The JavaScript language

Stanisław Polak, Ph.D.

Computer Systems Group

https://www.icsr.agh.edu.pl/~polak/

Notatki

Outline

Notatki

Organizational information

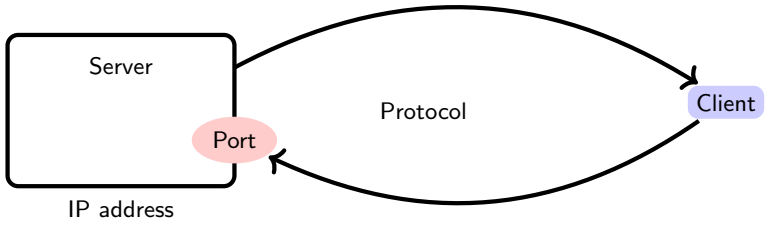
- Lectures every week — each of them is preparation for laboratory exercises
- Laboratory exercises — every week
- The final grade is calculated on the basis of the number of points:
  - A set of programming tasks to be performed in the classroom
  - A set of homework — the deadline for the solution: next classes
- Subject URL:
  `https://www.icsr.agh.edu.pl/~polak/jezyki/js/`
- Laboratory exercises URL:
  `https://polak.icsr.agh.edu.pl/`

Outline of laboratory exercises

1. CSS3 and creating responsive websites
2. JavaScript — data types, creating 2D graphics
3. DOM
4. NodeJS
5. Basics of the "Express.js" framework
6. AJAX
7. The "jQuery" programming library

Notatki

The Client-Server Model



Notatki

General rules for creating correct websites

- Formatting $\notin$ information
- The simplest means
  - HTML + CSS
  - HTML + CSS + JavaScript

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello World</title>
    <link rel="stylesheet" href="main.css"/>
  </head>
  <body>
    <main>
      <h1>Hello World</h1>
        <span class="name">SP</span> was here.
    </main>
    <footer>Stanisław Polak</footer>
  </body>
</html>
```

HTML

```css
main,  footer {
  display: block;
  background: gray;
  padding: 10px;
  margin: 1px;
}
.name {font-family: arial, verdana, sans-serif;}
```

main.css

Notatki

The first JS script

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Pierwszy skrypt JS</title>
  </head>
  <body>
    <h1>Pierwszy skrypt JS</h1>
    <script>
      document.write("Witaj w dokumencie");
      console.log("Witaj w konsoli");
    </script>
  </body>
</html>
```

HTML document with the content of JS script

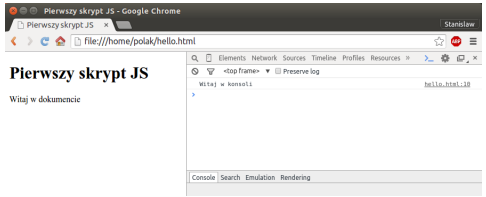


Figure: The result of the JS script execution after rendering the web page

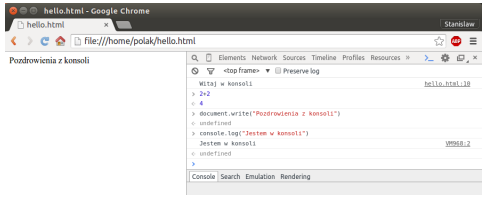


Figure: Using the JS console

Notatki

Embedding JS code

```
<!-- <script type="text/javascript;version=x.x"> -->
<script type="text/javascript">
<!--
  Treść skryptu JavaScript
  The content of the JavaScript script
-->
</script>
<noscript>
Twoja przeglądarka nie obsługuje JavaScript!<br>
Your browser does not support JavaScript!
</noscript>
```

JS internal script embedded in HTML

```
<script src="URL"></script>
<noscript>
Twoja przeglądarka nie obsługuje JavaScript!<br>
Your browser does not support JavaScript!
</noscript>
```

Referring to an external script

Notatki

The order in which scripts are executed

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<script>
function info(arg){
  var img = document.getElementsByTagName("img");
  var names='';
  var img_msg = '\nThe "img" element is unavailable\n'
  if(img.length != 0)
    img_msg = '\nThe "img" element is available\n';
  if(document.body){
    x=document.body.childNodes
    for (i=0;i<x.length;i++){
      if(x[i].nodeType == 1)
        names+=x[i].nodeName+", ";
    }
    console.log(arg+img_msg+"'document.body' contains elements: "+names);
  }
  else
    console.log(arg+img_msg+"'document.body' = NULL");
}
</script>
<script>info("Script1 1");</script>
<script src="a.js"></script>
</head>
<body onLoad="info('Script 4');">
<script>info("Script 2");</script>
<script src="b.js"></script>
<img src="image.jpg">
<script>info("Script 3");</script>
<script src="c.js"></script>
</body>
</html>
```

```
info('<file name>');
```

a.js, b.js, c.js

Console output:

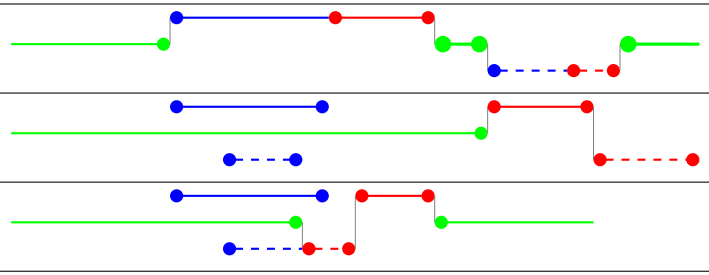
1. Script 1
   The "img" element is unavailable
   'document.body'=NULL
2. a.js
   The "img" element is unavailable
   'document.body'=NULL
3. Script 2
   The "img" element is unavailable
   'document.body' contains elements: SCRIPT,
4. b.js
   The "img" element is unavailable
   'document.body' contains elements: SCRIPT, SCRIPT,
5. <Displaying the image 'image.jpg'>
6. Script 3
   The "img" element is available
   'document.body' contains elements: SCRIPT, SCRIPT, IMG, SCRIPT,
7. c.js
   The "img" element is available
   'document.body' contains elements: SCRIPT, SCRIPT, IMG, SCRIPT, SCRIPT,
8. Script 4
   The "img" element is available
   'document.body' contains elements: SCRIPT, SCRIPT, IMG, SCRIPT, SCRIPT,

Notatki

Asynchronous or deferred execution of external scripts

Notatki

Tag	Document processing schema
`<script>`	Script 1 HTML parser Script 2
`<script defer>`	Script 1 HTML parser Script 2
`<script async>`	Script 1 HTML parser Script 2



Parsing Downloading Execution

Source: `http://peter.sh/experiments/asynchronous-and-deferred-javascript-execution-explained/`

Defining variables

Notatki

```
1  var x=42;
2  //or
3  y = 42; // Not recommended
4  var _y = 42;
5  var $if=42;
6  var róża = 42;
7  var 1a = 42; // identifier starts immediately after numeric literal
8  console.log(y); // 42
9  console.log(Y); // Y is not defined
10 if=42; //missing variable name
11 var y=42
12 console.log(typeof(y)); // number
13 y="42";
14 console.log(typeof(y)); // string
```

Defining constants

```
const PI = 3.1415926;
console.log(PI); // 3.1415926
console.log(typeof(PI)); // number
const PI = 3.14; // redeclaration of const PI
PI="3.24" // An attempt to overwrite a constant value
console.log(PI); // 3.1415926 that is, the attempt to overwrite failed
console.log(typeof(PI)); // number

var PI = 3.14 // redeclaration of const PI
var zmienna=1;
const zmienna=1; // redeclaration of var zmienna
```

Notatki

Special types of JS

Type: "null"

```
var empty=null;
console.log(typeof(empty)); // object

var empty1 = NULL; //NULL is not defined

if(empty)
  console.log("true");
else
  console.log("false");
// false

console.log(empty-1); // -1
```

Type: "undefined"

```
console.log(typeof(abc)); // undefined

var def;
console.log(typeof(def)); // undefined

function f(arg){
  console.log("arg="+arg)
}

var result = f()        // arg=undefined
console.log(result)  // undefined


if(def === undefined)
  console.log("Undefined");
else
  console.log("Defined");
// Undefined

if(def)
  console.log("true");
else
  console.log("false");
// false

console.log(def-1); // NaN
```

Notatki

Simple types and their object-related equivalents

Simple types	Object equivalent (Prototype)
boolean	Boolean
number	Number
string	String

```
1  console.log("2+2"); // "2 + 2" is of type (simple) 'string' => will write: 2 + 2
2  console.log("2+2".length); // Implicit conversion to the type (prototype) 'String' => will write: 3
3  /*
4  The above line is equivalent:
5      var objString = new String("2+2")
6      console.log(objString.length)
7  */
8  var str = "2+2";
9  console.log(str.charAt(1)); // +
10 console.log(str[1]); // +
11 console.log(str.charCodeAt(1)); // 43
12
13 var num = 1.987654
14 console.log(num.toPrecision(3)) //Implicit conversion to the type (prototype) 'Number' => will write:
       1.99
```

Notatki

Type: "boolean"

```
1 var dead=false;
2 var married=true;
3 console.log(typeof(dead)); // boolean
4 married=FALSE; //FALSE is not defined
```

Notatki

Conversion to type “boolean”

```
1 console.log(Boolean("abc")); // true
2 console.log(Boolean("")); // false
3 console.log(Boolean(10)); // true
4 console.log(Boolean(0)); // false
5 console.log(Boolean(null)); // false
6 console.log(Boolean(undefined)); // false␣
```

Notatki

Type: “number”

```
1  var price = 10.5;
2  var num1 = 2;
3  var num2 = 2.0;
4  var binary = 0b101; //0B101;
5  var octal = 0o77; //0077
6  var hexadecimal = 0xFF; //0XFF
7
8  console.log(typeof(price)); // number
9  console.log(typeof(num1)); // number
10 console.log(typeof(num2)); // number
11 console.log(num2); // 2
12 console.log(typeof(binary)); // number
13 console.log(binary); // 5
14 console.log(typeof(octal)); // number
15 console.log(octal); // 63
16 console.log(typeof(hexadecimal)); // number
17 console.log(hexadecimal); // 255
```

Notatki

Conversion to type: “number”

```
1  console.log(parseInt("3.14")); // 3
2  console.log(parseInt("3.94")); // 3
3  console.log(parseInt("3.94.1")); // 3
4  console.log(parseInt("3.94a")); // 3
5  console.log(parseInt("a3.94")); // NaN
6  console.log(parseFloat("3.14")); // 3.14
7  console.log(parseFloat("3.14.1")); // 3.14
8  console.log(parseFloat('0x10')); // 0
9  console.log(parseFloat('')); // NaN
10 console.log(parseFloat(' \r\n\t')); // NaN
11
12
13 console.log(parseInt("101",2)); // 5
14 console.log(parseInt("FF")); // NaN
15 console.log(parseInt("FF",16)); // 255
16
17 console.log(parseInt("FF - Firefox")); // NaN
18 console.log(parseInt("FF - Firefox",16)); // 255
19 console.log(parseInt("false")); // NaN
20 console.log(parseInt("false",16)); // 250 - "fa" has been changed to a number!
21
22 console.log(Number(null)); // 0
23 console.log(Number(undefined)); // NaN
24 console.log(Number(false)); // 0
25 console.log(Number(true)); // 1
26 console.log(Number("3.14")); // 3.14
27 console.log(Number("3.14.1")); // NaN
28 console.log(Number("3")); // 3
29 console.log(Number("3a")); // NaN
30 console.log(Number('0x10')); // 16
31 console.log(Number('')); // 0
32 console.log(Number(' \r\n\t'); // 0…
```

Notatki

Type: “string”

```
1  var last_name = "Polak";
2  var first_name = 'Stanisław';
3
4  console.log(typeof(last_name)); // string
5  console.log(typeof(first_name));  // string
6
7  console.log("First name=${first_name} Last name=${last_name}"); // First name=${first_name} Last name=${last_name}
8  console.log('First name=${first_name} Last=${last_name}'); // First name=${first_name} Last name=${last_name}
9
10 var a = 11 - "1" ;
11 console.log(a); // 10
12 var b = 11 + "1";
13 console.log(b); // 111
14
15 console.log(typeof(a)); // number
16 console.log(typeof(b)); // string
17
18 last_name[0]='W';
19 console.log(last_name); //"Polak" instead of "Wolak"
20
21 last_name = 'W' + last_name.substr(1);
22 console.log(last_name); // and now "Wolak"
```

Notatki

Type: “string”

Template strings

Untagged

```
var a = 2;
var str = `Variable 'a' has value ${a}, 2+2=${2+2}\n`
console.log(str);
/*********************************/
var str = `Line 1
Line 2`;
console.log(str);
```

Tagged

```
function tag(strings,val1,val2){
  result = "String 1:\t\t'"+strings[0]+"'\n";
  result += "Raw string 1:\t'"+strings.raw[0]+"'\n";
  result += "Value 1:\t\t"+val1+"\n";
  result += "String 2:\t\t'"+strings[1]+"'\n";
  result += "Raw string 2:\t'"+strings.raw[1]+"'\n";
  result += "Value 2:\t\t"+val2+"\n";
  return result;
}
/*************************************/
var a = 2;
var b = 3;
str = tag`a+b=\t${a+b}\n, a*b=${a*b}`;
console.log(str);
```

Notatki

Typ „string”

Sekwencje specjalne

- \b
- \f
- \n
- \r
- \t
- \v
- \'
- \"
- \\
- \xXX
- \uXXXX

```
console.log("a'\"\x63\'\u0105") // a'"cą
```

Example of use

Notatki

Conversion to type: “string”

```
var dead = true;
console.log(typeof(dead)); // boolean
var lancuch=dead.toString();
console.log(typeof(lancuch)); // string
console.log(lancuch); // true
var liczba = 0xFF;
console.log(0xFF.toString()); // "255"
liczba = 11;
console.log(liczba.toString()); // "11"
liczba = 11.9;
console.log(liczba.toString()); // "11.9"

var liczba=255;
console.log(liczba.toString(2)); // 11111111
console.log(liczba.toString(4)); // 3333
console.log(liczba.toString(8)); // 377
console.log(liczba.toString(16)); // ff

console.log(String(null)); // "null"
console.log(String(undefined)); // "undefined"
console.log(String(false)); // "false"
console.log(String(true)); // "true"
console.log(String(255)); // "255"
console.log(String(3.14)); // "3.14"
```

Notatki

Type: “symbol”

```
symbol1 = Symbol();
symbol2 = Symbol();
console.log(typeof(symbol1));   //symbol
console.log(symbol1 == symbol2) //false

symbol3 = Symbol('Symbol description');
symbol4 = Symbol('Symbol description');
console.log(symbol3); //Symbol(Symbol description)
console.log(symbol4); //Symbol(Symbol description)
console.log(symbol3 == symbol4); //false

symbol5 = Symbol.for("symbol3");
symbol6 = Symbol.for("symbol3");
console.log(symbol5); //Symbol(symbol3)
console.log(symbol6); //Symbol(symbol3)
console.log(symbol5 == symbol6); //true

var symbol7 = Symbol.for("uid");
console.log(Symbol.keyFor(symbol7)); // "uid"
var symbol8 = Symbol.for("uid");
console.log(Symbol.keyFor(symbol8)); // "uid"
var symbol9 = Symbol("uid");
console.log(Symbol.keyFor(symbol9)); // undefined
```

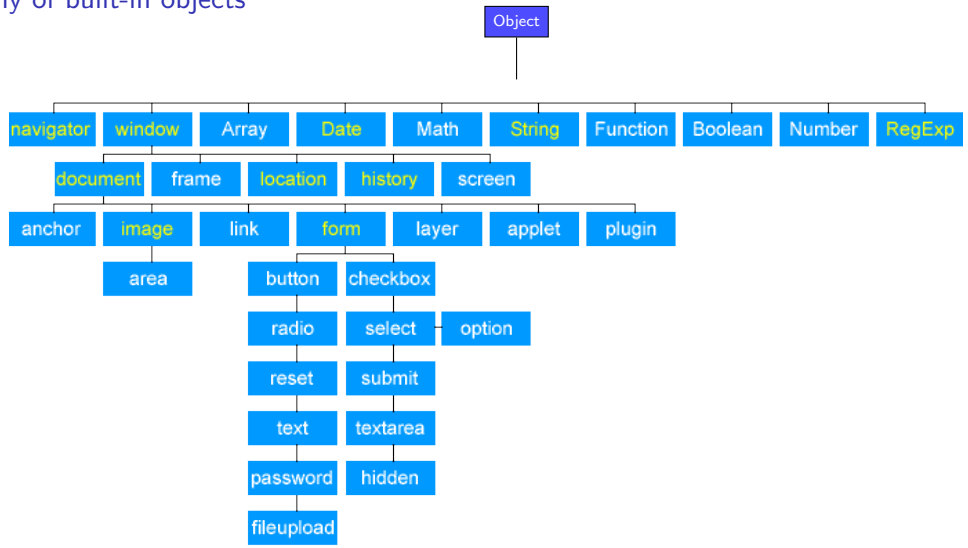
Notatki

Objects

