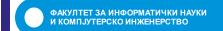
Auditory exercises 7

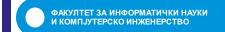
Internet programming

Ivan Kitanovski Bojan Ilijoski



Objects in JavaScript

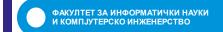
Attributes	Methods
car.name = Fiat	car.start()
car.model = 500	car.drive()
car.weight = 850kg	car.brake()
car.color = white	car.stop()
	car.name = Fiat car.model = 500 car.weight = 850kg



Objects in JavaScript (2)

■ The objects are variables that store more values

```
var car = {type:"Fiat", model:"500",
color:"white"};
```



Objects in JavaScript (3)

Access to the attributes of the object

Access to the methods of the object

```
objectName.methodName()
```

 If the method is accessed without (), then the definition of the function returns

```
var f = objectName.methodName;
```



Objects in JavaScript (4)

Constructor function

```
function Person(first, last, age, eyecolor) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eyecolor;
    this.name = function() {
        return this.firstName + " " + this.lastName;};
}
```

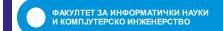
Adding a new attribute/method

```
person.nationality = "English";
Person.getAge = function() {return this.age;};
```



Example 1

```
var myObj = new Object(),
    str = 'myString',
    rand = Math.random(),
   obj = new Object();
                       = 'Dot syntax';
myObj.type
myObj['date created'] = 'String with space';
myObj[str]
            = 'String value';
myObj[rand]
                       = 'Random Number';
myObj[obj]
                       = 'Object';
myObj['']
                       = 'Even an empty string';
console.log(myObj);
//{type: "Dot syntax", date created: "String with space", myString: "String
value", 0.4700474686907987: "Random Number", [object Object]: "Object", ...}
```

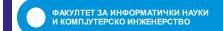


Example 2

```
// Animal properties and method encapsulation
var Animal = {
  type: 'Invertebrates', // Default value of properties
  displayType: function() { // Method which will display type of Animal
    console.log(this.type);
};
// Create new animal type called animal1
var animal1 = Object.create(Animal);
animal1.displayType(); // Output:Invertebrates
// Create new animal type called Fishes
var fish = Object.create(Animal);
fish.type = 'Fishes';
fish.displayType(); // Output:Fishes
```

Getters and Setters

```
var o = {
  a: 7,
 get b() {
    return this.a + 1;
  },
  set c(x) {
    this.a = x / 2;
console.log(o.a); // 7
console.log(o.b); // 8
0.c = 50;
console.log(o.a); // 25
```

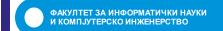


Deleting properties

```
// Creates a new object, myobj, with two
properties, a and b.
var myobj = new Object;
myobj.a = 5;
myobj.b = 12;
// Removes the a property, leaving myobj with
only the b property.
delete myobj.a;
console.log ('a' in myobj); // yields "false"
```

Comparing Objects

```
// Two variables, two distinct objects with the same properties
var fruit = {name: 'apple'};
var fruitbear = {name: 'apple'};
fruit == fruitbear; // return false
fruit === fruitbear; // return false
// Two variables, a single object
var fruit = {name: 'apple'};
var fruitbear = fruit; // assign fruit object reference to fruitbear
// here fruit and fruitbear are pointing to same object
fruit == fruitbear; // return true
fruit === fruitbear; // return true
fruit.name = 'grape';
console.log(fruitbear); // yields { name: "grape" } instead of { name: "apple"
```

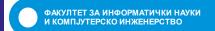


Reminder

```
var num = 0;
var obj = new String('0');
var str = '0';
console.log(num == num); // true
console.log(obj == obj); // true
console.log(str == str); // true
console.log(num == obj); // true
console.log(num == str); // true
console.log(obj == str); // true
console.log(null == undefined); // true
// both false, except in rare cases
console.log(obj == null);
console.log(obj == undefined);
```

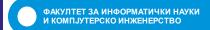
Creating objects - Object Literals

```
// This is an empty object initialized using the object literal
notation
var myBooks = {};
// This is an object with 4 items, again using object literal
var mango = {
color: "yellow",
shape: "round",
sweetness: 8,
howSweetAmI: function () {
console.log("Hmm Hmm Good");
```



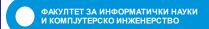
Creating objects - Object Constructor

```
var mango = new Object ();
mango.color = "yellow";
mango.shape= "round";
mango.sweetness = 8;
mango.howSweetAmI = function () {
console.log("Hmm Hmm Good");
```



Creating objects - Constructor Pattern

function Fruit (theColor, theSweetness, theFruitName, theNativeToLand) { this.color = theColor; this.sweetness = theSweetness; this.fruitName = theFruitName; this.nativeToLand = theNativeToLand; this.showName = function () { console.log("This is a " + this.fruitName); } this.nativeTo = function () { this.nativeToLand.forEach(function (eachCountry) { console.log("Grown in:" + eachCountry); });



Creating objects - Prototype Pattern

```
function Fruit () {
Fruit.prototype.color = "Yellow";
Fruit.prototype.sweetness = 7;
Fruit.prototype.fruitName = "Generic Fruit";
Fruit.prototype.nativeToLand = "USA";
Fruit.prototype.showName = function () {
console.log("This is a " + this.fruitName);
Fruit.prototype.nativeTo = function () {
            console.log("Grown in:" + this.nativeToLand);
```



