Telerik Academy

Functions

Reusable parts of Code

Telerik Software Academy

Learning & Development Team

http://academy.telerik.com

Telerik Academy

Table of Contents

Functions Overview

What is a function?

What is a Function?

- A function is a kind of building block that solves a small problem
  - A piece of code that has a name and can be called from the other code
  - Can take parameters and return a value
- Functions allow programmers to construct large programs from simple pieces

Why to Use Functions?

- More manageable programming
  - Split large problems into small pieces
  - Better organization of the program
  - Improve code readability and understandability
  - Enhance abstraction
- Avoiding repeating code
  - Improve code maintainability
- Code reusability
  - Using existing functions several times

Telerik Academy

Declaring and Creating Functions

Declaring and Creating Functions

- Each function has a name
  - It is used to call the function
  - Describes its purpose
- Functions in JavaScript does not have return type

```
function printLogo() {
    console.log('Telerik Corp.');
    console.log('www.telerik.com');
}
```

Function name

Declaring and Creating Functions (2)

- Each function has a body
  - It contains the programming code
  - Surrounded by { and }

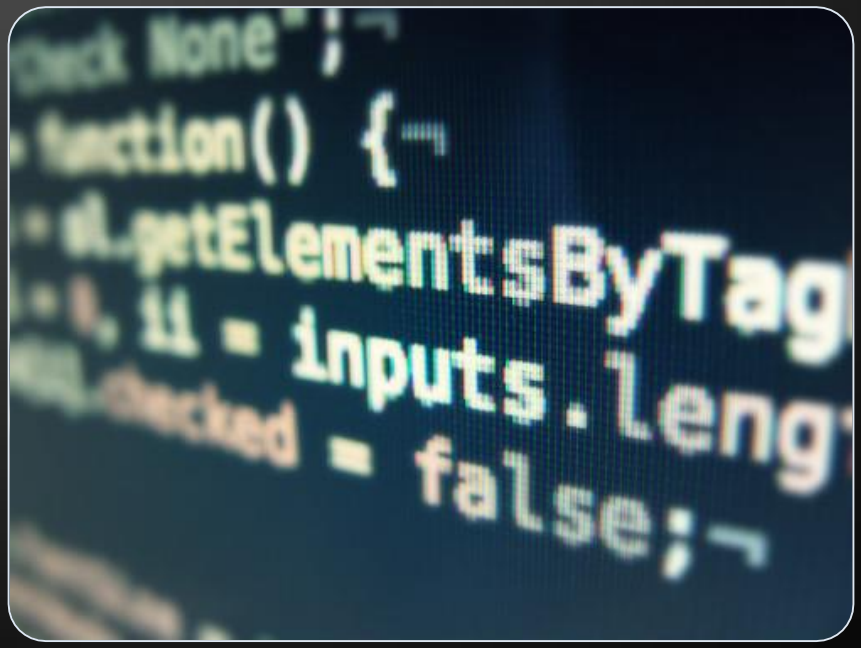
```
function printLogo() {
    console.log('Telerik Corp.');
    console.log('www.telerik.com');
}
```

Function body

Telerik Academy

Declaring and Creating Functions

Live Demo



Telerik Academy

Ways of Defining a Function

- Functions can be defined in three ways:
  - Using the constructor of the Function object

```
var print = new Function('console.log("Hello")');
```

  - By function declaration

```
function print() { console.log('Hello') };
```

  - By function expression

```
var print = function() { console.log('Hello') };
var print = function printFunc() { console.log('Hello') };
```

Telerik Academy

Calling Functions

Calling Functions

- To call a function, simply use:
  1. The function's name
  2. Parentheses
  3. A semicolon (;)

```
printLogo();
```

- This will execute the code in the function's body and will result in printing the following:

```
Telerik Corp.
www.telerik.com
```

Calling Functions (2)

