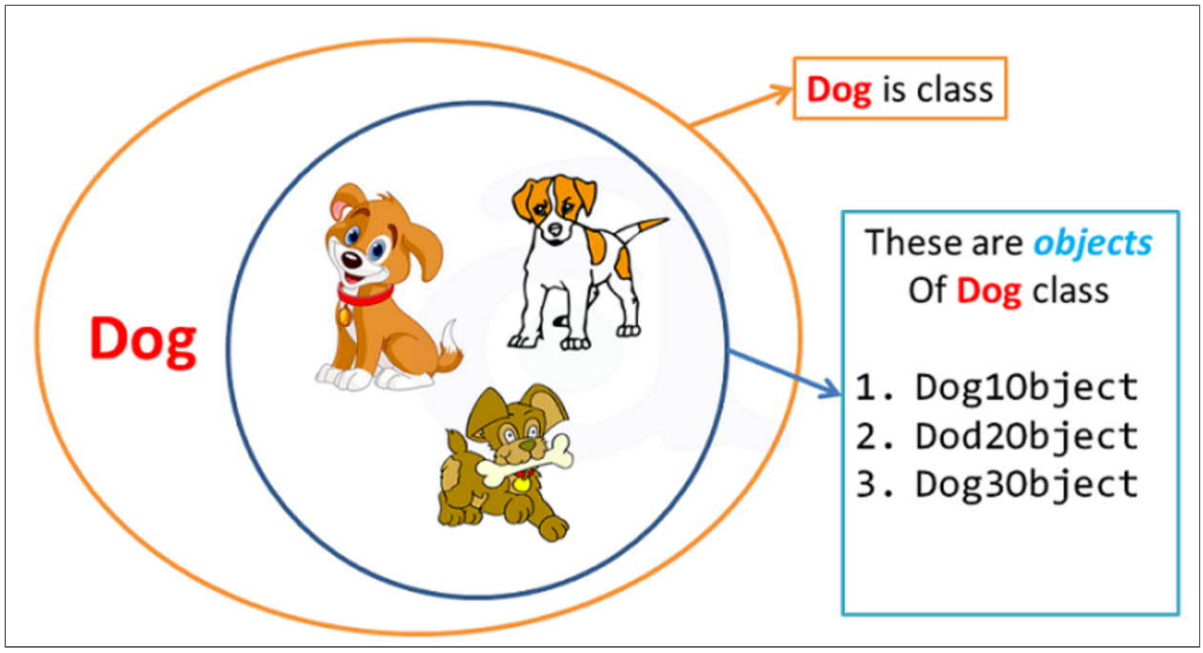
КЛАСОВЕ

- Всеки клас има цел и представлява шаблон / прототип за реален обект от живота, например Човек, Правоъгълник, Текстово поле
- Класът може да описва свойства (properties), както всички ние имаме цвят на очите, възраст, пол и т.н.
- Класът може да описва действия (method), например кучетата лаят
- Класът е шаблон / прототип за създаването на Обекти



ОБЕКТИ

- Представляват конкретен екземпляр (инстанция) на клас
- Всяка инстанция съдържа свой конкретен набор от свойства
- Едно куче може да е с жълта, черна или друга по цвят козина



ООП В JAVASCRIPT

- Прототипно ориентиран език
- В JS няма класове а протипи, които извършват аналогична функция *
- Обектите се създават и наследяват от обекти

```
* С новостите от ES6 се въвежда думата "class"
но тя е само синтактично подобрение.
Не променя имплементацията на езика.
```

КАК СЕ ДЕФИНИРА КЛАС

- Използват се функции за създаване на обекти
- Функцията служи за конструктор

```
function Person() { }

    var firstPerson = new Person(); //first instance of Person
```

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

    var firstPerson = new Person("Adam", 33); //first instance of P
    var secondPerson = new Person("Eva", 30); //second instance of
```

ES6 КЛАСОВЕ

- Използват се ключовата дума class
- Има дефиниран конструктор

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

    const peter = new Person("Peter", 18);
```

THIS

- Специален обект в JavaScript
- Стойността му сочи към обекта от където се извиква функцията. Контекста на функцията
- Тази стойност може да бъде подменена при нужда