```
1  //Create an instance of the (built-in) type 'Object'
2  var object1 = new Object();
3  var object2 = {a:1, b:10};
4  console.log(typeof(object1)); // object
5  console.log(typeof(object2)); // object
6
7  //Access to object properties
8  console.log(object1.constructor);    // function Object() { [native code] }
9  console.log(object1['constructor']); // function Object() { [native code] }
10 console.log(object2['constructor']); // function Object() { [native code] }
11 console.log(object2.a); // 1
12 console.log(object2['b']); // 10
13
14 //Use the symbol as the object's property
15 a = Symbol();
16 var object3 = {[a]:1, b:10}
17 console.log(object3[a]);   // 1
18 console.log(object3.a);    // undefined
19 console.log(object3['a']); // undefined
20 console.log(object3['b']); // 10
21 console.log(object3.b); // 10
```

Notatki

Hierarchy of built-in objects



Source: `http://www.visualtech.ca/javascript/javascript_object_model.php`

Notatki

The "Array" object

```
var tab1 = new Array(1,2,3);//equivalent to: var tab1=Array(1,2,3)
var tab2a = new Array(10);  //equivalent to: var tab2a = []; tab2a.length=10;
var tab2b = new Array("10");
var tab3 = [4,'abc',6];

console.log(tab1.length); // 3
console.log(tab2a.length); // 10
console.log(tab2b.length); // 1
console.log(tab3.length); // 3
console.log(tab1[0]); // 1
console.log(tab1.0); //missing ) after argument list
console.log(tab2a[0]); // undefined
console.log(tab2b[0]); // 10
console.log(tab3[1]); // abc
console.log(tab3[5]); // undefined

var tab4 = new Array(new Array(1,null,undefined),new Array('A','B','C'),new Array('x','y','z'));
console.log(tab4.length); // 3
console.log(tab4[0]); // 1,,
console.log(tab4[0][1]); // null
var tab1 = new Array(1,2,3);
var tab2 = [4,5,6];
console.log(tab1); // 1, 2 3
console.log(tab1.reverse()); // 3, 2, 1

var tab3 = tab1.concat(tab2);
console.log(tab3); // 1, 2, 3, 4, 5, 6
console.log(tab3.splice(1,2)); // 2, 3
```

Notatki

The "Map" object

```
var map = new Map();
emptyObject = {};
map.set("string","The value associated with the string");
map.set(1,{a:10});
map.set(emptyObject,1);

console.log(map); //Map { string: "The value associated with the string", 1: Object, Object: 1 }
console.log(map.get(1)); //Object { a: 10 }
console.log(map.get(2)); //undefined
console.log(map.get("string")); //"The value associated with the string"
console.log(map.get({})); //undefined
console.log(map.get(emptyObject)); //1
console.log(map.size); //3
map.delete("string");
console.log(map.size); //2

//Iteration of the hash
map.forEach((value,key,map) => {console.log("map["+key+"]="+value)});
/*
"map[1]=[object Object]"
"map[[object Object]]=1"
*/
//Conversion of the array into hash
var tab = [["key1","String"],["key2",5]];
map = new Map(tab);
console.log(map); //Map { key1: "String", key2: 5 }
```

Notatki

The "WeakMap" object

How the "WeakMap" objects differ from the "Map" objects

- They store weak references to the key
- Only objects can be keys
- The keys are not countable
- These objects cannot be iterated — see line 11

```
1  //Source: http://ilikekillnerds.com/2015/02/what-are-weakmaps-in-es6/-->
2  var map = new WeakMap();
3  var element1 = window;
4  var element2 = document.querySelector('body');
5
6  //We store two objects in our Weakmap
7  map.set(element1, 'window');
8  map.set(element2, 'myelement');
9
10 console.log(map.size); // undefined
11 //map.forEach((value,key,map) => {console.log("map["+key+"]="+value)});
12
13 // If we remove one of the elements, in this case element2, it will also be removed from our Weakmap
14 element2.parentNode.removeChild(element2);
15
16 // Delete the local reference
17 element2 = null;
18
19 console.log(map.get(element2)); // undefined
```

Notatki

The "Set" and "WeakSet" objects

Set

```
1  var set = new Set();
2  emptyObject = {};
3  set.add("string");
4  set.add("string");
5  set.add({a:10});
6  console.log(set .size); //2
7  set.forEach((value,key,set ) => {console.log("set ["+key+"]="+
       value)});
8  /*
9
10 "set[napis]=napis"
11 "set[[object Object]]=[object Object]"
12 */
13 set.delete("string")
14 console.log(set.size); //1
15 set.forEach((value,klucz,set ) => {console.log("set["+key+"]="
       +value)});
16 /*
17
18 "set [[object Object]]=[object Object]"
19 */
20 /*********************************/
21 set  = new WeakSet();
22 obj1 = {};
23 obj2 = obj;
24 set.add(obj1);
25 set.add(obj2);
26 console.log(set.has(obj1));
27 console.log(set.has(obj2));
```

WeakSet

```
1  var set  = new WeakSet();
2  var element = document.querySelector('body');
3  set.add(element);
4  console.log(set.has(element));// true
5
6  // If we remove the element 'element', it will also be removed
       from our Weakset
7  element.parentNode.removeChild(element);
8
9  //Delete the local reference
10 element = null;
11
12 console.log(set.has(element)); // false
```

Notatki

The "RegExp" object

```
1  function check(number) {
2    var re = /\d{7}/g; //'g' - return all matching fragments, not just the first one
3    //lub
4    var re=new RegExp("\\d{7}","g");
5
6    if(re.test(number))
7      console.log("The correct phone number");
8    else
9      console.log("The telephone number should consist of seven digits");
10 }
11
12 var number1="1234567";
13 var number2="12-34";
14
15 check(number1); // The correct phone number
16 check(number2); // The telephone number should consist of seven digits
```

Validation of phone number format

Notatki

The "Function" object

```
1 var adder = new Function("a", "b", "return a + b");
2 var result = adder(1,2);
3 console.log(result); // 3
4 console.log(adder.length); // 2
5 var obj = {x: 1, y:2};
6 adder = new Function("message",  "console.log(message+' ');return this.x + this.y");
7 console.log(adder.call(obj)); // undefined 3
8 console.log(adder.call(obj,'Value=')); // Value=3
```

Notatki

The "Math" object

```
...
<canvas id="canvas" width="400" height="100">
Your browser does not support the "canvas" element
</canvas>

<script>
var canvas = document.getElementById("canvas");
if (canvas.getContext) {
  var ctx = canvas.getContext('2d');
  var ox = 0, oy = 50;
  var t_min = 0, t_max = 10*Math.PI;
  var scale = 20, step = 200, inc = t_max/step;

  ctx.beginPath();
  for (var t=t_min; t<=t_max; t+=inc){
    y = scale * Math.sin(t);
    x = (t / t_max) * canvas.width;
    ctx.lineTo(ox+x, oy-y);
  }
  ctx.stroke();
</script>
...
```

Drawing a function $y = \sin(x)$ for $x \in [0, 10\pi]$

Notatki

The "Date" object

```
function JSClock() {
   var time = new Date()
   var hour = time.getHours()
   var minute = time.getMinutes()
   var second = time.getSeconds()
   var temp = "" + ((hour > 12) ? hour - 12 : hour)
   if (hour == 0)
      temp = "12";
   temp += ((minute < 10) ? ":0" : ":") + minute
   temp += ((second < 10) ? ":0" : ":") + second
   temp += (hour >= 12) ? " P.M." : " A.M."
   return temp
}
document.write(JSClock()); // 2:21:47 P.M. ␣
```

The current time is displayed in a 12-hour format

Notatki

The "navigator" object

```
1  //We assume that 'navigator.userAgent' contains the string "Mozilla/4.0 (compatible; MSIE 7.0; Windows
       NT 5.1; .NET CLR 2.0.50727)"
2
3  if (/MSIE (\d+\.\d+);/.test(navigator.userAgent)){ //check if the browser is MSIE x.x;
4   var ieversion=new Number(RegExp.$1) // $1 contains the version number, here: 7.0
5   if (ieversion>=8)
6    document.write("You're using IE8 or above")
7   else if (ieversion>=7)
8    document.write("You're using IE7.x")
9   else if (ieversion>=6)
10   document.write("You're using IE6.x")
11  else if (ieversion>=5)
12   document.write("You're using IE5.x")
13 }
14 else
15  document.write("n/a")
```

Identification of the IE browser version

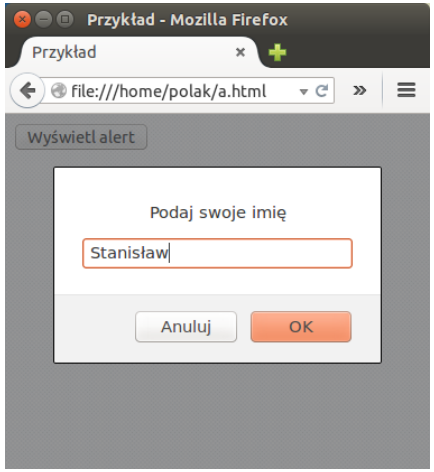
Notatki

The "window" object

The prompt() and alert() methods

```
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html;charset=UTF
       -8">
4  <title>Przykład</title>
5  <script>
6  function load_and_display_the_name(){
7    var first_name=window.prompt('Podaj swoje imię','');
8    window.alert("Witaj "+first_name); //Display welcome text
9  }
10 </script>
11 </head>
12 <body>
13 <form>
14 <button type="button" onClick="load_and_display_the_name();">Wy
       świetl alert</button><br>
15 </form>
16 </body>
17 </html>
```

Opening the input / output data window



Notatki

The "window" object

The setTimeout() and clearTimeout() methods

```
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
4  <title>An example</title>
5  <script>
6  function delayedAlert(time){
7    timeoutID = window.setTimeout(display, 2000);
8  }
9
10 function display(){
11   window.alert("Hello World!");
12   //timeoutID = window.setTimeout(display, 2000);
13 }
14
15 function stopExecution(){
16   window.clearTimeout(timeoutID);
17 }
18 </script>
19 </head>
20 <body>
21 <form>
22 <button type="button" onClick="delayedAlert();">Show alert</button><br>
23 <button type="button" onClick="stopExecution();">Cancel</button>
24 </form>
25 </body>
26 </html>
```

After 2 seconds, the alert window is displayed

Notatki

The "window" object

The setInterval() and clearInterval() methods

```
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
4  <title>An example</title>
5  <script>
6  function start_cyclic_execution() {
7    timeoutID = window.setInterval(display, 1000);
8  }
9
10 function display(){
11   console.log("Hello World!");
12 }
13
14 function stop_cyclic_execution(){
15   window.clearInterval(timeoutID);
16 }
17 </script>
18 </head>
19 <body>
20 <form>
21 <button type="button" onClick="start_cyclic_execution()">Start</button><br>
22 <button type="button" onClick="stop_cyclic_execution();">Stop</button>
23 </form>
24 </body>
25 </html>
```

Every second, a message is displayed on the console

Notatki

The "window" object

The requestAnimationFrame() and cancelAnimationFrame() methods

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
<title>An example</title>
<script>
function start_cyclic_execution() {
  requestID=window.requestAnimationFrame(display);
}

function display(){
  console.log("Hello World!");
  requestID=window.requestAnimationFrame(display);
}

function stop_cyclic_execution(){
  window.cancelAnimationFrame(requestID);
}
</script>
</head>
<body>
<form>
<button type="button" onClick="start_cyclic_execution()">Start</button><br>
<button type="button" onClick="stop_cyclic_execution()">Stop</button>
</form>
</body>
</html>
```

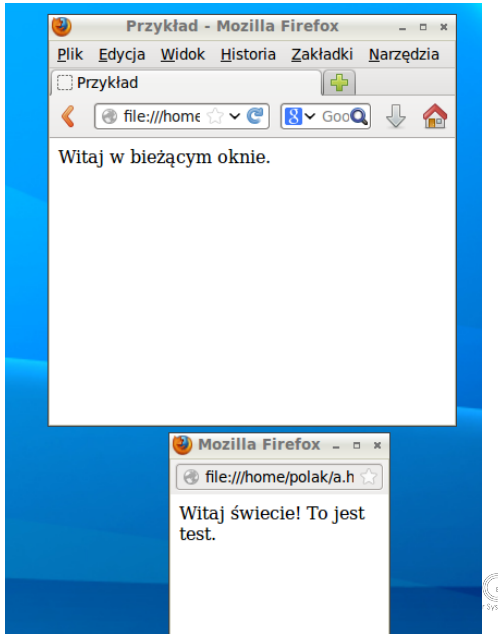
Periodically, a message is printed when the screen is refreshed

Notatki

The "document" object

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
      charset=UTF-8">
<title>Przykład</title>
</head>
<body>
Witaj
<script>
var newWindow=window.open('','','toolbar=no,
     scrollbars=no,width=200,height=150');
newWindow.document.open("text/html", "replace");
newWindow.document.writeln("Witaj świecie!");
newWindow.document.write("To jest test.");
newWindow.document.close();
document.write(" w bieżącym oknie.");

</script>
</body>
</html>
```

The text is displayed in a new window

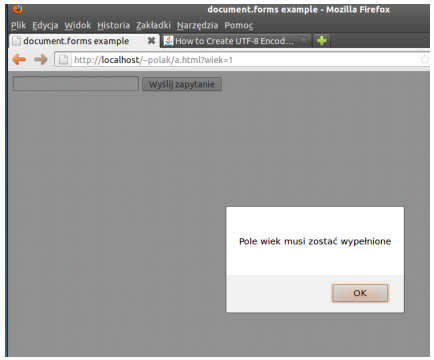


Notatki

The "form" object

```
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4  <title> document.forms example</title>
5  <script>
6  function check() {
7  var form = document.forms[0];
8  //var form = document.forms.form1;
9  //var form = document.forms['form1'];
10 var element  = form.elements[0];
11 //var element  = form.elements.wiek;
12 //var element  = form.elements['wiek'];
13 if (element.value == ""){
14   window.alert("Pole wiek musi zostać wypełnione");
15   return false;
16 }
17 else
18   return true;
19 }
20 </script>
21 </head>
22 <body>
23 <!-- <form ...  onSubmit="return false;"> -->
24 <form id="form1" action="" onSubmit="return check();">
25  <input name='wiek' type='text'>
26  <input type='submit'>
27 </form>
28 </body>
29 </html>
```

Suspending the submission of the form



Notatki

The "Image" object

```
1  <html>
2  <head>
3  <script language = "JavaScript">
4
5  function preloader(){
6    bigImage = new Image();
7    bigImage.src = "bigImage.jpg";
8  }
9  </script>
10 </head>
11 <body onLoad="javascript:preloader()">
12 <a href="#" onMouseOver="javascript:document.images[0].src='bigImage.jpg'">
13 <!--
14 lub tak:
15 <a href="#" onMouseOver="javascript:document.img01.src='bigImage.jpg'">
16 -->
17 <img name="img01" src="differentImage.jpg"></a>
18 </body>
19 </html>
```

Pre-loading a large image

Notatki

The “location” object

```
<script>
if (window.location.protocol == "http:") {
  var restUrl = window.location.href.substr(5);
  location.href = "https:" + restUrl;
}
</script>
```

Redirecting to the secure HTTPS protocol

Notatki

The “history” object

