oop- JS

hasOwnProperty - האם יש לו מאפיין כלשהו

אין אפשר לשאול האם יש לו פונק'. ✓

זה כמו property

יש 2 צורות להקצאת אובייקטים

1) var שם אובייקט = { - פה זה אובייקט אחד.

מאפיינים

}

2) function שם אובייקט (מקבל מאפיינים) {

this.מאפיין = מאפיין שקבלו

},

ואז אח"כ אפשר להקצות הרבה אובייקטים מסוגו: var שם = new שם אובייקט (~~לאובייקט~~ ערכים)

אם כותבים משתנה בלי var - זה נהיה גלובלי וכל ה script מכיר אותו.

לזו שימושו משתנה פרטי - באמצעות var, גם לפונקציות, אם יש פונק' בתוך פונק' בלי this זה גם פרטי.

משתנה רגיל - באמצעות this

פונקציות ~~משתנה~~ privileged : מחזירות את מאפיינים פרטיים או פונקציות פרטיות.

prototype - מתאימה, במקום שיהיה בתוך פונקציה הרבה פעמים, זה משהו פעם אחת.

של אובייקט

דרך שניה - ליצור הורשה - לדרוס פונקציה של ה base קיים.

.call(this, ✓)

=== משווה את הטיפוס וגם את הערך

object.create(prototype של שם

לדוג' דריסה של פרח בצמח:

returnplant.prototype.myfunction.call(this) + דברים של הבן

object reflection - מעבר על כל המאפיינים של האובייקט

object.defineProperty -> לוקח שם של אובייקט ומאפיינים

הוסף ל static - מאפיין . שם מחלקה

```
<script type="text/javascript">
//built in objects
var d = new Date();
var arr = new Array();
//constructor function
function Programmer(fName, lName, lang, salary){
    this.firstName = fName;
    this.lastName = lName;
    this.language = lang;
    this.salary = salary;
    this.getFullName = function(){
        return this.firstName + " " + this.lastName;
    }
}
//define objects by constructor function
var programmer1 = new Programmer("Sari", "Cohen", "angular", 10000);
var programmer2 = new Programmer("Brachi", "Levi", "react", 12000);
console.log(programmer1.language);
console.log(programmer2.salary);
console.log(programmer1.getFullName());
console.log(programmer2.getFullName());
//define array of objects by constructor function
var progArray = [programmer1, programmer2, new Programmer("Zipi", "Klein", 
  "C#", 7000)];

console.log(programmer1.hasOwnProperty("firstName")); //true
console.log(programmer2.hasOwnProperty("address")); //false

//object reflection
for(var key in programmer1){
    if(typeof programmer1[key] !== 'function'){// check if the property is not 
       function
        console.log(key + "-" + programmer1[key]); //print the key and value 
          of the proprty
    }
}

//define object literal notation
var webSite = {
    clientSide: "JavaScript",
    serverSide: "Java",
    dataBase: "SQL",
    getLangs: function(){
        return this.clientSide + " " + this.serverSide;
    }
}
console.log(webSite.getLangs());

//public, private and privileged function
function Programmer2(fName, lName){
    var firstName = fName; //private field
    this.lastName = lName; //public field
    var privateGetFirstName = function(){ //private field
         return firstName;
    }
    function privateFunction(){ //another way to define private field
```

```
        return true;
    }
    this.privilegedGetFirstName = function(){ //privileged function
        return getFirstName();
    }
}
var p1 = new Programmer2("Yossi", "Man");
console.log(p1.firstName); //undefined
// console.log(privateFunction()); //error - privateFunction is not defined

function Programmer3(lang, salary){
   var _lang = lang;
   var _salary = salary;
   Object.defineProperty(this, "lang", { //read only field
       get: function(){
         return _lang;
       },
   });
   Object.defineProperty(this, "salary", {
       get: function(){
          return _salary;
       },
       set: function(value){
          if(value < 0){
              alert("incorrect salary");
          }
          else{
              _salary = value;
          }
       }
   });
}
var p3 = new Programmer3("node.js", 8000);
p3.lang = "android";
console.log(p3.lang); // node.js
//the lang wasn't changed because this field doesn't have set function.
p3.salary = -50; //an error alert will appear

function Circle(radius){
   this.radius = radius;
   Circle.PI = 3.14; //static property
   this.calculateArea = function(){
       return Circle.PI * this.radius * this.radius;
   }
}
var circle1 = new Circle(5);
console.log(circle1.calculateArea()); //5 * 5 * 3.14

var s = new String("string1");
console.log(s.toUpperCase());
String.prototype.toUpperCase = function(){//override the function toUpperCase
    return "aaaa";
}
console.log(s.toUpperCase()); //aaaa

function Worker(salary){
   this.salary = salary;
}
Worker.prototype.getDetails = function(){
    return this.salary;
}
function Programmer4(salary, lang){
   Worker.call(this, salary);
   this.lang = lang;
}
Programmer4.prototype = Object.create(Worker.prototype);
Programmer4.prototype.getDetails = function(){
    return Worker.prototype.getDetails.call(this) + " " + this.lang;
}
var w = new Worker(5000);
var p = new Programmer4(10000, "swift");
console.log(w.getDetails());
console.log(p.getDetails());

</script>
```