- A function can be called from:
  - Any other function
  - Itself (process known as recursion)

```
function print(){
    console.log('printed');
}

function anotherPrint(){
    print();
    anotherPrint();
}
```

Don't do this at home

Telerik Academy

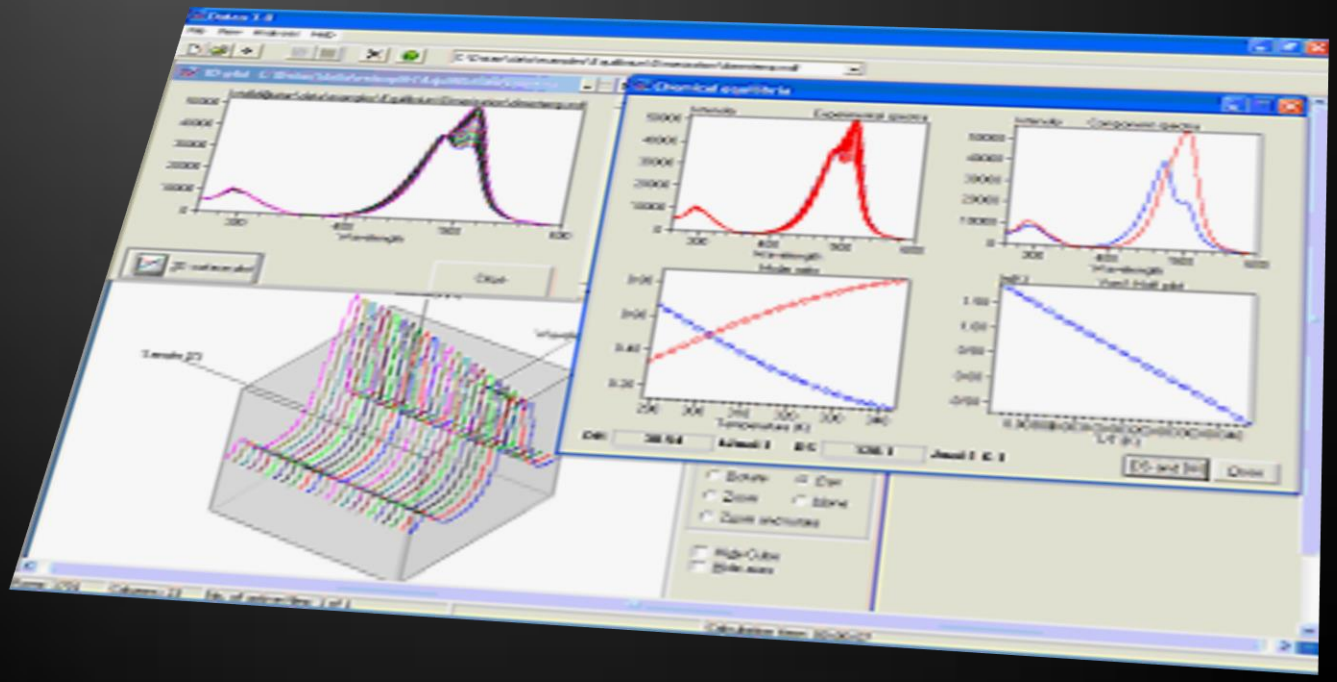
Declaring and Calling Functions

Live Demo

Telerik Academy

Functions with Parameters

Passing Parameters and Returning Values



Function Parameters

- To pass information to a function, you can use parameters (also known as arguments)
  - You can pass zero or several input values
  - Each parameter has name
  - Parameters are assigned to particular values when the function is called
- Parameters can change the function behavior depending on the passed values

Defining and Using Function Parameters

```
function printSign(number) {
    if (number > 0)
        console.log('Positive');
    else if (number < 0)
        console.log('Negative');
    else
        console.log('Zero');
}
```

- Function's behavior depends on its parameters
- Parameters can be of any type
  - Number, String, Object, Array, etc.
  - Even Function