```
history.back(); // equivalent to clicking the 'Back' button
history.go(-1); // equivalent to history.back();
```

Notatki

The "screen" object

```
1 if (screen.pixelDepth < 8) {
2   // use of the "low-color" version of the page
3 } else {
4   // use of the "full-color" version of the page
5 }␣
```

Notatki

Operators, "inherited" from Java

Exactly the same as in Java

- Arithmetic (+, -, *, /, ++, --, %)
- Conditional (?:)
- Bitwise (&, |, ^, ~, <<, >>, >>>)
- Logical (&&, ||, !)
- String (+)
- Assignment (=, *=,/=, %=, +=, -=, <<=, >>=, >>>=, &=, ^=, |=)
- Type (`instanceof`)
- `new` and `this`

Almost the same as in Java

- Comparison — additional: === and !==

```
1 document.write(true == 1); // true, because 'true' is
      converted to 1 and then compared
2 document.write("1" ==  1); // true, because 1 is converted to
      "1" and then compared
3 document.write("1" === 1); // false
4 document.write("1" !== 1); // true
5 document.write(1 === 1);   // true
```

Additional comparison operators

Notatki

Comma

```
a=1 , b=2;
document.write(a); // 1
document.write(b); // 2
for (var i = 0, j = 9; i <= 9; i++, j--)
  document.writeln("a[" + i + "][" + j + "]"); /* Output:
a[0][9]
a[1][8]
a[2][7]
a[3][6]
a[4][5]
a[5][4]
a[6][3]
a[7][2]
a[8][1]
a[9][0]
*/
```

Notatki

The `in` operator

```
var trees = new Array("redwood", "cedar", "oak", "maple");
0 in trees;        // true
3 in trees;        // true
6 in trees;        // false
"maple" in trees;  // false (the index number should be given, not the value after the given index)
"length" in trees; // true ('length' is the property of the object 'Array')

//Predefined objects
"PI" in Math;          // true
var myString = new String("coral");
"length" in myString;  // true

// User objects
var car = {brand: "Honda", model: "Accord", year: 1998};
"brand" in car; //  true
"model" in car; //  true
```

Notatki

The delete operator

```
x = 42;
var y = 43;
obj = new Number();
obj.h = 4;
delete x;       // returns true (you can delete a variable if it is implicitly defined)
delete y;       // returns false (the variable can not be deleted if it is defined using 'var')
delete Math.PI; // returns false (you can not delete predefined properties)
delete obj.h;   // returns true (you can delete user-defined properties)
delete obj;     // returns true (you can delete a variable if it is implicitly defined)

var trees = new Array("redwood", "cedar", "oak", "maple");
document.write(trees.length); // 4
document.write(trees[2]);     // oak
delete trees[2];
document.write(trees.length); // 4
document.write(trees[2]);     // undefined

trees[3] = undefined;
if (2 in trees)
  document.write("The element with index 2 exists in the table");
if (3 in trees)
  document.write("The element with index 3 exists in the table");
//Output: The element with index 3 exists in the table
```

Notatki

The typeof operator

```
var myFun = new Function("5 + 2");
var shape = "round";
var size = 1;
var today = new Date();

typeof myFun;           // returns "function"
typeof(shape);          // returns "string"
typeof size;            // returns "number"
typeof today;           // returns "object"
typeof does_not_exists; // returns "undefined"
typeof true;            // returns "boolean"
typeof null;            // returns "object"
typeof 62;              // returns "number"
typeof 'Hello world';   // returns "string"
typeof Math;            // returns "object"
typeof Array;           // returns "function"
```

Notatki

The void operator

```
<A HREF="javascript:void(0)">Click here so that nothing will happen</A>
<A HREF="javascript:void(document.form.submit())">Click here for the form to be approved</A>
```

Notatki

The for ... in instruction

```
var car = {brand: "Ferrari",color:"red",
     number_of_doors:2};
for(counter in car)
    console.log(car[counter]);
```

Iteration of the object

```
var tab = new Array(1,2,3);
tab.newProperty = 123;
for(counter in tab)
    console.log("tab["+counter+"]="+tab[counter]);
```

Iteration of the array — not recommended

Notatki

The `for ... of` instruction

```
1 var tab = new Array('a',2,3);
2 tab.newProperty = 'b';
3 for(counter of tab)
4   console.log(counter);
```

```
1 var map = new Map();
2 emptyObject = {};
3 map.set("string","something");
4 map.set(1,{a:10});
5 map.set(emptyObject,1);
6 for (var [key, value] of map) {
7   console.log(key + " = " + value);
8 }
```

Notatki

The `with` instruction

```
1 var car = {brand:"Ferrari", color:"red", number_of_doors:2};
2 brand = "Fiat";
3
4 with(car){
5   console.log(brand);           // Ferrari
6   console.log(color);           // red
7   console.log(number_of_doors); // 2
8 }␣
```

Notatki

Handling exceptions

Instructions: `throw`, `try`, `catch` and `finally`

```
//Creating an object representing an exception
function Exception (message) {
  this.message = message;
  this.name    = "Negative";
}

try{
  var age = -1;
  if(age <0){
    var exception=new Exception("Age can not be a negative number");
    throw exception;
  }
  console.log("It will not be written anymore");
}
catch (e if e.name =="Negative") { console.log(e.message);}
catch (e)  {/* handling exceptions that were not previously captured */}
finally{/*instructions to be always done */}
```

Notatki

Functions

- Defining (dissimilar to Java) — using the keyword `function`; calling functions and returning values by function — similar to Java
- They let you define objects' prototype

```
function multiply(a, b=1){
  var c = a * b;
  a = 0;
  b = 0;
  return c;
}
function f() { return [1, 2, 3] }

a = 2;
b = 2;
var result = multiply(a,b);
console.log(result);//  4…
var result = multiply(a);
console.log(result);// 2

var x, y, z;
[x, y, z] = f(); // Returning many values
[x, y, z] = (function f() { return [1, 2, 3] })(); //
    Simultaneous defining and calling functions
console.log(x); // 1
/*****************************/
const constant = 1;
var variable = 2;
function constant(){} //Redeclaration of const constant
function variable(){}
variable(); //variable is not a function
```

Example functions

```
function change(x,object1,object2){
  x = 2;
  object1.brand = "Fiat";
  object2 = {brand: "Skoda"};
}

var car1 = {brand: "Ferrari"};
var car2 = {brand: "Ferrari"};
var variable = 1;
console.log(variable);     // 1
console.log(car1.brand);   // Ferrari
console.log(car2.brand);   // Ferrari
change(variable,car1,car2);
console.log(variable);     // 1
console.log(car1.brand);   // Fiat
console.log(car2.brand);   // Ferrari
```

Passing simple and complex types

Notatki

Anonymous functions

```
//Procedural function
function hello1(who) {
  return 'Hello '+who;
}
/**********************/
console.log(hello1('world'));// "Hello world"
/**********************/
/**********************/
// Function as a variable
var hello2 = function (who) {return 'Hello '+who};
// or
var hello2 = (who) => {return 'Hello '+who};
// or
var hello2 = (who) => 'Hello '+who;
/*******************************/
var hello3 = function() {
  console.log('Hello');
  console.log('World');
}
// or
var hello3 = () => {
  console.log('Hello');
  console.log('World');
}
console.log(hello2('world'));// "Hello world"
hello3(); // "Hello"
          // "World"
```

```
function Person() {
  // The Person () constructor defines 'this' as an
       instance of itself
  this.age = 0;
  this.salary = 0;

  setInterval(function () {
    //Here 'this' ⟺ object 'window' that is, it is
         different from 'this' defined in the Person
         constructor
    this.age++;
    console.log("Age="+this.age);
  }, 1000);

  setInterval(() => {
    this.salary++;//Here 'this' is a Person object
    console.log("Salary="+this.salary);
  }, 1000);
}
var person = new Person();
```

Lexical `this`

Notatki

Closures

```
function init() {
  var name = "Polak";

  function displayName() {
    console.log(name);
  }
  displayName();
}
init(); // Polak
```

```
function createFunction() {
  var name = "Polak";

  function displayName() {
    console.log(name);
  }
  return displayName;
}

var myFunction = createFunction();
myFunction(); // Polak
```

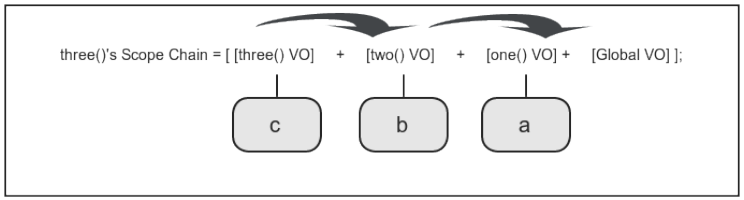
```
function multiply_by(x) {
  return function(y) {
    /* the function uses two variables:
       y - available to the user
       x - defined only inside the 'multiply_by()' function
    */
    return x * y;
  };
}

var product_5_by = multiply_by(5); //the parameter 'x' is assigned the value 5

console.log(product_5_by(12)); // will be written 5 * 12 or 60
```

Example of closure use

Notatki

Scope chain

```
1  function one(){
2    var a = 1;
3    two();
4
5    function two(){
6      var b = 2;
7      three();
8
9      function three() {
10       var c = 3;
11       console.log(a + b + c); //6
12     }
13   }
14 }
15 one();
```



Source: http://davidshariff.com/blog/javascript-scope-chain-and-closures/

Notatki

Variables in functions

```
1  function fun(x){
2    a = ++x;
3    b = 10;
4  }
5  /*************************/
6  a = 0; // <=> var a = 0;
7  console.log(a); // 0
8  console.log(b); //b is not defined
9  fun(a);
10 console.log(a); // 1
11 console.log(b); // 10
```

```
1  function fun(x){
2    var a = ++x;
3    var b = 10;
4  }
5  /*************************/
6  a = 0; // <=> var a = 0;
7  console.log(a); // 0
8  console.log(b); //b is not defined
9  fun(a);
10 console.log(a); // 0
11 console.log(b); //b is not defined
```

```
1  let a=1;
2  console.log(a); // 1
3  /*****************************/
4  for (let i = 0; i<10; i++) {
5    console.log(i);
6  // 1, 2, 3, 4 ... 9
7  }
8  console.log(i); // i is not defined
```

Expression 'let'

```
1  function fun()
2  {
3    var a = 3;
4    var b = 4;
5    if (a === 3) {
6      let a = 10; // another variable 'a'. Range - interior of the 'if
           ' block
7      var b = 11; // the same variable 'b' as above. Range - interior
           of the 'fun' function
8      console.log(a);  // 10
9      console.log(b);  // 11
10   }
11   console.log(a); // 3
12   console.log(b); // 11
13 }
14 fun();
```

'let' vs. 'var'

Notatki

Functions with variable number of arguments

```
function write() {
  // go after all arguments
  for (var i=0; i<arguments.length; i++)
    console.log(arguments[i]);
}

write("A","B","C");
```

```
function write(a, ...others) {
  console.log(a);
  console.log(Array.isArray(others));
  for(counter in others)
    console.log("others["+counter+"]="+others[
        counter]);

  console.log(arguments.length); //SyntaxError: '
      arguments' object may not be used in
      conjunction with a rest parameter
}

write("A","B","C");
```

Notatki

Document Object Model

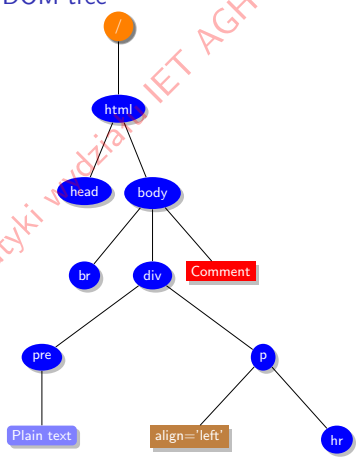
General characteristics

- DOM — Document Object Model
- Document — tree of objects
- Software interface (API) for HTML and XML documents
- A set of properties and methods for manipulating the above mentioned documents

```
<html>
  <head></head>
  <body>
    <br>
    <div>
      <pre>Plain text</pre>
      <p align='left'>
        <hr/>
      </p>
    </div>
    <!-- Comment -->
  </body>␣
</html>
```

HTML document

DOM tree



Notatki

Basic properties of nodes

The type of node	nodeType	nodeName	nodeValue
Element	Node.ELEMENT_NODE (1)	Name of the tag in uppercase letters	null
Attribute	Node.ATTRIBUTE_NODE (2)	The name of the attribute	The value of the attribute
Text	Node.TEXT_NODE (3)	#text	The text
Comment	Node.COMMENT_NODE (8)	#comment	The comment
Document	Node.DOCUMENT_NODE (9)	#document	null

Notatki

Displaying information about a single 'element' node

Methods: `getElementById()` / `querySelector()`

```
1  <html>
2    <head>
3      <title>An example</title>
4      <script>
5  function start(){
6    var element = document.getElementById("elem1");
7    # or
8    var element = document.querySelector("#elem1");
9
10   window.alert(element);            // [object HTMLBodyElement]
11
12   window.alert(element.nodeType);   // 1
13   window.alert(element.nodeName);   // BODY
14   window.alert(element.nodeValue);  // null
15 }
16     </script>
17   </head>
18   <body id="elem1" onLoad="start();">
19   </body>
20 </html>
```

Notatki

Displaying information about several 'element' nodes

Methods: `getElementsByTagName()` / `querySelectorAll()`

```
<html>
  <head>
    <title>An example</title>
    <script>
function start(){
  var elements = document.getElementsByTagName("td");
  # or
  var elements = document.querySelectorAll("tr td");
  window.alert(elements);               // [object HTMLCollection]
  window.alert(elements.length);        // 4
  window.alert(elements[0]);            // [object HTMLTableCellElement]
  window.alert(elements[0].nodeType);   // 1
  window.alert(elements[0].nodeName);   // TD
  window.alert(elements[0].nodeValue); // null
  window.alert(elements[1]);            // [object HTMLTableCellElement]
  window.alert(elements[1].nodeType);   // 1
  window.alert(elements[1].nodeName);   // TD
  window.alert(elements[1].nodeValue); // null
}
    </script>
  </head>
  <body onLoad="start();">
    <table>
      <tr><td>a</td><td>b</td></tr>
      <tr><td>c</td><td>d</td></tr>
    </table>
  </body>
</html>
```

Notatki

Retrieving descendants of 'element' and 'text' nodes

The `childNodes` property

```
<html>
  <head>
    <title>An example</title>
    <script>
function start(){
  var elements = document.getElementsByTagName("tr");
  for(var i=0 ; i<elements.length ; i++){
    var descendants = elements[i].childNodes; // the type of 'descendants' object is "NodeList"
    for (var j=0;j<descendants.length;j++){
     var string = descendants[j].nodeName + ": " + descendants[j].childNodes[0].nodeValue;
     window.alert(string);
    }
  }
}
/* Output:
TD: a
TD: b
TD: c
TD: d
*/
    </script>
  </head>
  <body onLoad="start();">
  <table border="1">
    <tr><td>a</td><td>b</td></tr>
    <tr><td>c</td><td>d</td></tr>
  </table>
  </body>
</html>
```

Notatki

Support for element attributes

The attributes property, and the setAttribute() & the removeAttribute() methods

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>An examples</title>
   <script>
function change(thickness){
  var element = document.getElementById("elem1"); // "HTMLTableElement" type object
  window.alert(element.getAttribute('border')); // 1
  window.alert(element.getAttribute('id')); // elem1
  element.setAttribute('border',thickness);

  //You can do it anyway
  var attributes = element.attributes; //obiekt typu "NamedNodeMap" type object
  window.alert(attributes.border.value); // 1
  window.alert(attributes.id.value);     // elem1
  attributes.border.value = thickness;
}

function delete(){
  var element = document.getElementById("elem1");
  element.removeAttribute('border');
}
   </script>
  </head>
  <body>
    <table border="1" id="elem1">
      <tr><td>a</td><td>b</td></tr>
      <tr><td>c</td><td>d</td></tr>
    </table>
   <form>
     <input type="button" value="Change the thickness" onClick="change(2);">
     <input type="button" value="Delete" onClick="delete();">
   </form>
  </body>
</html>
```

