JavaScript Objects

objects - אובייקטים

לאובייקט יש פונקציות ומאפיינים לדוגמא

```
let person = {
    firstName: 'John', lastName: 'Deer', getFullName: function () {
        return this.firstName + this.lastName;
    }
};

alert(person.firstName);

alert(person.getFullName());
```

ל this יש משמעויות רבות ב JS אבל בתוך פונקציה של אוביקט הוא מצביע על האוביקט שהפעיל את הפונקציה

גישה למאפיין או פונקציה של אובייקט נעשית בעזרת שם האובייקט ונקודה

בהגדרת האובייקט אפשר לרשום את firstName , lastName עם מרכאות

http://www.nathankrasney.com/

אובייקט וזיכרון

חשוב להבין שיש כאן שני חלקים :

- הקצאת מקום לאוביקט קרי כתובת - על ה stack
- הקצאת מקום לאיברים של האובייקט - על ה heap

JS מבצע את הקצאת זיכרון בצורה אוטומטית

JS מבצע את שחרור הזכרון בצורה אוטומטית בעזרת garbage collector

מערך של אובייקטים

ראה דוגמא כאן

value type מול reference type

Value type

- Number
- Boolean
- String
- Symbol(es6)

Reference type

- Object
- Array
- Function
- Date

הערך בתא הוא **הכתובת** אליה המשתנה מצביע.
המשתנה מוקצה ב stack והמידע עצמו באזור
זכרון שנקרא heap זה זיכרון פחות מהיר אבל בעל
הרבה יותר מקום ביחס ל stack

הערך בתא הוא **הערך** של המשתנה
המשתנה מוקצה באזור זכרון שנקרא stack
זה זיכרון מהיר אך מאוד מוגבל במקום

http://www.nathankrasney.com/

השוואה בין אובייקטים

להבין את מה משווים :

את הערך של הכתובת

או את מה שהאובייקט מצביע עליו

Reference type vs value type - 1 דוגמא

```
let a=1,b,obj1 = {name : 'Avi'},obj2;
b=a;
a=2;
console.log(b,b==a);
obj2=obj1;
obj1.name='Yossi';
console.log(obj2,obj1 == obj2);
```

השמת ערך !!!

מה אתם מצפים לקבל ?

השמת כתובת !!!

מה אתם מצפים לקבל ?

איך תראה תמונת הזיכרון הכוללת את התאים a,b,obj1,obj2 , **בלי ההבנה של זה לא תבינו מה קורה**

http://www.nathankrasney.com/

Reference type vs value type - 2 דוגמא

```
let array1,array2 = [1,2,3,6];
array1 = array2;
array1[0]=8;
array2[1]=18;
console.log(array1 == array2);
console.log(array1,array2)
```

מה אתם מצפים לקבל

איך תראה תמונת הזיכרון הכוללת את התאים array1 , array2, **בלי ההבנה של זה לא תבינו מה קורה**

http://www.nathankrasney.com/

Reference type vs value type - 3 דוגמא

```
let array1 = [{ name: "Jim", age: 33 }, { name: "John", age: 22 }],
 array2 = [1, 2, 3, 6],
 a,
 obj1;

a = array2[2];
a = 44;
console.log(array2);

obj1 = array1[1];
obj1.age = 55;
console.log(array1);
```

מה אתם מצפים לקבל

איך תראה תמונת הזיכרון הכוללת את התאים obj1 , a , array2 , array1 ,
בלי ההבנה של זה לא תבינו מה קורה

http://www.nathankrasney.com/

שימוש בפונקציה כמשתנה

```
const val = function (num1 , num2){
   return num1+num2;
}

console.log(val(1,2));
```

יצירת אוביקט - constructor function \ factory function

```
function Person  (name ,age){
   this.name = name;
   this.age = age;
  this.write = function(){
      console.log(`name : ${this.name} , age : ${this.age}`)
   }
}

let jim = new Person('Jim' , 33);
let jane = new Person('Jane' , 21);
console.log(jim.name);
jim.write();
```