THIS В КОНТЕКСТ НА КЛАС / ПРОТОТИП

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

    const peter = new Person("Peter", 18);

    peter.printName();
```

THIS В ГЛОБАЛНИЯ КОНТЕКСТ НА BROWSER WINDOW

```
console.log(this); // this === window
    // Window {parent: Window, postMessage: ƒ, blur: ƒ, focus: ƒ, c
```

```
function printThis() {
        console.log(this);
    }

    printThis();

    window.printThis();
```

В КОНТЕКСТ НА ОБЕКТ СЪЗДАДЕН ОТ LITERAL

```
const myDog = {
        name: "Johny",
        barks: function (message) {
            console.log(`${this.name} barks '${message}
        }
    }

    myDog.barks("Niom niom");
```

ПОДМЯНА НА THIS

- Използва се когато искате един метод да се държи по различен начин в различни контексти
- Може да се подмени с някой от методите bind(), call() и apply()

ПОДМЯНА НА THIS

С call() или apply()

- Видяхме че this метод сочи към глобалния прозорец.

```
function talk (message) {
        console.log(`${this.name} says '${message}'`);
    }
    talk("Good morning!");
```

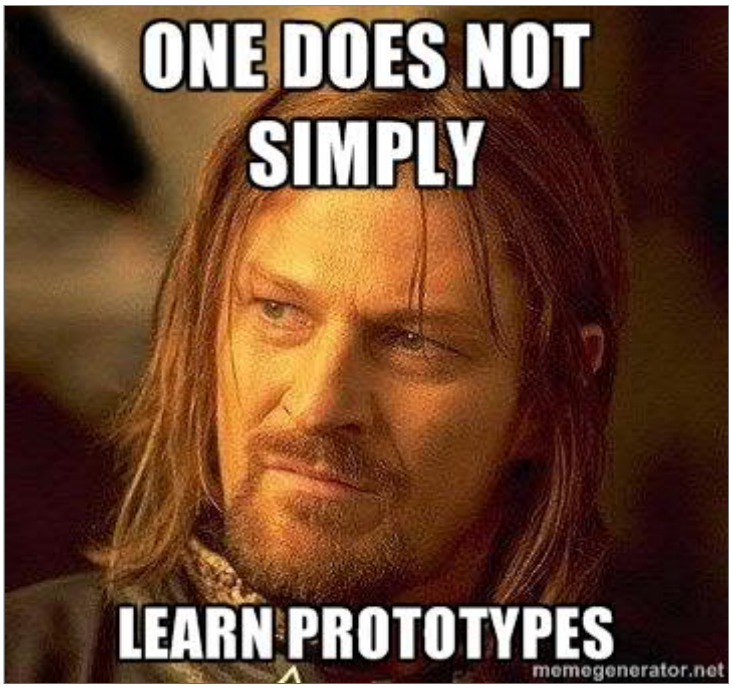
```
const person1 = new Person("Simone");

    talk.call(person1, "Good morning!"); // talk.apply(person1, ["G
```

ПОДМЯНА НА THIS

- Когато искате да дадете контекст на даден метод, но не искате да го викате, можете да използвате ”bind()”

```
function talk (message) {
        console.log(`${this.name} says '${message}'`);
    }

    const person2 = new Person("Anton");

    var person2Talk = talk.bind(person2);

    // ...
    // do something else
    // ...

    person2Talk("Hello");
```

PROTOTYPE/ПРОТОТИП

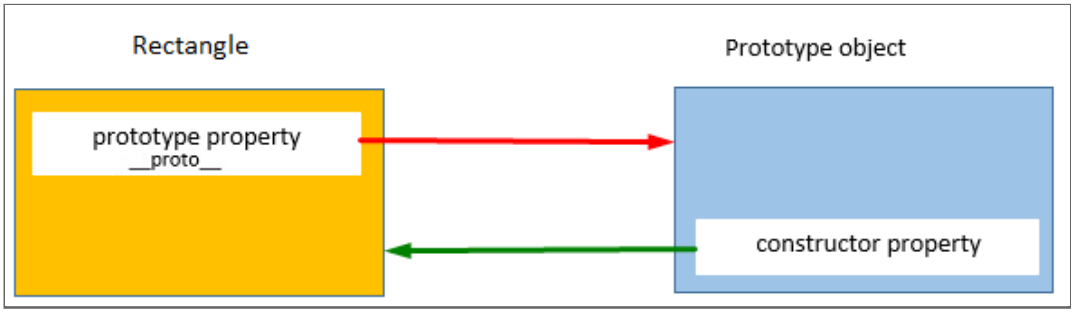


- Прототип (Prototype) е обект,който има свои пропъртита и методи
- При създаване на клас, JS engine-а му добавя пропърти prototype
- Всеки клас достъпва своя прототип чрез думата prototype (Class.prototype)

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
}
console.log(Rectangle.prototype); //{constructor: ƒ }
console.log(typeof Rectangle); //function
```