Notatki

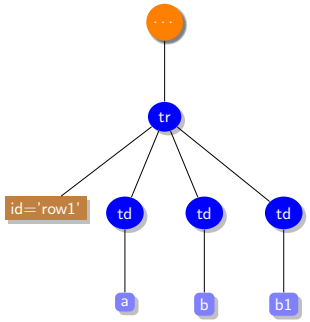
Inserting a new table cell at the end of a row

Methods: createElement() , createTextNode() and appendChild()

```
<html>
  <head>
    <title>An example</title>
    <script>
function insert(){
  var newTD = document.createElement("td");
  var newTextNode = document.createTextNode("b1");
  newTD.appendChild(newTextNode);
  var element = document.getElementById("row1");
  element.appendChild(newTD);
}
    </script>
  </head>
  <body>
    <table border="1">
      <tr id="row1"><td>a</td><td>b</td></tr>
      <tr><td>c</td><td>d</td></tr>
    </table>
   <form>
     <input type="button" value="Insert" onClick="insert();">
   </form>
  </body>
</html>
```

a	b
c	d

a	b	b1
c	d	



Notatki

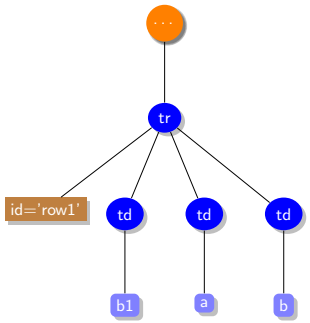
Inserting a new table cell at the beginning of a row

The `insertBefore()` method

```
1  <html>
2    <head>
3      <title>An example</title>
4      <script>
5  function insert(){
6    var newTD = document.createElement("td");
7    var newTextNode = document.createTextNode("b1");
8    newTD.appendChild(newTextNode);
9    var element = document.getElementById('row1');
10   var refTD = element.getElementsByTagName("td").item(0);
11   element.insertBefore(newTD,refTD);
12 }
13     </script>
14   </head>
15   <body>
16     <table border="1">
17       <tr id='row1'><td>a</td><td>b</td></tr>
18       <tr><td>c</td><td>d</td></tr>
19     </table>
20     <form>
21       <input type="button" value="Insert" onClick="insert();">
22     </form>
23   </body>
24 </html>
```

a	b
c	d

b1	a	b
c	d	



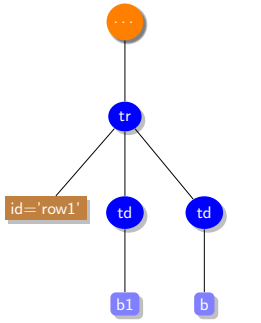
Notatki

Replacing a table cell

The `replaceChild()` method

```
1  <html>
2    <meta http-equiv="Content-Type" content="text/html; charset=
         UTF-8">
3    <head>
4      <title>An example</title>
5      <script>
6  function replace(){
7    var newTD = document.createElement("td");
8    var newTextNode = document.createTextNode("b1");
9    newTD.appendChild(newTextNode);
10   var element = document.getElementById('row1');
11   var refTD = element.getElementsByTagName("td").item(0);
12   element.replaceChild(newTD,refTD);
13 }
14     </script>
15   </head>
16   <body>
17     <table border="1">
18       <tr id='row1'><td>a</td><td>b</td></tr>
19       <tr><td>c</td><td>d</td></tr>
20     </table>
21     <form>
22       <input type="button" value="Replace" onClick="replace();
             ">
23     </form>
24   </body>
25 </html>
```

a	b
c	d



Notatki

Access to CSS styles

The style object

- CSS feature ⟼ Property of the style object
  - background-color ⟼ style.backgroundColor
  - border-top-width ⟼ style.borderTopWidth
  - Exceptions:
    - float ⟼ style.cssFloat
    - class ⟼ style.className
    - for ⟼ style.htmlFor

a	b
c	d

```
1  <html>
2    <head>
3      <meta http-equiv="Content-Type" content="text/
           html; charset=UTF-8" />
4      <title>An example</title>
5      <script>
6  function color(value){
7    var element = document.getElementById("cell1");
8    element.style.backgroundColor = value;
9  }
10     </script>
11   </head>
12   <body>
13     <table border="1">
14       <tr><td id="cell1">a</td><td>b</td></tr>
15       <tr><td>c</td><td>d</td></tr>
16     </table>
17     <form>
18       <input type="button" value="Red" onClick="
             color('#FF0000');">
19       <input type="button" value="Green" onClick="
             color('#00FF00');">
20     </form>
21   </body>
22 </html>␣
```

Notatki

Changing the background color after clicking a table cell

Event handling

```
1  <!DOCTYPE html>
2  <meta charset="UTF-8">
3  ...
4      <script>
5    function changeColor(){
6      var cell = document.getElementById("cell");
7      cell.style.backgroundColor='red';
8    }
9    /*********************************/
10   function displayAlert(){
11       alert("displayAlert()");
12   }
13   /*********************************/
14   function load() {
15     var el = document.getElementById("table");
16     el.addEventListener("click", displayAlert, false); //First, 'changeColor ()' will be executed,
           followed by 'displayAlert ()'. If the third parameter is 'true' then 'displayAlert ()' will be
            executed first, then 'changeColor ()'
17     var el = document.getElementById("cell");
18     el.addEventListener("click", changeColor, false);
19   }
20     </script>
21     ...
22   <body onload="load();">
23     <table id="table" border="1">
24       <tr><td id="cell">One</td></tr>
25       <tr><td>Two</td></tr>
26     </table>
27   </body>
28 </html>
```

Notatki

Web Components

Components

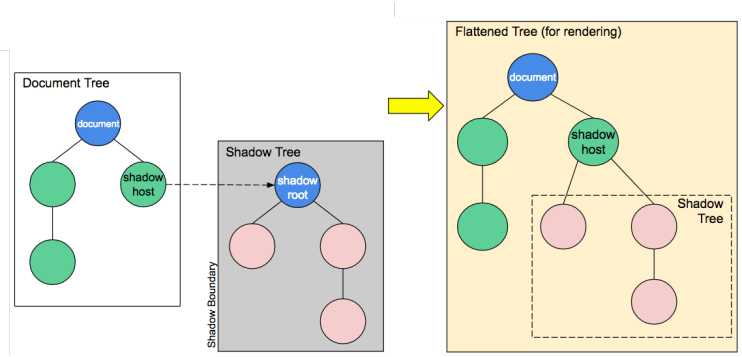
- Shadow DOM
- Custom elements
- HTML templates

Stages of building your own component

1. Template creation
2. A component class creation
3. Addition of the Shadow DOM tree
4. Registration of a new element
5. Expanding class logic

Notatki

Shadow DOM



Source: `https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM`

Notatki

Shadow DOM

Example

```
<html>
  <head>
    <style>
     .selected {background-color: red }
    </style>
  </head>
  <body>
    <div id='host'>Hello</div>
    <div class='selected'>World</div>
    <script>
      var host = document.querySelector('#host');
      var root = host.attachShadow({mode: 'closed'});
      console.log(host.shadowRoot);
      console.log(root);
      console.log(root.host);
      var div = document.createElement('div');
      div.textContent = "New content";
      div.className = "selected";
      root.appendChild(div);
      console.log(document.querySelectorAll('div').length)
          ;
      console.log(root.querySelectorAll('div').length);
    </script>
  </body>
</html>
```

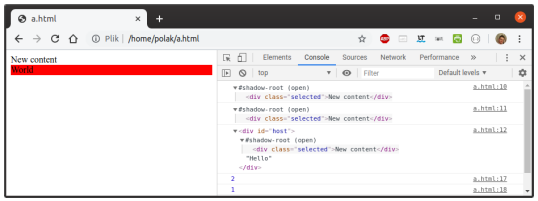


Figure: The result of the script execution

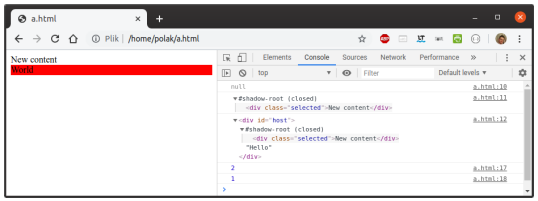


Figure: The result of the script execution

Notatki

Custom elements

Autonomous custom elements

```
...
<colored-text color="red"> Hello </colored-text>
<colored-text color="green"> Hello </colored-text>

<script>
  class ColoredText extends HTMLElement {
    constructor() {
      super();
      const color = this.getAttribute('color');
      // Create a shadow root
      const shadow = this.attachShadow({mode: 'open'
          });
      const p = document.createElement('p');
      p.textContent = this.textContent;
      // Create some CSS to apply to the shadow dom
      const style = document.createElement('style');
      style.textContent = `p { color: ${color} }`;
      // attach the created elements to the shadow
          dom
      shadow.appendChild(style);
      shadow.appendChild(p);
    }
  }

  customElements.define('colored-text', ColoredText);
</script>
...
```

It will be displayed

Hello

Hello

Customized built-in elements

```
...
<p is="colored-text" color="blue">Hello</p>

<script>
  class ColoredText extends HTMLParagraphElement {
    constructor() {
      super();
      const color = this.getAttribute('color');
      // Create a shadow root
      const shadow = this.attachShadow({mode: 'open'
          });
      const p = document.createElement('p');
      p.textContent = this.textContent;
      // Create some CSS to apply to the shadow dom
      const style = document.createElement('style');
      style.textContent = `p { color: ${color} }`;
      // attach the created elements to the shadow
          dom
      shadow.appendChild(style);
      shadow.appendChild(p);
    }
  }

  customElements.define("colored-text", ColoredText, extends:
      "p");
</script>
...
```

It will be displayed

Hello

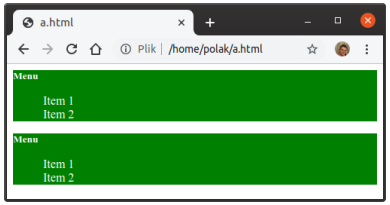
Notatki

HTML templates

The „template" element

```
<html>
  <body>
    <template>
      <style>
        nav {
          color: white;
          background-color: green;
        }

        h1 {
          font-size: smaller;
        }

        menuitem {
           display: block;
        }
      </style>
      <nav>
        <h1>Menu</h1>
        <menu type="list">
          <menuitem>Item 1</menuitem>
          <menuitem>Item 2</menuitem>
        </menu>
      </nav>
    </template>

    <my-menu> </my-menu>
    <my-menu> </my-menu>
```

```
    <script>
      customElements.define('my-menu',
        class extends HTMLElement {
          constructor() {
            super();
            let templateContent = document.
                querySelector('template').
                content;
            const shadowRoot = this.
                attachShadow({ mode: 'open'
                }).appendChild(
                templateContent.cloneNode(
                true));
          }
        })
    </script>
  </body>
</html>
```



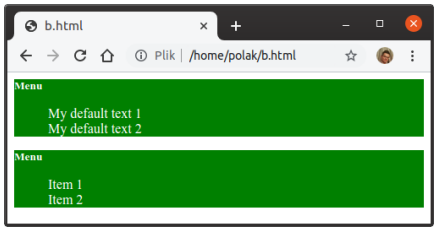
Notatki

HTML templates

The „slot" element

```
...
<template>
  <style>
  ...
  </style>
  <nav>
    <h1>Menu</h1>
      <menu type="list">
        <menuitem>
          <slot name="item1">My default text 1</
              slot>
        </menuitem>
        <menuitem>
          <slot name="item2">My default text 2</
              slot>
        </menuitem>
      </menu>
  </nav>
</template>

<my-menu> </my-menu>

<my-menu>
  <span slot="item1">Item 1</span>
  <span slot="item2">Item 2</span>
</my-menu>
```

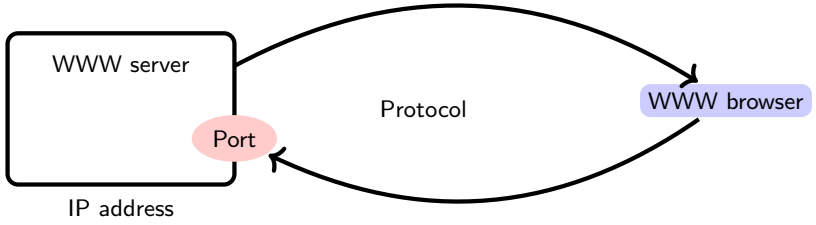
```
<script>
...
// Ta sama zawartość co na poprzednim slajdzie
      / The same content as on the previous
     slide
</script>
...
```



Notatki

Client-Server Model

WWW service



Notatki

Selected HTTP protocol commands (methods)

The "GET" command

Request URL:
`http://www.icsr.agh.edu.pl/index.html`

```
1 GET /index.html HTTP/1.1
2 Host: www.icsr.agh.edu.pl
3
```

Request

```
1  HTTP/1.1 200 OK
2  Date: Mon, 09 Aug 2013 17:02:08 GMT
3  Server: Apache/2.4.4 (UNIX)
4  Content-Length: 1776
5  Content-Type: text/html; charset=utf-8
6
7  <!DOCTYPE html>
8  <html>
9  ...
10 </html>
```

Response

The "POST" command

Request URL: `http://www.icsr.agh.edu.pl/cgi-bin/search.cgi`

```
1 POST /cgi-bin/search.cgi HTTP/1.1
2 Host: www.icsr.agh.edu.pl
3 Content-Length: 46
4
5 query=alpha+complex&casesens=false&cmd=submit
```

Request

```
1  HTTP/1.1 200 OK
2  Date: Mon, 09 Aug 2013 17:02:20 GMT
3  Server: Apache/2.4.4 (UNIX)
4  Content-Length: 1776
5  Content-Type: text/html; charset=utf-8
6  Connection: close
7
8  <!DOCTYPE html>
9  <html>
10 ...
11 </html>
```

Response

Notatki

Sending data from the HTML form

Notatki

Approving the form ⟼ data encoding ⟼ sending the data to a web server

```
1 <form method="..." enctype="..." action="...">
2 ...
3 </form>
```

- GET
- POST

- application/x-www-form-urlencoded
- multipart/form-data

Encoding procedure „application/x-www-form-urlencoded”

Example

Notatki

```
1 <form action="http://www.serwer.com/script">
2 Login:    <input name="login" type="TEXT"><br>
3 Password: <input name="password" type="PASSWORD">
4 </form>
```

HTML document

Login: Jan

Password: Kowalski (Nowak)

```
1 login=Jan&password=Kowalski+%28Nowak%29
```

Encoded data

Encoding procedure „multipart/form-data”

Example

```
1 <form  action="..." method="POST" enctype="multipart/form-data">
2 <input name="login" type="TEXT">
3 <input name="password" type="PASSWORD">
4 <input name="file" type="FILE"  accept="image/jpeg,image/gif">
5 </form>
```

Jan

Kowalski (Nowak)

image.jpg

```
1  POST /skrypt HTTP/1.0
2  Content-Length: 775
3  Content-Type: multipart/form-data; boundary=--------------------------815276501818664599101790669 2
4
5  ----------------------------815276501818664599101790669 2
6  Content-Disposition: form-data; name="login"
7
8  Jan
9  ----------------------------815276501818664599101790669 2
10 Content-Disposition: form-data; name="password"
11
12 Kowalski (Nowak)
13 ----------------------------815276501818664599101790669 2
14 Content-Disposition: form-data; name="file"; filename="image.jpg"
15 Content-Type: image/jpeg
16 Content-Transfer-Encoding: binary
17
18 The content of 'image.jpg'
19 ----------------------------815276501818664599101790669 2
```

Notatki

Node.js

General characteristics

- Provides JavaScript on the server side
- Uses V8 JavaScript Engine
- System for creating network services with asynchronous I/O
- Uses the event-driven programming paradigm
- It is well-suited for writing applications that require real-time communication between the browser and the server
- A single instance of Node.js acts as a single thread

Event loop — entity that handles / processes external events and converts them to callback functions

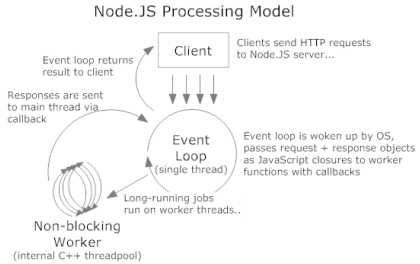


Figure: The diagram of the event loop operation in Node.js

Source: http://www.aaronstannard.com/post/2011/12/14/Intro-to-NodeJS-for-NET-Developers.aspx