פונקצית הבנאי נכתבת בדרך כלל באות גדולה לדוגמא Person

לכל פונקציה יש אובייקט this אשר מצביע לאובייקט הגלובלי - window עבור הדפדפן , אבל new גורם לו להצביע אל האובייקט שנוצר

האופרטור new יוצר אובייקט ריק ואז פונקצית הבנאי מופעלת. האובייקט שנוצר והאובייקט jim מצביעים לאותו מקום בזיכרון heap

http://www.nathankrasney.com/

ירושת אובייקט - prototype property

- לכל אובייקט JS יש מאפיין פרטי בשם prototype אשר מאפשר ירושה. המאפיין הזה מצביע אל אובייקט בשם prototype
- אובייקט יכול לשים תחת מאפיין prototype את כל המאפיינים והפונקציות שהוא מאפשר לרשת
- כל אובייקט יורש by default אובייקט בשם Object ויכול להשתמש בפונקציות שלו לדוגמא hasOwnProperty
- ירושה בין אובייקטים מתאפשרת באמצעות מושג שנקרא prototype chain
- שפות מונחות עצמים כמו C++ וכיוצא באלה תומכות בירושה של מחלקות ואילו JS תומך בירושת אוביקטים. בשני המקרים המטרה לעשות code reuse ז"א להשתמש בקוד קיים ולא לכתוב אותו מחדש

דוגמא לשימוש ב prototype לשיתוף מאפיין

```
var Person = function (name ,age){
  this.name = name;
  this.age = age;
  this.write = function(){
     console.log(`name : ${this.name} , age : ${this.age}`)
  }
}
Person.prototype.nationality = “Israel”;

let jim = new Person('Jim' , 33);
let jane = new Person('Jane' , 21);

console.log(jim.nationality,jane.nationality);
```

Every object instantiated using new Person will have nationality as property. **Benefit : less memory consumption , less code to write**

Will console Israel Israel

http://www.nathankrasney.com/

Prototype in the console and prototype chain

Add console.log(jim) to previous sample and you can see the nationality under **__proto__** which stand for the prototype

```
▼Person {name: "Jim", age: 33, write: ƒ}
    age: 33
    name: "Jim"
  ▶write: ƒ ()
  ▼__proto__:
    nationality: "Israel"
    ▶constructor: ƒ (name ,age)
    ▼__proto__:
      ▶constructor: ƒ Object()
      ▶hasOwnProperty: ƒ hasOwnProperty()
      ▶isPrototypeOf: ƒ isPrototypeOf()
      ▶propertyIsEnumerable: ƒ propertyIsEnumerable()
      ▶toLocaleString: ƒ toLocaleString()
      ▶toString: ƒ toString()
      ▶valueOf: ƒ valueOf()
      ▶__defineGetter__: ƒ __defineGetter__()
      ▶__defineSetter__: ƒ __defineSetter__()
      ▶__lookupGetter__: ƒ __lookupGetter__()
      ▶__lookupSetter__: ƒ __lookupSetter__()
      ▶get __proto__: ƒ __proto__()
      ▶set __proto__: ƒ __proto__()
```

Notice that there is another __proto__ and this belong to Object which is inherited by ANY object default. This is called prototype chain

http://www.nathankrasney.com/

שימוש ב __proto__ להצגת מאפיינים ופונקציות של מערך

```
let array = [1 , 5, 8];
console.log(array);
```

```
▼Array(3)
  0: 1
  1: 5
  2: 8
  length: 3
  ▼__proto__: Array(0)
    ▶concat: ƒ concat()
    ▶constructor: ƒ Array()
    ▶copyWithin: ƒ copyWithin()
    ▶entries: ƒ entries()
    ▶every: ƒ every()
    ▶fill: ƒ fill()
    ▶filter: ƒ filter()
    ▶find: ƒ find()
    ▶findIndex: ƒ findIndex()
    ▶flat: ƒ flat()
    ▶flatMap: ƒ flatMap()
    ▶forEach: ƒ forEach()
    ▶includes: ƒ includes()
    ▶indexOf: ƒ indexOf()
    ▶join: ƒ join()
    ▶keys: ƒ keys()
    ▶lastIndexOf: ƒ lastIndexOf()
    length: 0
    ▶map: ƒ map()
    ▶pop: ƒ pop()
```