PROTOTYPE/ПРОТОТИП

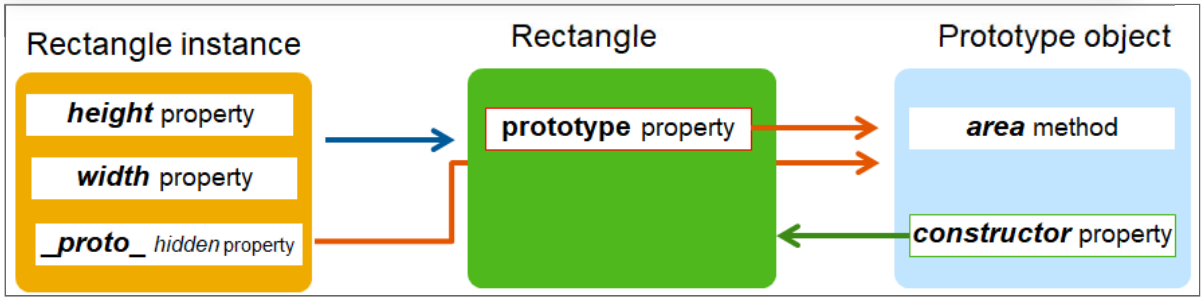
- Прототип (Prototype) има пропърти constructor по дефолт
- Пропъртито constructor сочи към функцията, която е собственик на prototype



PROPERTY НА ИНСТАНЦИЯТА

- Всяка инстанция от даден тип споделя прототипа му

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
    area() { return 2 * this.height + 2 * this.width; }
};
let r1 = new Rectangle(10, 15);
let r2 = new Rectangle(11, 14);
console.log(r1.height); // prints 10
console.log(r2.width); // print 14
console.log(r1);
```



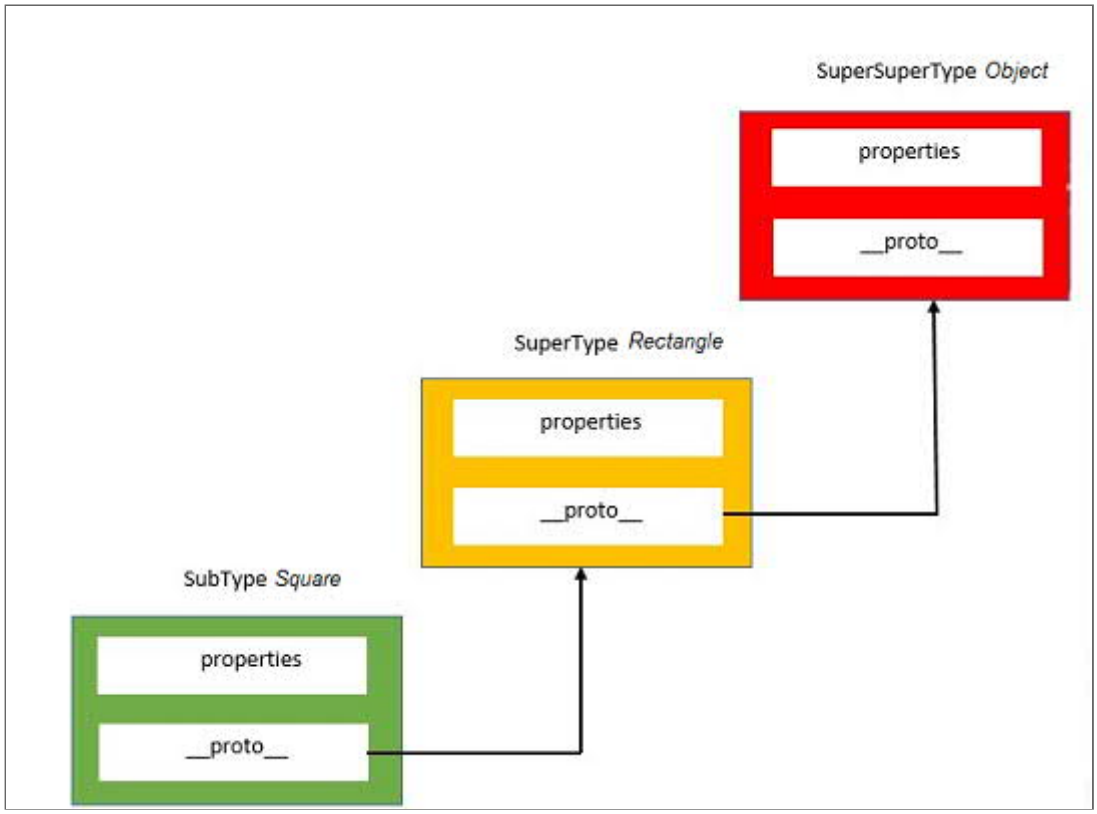
ДОБАВЯНЕ НА МЕТОД В ПРОТОТИПА

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
    area() { return 2 * this.height + 2 * this.width; }
};

Rectangle.prototype.sayHello = function () { console.log("Hello");
```