Notatki

The "Hello World" example

```
1 #!/usr/bin/node
2 console.log("Hello World");
```

hello.js

Notatki

Input and output of data

```
1  process.stdout.write('1');
2  process.stdout.write('2');
3  console.log(3)
4
5  /*
6  console.log = function(d) {
7    process.stdout.write(d + '\n');
8  };
9  */
10
11 process.stdin.setEncoding('utf8');
12 process.stdout.write('Enter data - pressing ^ D will finish entering them\n');
13 process.stdin.on('readable', function() {
14   var chunk = process.stdin.read();
15   if (chunk !== null) {
16     process.stdout.write('Read: ' + chunk);
17   }
18 });
19 console.log("The end of the script has been reached");
```

script.js

Notatki

Access to environment variables, command line support

```
1 //Reading the value of the 'HOME' environment variable
2 console.log("Your home directory is: "+process.env['HOME']);
3
4 //Displays the value of the command line arguments
5 console.log("The command line arguments are:");
6 process.argv.forEach(function(value, index, array) {
7   console.log('\t'+index + ': ' + value);
8 });
```

script.js

Notatki

Module

Creation of module

```
1  var myModule = require('myModule');
2  //myModule = require("./myModule");
3  /*********************************/
4  console.log(myModule.variable1); //undefined
5  console.log(myModule.variable2); //2
6  console.log(myModule.variable3); //undefined
7  console.log(myModule.variable4); //4
8  console.log(myModule.fun1());    //"fun1"
9  console.log(myModule.fun2());    //Error
10 /*********************************/
11 console.log(myModule); //{ variable2: 2, variable4: 4, fun1: [Function]
       }
12 console.log(myModule()); //Error
13
```

script.js

```
1 $ export NODE_PATH='dir_1:dir_2:...:dir_N'
```

```
1  variable1 = 1;
2  exports.variable2 = 2;
3  var variable3 = 3;
4  var variable4;
5  module.exports.variable4 = 4;
6  var exports.variable5 = 5; //SyntaxError: Unexpected token .
7  exports.fun1 = function () {
8    return "fun1";
9  };
10 fun2 = function () {
11   return "fun2";
12 };
13 console.log(module);
14 /* Module {
15   ...,
16   exports: { variable2: 2, variable4: 4, fun1: [Function] },
17   ... } */
18 exports = function() {
19   return "fun3";
20 }
21 module.exports = function() {
22   return "fun4";
23 }
24 console.log(module.exports); //{ variable2: 2, variable4: 4, fun1:
       [Function] }
25 console.log(exports);       //{ variable2: 2, variable4: 4, fun1:
       [Function] }
26 console.log(module.exports == exports); //true
```

node_modules/myModule.js

Notatki

Module

Using modules

Package support

```
$ npm install package_name
# Modules ⟼ ./node_modules/
# Executables ⟼ ./node_modules/.bin/
# Manuals ⟼ are not installed

$ npm install --global  package_name
# Modules ⟼ {prefix}/lib/node_modules/
# Executables ⟼ {prefix}/bin/
# Manuals ⟼ {prefix}/share/man/
# {prefix} = e.g. /usr

$ npm link package_name
#  Executes: ln -s {prefix}/lib/node_modules/package_name/ ./node_modules/
```

Notatki

File support

```
1  var   fs = require("fs");
2  // Check if the file exists
3  try{ fs.accessSync('file.txt'); }
4  catch(err){ fs.writeFileSync('file.txt', '1'); }
5  fs.readFile('file.txt', 'utf-8', function (error,
       data) {
6      if (error) throw error;
7      console.log("Read value: "+data);
8  });
9
10 fs.writeFile('file.txt', '2', function (error) {
11     if (error) throw error;
12     console.log('The value 2 has been saved');
13 });
```

Invalid version

```
1  var   fs = require("fs");
2  try{ fs.accessSync('file.txt') }
3  catch(err){ fs.writeFileSync('file.txt', '1'); }
4
5  fs.readFile('file.txt', 'utf-8', function (error,
       data) {
6    if (error) throw error;
7    console.log("Read value: "+data);
8
9    fs.writeFile('file.txt', '2', function (error) {
10     if (error) throw error;
11     console.log('The value 2 has been saved');
12   });
13 });
```

Correct version

Notatki

SQLite 3 database support

```
1  var sqlite3 = require('sqlite3');
2
3  var db = new sqlite3.Database(':memory:'); //returns 'Database' object
4
5  db.serialize(function() {
6    db.run("CREATE TABLE products (info TEXT)");
7    var stmt = db.prepare("INSERT INTO products VALUES (?)");//returns  'Statement' object
8    for (var i = 0; i < 2; i++) {
9      stmt.run("Product " + i);
10   }
11   stmt.finalize();
12
13   jsonData = { products: [] };
14   db.each("SELECT rowid AS id, info FROM products", function(err, row) {
15       jsonData.products.push({ id: row.id, info: row.info });
16     }, function () {
17       console.log(JSON.stringify(jsonData)); //JSON.stringify - built-in JS function
18     }
19   );
20 });
21 db.close();
```

bd.js

Notatki

C++ Addons

The "Hello World" example

```
1  /*
2  The following program is equivalent to the following JS code:
3    exports.hello = function() { return 'world'; };
4    */
5
6  #include <node.h>
7  using namespace v8;
8
9  void Method(const FunctionCallbackInfo<Value>& args) {
10   Isolate* isolate = args.GetIsolate();
11   args.GetReturnValue().Set(String::NewFromUtf8(isolate, "world"));
         //Specify what function returns
12 }
13
14 void init(Local<Object> exports) {
15   NODE_SET_METHOD(exports, "hello", Method); //Associate the name '
         hello' with the above C++ method and export it
16 }
17
18 NODE_MODULE(NODE_GYP_MODULE_NAME, init) //there is no semicolon
```

hello.cc

```
1  {
2    "targets": [
3      {
4        "target_name": "hello",
5        "sources": [ "hello.cc" ]
6      }
7    ]
8  }
```

binding.gyp

Installation of the "node-gyp" program

```
$ npm install --global node-gyp
```

Compilation and program execution

```
$ node-gyp configure
$ node-gyp build
$ node hello.js
world
```

```
1  var addon = require('./build/Release/hello');
2  console.log(addon.hello());
```

hello.js

Notatki

HTTP support

Script skeleton

```
1  var http = require("http");
2
3  function requestListener(request, response) {
4    console.log("A request from the client has appeared");
5    response.writeHead(200, {"Content-Type": "text/plain"});
6    response.write("Hello World");
7    response.end();
8  }
9  var serwer = http.createServer(requestListener);
10 serwer.listen(8080);
11 console.log("Server started");
```

server.js

```
1 var http = require("http");
2
3 http.createServer(function(request, response) {
4   console.log("A request from the client has appeared");
5   response.writeHead(200, {"Content-Type": "text/plain"});
6   response.write("Hello World");
7   response.end();
8 }).listen(8080);
9 console.log("Server started");
```

Alternative version

Testing the operation of the script

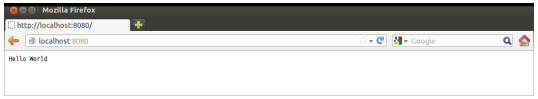


Figure: In the web browser

Notatki

URL parameter support

```
1  var url = require("url");
2  ...
3  function requestListener(request,response) {
4    console.log("A request from the client has appeared");
5    var url_parts = url.parse(request.url, true); //Pass true as the second argument to also parse the query string using the
         querystring module. Defaults to false.
6    console.log(url_parts);
7
8    response.writeHead(200, {"Content-Type": "text/plain"});
9    response.write('\n');
10   response.write(url_parts.pathname+'\n');
11   response.write('Login: '+url_parts.query['login']+'\n');
12   response.write('Password: '+url_parts.query['password']+'\n');
13   response.end();
14 }
15 ...
```

server.js

Output

Notatki

Form support

The "application/x-www-form-urlencoded" encoding support

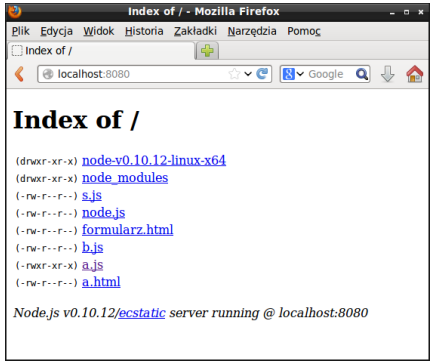
```
var qs = require('querystring');
...
function requestListener(request,response) {
  var url_parts = url.parse(request.url,true);
  if(url_parts.pathname == '/form'){ //generating the form
    response.writeHead(200, {"Content-Type": "text/html"});
    response.write('<form method="POST" action="/submit">');
    response.write('<input name="login" value="Jan">');
    response.write('<input name="password" value="Kowalski (Nowak) ąę">');
    response.write('<input type="submit">');
    response.write('</form>');
    response.end();
  }
  if(url_parts.pathname == '/submit') { //processing of the form content
    if(request.method=='GET') {
      response.writeHead(200, {"Content-Type": "text/plain"});
      response.write(url_parts.query['login']+'\n'); //the browser will write: "Jan\n"
      response.write(url_parts.query['password']+'\n'); //the browser will write: "Kowalski (Nowak) ąę\n"
      response.end();
    }
    else if(request.method=='POST') {
      var body='';
      request.on('data', function (data) {
        body +=data;
      });
      request.on('end',function(){
        var data =  qs.parse(body); //body contains "login=Jan&password=Kowalski+%28Nowak%29+%C4%85%C4%99"
        response.writeHead(200, {"Content-Type": "text/plain"});
        response.write(data['login']+'\n'); //the browser will write: "Jan\n"
        response.write(data['password']+'\n'); //the browser will write: "Kowalski (Nowak) ąę\n"
        response.end();
      });
    }
  }
}
...
```

Notatki

A stand-alone web server

Installation and startup

```
$ npm install http-server -g

$ http-server -h
usage: http-server [path] [options]

options:
  -p                Port to use [8080]
  -a                Address to use [0.0.0.0]
  -d                Show directory listings [true]
  -i                Display autoIndex [true]
  -e --ext          Default file extension if none supplied [none]
  -s --silent       Suppress log messages from output
  -h --help         Print this list and exit.
  -c                Set cache time (in seconds). e.g. -c10 for 10
                      seconds.
                    To disable caching, use -c-1.

$ http-server
Starting up http-server, serving ./ on port: 8080
Hit CTRL-C to stop the server
```



Notatki

Creating the application skeleton

```
$ mkdir MySite
$ cd MySite
$ vi  package.json
$ npm install #installing dependencies
```

```
{
    "name": "MySite",
    "version": "0.0.1",
    "private": "true",
    "dependencies": {
        "express": "*",
        "pug": "*",
        "morgan": "*"
    }
}
```

package.json

Notatki

The main file

```
var express = require('express'),
    logger = require('morgan');

var app = express();
var router = express.Router();
app.set('views', __dirname + '/views');
app.set('view engine', 'pug');
app.use(logger('dev'));
app.use(express.static(__dirname + '/public'));
router.get('/', function (req, res) {
  res.render('index',
    { title : 'Przykład' }
  )
});
app.use('/', router);
app.listen(3000);
```

app.js

Notatki

"Pug" files

```
1  extends layout.pug
2
3  block content
4    p
5      | Witaj Świecie
6      | Witaj Świecie
7    p
8      | Witaj Świecie
9
10   block sidebar
11     h1 Nagłówek
12     p
13       | Treść ramki
```

views/index.pug

```
1  doctype html
2  html
3    head
4      title #{title}
5      link(rel='stylesheet', href='/stylesheets/style.
           css')
6    body
7      header
8        h1 Moja strona
9      main
10       block content
11     aside
12       block sidebar
13     footer
14       p Aplikacja stworzona w oparciu o framework
             Express
```

views/layout.pug

Notatki

"CSS" files

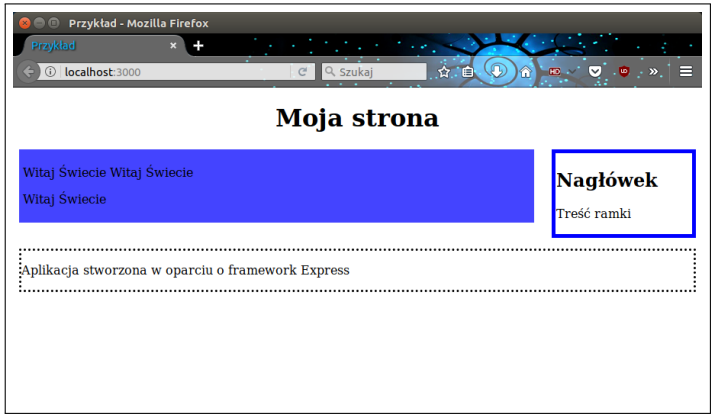
```
1  aside {
2    float: right;
3    border: 5px solid blue;
4    padding: 1px;
5    margin-bottom: 14px;
6    width: 20%;
7  }
8  main {
9    float: left;
10   background-color: #44f;
11   padding: 5px;
12   width: 75%;
13 }
14 footer {
15   clear:both;
16   border-style: dotted;
17 }
18 header {
19   text-align: center;
20 }
```

public/stylesheets/style.css

Notatki

Starting the application

```
$ node app
GET / 304 847.588 ms - -
GET /stylesheets/style.css 304 4.478 ms - -
```



Notatki

The express command

```
$ npm install --global express-generator
$ express --help
  Usage: express [options] [dir]

  Options:

    -h, --help          output usage information
        --version       output the version number
    -e, --ejs           add ejs engine support
        --pug           add pug engine support
        --hbs           add handlebars engine support
    -H, --hogan         add hogan.js engine support
    -v, --view <engine> add view <engine> support (ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)
    -c, --css <engine>  add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain
        css)
        --git           add .gitignore
    -f, --force         force on non-empty directory
```

Notatki

Generating the "Hello World" application

Using the command express

```
$ express  --view=pug MySite
  create : MySite
  create : MySite/package.json
  create : MySite/app.js
  create : MySite/public
  create : MySite/routes
  create : MySite/routes/index.js
  create : MySite/routes/users.js
  create : MySite/views
  create : MySite/views/index.pug
  create : MySite/views/layout.pug
  create : MySite/views/error.pug
  create : MySite/bin
  create : MySite/bin/www
  create : MySite/public/javascripts
  create : MySite/public/images
  create : MySite/public/stylesheets
  create : MySite/public/stylesheets/style.css

  install dependencies:
    $ cd MySite && npm install

  run the app:
    $ DEBUG=mysite:* npm start
```

Notatki

The 'package.json' file

```
{
  "name": "mysite",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.16.0",
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.0",
    "express": "~4.14.1",
    "morgan": "~1.7.0",
    "pug": "~2.0.0-beta10",
    "serve-favicon": "~2.3.2"
  }
}
```

Notatki

The main file (app.js)

```
1 ...
2 var index = require('./routes/index');
3 var users = require('./routes/users');
4
5 var app = express();
6 ...
7 app.use('/', index);
8 app.use('/users', users);
9 ...
```

Notatki

Files with route definitions

```
1 var express = require('express');
2 var router = express.Router();
3
4 /* GET home page. */
5 router.get('/', function(req, res, next) {
6   res.render('index', { title: 'Express' });
7 });
8
9 module.exports = router;
```

routes/index.js

```
1 var express = require('express');
2 var router = express.Router();
3
4 /* GET users listing. */
5 router.get('/', function(req, res, next) {
6   res.send('respond with a resource');
7 });
8
9 module.exports = router;
```

routes/user.js

Notatki

Other generated files

```
extends layout.pug

block content
  h1= title
  p Welcome to #{title}
```

views/index.pug

```
doctype html
html
  head
    title= title
    link(rel='stylesheet', href='/stylesheets/style.
        css')
  body
    block content
```

views/layout.pug

```
body {
  padding: 50px;
  font: 14px "Lucida Grande", Helvetica, Arial, sans-serif;
}

a {
  color: #00B7FF;
}
```

public/stylesheets/style.css

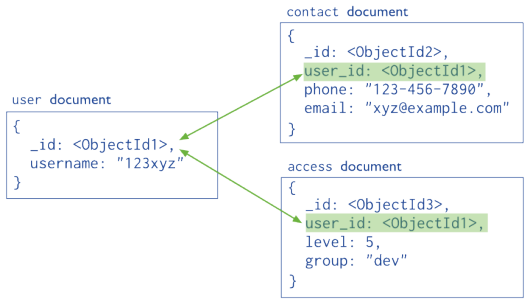
```
$ cd MySite && npm install
$ npm start
```

