JavaScript Basics

The basics are the same as in C, C++, Java, C#

It is case sensitive.
Semicolon ; marks the end of a statement. (Note: A newline does it too.)
Equal sign = assigns a new value to the variable: a = b + c
Double equal sign == tests the equality, operator != test the inequality (used mainly in if statement)
Product *, division /, remainder %, AND &&, OR ||
Single-line comment starts with //, multiple-line comment is /* between this */
Functions are called by using parentheses, i.e. name(argument);

Where to put your JavaScript

You have to put JavaScript to a HTML file, i.e. to a web page. But it's very simple.

```
<html>
<body>
<script>
  document.write("Hello World!");
</script>
</body>
</html>
```

Variables

A variable is a named memory cell. It's either global, or inside a function, or inside an object.
Datatypes are automatic, all numbers are double (i.e. 8-byte FPU).
A new variable is declared using var keyword like this: **var a = 20**
Beware! If you write just a = 20, it will usually work as well. But it sometimes it behaves unexpectedly! (The reason is that JavaScript is a dynamic language. So please always use var keyword to declare a new variable.)

Text and Writing

You can write string between "quotation marks" or 'apostrophes' (as in HTML).
document.write("html or text") sends the text to the output. It can be HTML code. In the fact it always is HTML code, so you need to write **<br>** to end the line, **&lt;** to write <, **&gt;** to write >.
Strings can be appended by **+** operator, e.g. "Hello " + "World"

Repetition of code

JavaScript supports all usual repetition commands and a few more. Let’s show two basic commands **while** and **for**, this examples shows the numbers 0 to 9:

```
var a = 0;
while(a <= 9) {
  document.write(a);
  a++; //increment a by one
}

for(var b = 0; b <= 9; b++) {
  document.write(b);
}
```

Conditional execution

Conditional statement **if** is written like while, the **else** branch is optional.

```
if(a > 0) document.write("a is positive");
if(b == 0) document.write("b is zero "); else document.write("b isn’t zero ");
```

Functions

Unlike most other languages, JavaScript allows so-called global code. It means that we can write commands directly into <script> tag. But we can also use functions (and we usually do).
A function is a named part of our program. Function can have parameters and/or return a value. The definition of a function is syntactically very simple (thanks to the fact JavaScript is a dynamic language).

```
function jméno() {
  //the commands are put here
  //each particular command ends with a semicolon or by a newline
}
```

A function usually has some input parameters. And it can return result by using **return** command. (Unlike other languages, we don’t need to write void anywhere in java when not returning a value from function.)

```
function SumOfThree (a,b,c) {
  return a+b+c;
}
```

When we define a function like this, we have to call it to execute this part of program.

```
document.write(SumofThree(2,3,5));
```

Exercises – simple algorithms

1. Print numbers 1 to 10 on a single line. Add a space behind each number to enhance readability.

2. Print even numbers 2 to 20, each on its own line.

3. Print a triangle made from stars. Each line has one more star than the previous line.
```
*
**
***
****
*****
```

4. Print a shrinking triangle like this:
```
*****
****
***
**
*
```

5. Modify each of the two preceding triangle programs to be a function with parametrized triangle size (to let us print bigger or smaller triangles).

Arrays

An array is a series of memory cells, which can be accessed by their index (ranking in the array).
New array is created this way: **var p = new Array(size);**
Particular array cells are indexed from 0 to size–1.
Particular array cells are accessed using square brackets: p[0], p[1], etc.
Array can be passed to a function as a whole. It is passed by reference, i.e. the called function gains direct access to the original array, not just its own local copy!!
Array can be cloned by slice method: **var thecopy = theoriginal.slice(0);**

Structured data

Thanks to being a dynamic language, JavaScript allows a very easy composition of data into structures.

person = { name: "James", surname: "Smith", born: 1990 }

We just created a variable called person, consisting of three data fields. We use dot **.** to access inner data fields (similarly to other programming languages), i.e. person.name, person.surname, person.born.

Objects

When we want to add also some functionality to our structured data, we use objects. An object is de facto a data structure supplemented by functions/methods, which work with these data. There are no classes in JavaScript, just objects. So we directly create one object and use it as a template/prototype, other objects of the same type are created by cloning the prototype. (This class-less object creation is called **prototyping**.)

The following example shows the object syntax. We create an object named Queue. It contains two variables named first and last, they are references to the start and the end of the queue. They are initialized to **null**, that means the reference variable is empty, i.e. points to nothing. Furthermore, the object contains three methods Put, Get, IsEmpty. (As you can see in this example, JavaScript is a functional language, so objects are in the fact functions too. Methods in these objects are in fact just variables pointing to functions.)

```
function Queue() {

 this.first = null;
 this.last = null;

 this.Put = function(value) {
  //adds a new item to the queue
 }

 this.Get = function() {
  //removes an item from the queue
}

 this.IsEmpty = function() {
  //returns true is queue is empty
  //note that triple-equal === operator is intended here, because we compare with null
  return this.first===null
 }
}
```

Now this Queue function-object is a prototype for creation of other queue objects using **new**.

```
var f = new Queue()
f.Put(10)
f.Put(20)
f.Put("nazdar")

var g = new Queue()
g.Add(1)
g.Add(2)
```

Exercises – arrays and objects

1. Write a function to print out the contents of an array passed as a parameter to this function. Then create and array with numbers 1 to 10 and print it out with the newly created function.

2. Complete the above Queue skeleton to the fully functional queue.

3. Analogously to the queue, write the Stack class.

4. Write Sort function to sort the number array. (JavaScript does not distinguish data types, so the function will be usable for strings as well.)