http://www.nathankrasney.com/

דוגמא לשימוש ב prototype לשיתוף פונקציה

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

Person.prototype.write = function() {
      console.log(`name : ${this.name} , age : ${this.age}`);
   }

let person1 = new Person("Nathan" , 33);
let person2 = new Person("Nitzan" , 29);
person1.write();
person2.write();
```

Benefit : less memory consumption , less code to write

http://www.nathankrasney.com/

Create object using Object.create

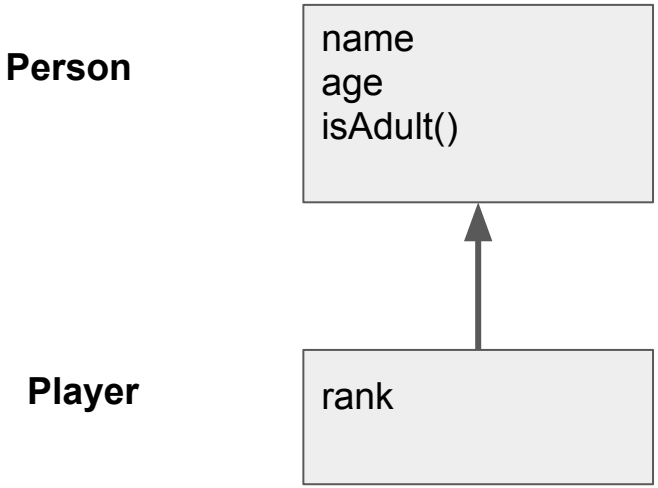
```
var personProto = {
    name : 'some person' , age : 22 , write : function(){
      console.log(`name : ${this.name} , age : ${this.age}`)
     }
    }

var jim = Object.create(personProto);
jim.name = 'jim';
jim.age = 33;
jim.write();
```

פחות בשימוש מ
function constructor

http://www.nathankrasney.com/

דוגמא קלאסית לירושה - אבל מימוש מלא חסר



http://www.nathankrasney.com/

מימוש הדוגמא הקודמת

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

Person.prototype.isAdult = function(){
   return age >= 18;
}

function Player(name ,age,rank){
    this.rank = rank;
   Person.call(this,name,age);
`  // TBD , how to inherit isAdult ?????????
}
```

קל לממש את הירושה זאת
באמצעות class ונראה זאת בהמשך
ב react

http://www.nathankrasney.com/

isPrototypeOf

Checks if an object exist in another object :

```
const Animal = {
    isAnimal: true
  };
const Lion = Object.create(Animal);
Lion.isLion = true;

console.log(Animal.isPrototypeOf(Lion)); //true
console.log(Lion.isPrototypeOf(Animal)); //false
console.log(Lion);
console.log(Animal);
```

```
19:18:11.158 true
19:18:11.158 false
19:18:11.158 ▼{isLion: true}
               isLion: true
             ▼__proto__:
                isAnimal: true
               ▶__proto__: Object
19:18:11.159 ▼{isAnimal: true}
               isAnimal: true
             ▶__proto__: Object
>
```

http://www.nathankrasney.com/

instanceof

The **instanceof operator** tests whether the prototype property of a constructor appears anywhere in the prototype chain of an object.

```
function Person(name,age){
        this.name=name;
        this.age = age;
    }
const p = new Person('Nathan',44);
let n=1;
console.log(p instanceof Person); // true
console.log(n instanceof Person); // false
```