Instalowanie zależności i uruchamianie aplikacji

Notatki

The data model in MongoDB

References



Embedded documents

Notatki

Installing dependencies

```
$ vi  package.json
$ npm install # installing dependencies
```

```
...
"dependencies": {
  ...
  "monk": "*"
}
...
```

package.json

Notatki

Creating database content

```
$ mongo
MongoDB shell version: 2.6.3
connecting to: test
> use testbase1
switched to db testbase1
> db.datacollection.insert({"title":"Express & Mongo"})
> db.datacollection.insert({"title":"Express"})
> db.datacollection.find()
{ "_id" : ObjectId("52f8bb757bade7e2c4741741"), "title" : "Express & Mongo" }
{ "_id" : ObjectId("52f8bd407bade7e2c4741742"), "title" : "Express" }
> db.datacollection.find({"title":"Express"})
{ "_id" : ObjectId("52f8bd407bade7e2c4741742"), "title" : "Express" }
```

Notatki

Modifications

```
...
var bodyParser = require('body-parser');
// New code
var monk = require('monk');
var db = monk('localhost:27017/testbase1');
...
//new code
app.use(function(req,res,next){
    req.db = db;
    next();
});
//existing code
app.use('/', index);
app.use('/users', users);
...
```

app.js

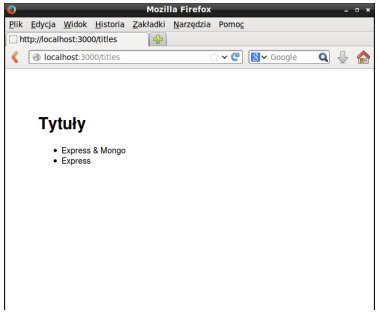
```
extends layout.pug

block content
  h1 Tytuły
  ul
     each element in titlelist
          li #{element.title}
```

views/titlelist.pug

Notatki

```
...
router.get('/titles', function(req, res) {
    var db = req.db;
    var collection = db.get('datacollection');
    collection.find({},{},function(e,docs){
        res.render('titlelist', {
            "titlelist" : docs
        });
    });
});

module.exports = router;
```

routes/index.js



AJAX

- AJAX (**A**synchronous **Ja**vaScript and **X**ML)
- AJAX = HTML + CSS + DOM + XMLHttpRequest + XML + JavaScript
- Capabilities:
  - Sending a query to the server without reloading the page
  - The application can make quick, incremental updates to the user interface without the need to reload the entire page in the browser
  - Parsing and working with XML documents
- Is it always worth using AJAX?

Notatki

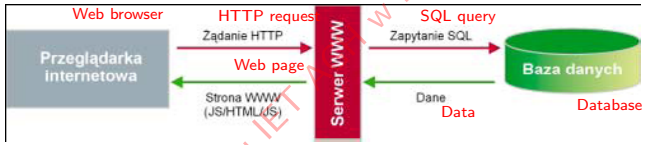


Figure: The scheme of a typical website

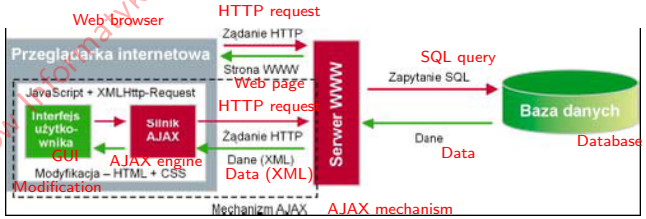


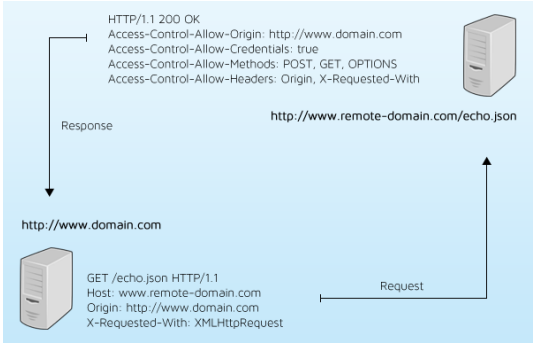
Figure: The scheme of the AJAX website

Request

```
1  var xhr;
2
3  xhr = new XMLHttpRequest();
4  if (!xhr) {
5    alert('I can not create an XMLHTTP object instance
         ');
6    return;
7  }
8  xhr.onreadystatechange = function() { alertContents(xhr
       ); };
9  // xhr.onreadystatechange = () => alertContents(xhr);
10 xhr.open('GET', "/requested_file.html", true);
11 xhr.send(null);
12 //xhr.open('POST, "/script.cgi", true);
13 //xhr.setRequestHeader('Content-Type','application/x-
       www-form-urlencoded');
14 //xhr.send('field1=value1&field2=value2&...');
```

```
1 <table>
2 <tr><td>Hello</td></tr>
3 </table>
```

requested_file.html

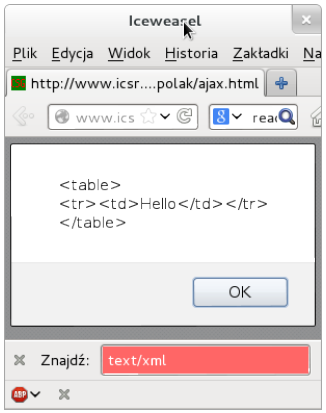


Source: https://zinoui.com/blog/cross-domain-ajax-request

Notatki

Handling the response

```
1  function alertContents(xhr) {
2    try {
3      if (xhr.readyState == 4) {
4        if (xhr.status == 200) {
5          alert(xhr.responseText);
6          var xmldoc = xhr.responseXML;
7          var root_node = xmldoc.getElementsByTagName(
               'td').item(0);
8          alert(root_node.firstChild.data);
9        }
10       else {
11         alert('There was a problem with this task.')
               ;
12       }
13     }
14   } catch( e ) { alert('An exception caught: ' + e.
         description); }
15 }
```



Notatki

Promises

- Represent / Store the results of an asynchronous operation[1]
- The results of the operation may not be available yet, but they will be
- Promises are chainable

States of promises

- pending
- fulfilled
- rejected
- settled

[1] Return value (in the case of success) or error (in the case of failure)

Notatki

Promises

Example of use

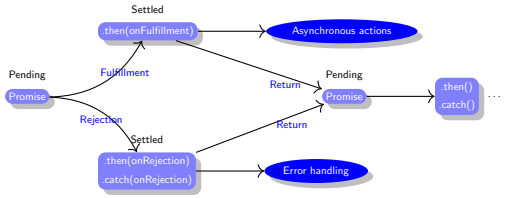
```
1  /*** Creating a promise ***/
2  function createPromise() {
3    return new Promise(function(resolve, reject) {
4      // Calculations performed asynchronously
5      setTimeout(function() {
6        var divider = Math.floor(Math.random() * 3);
7        if(divider != 0)
8          resolve(10/divider); // Fulfill a promise
9        else
10         reject("Attempt to divide by 0"); // Reject a promise
11     }, 2000);
12   });
13 }
14
15 /*** The use of the promise ***/
16 function usePromise() {
17   createPromise() //An instance of the promise has been
         created
18   .then(function(result) {
19     console.log('Division result:', result);
20   })
21   .catch(function(error) {
22     console.log('An error occurred!', error);
23   });
24 }
25
26 usePromise();
```

Output

```
'Division result:' 5
```

Output

```
"An error occurred!" "Attempt to divide by 0"
```



Notatki

Promises

The async / await syntax

That's how promises have been used so far

```
1  function usePromise() {
2    createPromise()
3      .then(function (result) {
4        console.log('Division result : ', result);
5      })
6      .catch(function (error) {
7        console.log('An error occurred! ', error);
8      });
9  }
10
11 usePromise();
```

Using async / await

```
1  async function usePromise() {
2    try {
3      const result = await createPromise();
4      console.log('Division result : ', result);
5    } catch (error) {
6      console.log('An error occurred! ', error);
7    }
8  }
9
10 usePromise();
```

Notatki

Fetch API

General characteristics

- A JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses
- Differences from AJAX:
  - It uses promises and not callbacks
    - A promise that is the result of a query will not reject on HTTP error status even if the response is an HTTP 404 or 500
  - By default, function to execute queries will not send or receive any cookies from the server

Notatki

Fetch API

Example of use

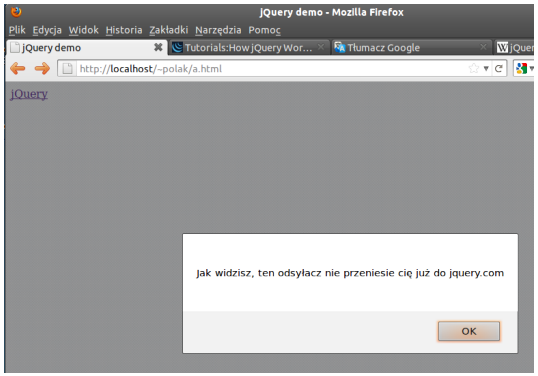
```
1  ...
2  <script>
3    const header = new Headers();
4    header.append('Content-Type', 'text/plain');
5    const request = new Request('http://localhost:8000/document.html',
6      {
7        method: 'GET',
8        headers: header,
9      });
10   fetch(request).then(response => {
11     if (response.status !== 200)
12       return Promise.reject('The query failed');
13     else {
14       console.log(response); /*Response {
15                                         type: "basic",
16                                         url: "http://localhost:8000/document.html",
17                                         redirected: false,
18                                         status: 200,
19                                         ok: true,
20                                         statusText: "OK",
21                                         headers: Headers,
22                                         bodyUsed: false
23                                       } */
24       //console.log(response.arrayBuffer()); //Promise { <state>: "pending" }
25       //console.log(response.blob()); //Promise { <state>: "pending" }
26       //console.log(response.json()); //Promise { <state>: "pending" }
27       //console.log(response.text()); //Promise { <state>: "pending" }
28       //console.log(response.formData()); //Promise { <state>: "pending" }
29       response.text().then(function (text) {
30         console.log(text); //The content of the current file
31       });
32     } //if (response.status !== 200)
33   }).catch(error => console.error(error))
34 </script>
35 ...
```

document.html

Notatki

The basics of the "jQuery" library

```
1  <html>
2   <head>
3     <meta charset="utf-8">
4     <title>jQuery demo</title>
5   </head>
6   <body>
7     <a href="http://jquery.com/">jQuery</a>
8     <script src="http://code.jquery.com/jquery-3.4.1.
          min.js"></script>
9     <script >
10      //$(document).ready(function(){
11      jQuery(document).ready(function(){
12        $("a").click(function(event)){
13          alert("Jak widzisz, ten odsyłacz nie
                przeniesie cię już do jquery.com");
                //As you can see, this link will not
                take you to jquery.com anymore
14          event.preventDefault();
15        });//$("a").ready(function()
16      });  //jQuery(document).ready(function()
17    </script>
18  </body>
19 </html>
```



Notatki

Obtaining text content

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p { color:blue; margin:8px; }
      b { color:red; }
    </style>
    <script src="http://code.jquery.com/jquery-3.4.1.min.js"></script>
  </head>
  <body>
    <p><b>Test</b> Paragraph.</p>

    <p></p>
    <script >
      var str = $("p:first").text(); //The variable 'str' contains the string 'Test Paragraph'
      $("p:last").html(str);
    </script>
  </body>
</html>
```

Test Paragraph.
Test Paragraph.

Notatki

Access to CSS styles

Equivalent to the DOM example

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>An example</title>
    ...
  </head>
  <body>
    <table border="1">
      <tr><td id="cell1">a</td><td>b</td></tr>
      <tr><td>c</td><td>d</td></tr>
    </table>
    <form>
      <input type="button" value="Red" onClick="color('#FF0000');">
      <input type="button" value="Green" onClick="color('#00FF00');">
    </form>
  </body>
</html>
```

a	b
c	d

```
...
<script>
  function color(value){
    $('#cell1').css("backgroundColor",value);
  }
</script>
```

Use of jQuery methods

```
<script>
  function color(value){
    var element = document.getElementById("cell1");
    element.style.backgroundColor = value;
  }
</script>
```

Use of DOM methods

Notatki

Adding a new element to the set of matched elements

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-3.4.1.min.js"></script>
  </head>
  <body>
    <p>Hello</p><span>Hello Again</span>
    <script >
      $("p").add("span").css("background", "yellow");
    </script>
  </body>
</html>
```

Hello

Hello Again

Notatki

Inserting a new table cell at the end of the row

Equivalent to the DOM example

```
<html>
  <head>
    <title>An example</title>
    ...
  </head>
  <body>
    <table border="1">
      <tr id="row1"><td>a</td><td>b</td></tr>
      <tr><td>c</td><td>d</td></tr>
    </table>
    <form>
      <input type="button" value="Insert" onClick="insert();">
    </form>
  </body>
</html>
```

a	b
c	d

a	b	b1
c	d	

```
...
<script>
  function insert(){
    $('#row1').append($('<td>').text('b1'));
  }
</script>
...
```

Use of jQuery methods

```
...
<script>
  function insert(){
    var newTD = document.createElement("td");
    var newTextNode = document.createTextNode("b1");
    newTD.appendChild(newTextNode);
    var element = document.getElementById("row1");
    element.appendChild(newTD);
  }
</script>
...
```

Use of DOM methods

Notatki

Animating text

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        background-color:#bca;
        width:100px;
        border:1px solid green;
      }
    </style>
    <script src="http://code.jquery.com/jquery-3.4.1.min.js"></script>
  </head>
  <body>
    <button id="go">&raquo; Run</button>

    <div id="block">Hello!</div>
    <script>
      /* Using many types of units in one animation. */

      $("#go").click(function(){
        $("#block").animate({
          width: "70%",
          opacity: 0.4,
          marginLeft: "0.6in",
          fontSize: "3em",
          borderWidth: "10px"
        }, 1500 );
      });
    </script>
  </body>
</html>
```

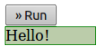
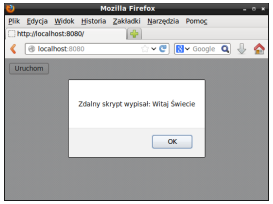