Printing arrays

```
// Create a new school object with 3 own properties: schoolName,
schoolAccredited, and schoolLocation.
var school = {schoolName:"MIT", schoolAccredited: true,
schoolLocation: "Massachusetts"};
//Use of the for/in loop to access the properties in the school
object
for (var eachItem in school) {
console.log(eachItem); // Prints schoolName, schoolAccredited,
schoollocation
```

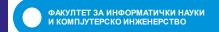


Printing properties

```
function HigherLearning () {
this.educationLevel = "University";
   Implement inheritance with the HigherLearning constructor
var school = new HigherLearning ();
school.schoolName = "MIT";
school.schoolAccredited = true;
school.schoolLocation = "Massachusetts";
//Use of the for/in loop to access the properties in the school object
for (var eachItem in school) {
console.log(eachItem); // Prints educationLevel, schoolName,
schoolAccredited, and schoolLocation
```

Computed properties

```
let fruit = prompt("Which fruit to buy?", "apple");
let bag = {
  [fruit]: 5, // the name of the property is taken from
the variable fruit
};
alert( bag.apple ); // 5 if fruit="apple"
//same with
let bag = {};
// take property name from the fruit variable
bag[fruit] = 5;
```

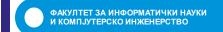


 Write a JavaScript program that will display all attributes of a given object.

```
function _keys(obj) {
    if (!isObject(obj)) return [];
    if (Object.keys) return Object.keys(obj);
    var keys = [];
    for (var key in obj) keys.push(key);
    return keys;
}

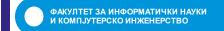
function isObject(obj) {
    var type = typeof obj;
    return type === 'function' || type === 'object' && !!obj;
}

console.log(_keys({red: "#FF0000", green: "#00FF00", white: "#FFFFFFF"}));
```



Write a JavaScript program that will delete the attribute rollno from the following object.

```
var student = {
    name : "David Rayy",
    sclass : "VI",
    rollno : 12
};
```



```
var student = {
  name : "David Rayy",
  sclass : "VI",
  rollno : 12
};

console.log(student);

delete student.rollno;

console.log(student);
```

 Write JavaScript program that in console will print the data for the following objects (book, author and status).

```
var library = [
       author: 'Bill Gates',
       title: 'The Road Ahead',
       readingStatus: true
   },
       author: 'Steve Jobs',
       title: 'Walter Isaacson',
       readingStatus: true
   },
       author: 'Suzanne Collins',
       title: 'Mockingjay: The Final Book of The Hunger Games',
       readingStatus: false
   }];
```

```
var library = [
        title: 'Bill Gates',
        author: 'The Road Ahead',
        readingStatus: true
    },
        title: 'Steve Jobs',
        author: 'Walter Isaacson',
        readingStatus: true
    },
        title: 'Mockingjay: The Final Book of The Hunger Games',
        author: 'Suzanne Collins',
        readingStatus: false
}];
for (var i = 0; i < library.length; i++) {</pre>
    var book = "'" + library[i].title + "'" + ' by ' + library[i].author + ".";
    if (library[i].readingStatus) {
      console.log("Already read " + book);
    } else {
     console.log("You still need to read " + book);
```

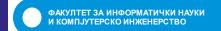


Write a JavaScript function that will transform an object into a list of '[key,value]' pairs.

```
Input:
{red: "#FF0000", green: "#00FF00", white: "#FFFFFF"}

Output:
[["red", "#FF0000"], ["green", "#00FF00"], ["white", "#FFFFFF"]]
```

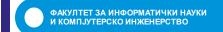
```
function key_value_pairs(obj) {
    var keys = keys(obj);
    var length = keys.length;
    var pairs = Array(length);
    for (var i = 0; i < length; i++)</pre>
      pairs[i] = [keys[i], obj[keys[i]]];
    return pairs;
function keys(obj) {
    if (!isObject(obj)) return [];
    if (Object.keys) return Object.keys(obj);
    var keys = [];
    for (var key in obj) keys.push(key);
    return keys;
function isObject(obj) {
    var type = typeof obj;
    return type === 'function' || type === 'object' && !!obj;
console.log(key_value_pairs({red: "#FF0000", green: "#00FF00", white: "#FFFFFF"}));
```



 Write a JavaScript program that will calculate the perimeter and area of a circle.

 Note: Create two methods for perimeter and area. The radius will be given as input for the user.

```
function circle(radius) {
    this.radius = radius;
    this.area = function () {
        return Math.PI * this.radius * this.radius;
    };
    this.perimeter = function () {
        return 2*Math.PI*this.radius;
    };
var c = new circle(3);
console.log('Area =', c.area().toFixed(2));
console.log('perimeter =', c.perimeter().toFixed(2));
```



Write a JavaScript program that will show the clock work.

Output:

```
"14:37:42"
```

"14:37:43"

"14:37:44"

"14:37:45"

"14:37:46"

"14:37:47"

```
function my Clock() {
  this.cur date = new Date();
  this.hours = this.cur_date.getHours();
  this.minutes = this.cur date.getMinutes();
  this.seconds = this.cur date.getSeconds();
my Clock.prototype.run = function () {
  setInterval(this.update.bind(this), 1000);
};
my Clock.prototype.update = function () {
  this.updateTime(1);
  console.log(this.hours + ":" + this.minutes + ":" + this.seconds);
};
my Clock.prototype.updateTime = function (secs) {
 this.seconds+= secs;
  if (this.seconds >= 60) {
     this.minutes++;
     this.seconds= 0;
  if (this.minutes >= 60) {
     this.hours++;
     this.minutes=0;
  if (this.hours >= 24) {
     this.hours = 0;
};
var clock = new my_Clock();
clock.run();
```