Defining and Using Function Parameters (2)

- Functions can have as many parameters as needed:

```
function printMax(number1, number2) {
    var max = number1;
    if (number2 > number1)
        max = number2;
    console.log('Maximal number: ' + max);
}
```

Calling Functions with Parameters

- To call a function and pass values to its parameters:
  - Use the function's name, followed by a list of expressions for each parameter
- Examples:

```
printSign(-5);
printSign(balance);
printSign(2 + 3);

printMax(100, 200);
printMax(oldQuantity * 1.5, quantity * 2);
```

Functions Parameters – Example

```
function printSign(number) {
  if (number > 0)
    console.log('The number ' + number + ' is positive.');
  else if (number < 0)
    console.log('The number ' + number + ' is negative.');
  else
    console.log('The number ' + number + ' is zero.');
}

function printMax(number1, number2) {
  var max = number1;
  if (max > number1) {
    max = number2;
  }
  console.log('Maximal number: ' + max);
}
```

Telerik Academy

Function Parameters

Live Demo

Printing Triangle – Example

- Creating a program for printing triangles as shown below:

```
n=5 →
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
n=6 →
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

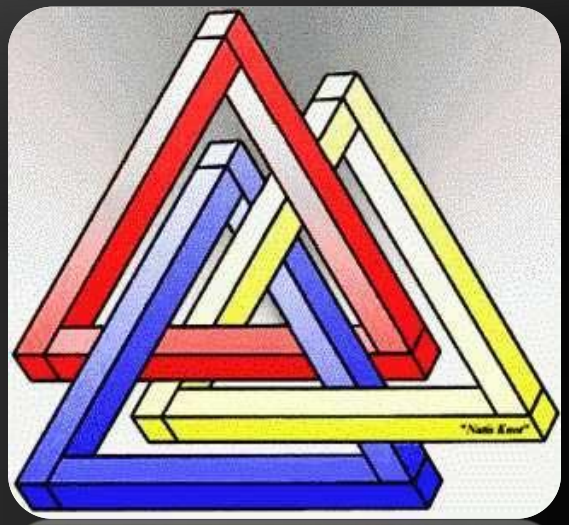
Printing Triangle – Example

```
var n = read('input-tb');

var line;
for (line = 1; line <= n; line++)
   printLine(1, line);
for (line = n-1; line >= 1; line--)
   printLine(1, line);

function printLine(start, end) {
   var line='';
   for (var i = start; i <= end; i++){
      line += ' ' + i;
   }
   console.log(line);
}
```

Telerik Academy



Printing Triangle

Live Demo

Telerik Academy

The arguments Object

arguments Object

- Every function have a special object called `arguments`
  - It holds information about the function and all the parameters passed to the function
  - No need to be explicitly declared
    - It exists in every function

```
function printArguments() {
  for(var i in arguments) {
    console.log(arguments[i]);
  }
}
printArguments(1, 2, 3, 4); //1, 2, 3, 4
```

Telerik Academy

The arguments Object

Live Demo

Telerik Academy

Returning Values From Functions

Telerik Academy

Returning Values from Functions

- Functions in JavaScript may or may not return a value
  - The return value can be of any type
    - `Number`, `String`, `Object`
  - If no value is returned, the caller gets value "`undefined`"

```
var head = arr.shift();
```

```
var price = getPrice() * quantity * 1.20;
```

```
var noValue = arr.sort();
```

Defining Functions That Return a Value

- Functions can return any type of data (`number`, `string`, `object`, etc…)
- Use `return` keyword to return a result

```
function multiply(firstNum, secondNum) {
    return firstNum * secondNum;
}
```

The return Statement

- The return statement:
  - Immediately terminates function's execution
  - Returns specified expression to the caller
  - Example:

```
return -1;
```

- To terminate function execution, use just:

```
return;
```

- Return can be used several times in a function body
  - To return a different value in different cases

Telerik Academy

Return Value

Live Demo

Sum Even Numbers– Example

- Calculate the sum of all even numbers in an array

```
function sum(numbers) {
  var sum = 0;
  for (var i in numbers) {
    if(numbers[i] % 2 == 0) {
      sum += numbers[i];
    }
  }
  return sum;
}
```

if (!numbers[i]%2) is also valid

Telerik Academy

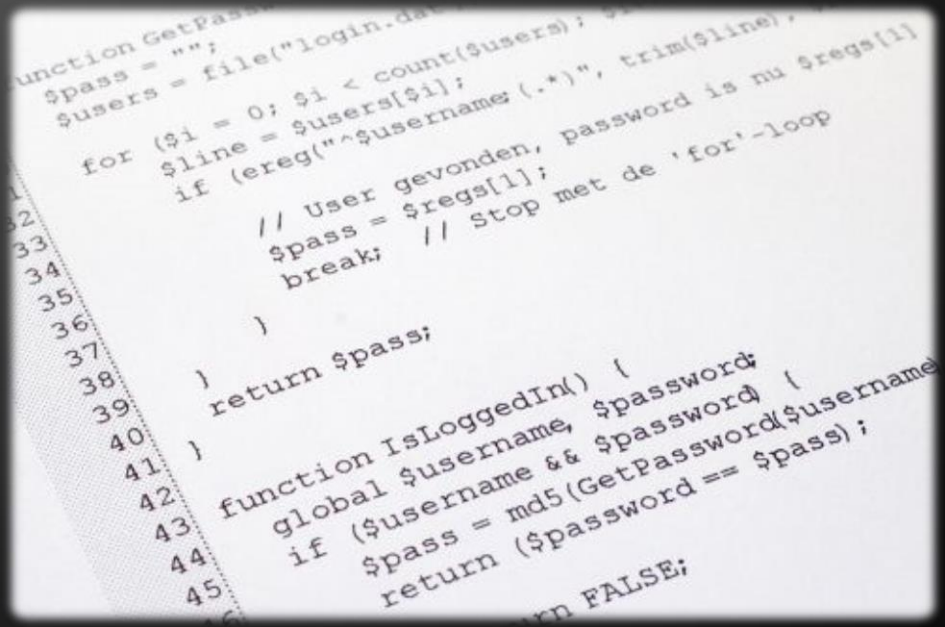
Sum of Even Numbers

Live Demo

Telerik Academy

Function Scope

Scope of variables and functions



Telerik Academy

Function Scope

- Every variable has its scope of usage
  - A scope defines where the variable is accessible
  - Generally there are local and global scope

```
<script>
var arr = [1, 2, 3, 4, 5, 6, 7];
function countOccurences (value) {
  var count = 0;
  for (var i=0; i < arr.length; i++) {
    if (arr[i] == value) {
      count++;
    }
  }
  return count;
}
</script>
```

arr is in the global scope (it is accessible from anywhere)

count is declared inside countOccurences and it can be used only inside it

Try removing the "var" before count

Telerik Academy

Function Scope

Live Demo

Telerik Academy

Function Overloading

Many functions with the same name

Telerik Academy

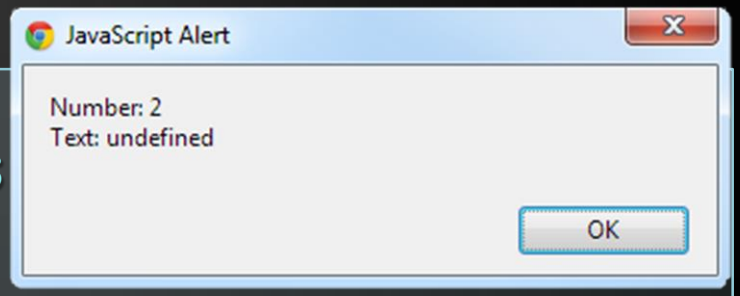
Function Overloading

- JavaScript does not support function overloading
  - i.e. only one function with a specified name can exists in the same scope