Figure: Pierwsza klatka animacji

Figure: Ostatnia klatka animacji

Notatki

Support for AJAX technology




```
...
<script>
  $(document).ready(function( ){
    $('#button1').click(function(){
      $.ajax({
        type: "GET",
        url: "http://localhost:8080/hello",
        success: function(msg){
          alert( "Zdalny skrypt wypisał: " + msg ); //The
                remote script was written:
        }
      });
    });
  });
</script>
...
<form action="#">
    <input type="button" value="Uruchom" id="button1" />
</form>
...
```

form.html

```
var http = require("http");
var url = require("url");
var fs = require('fs');

function requestListener(request, response) {
  var url_parts = url.parse(request.url,true);

  if(url_parts.pathname == '/hello'){
    response.writeHead(200, {"Content-Type": "text/plain
          "});
    response.write("Witaj Świecie");
    response.end();
  }
  else {
    var filename="form.html"
    var stat = fs.statSync(filename);
    response.writeHead(200, {
      'Content-Type': 'text/html',
      'Content-Length': stat.size
    });
    var readStream = fs.createReadStream(filename);
    readStream.pipe(response);
  }
}
var server = http.createServer(requestListener)
server.listen(8080);
```

node.js

Notatki

The “Hello World” example

```
1 console.log("Hello World");
```

script.ts

```
1 console.log("Hello World");
```

script.js

Notatki

Error handling

Typing error

```
1 let alive: boolean = 'abc';
2 console.log(alive);
```

script.ts

```
1 let alive = 'abc';
2 console.log(alive);
```

script.js

Syntax error

```
1 let if = 2;
2 console.log(if);
```

script.ts

```
1 let ;
2 if ( = 2)
3     ;
4 console.log();
5 if ()
6     ;
```

script.js

Notatki

Declaring the type of variable

```
let alive:boolean = true;
let age:number = 48;
let name:string = 'Kowalski';

let names:string[] = ['Jan','Jerzy'];
let children_age:Array<number> = [1, 20, 3];
let parents_age:Array<number> = [40, "forty one"]; //... Type 'string' is not assignable to type 'number'.

let tuple:[string,number,boolean];
tuple = ['1',2,true];
console.log(tuple[2]); // true
tuple = ['1',2,true,4]; // OK
tuple = ['1',2]; //... Type '[string, number]' is not assignable to type '[string, number, boolean]'. ...
tuple = [1,2,3]; //... Type '[number, number, number]' is not assignable to type '[string, number, boolean]'.

enum Eyes {Blue, Green = 4, Grey, Brown};
let eye_color = Eyes.Blue;
console.log(Eyes[0]); //  Blue
console.log(Eyes[4]); //  Green
console.log(Eyes[5]); //  Grey
```

```
let anything:any = 4;
anything = "String"; //OK
anything = true; //OK
anything.x; //OK
anything(); //OK
new anything(); //OK
let string: string = anything; //OK

let something: unknown = 4;
something = "String"; //OK
something = true; //OK
something.x; //... Object is of type 'unknown'
something(); //... Object is of type 'unknown'
new something(); //... Object is of type 'unknown'
let string2: string = something; // ... Type 'unknown' is not assignable to type 'string'.

function print(message): void {
  console.log(message);
}

function exception(message): never {
  throw new Error(message);
}

function loop(): never  {
  while(true){}
}
```

Notatki

Defining the type of variables

```
let string:string;
string = 1; //Error, you can not assign a number to a string type variable

let number = 1;
number = '2'; //Error, you can not assign a string to a numeric type variable
```

script.ts

```
let string:string;
string =  <any> 1; //Now it is OK
//or
string =  1 as any;
let number = 1;
number = <any> '2'; //Now it is OK
//or
number = '2' as any; //Now it is OK
```

script.ts

Notatki

Interfaces

Creating a complex type

```
interface Person {
  first_name: string; //mandatory
  last_name: string;  //mandatory
  age?: number;       //optional
}
/*****************************/
let user: Person;
/*****************************/
user = {
  first_name: 'Jan',
  last_name: 'Kowalski'
}
/*****************************/
user = {
  first_name: 'Jan',
  last_name: 'Kowalski',
  age: '40'//Error: wrong value type
}
/*****************************/
user = {
 // Error: 'last_name' is not specified
 first_name: 'Jan'
}
/*****************************/
user = {
  first_name: 'Jan',
  last_name: 345, //Error: wrong value type
  age: 'teenager'
}
```

script.ts

Notatki

Interfaces

Specifying the types of functions

```
interface stringFunction {
  (param1:string, param2:string): string;
}
/*********************************************/
let addStrings: stringFunction;
let addNumbers: stringFunction;
/*********************************************/
addStrings = function(string1: string,string2: string) {
  return string1+string2;
} //OK

addNumbers = function(number1: number, number2: number) {
  return number1 + number2;
} /*...
error TS2322: Type '(number1: number, number2: number) => number' is not assignable to type '
    stringFunction'.
  Types of parameters 'number1' and 'param1' are incompatible.
    Type 'number' is not assignable to type 'string'.

*/
/*********************************************/
addStrings('Jan','Kowalski'); //OK
addStrings('Kowalski'); //... error TS2346: Supplied parameters do not match any signature of call
    target.
addStrings(1,2); //... error TS2345: Argument of type 'number' is not assignable to parameter of type '
    string'.
```

script.ts

Notatki

Interfaces

Indexed type

```
interface hashString {
     [index: string]: string;
}
let parameters: hashString;

parameters['server'] = 'HP'; // OK
let bar:number = parameters['server']; //... error TS2322: Type 'string' is not assignable to type '
     number'.
parameters['server'] = 234; //... error TS2322: Type 'number' is not assignable to type 'string'.
```

script.ts

Notatki

Interfaces

Class types

```
class Person {
  id:number;
}

interface Validator{
  checkIdUniqueness(s: number): boolean;
}

class Employee extends Person implements Validator {}
```

script.ts

Notatki

Classes

Modifiers

```typescript
class Person {
  protected _id:number;
  readonly first_name:string;
  readonly last_name:string = "Anonym";

  get id(): number {
    return this._id;
  }

  set id(value: number) {
    this._id = value
  }

  constructor(first_name:string,name:string) {
    this.first_name = first_name;
    this.last_name = last_name;
  }
}
```

```typescript
class Employee extends Person  {
    constructor(id:number,first_name:string,
        last_name:string) {
    super(first_name,last_name);
    this.id=id;
  }
}

let Employee = new Employee(1,"Stanisław","
    Polak");
console.log(Employee.id);        //1
console.log(Employee._id);        //Property '
    _id' is protected and only accessible
    within class 'Person' and its subclasses.
console.log(Employee.first_name);      //Stanis
    ław
console.log(Employee.last_name); //Polak
Employee.first_name = "Jan"; //...Left-hand
    side of assignment expression cannot be a
     constant or a read-only property.
```

script.ts

Notatki

Classes and generic types

```typescript
class CustomCollection<T> {
  private itemArray: Array<T>;

  constructor() {
    this.itemArray = [];
  }

  Add(item: T) {
    this.itemArray.push(item);
  }

  GetFirst(): T {
    return this.itemArray[0];
  }
}
/*******************************/
class User {
  public Name;
}
/*******************************/
class Message {
  public Message;
}
```

```typescript
class MyApp {
  constructor() {
    let myUsers = new CustomCollection<User>();
    let myMessages = new CustomCollection<Message>();

    let user: User = myUsers.GetFirst();       // OK
    let message: Message = myUsers.GetFirst(); // Error because of
        the Generic type validation.
    myUsers.Add(new Message());                // Error because of
        the Generic type validation.
  }
}
```

script.ts

Source: https://gist.github.com/abergs/5817818

Notatki

Decorators

```typescript
function f() {
  console.log("f(): evaluated");
  return function (target, propertyKey: string, descriptor:
        PropertyDescriptor) {
    console.log("f(): called");
    console.log(target);
    console.log();
    console.log(propertyKey);
    console.log();
    console.log(descriptor);
  }
}

function g(value) {
  console.log("g("+value+"): evaluated");
  return function (target, propertyKey: string, descriptor:
        PropertyDescriptor) {
    console.log("g("+value+"): called");
  }
}

class C {
  @f()
  @g('abc')
  method() {}
}
```

script.ts

Source: https://github.com/Microsoft/TypeScript-Handbook/blob/master/pages/Decorators.md

Notatki

Namespace

Single-file

```typescript
namespace A {
  var a:string = 'abc';
  export class Twix {
    constructor() {
      console.log('Twix');
    }
  }

  export class PeanutButterCup {
    constructor() {
      console.log('PeanutButterCup');
    }
  }

  export class KitKat {
    constructor() {
      console.log('KitKat');
    }
  }
}
let o1 = new A.Twix(); // Twix
let o2 = new A.PeanutButterCup(); // PeanutButterCup
let o3 = new A.KitKat(); // KitKat
console.log(A.a); //...error TS2339: Property 'a'
      does not exist on type 'typeof A'.
```

script.ts

Source: http://stackoverflow.com/questions/30357634/how-do-i-use-namespaces-with-typescript-external-modules

Notatki

Namespace

Multiple-file

```ts
namespace A {
  export class Twix { ... }
}
```

global1.ts

```ts
namespace A {
  export class PeanutButterCup { ... }
}
```

global2.ts

```ts
namespace A {
  export class KitKat { ... }
}
```

global3.ts


```ts
/// <reference path="global1.ts" />
/// <reference path="global2.ts" />
/// <reference path="global3.ts" />
let o1 = new A.Twix();
let o2 = new A.PeanutButterCup();
let o3 = new A.KitKat();
```

script.ts

Notatki

Modules

Exporting

```ts
export class Twix {
  constructor() {
    console.log('Twix');
  }
}
export {Twix as Raider};
```

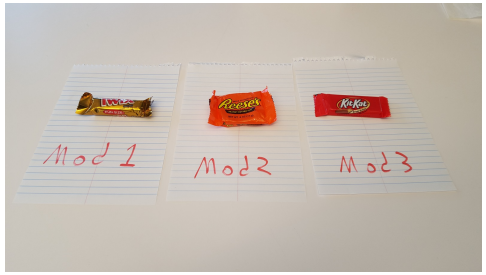
Mod1.ts

```ts
class PeanutButterCup {
  constructor() {
    console.log('PeanutButterCup');
  }
}
export {PeanutButterCup};
```

Mod2.ts

```ts
export  class KitKat {
  constructor() {
    console.log('KitKat');
  }
}
```

Mod3.ts



Source: http://stackoverflow.com/questions/30357634/how-do-i-use-namespaces-with-typescript-external-modules

Notatki

Modules

Importing

```ts
export class Twix {...}
export {Twix as Raider};
```

Mod1.ts

```ts
class PeanutButterCup {...}
export {PeanutButterCup};
```

Mod2.ts

```ts
export class KitKat {...}
```

Mod3.ts

```ts
import {Twix, Raider} from './Mod1';
import {PeanutButterCup} from './Mod2';
import {KitKat} from './Mod3';
import {KitKat as KitKatChunKy} from './Mod3';
import * as  Mars from './Mod1';

let o1 = new Twix(); //  Twix
let o2 = new Raider(); // Twix
let o3 = new PeanutButterCup(); //  PeanutButterCup
let o4 = new KitKat(); //  KitKat
let o5 = new KitKatChunKy(); //  KitKat
let o6 = new Mars.Twix(); //  Twix
let o7 = new Mars.Raider(); //  Twix
```

script.ts

Notatki

Namespaces in modules

```ts
export namespace A {
  export class Twix { ... }
}
```

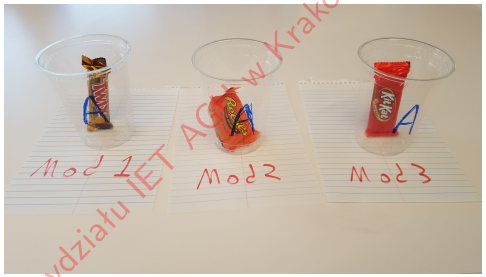
Mod1.ts

```ts
export namespace A {
  export class PeanutButterCup { ... }
}
```

Mod2.ts

```ts
export namespace A {
  export class KitKat { ... }
}
```

Mod3.ts



Source: http://stackoverflow.com/questions/30357634/how-do-i-use-namespaces-with-typescript-external-modules

Notatki

Modules

Importing in the “NodeJS” style

```
1 class Twix {
2   constructor() {
3     console.log('Twix');
4   }
5 }
6 export = Twix;
```

Mod.ts

```
1 import Twix = require('./Mod');
2 let o = new Twix();
```

script.ts

Notatki

Creating a project

```
1  {
2    "compilerOptions": {
3      "module": "commonjs",
4      "target": "es5",
5      "noImplicitAny": false,
6      "sourceMap": false
7    },
8    "files": [
9      "script.ts"
10   ]
11 }
```

tsconfig.json

Notatki

Declaration files

```ts
import area = require('./area'); //including 'area.d.ts'
let radius = 2;
console.log(`The area of the circle with radius ${radius} is ${area(radius)}`);
```

script.ts

```ts
function area(radius){
  return Math.PI*Math.pow(radius,2);
}
module.exports = area;
```

area.js

```ts
declare function area(radius: number) : number;
export = area
```

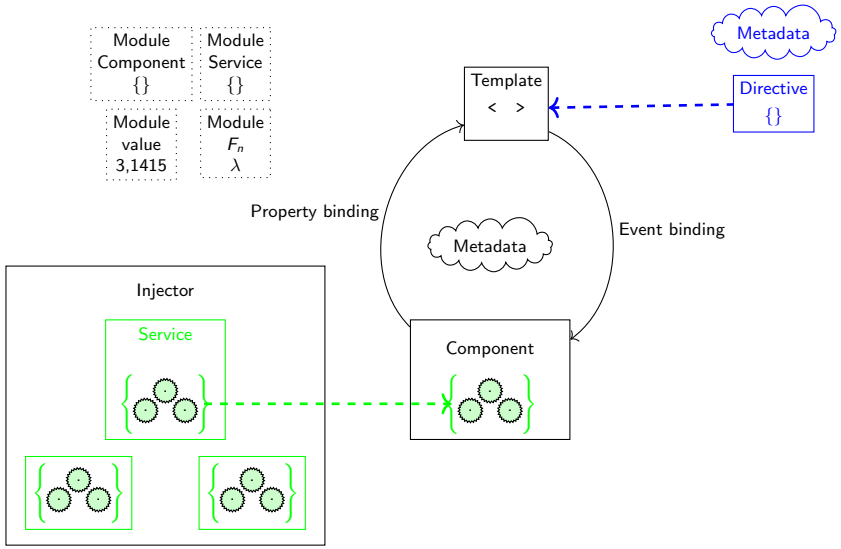
area.d.ts

```ts
"use strict";
var area = require("./area");
var radius = 2;
console.log("The area of the circle with radius " + radius + " is " + area(radius));
```

script.js

Notatki

Architecture



Notatki

Generating the application skeleton

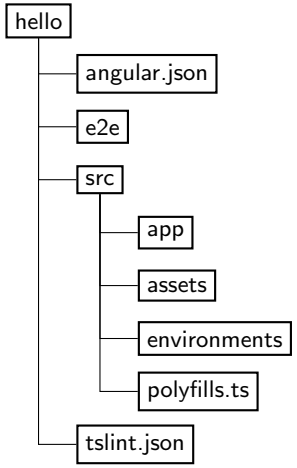
```
$ npm install --global @angular/cli
$ ng new hello
$ cd hello
$ ng serve
```

□

Notatki

The structure of the project

Important files and directories



Notatki

Component and module

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'app';
10 }
```

src/app/app.component.ts

```
1  import { BrowserModule } from '@angular/platform-browser'
       ;
2  import { NgModule } from '@angular/core';
3
4  import { AppComponent } from './app.component';
5
6  @NgModule({
7    declarations: [AppComponent],
8    imports: [BrowserModule],
9    providers: [],
10   bootstrap: [AppComponent]
11 })
12 export class AppModule { }
```

src/app/app.module.ts

Notatki

The main file

```
1  import { enableProdMode } from '@angular/core';
2  import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4  import { AppModule } from './app/app.module';
5  import { environment } from './environments/environment';
6
7  if (environment.production) {
8    enableProdMode();
9  }
10
11 platformBrowserDynamic().bootstrapModule(AppModule)
12   .catch(err => console.log(err));;
```

src/main.ts

Notatki

Template

```
<!--The content below is only a placeholder and can be
      replaced.-->
<div style="text-align:center">
  <h1> Welcome to {{title}}!!  </h1>
...
```

src/app/app.component.html

