JavaScript
Обекти

Въведение

Обект: структура от данни, която обединява в едно цяло данните и действията (функциите), които могат да се извършват върху тях.

Данните в един обект се наричат *свойства* (properties), а функциите – *методи* (methods).

Object = properties + methods

Създаване (дефиниране) на обект

- Създаване на единичен обект:

  Два идентични като функционалност подхода – предпочита се втория вариант:

  1. Object notation (почти не се използва на практика):

```
var obj = new Object();
```

  2. Curly Brace notation (object literal notation):

```
var obj = {};
```

- Чрез функция (която играе ролята на конструктор):

```
var obj = new Constructor(){};
```

// Внимавайте да не объркате с първия вариант даден по-горе!

NB: Обектите са и променливи!

Дефиниране на обект със свойства

```
var patient = {

       first_name     : "Maria",

       age            : 26,

       past_illnesses : ['variola', 'tonsillitis', 'bronchitis'],

       idiosyncrasies : 'red',   // no interpolation

       'new'          : 'bag'    // note the quotes and the missing comma !

};

console.log('patient:', patient);
```

Двойките *свойство : стойност* се разделят със запетая, като не се препоръчва да се слага запетая след последния елемент!

Достъп до свойствата на обект

```
// Създаваме нов, празен обект:
var patient = {};
```

1. Property access by dot notation:

```
   // дефинираме свойството first_name;
   patient.firstName = 'Maria';
```

2. Property access by square brackets notation:

```
   // дефинираме свойството age:
   patient["age"] = 26;
```

Разликата?

Чрез квадратните скоби можем да използваме променливи за да зададем свойство:

```
var idiosyncrasies = 'hair';

patient[idiosyncrasies] = 'red';
```

което е еквивалентно на:

`patient.hair = 'red';` или `patianet[„hair“] = 'red';`

Но не можем да направим същото чрез:

```
patient.idiosyncrasies = 'red'; //ReferenceError:
idiosyncrasies is not defined
```

Дефиниране на методи

Методите са функции (анонимни), които се съхраняват в обекта като негови свойства!

```
var person = {
    // свойства:
    first_name: "Maria",
    last_name:  "Petrova",
    // методи:
    full_name: function() {
      return this.first_name + ' ' + this.last_name;
    }
};

// извикване на метод
console.log( "Person's full name is:", person.full_name() );
```

NB: **This** еднозначно обозначава (реферира) обекта person!

Итерация върху свойствата на обект

```
for (var prop in obj) {
      console.log( prop, "=", obj[prop] );
};
```

Упражнение: дефинирайте обект 'Person' и изпробвайте дадения вариант върху него.

Конструктор (на обекти)

Какво е конструктор на обект в JS?

Функция, чрез която създаваме обект.

Използва се, когато искаме да създадем множество еднотипни обекти

(*аналогия*: обектът е курабийка, а конструкторът - форма за курабийки).

Дефиницията по нищо не се различава от дефиницията на стандартна функция.

Разликата е в извикването: конструкторът винаги се извиква с '**new**' пред името на функцията.

Конструктор (на обекти)

Пример (http://codepen.io/webdesigncourse/pen/mVWNzN?editors=001):

```
function alien( name, fingers ){
  this.name = name;
  this.fingers = fingers;
}

var alf = new alien( 'alf', 8 );
var alien2 = new alien( 'zx42', 33 );
// и т.н.
```

Функция създаваща обекти (Factory Pattern)

Пример (https://codepen.io/webdesigncourse/pen/MoxXJz):

```
function aliansFactory(name, fingers){
    var obj = {};

    obj.name = name || 'anonymous';
    obj.fingers = fingers || '5';

    return obj;
}

var alf = aliansFactory( 'alf', 8 );
var alien2 = aliansFactory( 'zx42', 33 );


console.dir(alf);
console.dir(alien2);
```