```
function print(number) {
  console.log('Number: ' + number);
}

function print(number,text) {
  console.log('Number: ' + number + '\nText: ' + text);
}

print(2);
```



  - The second print overwrites the first one

Telerik Academy

False Function Overloading

Live Demo

Telerik Academy

Function Overloading (2)

- Function overloading in JavaScript must be faked
  - i.e. make it look like overloading
- Many ways of fake function overloading exist
  - Different number of parameters
  - Different type of parameters
  - Optional parameters

Telerik Academy

Function Overloading - Different Number of Parameters

- Overloading functions with different number of parameters is done by a simple switch by the length of the arguments

```
function printText (number, text) {
  switch (arguments.length) {
    case 1 : console.log ('Number :' + number); break;
    case 2 :
      console.log ('Number :' + number);
      console.log ('Text :' + text);
      break;
  }
}
printText (5); //logs 5
printText (5, 'Lorem Ipsum'); //logs 5 and Lorem Ipsum
```

Telerik Academy

Function Overloading - Different Number of Parameters

Live Demo

Telerik Academy

Function Overloading: Different Types of Parameters

- Overloading functions with different type of the parameters is done with a switch on the type of the parameter

```
function printValue (value) {
  var log = console.log;
  switch (typeof value) {
    case 'number' : log ('Number: ' + value); break;
    case 'string' : log ('String: ' + value); break;
    case 'object' : log ('Object: ' + value); break;
    case 'boolean' : log ('Number: ' + value); break;
  }
}
printValue (5);
printValue ('Lorem Ipsum');
printValue ([1, 2, 3, 4]);
printValue (true);
```

Telerik Academy

Function Overloading - Different Number of Parameters

Live Demo

Telerik Academy

Function Overloading with Default Parameters

- In JavaScript all parameters are optional
  - i.e. functions can be invoked without them
- Yet, there is a reason behind requesting parameters
  - Maybe the function's behavior depends on it?

Telerik Academy

Function Overloading with Default Parameters (2)

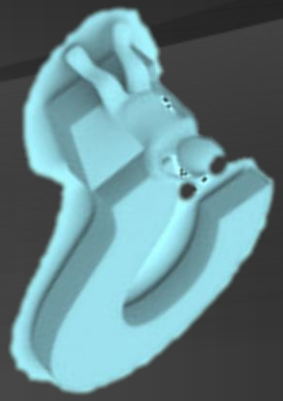
- Default parameters are checked in the function body
  - If the parameter is not present - assign a value

```
//only the str parameter is required
function substring (str, start, end){
  start = start || 0;
  end = end || str.length;
  //function code
}
```

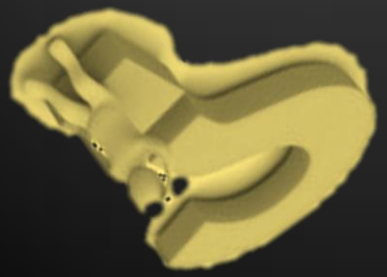
Telerik Academy

Function Overloading with Default Parameters

Live Demo



Questions?



Exercises

1. Write a function that returns the last digit of given integer as an English word. Examples: 512 → "two", 1024 → "four", 12309 → "nine"
2. Write a function that reverses the digits of given decimal number. Example: 256 → 652
3. Write a function that finds all the occurrences of word in a text
   - The search can case sensitive or case insensitive
   - Use function overloading
4. Write a function to count the number of divs on the web page

Exercises (2)

5. Write a function that counts how many times given number appears in given array. Write a test function to check if the function is working correctly.
6. Write a function that checks if the element at given position in given array of integers is bigger than its two neighbors (when such exist).
7. Write a function that returns the index of the first element in array that is bigger than its neighbors, or -1, if there's no such element.
   - Use the function from the previous exercise.