```
...
@Component({
  selector: 'app-root',
  ...
})
export class AppComponent {
  title = 'app';
}
```

src/app/app.component.ts

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello</title>
    <base href="/">

    <meta name="viewport" content="width=device-width, initial
          -scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
  </head>
  <body>
    <app-root></app-root>
  </body>
</html>
```

src/index.html

Notatki

The "Karma' environment for running unit tests

Configuration

```
...
module.exports = function (config) {
  config.set({
    ...
    frameworks: ['jasmine', '@angular-devkit/
          build-angular'],
    plugins: [
      require('karma-jasmine'),
      require('karma-chrome-launcher'),
      ...
    ],
    ...
  });
};
```

src/karma.conf.js

```
// This file is required by karma.conf.js and loads
     recursively all the .spec and framework files

import 'zone.js/dist/zone-testing';
import { getTestBed } from '@angular/core/testing';
import {
BrowserDynamicTestingModule,
platformBrowserDynamicTesting
} from '@angular/platform-browser-dynamic/testing';

declare const require: any;

// First, initialize the Angular testing environment
    .
getTestBed().initTestEnvironment(
BrowserDynamicTestingModule,
platformBrowserDynamicTesting()
);
// Then we find all the tests.
const context = require.context('./', true, /\.spec
     \.ts$/);
// And load the modules.
context.keys().map(context);
```

src/test.ts

Notatki

The "Karma' environment for running unit tests

Tests

```
...
describe('AppComponent', () => {
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  }));

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app).toBeTruthy();
  });

  it("should have as title 'hello'", () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app.title).toEqual('app');
  });

  it(('should render title in a h1 tag', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.debugElement.nativeElement;
    expect(compiled.querySelector('h1').textContent).toContain('Welcome to hello!'
          );
  });
});
```

src/app/app.component.spec.ts

```
$ ng test
...
24 05 2019 14:09:36.410:INFO [karma-server
      ]: Karma v4.0.1 server started at
      http://0.0.0.0:9876/
24 05 2019 14:09:36.411:INFO [launcher]:
      Launching browsers Chrome with
      concurrency unlimited
24 05 2019 14:09:36.431:INFO [launcher]:
      Starting browser Chrome
...
Chrome 74.0.3729 (Linux 0.0.0): Executed 3
      of 3 SUCCESS (0.255 secs / 0.239 secs
      )
TOTAL: 3 SUCCESS
TOTAL: 3 SUCCESS
```

Starting the test

Notatki

The "Protractor" environment for running e2e tests

Configuration

```
// Protractor configuration file, see link for more information
// https://github.com/angular/protractor/blob/master/lib/config.ts

/*global jasmine */
var SpecReporter = require('jasmine-spec-reporter');

exports.config = {
  allScriptsTimeout: 11000,
  specs: ['./src/**/*.e2e-spec.ts'],
  capabilities: {'browserName': 'chrome'},
  directConnect: true,
  baseUrl: 'http://localhost:4200/',
  framework: 'jasmine',
  jasmineNodeOpts: {
    showColors: true,
    defaultTimeoutInterval: 30000,
    print: function() {}
  },
  onPrepare() {
    require('ts-node').register({
      project: require('path').join(__dirname, './tsconfig.e2e.json')
    });
    jasmine.getEnv().addReporter(new SpecReporter({ spec: { displayStacktrace: true } }));
  }
};
```

e2e/protractor.conf.js

Notatki

The "Protractor" environment for running e2e tests

Test

```
1  import { AppPage } from './app.po';
2  import { browser, logging } from 'protractor';
3
4  describe('workspace-project App', () => {
5    ...
6    it('should display welcome message', () => {
7      page.navigateTo();
8      expect(page.getTitleText()).toEqual('Welcome to app!');
9    });
10   ...
11 });
```

e2e/src/app.e2e-spec.ts

```
1  import { browser, by, element } from 'protractor';
2
3  export class AppPage {
4    navigateTo() {
5      return browser.get(browser.baseUrl) as Promise<any>;
6    }
7
8    getTitleText() {
9      return element(by.css('app-root h1')).getText() as Promise
           <string>;
10   }
11 }
```

e2e/src/app.po.ts

```
1  $ ng e2e
2  ...
3  14:23:10] I/file_manager - creating folder /home/polak/
       hello/node_modules/webdriver-manager/selenium
4  [14:23:10] I/config_source - curl -o/home/polak/hello/
       node_modules/webdriver-manager/selenium/chrome-
       response.xml https://chromedriver.storage.
       googleapis.com/
5  ...: Compiled successfully.
6  [14:23:11] I/downloader - curl -o/home/polak/hello/
       node_modules/webdriver-manager/selenium/
       chromedriver_74.0.3729.6.zip https://chromedriver
       .storage.googleapis.com/74.0.3729.6/
       chromedriver_linux64.zip
7  [14:23:12] I/update - chromedriver: unzipping
       chromedriver_74.0.3729.6.zip
8  [14:23:12] I/update - chromedriver: setting permissions
       to 0755 for /home/polak/hello/node_modules/
       webdriver-manager/selenium/chromedriver_74
       .0.3729.6
9  [14:23:13] I/launcher - Running 1 instances of WebDriver
10 [14:23:13] I/direct - Using ChromeDriver directly...
11 Jasmine started
12
13 workspace-project App
14 ... should display welcome message
15
16 Executed 1 of 1 spec SUCCESS in 1 sec.
17 [14:23:18] I/launcher - 0 instance(s) of WebDriver still
       running
18 [14:23:18] I/launcher - chrome #01 passed
```

Starting the test

Notatki

Application concept

Assumptions

- The app, at the very beginning, displays:
  - default title („Welcome to app!")
  - list of products in the form of a HTML table
  - button with the word „Ustaw tytuł" (Eng. Set title)
- Displaying of the list of products and the button corresponds to a separate component
- The content of the title and table headings determines the main component
- Pressing the button modifies the title: "Welcome to Lista produktów!" instead of "Welcome to app!"
- The products are stored in the program in the form of an array of objects
- The page has to be responsive

Required ingredients

- The main module
- Components:
  - main (parent)
  - child (subordinate), displaying the list of products
- A class that represents the product
- A service that provides the content of the list of products

Notatki

Generating skeletal files

```
$ ng new shop
$ cd shop
$ ng generate class Product # Create a class skeleton
CREATE src/app/product.spec.ts (158 bytes)
CREATE src/app/product.ts (25 bytes)
$ ng generate service Products # Create a service skeleton
CREATE src/app/products.service.spec.ts (343 bytes)
CREATE src/app/products.service.ts (137 bytes)
$ ng generate component ProductsList # Create the component's skeleton
CREATE src/app/products-list/products-list.component.css (0 bytes)
CREATE src/app/products-list/products-list.component.html (32 bytes)
CREATE src/app/products-list/products-list.component.spec.ts (671 bytes)
CREATE src/app/products-list/products-list.component.ts (296 bytes)
UPDATE src/app/app.module.ts (422 bytes)
```

Notatki

Some of the generated skeletal files

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class ProductsService {

  constructor() { }
}
```

src/app/products.service.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-products-list',
  templateUrl: './products-list.component.html',
  styleUrls: ['./products-list.component.css']
})
export class ProductsListComponent implements OnInit {
  constructor() { }
  ngOnInit() { }
}
```

src/app/products-list/products-list.component.ts

```
<p>products-list works!</p>
```

src/app/products-list/products-list.component.html

```
export class Product {
}
```

src/app/product.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { ProductsListComponent } from './products-list/
      products-list.component';

@NgModule({
  declarations: [
    AppComponent,
    ProductsListComponent
  ],
  imports: [
    BrowserModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

src/app/app.module.ts

Notatki

Initial activities

Enabling responsiveness

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Shop</title>
    <base href="/">
    <link rel="stylesheet"
          href="https://stackpath.bootstrapcdn.com/bootstrap/
          4.3.1/css/bootstrap.min.css">
    <meta name="viewport" content="width=device-
          width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="
          favicon.ico">
  </head>
  <body class="container">
    <app-root></app-root>
    <script src="https://code.jquery.com/jquery-3.3.1.slim.
          min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/
          popper.js/1.14.7/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/
          bootstrap/4.3.1/js/bootstrap.min.js"></script>
  </body>
</html>
```

src/index.html

Embedding the child component

```
<h1>Welcome to  {{title}} </h1>
<app-products-list> </app-products-list>
```

src/app/app.component.html

```
...
@Component({
  selector: 'app-products-list',
  ...
})
...
```

src/app/products-list/products-list.component.ts

Notatki

Data transfer between components

Property binding and event binding

```
...
export class AppComponent {
  productsHeaders : string[] = ['Nazwa','Opis','Cena'
      ];
  setTitle (title: string){
    title = 'shop';
  }
}
```

src/app/app.component.ts

```
<h1> {{title}} </h1>
<app-products-list
  [headers] = 'productsHeaders'
  <!-- bind-headers="productsHeaders" -->
  (changeTitle) = 'setTitle($event)' >
  <!-- on-changeTitle="setTitle($event)" -->
</app-products-list>
```

src/app/app.component.html

```
import { ..., Input, Output, EventEmitter }
    from '@angular/core';

export class ProductsListComponent implements
    OnInit {
  @Input() headers: string[];
  @Output() changeTitle: EventEmitter<string> =
      new EventEmitter<string>();
  ...
  onClick() {
    this.changeTitle.emit ('Lista produktów');
  }
}
```

src/app/products-list/products-list.component.ts

```
<ng-template ... [ngForOf]=" headers ">
  ...
</ng-template>
...
<button (click)= 'onClick()' >Ustaw tytuł</button>
<!-- ... on-click="onClick()" ... -->
```

src/app/products-list/products-list.component.html

Notatki

A class that represents the product

```
export class Product {
  id: number;
  name: string = '';
  description: string = '';
  price: number;
}
```

src/app/product.ts

Notatki

Implementation of the product list delivery service

```
import { Injectable } from '@angular/core';
import { Product } from './product';

const PRODUCTS: Product[] = [
  {
    id: 1,
    name: 'spadochron',
    description: 'Rewelacyjny spadochron dla nurków! Znakomicie zapobiega zderzeniu się z dnem',
    price: 250.99
  },
  {
    id: 2,
    name: 'wanna',
    description: 'Dzięki wbudowanym drzwiom już więcej nie pośliźniesz się wychodząc z wanny',
    price: 599.80
  },
];
/*******************************/
@Injectable()
export class ProductsService {
  getProducts(): void {
    return PRODUCTS;
  }
}
```

src/app/products.service.ts

Notatki

Injecting the service

```
...
import { ProductsService } from '../products.service';
import { Product } from '../product';
...
export class ProductsListComponent implements OnInit {
  products:Product[];

  //Injecting the 'ProductsService' service - after the injection, the service will be available using the expression 'this.productsService'
  constructor(private productsService: ProductsService) {}

  ngOnInit(): Product[] {
    this.products =  this.productsService.getProducts();
  }
}
```

src/app/products-list/products-list.component.ts

```
...
import { ProductsService }  from './products.service';

@NgModule({
  ...
  providers: [ProductsService],
  ...
})
...
```

src/app/app.module.ts

Notatki

Displaying the list of products

```
<table class="table">
  <tr>
    <ng-template ngFor let-header [ngForOf]="headers">
      <!-- duplicated table cell -->
      <th>{{header}}</th>
    </ng-template>
  </tr>
  <ng-template ngFor let-product [ngForOf]="products">
    <!-- duplicated table row -->
    <tr>
      <td>{{product.name}}</td>
      <td>{{product.description}}</td>
      <td [innerHTML]="product.price"></td>
    </tr>
  </ng-template>
</table>
```

src/app/products-list/products-list.component.html

□

```
<table class="table">
  <tr>
    <th *ngFor="let header of headers">{{header}}</th>
  </tr>
  <tr *ngFor="let product of products">
    <td>{{product.name}}</td>
    <td>{{product.description}}</td>
    <td [innerHTML="product.price"]></td>
  </tr>
</table>
```

src/app/products-list/products-list.component.html

Notatki

The concept of application development

Assumptions

- The application displays a list of product names in the form of an unnumbered HTML list
- The number of products displayed on the page can be changed using a slider
- Clicking on the product name results in the display of detailed information about this product
- The page title is defined by the main component, and header titles are included in the template — without communication between components

Required new ingredients

- Components:
  - displaying product details
- Services:
  - providing information about the quantity of products
  - providing information about the selected (clicked) product

Notatki

Generating component core files

```
$ ng generate component ProductDetails
installing component
  create src/app/product-details/product-details.component.css
  create src/app/product-details/product-details.component.html
  create src/app/product-details/product-details.component.spec.ts
  create src/app/product-details/product-details.component.ts
  update src/app/app.module.ts
```

Notatki

Implementation of new services

```
...
@Injectable()
export class ProductsService {
  ...
  getProductsLength(): number {
    return PRODUCTS.length
  }
  /*****************************/
  getProduct(id:number): Product {
    return PRODUCTS.find((element) => element.id == id);
  }
}
```

src/app/products-list/products.service.ts

Notatki

Displaying the list of products

Bidirectional data binding

```
...
export class ProductsListComponent implements OnInit {
  ...
  productsLength: number; // the size of the product table
  pNumber: number; // the number of products on the page
  ...
  ngOnInit(): void {
    ...
    this.productsLength = this.productsService.getProductsLength
          ();
    this.pNumber = this.productsLength;
  }
}
```

src/app/products-list/products-list.component.ts

□

```
<input ... type='range' min='1' max='{{productsLength}}'
  [(ngModel)]='pNumber'> {{pNumber}}
  <!-- What is above can be written as follows: -->
  <!-- <input ... bindon-ngModel='pNumber'> -->
  <!-- and is a syntactic sugar of such a construction:  -->
  <!-- <input [ngModel]="pNumber" (ngModelChange)="pNumber=
        $event"> -->
  <!-- and this in turn is such a construction: -->
  <!-- <input [value]="pNumber" (input)="pNumber=$event.target
        .value"> -->
<ul>
  <li *ngFor='let product of products|slice:0:pNumber'>
    <!-- Creating a link with the form: "/product/N", where 'N
          ' is a natural number (product id)-->
    <a [routerLink]="['/product',product.id]">{{product.name}}</a></
        li>
</ul>
```

src/app/products-list/products-list.component.html

□

Notatki

Displaying product details

```
import { ActivatedRoute } from '@angular/router';
...
export class ProductDetailsComponent implements
     OnInit {
  product: Product;

  constructor(private productsService:
       ProductsService, private route:
       ActivatedRoute) { }

  ngOnInit() {
    let id = +this.route.snapshot.params['id'];
    this.product = this.productsService.getProduct(id);
  }
}
```

src/app/product-details/product-details.component.ts

```
<h2>{{product.name}}</h2>
<table class="table">
  <tr>
    <th>Opis</th>
    <th>Cena</th>
  </tr>
  <tr>
    <td>{{product.description}}</td>
    <td>{{product.price}}</td>
  </tr>
</table>
```

src/app/product-details/product-details.component.ts

□

Notatki

Routing configuration

```
...
import { RouterModule }   from '@angular/router';
...
@NgModule({
  ...
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule,
    RouterModule.forRoot([
      { path: '', component: ProductsListComponent },
      { path: 'product/:id', component:
           ProductDetailsComponent }
    ])
  ],
  ...
})
...
```

src/app/app.module.ts

```
...
export class AppComponent {
  title = 'Sklep';
}
```

src/app/app.component.ts

```
<h1>{{title}}</h1>
<router-outlet></router-outlet>
```

src/app/app.component.html

Notatki

Sources I

- Extending Native DOM Elements with Web Components. URL: https://blog.revillweb.com/extending-native-dom-elements-with-web-components-233350c8e86a.
- Thomas Fuchs. Scriptaculous Documentation. URL: http://madrobby.github.com/scriptaculous/.
- Google. Angular Guide. URL: https://angular.io/docs/ts/latest/guide/.
- Paweł Grzesiak. Ajax w kilka minut. URL: http://internetmaker.pl/artykul/723,1,ajax_w_kilka_minut.html.
- Patrick Hunlock. Functional Javascript. URL: http://www.hunlock.com/blogs/Functional_Javascript.
- MongoDB Inc. The MongoDB Manual. URL: http://docs.mongodb.org/manual/.
- Joyent. Node.js Manual & Documentation. URL: http://nodejs.org/api/.
- Microsoft. TypeScript handbook. URL: http://www.typescriptlang.org/Handbook.
- Mozilla. AJAX. URL: https://developer.mozilla.org/pl/AJAX.
- mozilla.org. JavaScript Guide. URL: https://developer.mozilla.org/en/JavaScript/Guide.
- Valerio Proietti. MooTools API Documentation. URL: http://mootools.net/docs/core.

Notatki

Sources II

- jQuery Project. jQuery API. URL: http://api.jquery.com/.
- Basarat Ali Syed. TypeScript Deep Dive. URL: http://basarat.gitbooks.io/typescript/.
- What are HTML Custom Elements. URL: https://medium.com/recraftrelic/what-are-html-custom-elements-c6ffea9c4244.
- wikibooks.org. JavaScript. URL: http://en.wikibooks.org/wiki/JavaScript.
- Wikipedia. URL: http://pl.wikipedia.org/.

Notatki