СТАТИЧНИ PROPERTIES

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
    area() { return 2 * this.height + 2 * this.width; }
}

Rectangle.sides = 4;
console.log(Rectangle.sides); // prints 4
```

НАСЛЕДЯВАНЕ



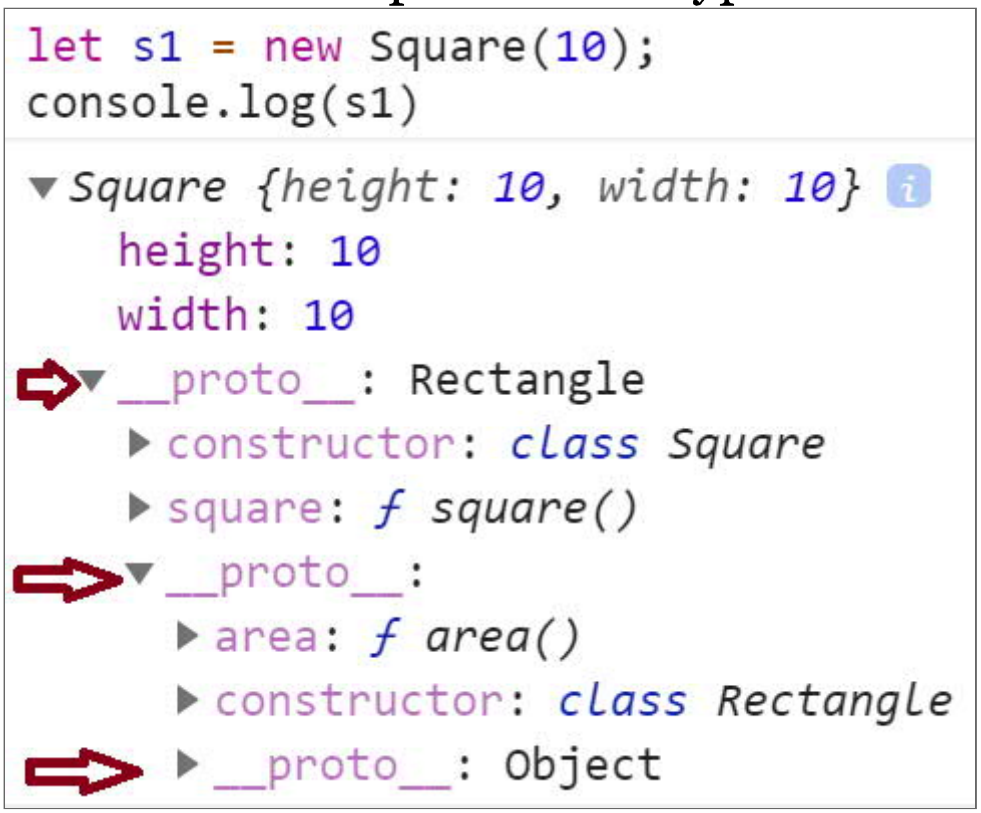
- Всички обекти наследяват класа Object

НАСЛЕДЯВАНЕ

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
    area() { return 2 * this.height + 2 * this.width; }
}
class Square extends Rectangle {
    constructor(size) {
        super(size, size);
    }
    square() { return this.height * this.height; }
}

let s1 = new Square(10);
console.log(s1.height); // prints 10
console.log(s1.width); // print 10
console.log(s1.square()); // prints 100
console.log(s1.area()); // print 40
```

