JAVASCRIPT

Part 2

Functions

Quite often we need to perform a similar action in many places of the script.

For example, we need to show a nice-looking message when a visitor logs in, logs out and maybe somewhere else.

Functions are the main “building blocks” of the program. They allow the code to be called many times without repetition.

Advantage

Code reusability

Less coding

Function Example

```
<script>
function msg(){
alert("hello! this is message");
}
</script>
```

The `function` keyword goes first, then goes the *name of the function*, then a list of *parameters* between the parentheses (comma-separated, empty in the example above) and finally the code of the function, also named “the function body”, between curly braces.

```
function name(para)
{
...body...
 }
```

Our new function can be called by its name

```
<script>
function msg(){
alert("hello! this is message");
}
msg();
msg();
</script>
```

Functions are actions. So their name is usually a verb. It should be brief, as accurate as possible and describe what the function does,

It is a widespread practice to start a function with a verbal prefix which vaguely describes the action.

Function Arguments

```
function getcube(number){
alert(number*number*number);
}
getcube(5);
```

Default values

If a parameter is not provided, then its value becomes `undefined`.

```
function showMessage(from, text = "no text given") {
 alert( from + ": " + text );
 }
 showMessage("Ann"); // Ann: no text given
```

Function with Return Value

We can call function that returns a value and use it in our program.

```
function getInfo(){
return "hello john! How r u?";
}
document.write(getInfo());
```

Local variables

A variable declared inside a function is only visible inside that function.

```
function showMessage() {
let message = "Hello, I'm JavaScript!"; // local variable
alert( message );
 }
showMessage();
alert( message );
```

Outer Variables

A function can access an outer variable as well, for example

```
let userName = 'John’;
 function showMessage() {
let message = 'Hello, ' + userName;
alert(message);
 }
showMessage();
```

```
let userName = 'John'; function
showMessage() { userName = "Bob";
let message = 'Hello, ‘ +userName;
 alert(message);
 } alert( userName );
showMessage();
alert( userName );
```

Global variables
Variables declared outside of any function, such as the outer `userName` in the code above, are called *global*.
Global variables are visible from any function (unless shadowed by locals).
It’s a good practice to minimize the use of global variables

The “null” value

The special `null` value does not belong to any of the types described above.
It forms a separate type of its own which contains only the `null` value:

```
let age = null;
```

The “undefined” value

The special value undefined also stands apart. It makes a type of its own, just like null.
The meaning of undefined is “value is not assigned”.
If a variable is declared, but not assigned, then its value is undefined:

Number

The *number* type represents both integer and floating point numbers. Besides regular numbers, there are so-called “special numeric values” which also belong to this data type: `Infinity`, `-Infinity` and `NaN`.

- `Infinity` represents the mathematical Infinity ∞. It is a special value that’s greater than any number.

```
alert( 1 / 0 ); // Infinity

alert( Infinity );

alert( "not a number" / 2 );
```

Strings

A string in JavaScript must be surrounded by quotes

```
let str = "Hello";
let str2 = 'Single quotes are ok too’;
let phrase = `can embed another ${str}`;
```

The expression
inside ${…}
is evaluated and
the result becomes a part
of the string

The syntax of creating string object using new keyword is given below:

```
var stringname=new String("hello javascript string");
document.write(stringname);
```

Methods	Description
charAt()	It provides the char value present at the specified index.
concat()	It provides a combination of two or more strings
indexOf()	It provides the position of a char value present in the given string.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
toLowerCase()	It converts the given string into lowercase letter.
split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.

charAt(index) Method

```
var str="javascript";
document.write(str.charAt(2));
```

concat(str) Method

```
var s1="javascript ";
var s2="concat example";
var s3=s1.concat(s2);
document.write(s3);
```

indexOf(str) Method

```
var s1="javascript index test";
var n=s1.indexOf(“test");
document.write(n);
```

toLowerCase() Method

```
var s1="JavaScript toLowerCase";
var s2=s1.toLowerCase();
document.write(s2);
```

String slice(beginIndex, endIndex) Method

```
var s1="abcdefgh";
var s2=s1.slice(2,5);
document.write(s2);
```

trim() Method

```
var s1="     javascript trim     ";
var s2=s1.trim();
document.write(s2);
```

split() Method

```
var str="This is Javascript example";
document.write(str.split(" "));
```

substr() Method

```
var str="JavaScript";
document.writeln(str.substr(0,4));
```