ООП в JavaScript

- Прототипна верига
- This
- Принципи

Но преди това малко въпроси

- Какво беше hoisting ?
- Какво връщат функциите без return?
- Как работи тройния оператор ?
- Кои типове данни са референтни ?
- typeof NaN ?

Още малко функции

- Видове функции.
  - Function declaration : function printMsg(msg){return msg}
  - Function expression: var printMsg = function(msg){return msg}
  - IIFI : (function(){ console.log('executed') })()
    - Как работят ?
    - За какво се използват ?

Финкциите са обекти

- Пропъртита и методи принадлежащи на всяка функция
  - length
  - name
  - call()
  - apply()
  - toString() (принадлежи на всичко което е обект в JS)
  - Какво е arguments ?

Какво е scope ?

- Function scope
- Global scope

Какво е ООП ???

- ООП е парадигма в програмирането, която ни дава допълнителна абстракция върху кода.
- Кода ни трябва да се състои от, взаимодействащи си обекти, а не обикновена функциналност.
  - Всеки обект има специялно предназначение или роля.
  - Всеки обект може да съдържа в себе си други обекти.

Класическо ООП

- JavaScript използва функциите за създаване на обекти.
- Функциите играят ролята на конструктори и зъздават обекти, когат бъдат извикани с new преди името на функцията.
  - Тези функции нямат за цел да свършат някаква работа, те се използват за шаблони за създаване на обекти.

```
function Person() {  };
var gosho = new Person();
```

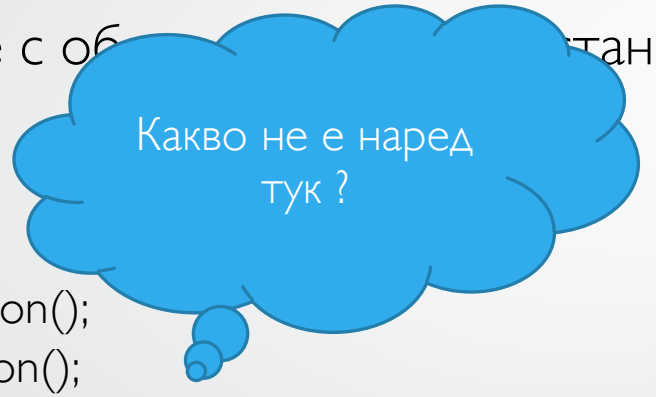
Създаване на обекти

- Когато създаваме с обект new всяка инстанция е независима една от друга

```
var gosho = new Person();
var maria = new Person();
maria.name = "Maria";
gosho.name = "Georgi";
```

Създаване на обекти

- Когато създаваме с об танция е независима една от друга




```
var gosho = new Person();
var maria = new Person();
maria.name = "Maria";
gosho.name = "Georgi";
```

Правилния начин за създаване на обекти

```
function Person(name, age){
        this._name  =  name;
        this._age  =  age;
         console.log('Name: ' + this._name + ' age: ' + this._age);
}
var gosho = new Person('Georgi', 23);
var maria = new Person('Maria', 44);
```

Всеки клас трябда да дефинира всичко за да работи, без да му се добавя нищо допулнително .

this в контекста на javascript

Какво е "this" ?

За какво се използва ?

"this" във функция.

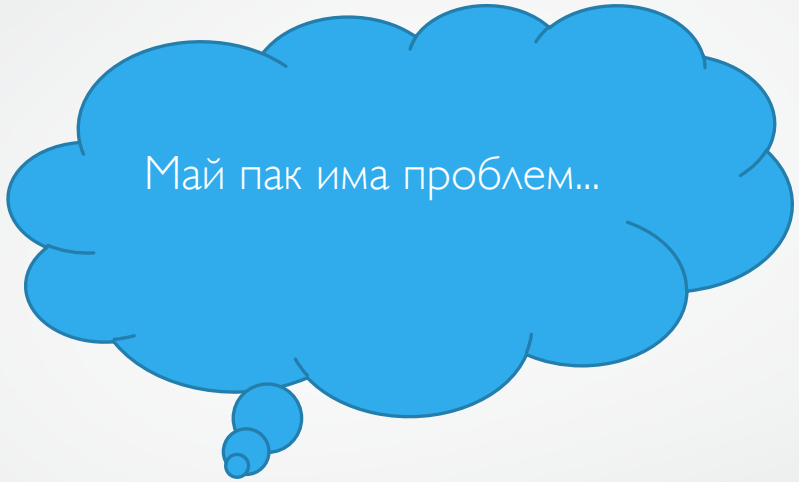
"this" в обект.

Кога "this" е "window" ?

Как добавяме методи на класа ?

```
function Person(name, age){
        this._name = name;
        this._age = age;
        this.sayHello = function(){
                console.log('Hello Im' + this._name + ' and Im ' + this._age + ' years
old')
         }
}
```

Как добавяме методи на класа ?




```
function Person(name, age){
        this._name = name;
        this._age = age;
        this.sayHello = function(){
                console.log('Hello Im' + this._name + ' and Im ' + this._age + ' years
old')
        }
}
```

Какво е prototype ???

- Всички функции имат prototype.
- Това е най-обикновен обект, който се споделя от всички инстанции на дадения клас.
- Дава достъп до this-а на конструктора.
- Използва се за добавяне на методи, своиства и при наследяване.

Какво печелим и какво губим с използването на prototype ?

- Печелим performance.
- Губим encapsulation.

Наследяване и прототипна верига

- Защо ни е наследяването ?
- Как работи прототипната верига ?
- Какво е constructor stealing ?
- Малко абстракция.

Пинципите на ООП

- Енкапсулация
- Наследяване
- Абстракция
- Полиморфизъм

Енкапсулация

Това е способността един обект да бъде контейнер за съхранение на свойства и методи. Идеята е да имаме данни, които да не могат да бъдат доступвани от друг код извън класа.

Наследяване

Позволява ни да правим ерархия от класове, с което можем да преизползваме код от вече наследени класове.

Абстракция

Това са класове, които се използват само за, като в тях може да имаме методи които сме задължени да опишем в наследяващите ги класове. От абстрактните класове не трябва да се правят инстанции.

Полиморфизъм

В превод от гръци означава 'многоформен'. Полиморфизъм се наблюдава, когато при наследяване имаме два класа с едно и също, който се реализира по различен начин и преви нещо различно.