PROTOTYPE CHAIN

- Обектите в JavaScript имат само един prototype
- Прототипът също има такъв, който от своя страна също има такъв
- Това се нарича “Prototype chain“



PROTOTYPE CHAIN

Когато достъпваме property на обект:

- Търси се даденото property в обекта
- Ако там няма такова, се търси в неговия prototype, ако не – в прототипа на прототипа
- При достигане на дъното/края на prototype chain, се връща undefined

КАК РАБОТИ PROTOTYPE CHAIN - ДЕМО

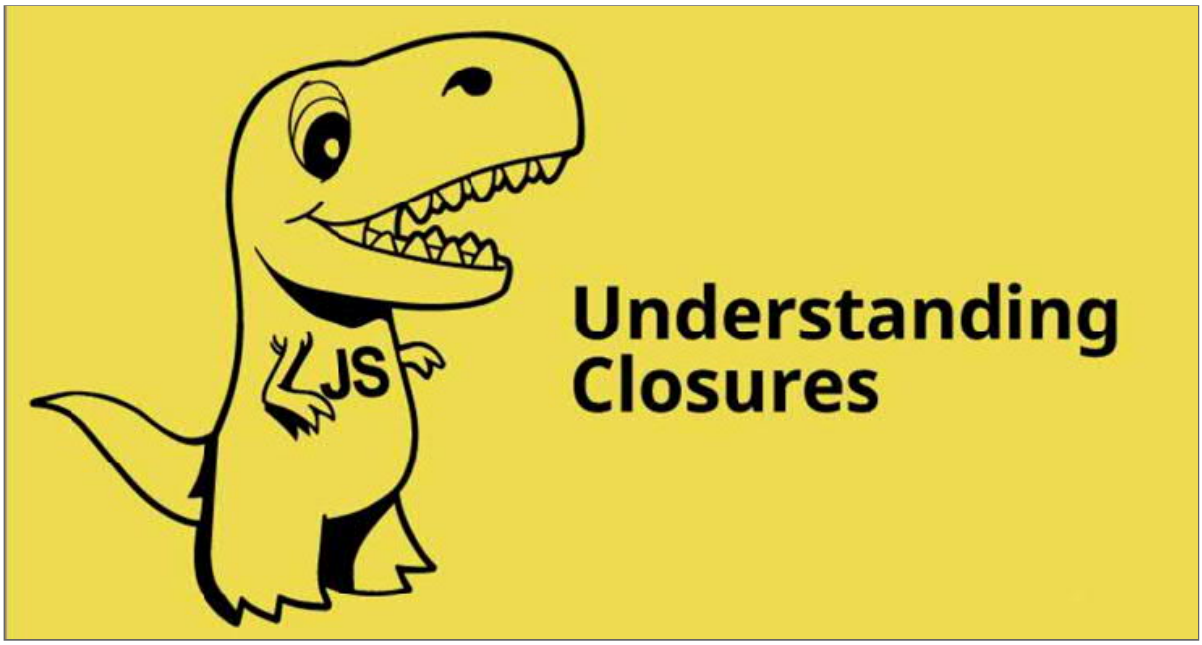
```
class Person {
    constructor(name) {
        this.name = name;
    }
}
Person.prototype.age = 25;

let oFirstPerson = new Person("Adam");

console.log(oFirstPerson.name);
console.log(oFirstPerson.age);
console.log(oFirstPerson.valueOf());
console.log(oFirstPerson.nonExistingProperty);
```

ИЗВИКВАНЕ НА РОДИТЕЛСКИ МЕТОД

```
class Person {
    constructor(name) {
        this.name = name;
    }
    introduce() {
        console.log(`I am a first born named ${this.name}`);
    }
}

class Student extends Person {
    introduce() {
        super.introduce();
        console.log(`I am a student`);
    }
}

let o1 = new Person("Adam");
let o2 = new Student("Jessie");
console.log(o1.introduce());
console.log(o2.introduce());
```


SAY PROTOTYPE ONE MORE TIME
I WILL FIND YOU
AND I WILL KILL YOU
VIA 9GAG.COM
imgflip.com

CLOSURES



КАКВО Е CLOSURE?

Значение

- Функция във функция
- Запазва контекста, в който е дефинирана вътрешната

```
function outer() {
    let x = 5;
    function inner(y) {
        return x + y;
    }
    console.log(inner(10));
}

outer();
```

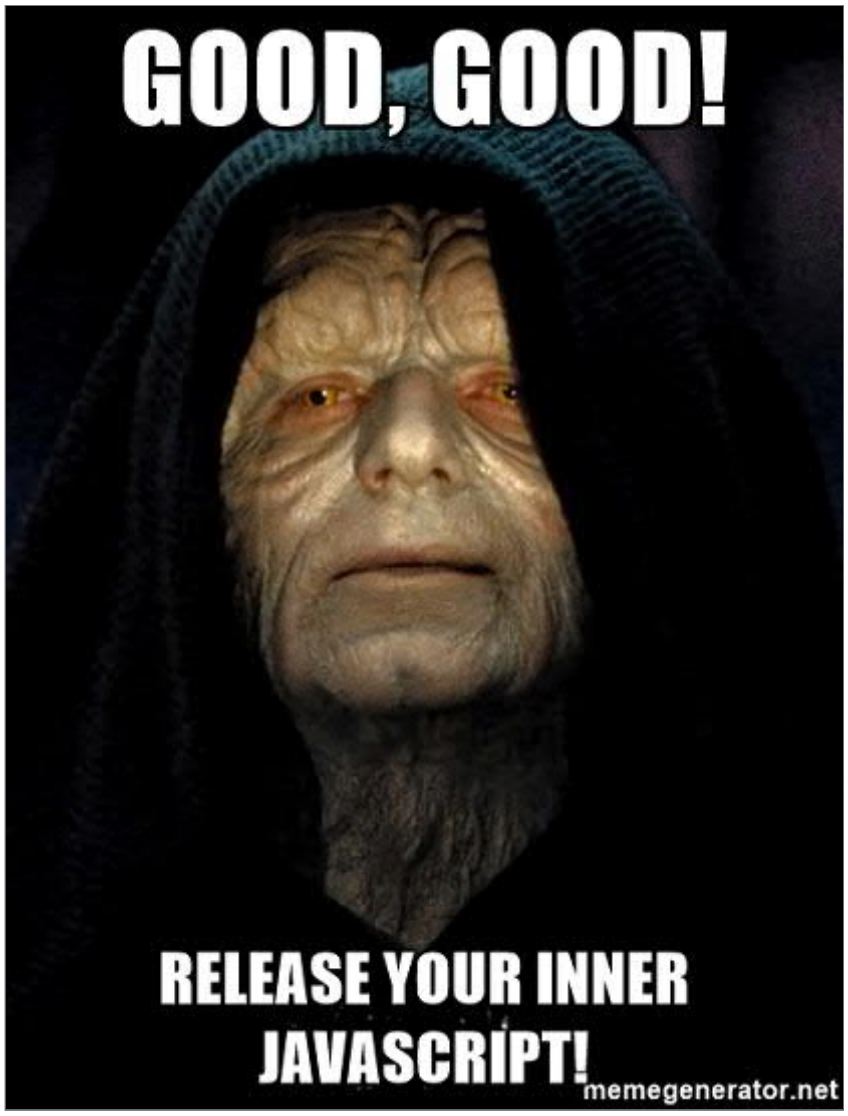
КАКВО Е CLOSURE?

Употреба

- Скриване на информация (private променливи и методи)

```
function outer() {
    let x = 5;
    return function inner(y) {
        return x + y;
    }
}

let newFunction = outer();
console.log(newFunction(4));
```

ВЪПРОСИ?


GOOD, GOOD!
RELEASE YOUR INNER JAVASCRIPT!
memegenerator.net

БЛАГОДАРИМ ВИ!

names: "Владислав Илиев
emails: **" vladislav.iliev@sap.com**