

Język JavaScript

Dr inż. Stanisław Polak

Grupa Systemów Komputerowych

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



Dr inż. Stanisław Polak

1

Notatki



Plan wykładu

► Podstawy języka JavaScript

Wstęp

Typy danych

Operatory

Instrukcje

Funkcje

- Obiektowy model dokumentu HTML
- Komponenty Webowe
- Podstawowe zagadnienia związane z usługą WWW
- Środowisko uruchomieniowe NodeJS

► Framework „Express.js”

Wprowadzenie

Podstawy

Obsługa protokołu HTTP

- Technika „AJAX” oraz interfejs Fetch API
- Biblioteka „jQuery”
- Podstawy języka TypeScript

► Framework „Angular”

Podstawy

Aplikacja „Hello World”

Aplikacja „Sklep”

Notatki



Dr inż. Stanisław Polak

2

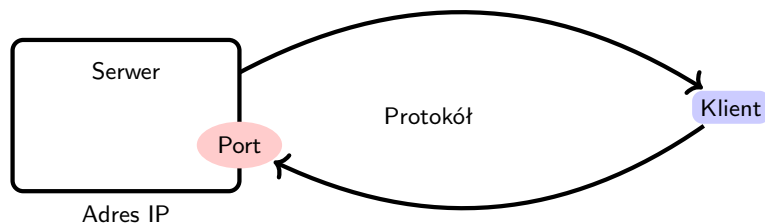


- ▶ Wykłady — co dwa tygodnie — przygotowanie do danego ćwiczenia
- ▶ Ćwiczenia laboratoryjne — co dwa tygodnie, przez cały semestr
- ▶ Ocena końcowa obliczana na podstawie ilości punktów za wykonanie:
 - ▶ Zestawu zadań programistycznych, przeznaczonych do wykonania na zajęciach
 - ▶ Zestawu zadań domowych — termin oddania: następne zajęcia
- ▶ Strona przedmiotu:
<https://www.icsr.agh.edu.pl/~polak/jezyki/js/>
- ▶ Strona ćwiczeń laboratoryjnych:
<https://polak.icsr.agh.edu.pl/>

Plan ćwiczeń

1. CSS3 oraz tworzenie stron responsywnych
2. JavaScript — typy danych, tworzenie grafik 2D
3. Standard DOM
4. Serwer „Node.js”
5. Framework Express.js — podstawy
6. Technologia AJAX
7. Biblioteka programistyczna „jQuery”

Model klient-serwer



Ogólne zasady tworzenia poprawnych stron WWW

- ▶ Formatowanie ≠ informacja
- ▶ Najprostsze środki
 - ▶ HTML + CSS
 - ▶ HTML + CSS + JavaScript

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>Hello World</title>
6     <link rel="stylesheet" href="main.css"/>
7   </head>
8   <body>
9     <main>
10      <h1>Hello World</h1>
11      <span class="name">SP</span> was here.
12    </main>
13    <footer>Stanisław Polak</footer>
14  </body>
15 </html>

```

HTML

```

1 main, footer {
2   display: block;
3   background: gray;
4   padding: 10px;
5   margin: 1px;
6 }
7 .name {font-family: arial, verdana, sans-serif;}

```

main.css

Notatki



Dr inż. Stanisław Polak

5

Podstawy języka JavaScript

Wstęp

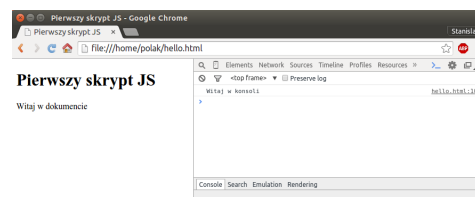
Pierwszy skrypt JS

```

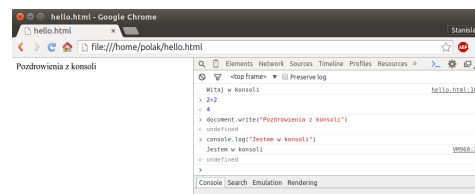
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Pierwszy skrypt JS</title>
5   </head>
6   <body>
7     <h1>Pierwszy skrypt JS</h1>
8     <script>
9       document.write("Witaj w dokumencie");
10      console.log("Witaj w konsoli");
11    </script>
12  </body>
13 </html>

```

Dokument HTML z treścią skryptu JS



Rysunek: Wynik wykonania skryptu JS po zakończeniu renderowania strony WWW



Rysunek: Korzystanie z konsoli JS

Notatki



Dr inż. Stanisław Polak

8

Osadzanie kodu JS

```

1 <!-- <script type="text/javascript;version=x.x" -->
2 <script type="text/javascript">
3 <!--
4     Treść skryptu JavaScript
5     The content of the JavaScript script
6 -->
7 </script>
8 <noscript>
9 Twoja przeglądarka nie obsługuje JavaScript!  

10 Your browser does not support JavaScript!
11 </noscript>_

```

Skrypt wewnętrzny osadzony w HTML

```
1 <script src="URL"></script>
2 <noscript>
3 Twoja przeglądarka nie obsługuje JavaScript!<br>
4 Your browser does not support JavaScript!
5 </noscript>_
```

Odwołanie do skryptu zewnętrznego

Notatki

Dr inż. Stanisław Polak

9



Kolejność wykonywania skryptów

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5 <script>
6 function info(arg){
7     var img = document.getElementsByTagName("img");
8     var names="";
9     var img_msg = '\nThe "img" element is unavailable\n'
10    if(img.length != 0)
11        img_msg = '\nThe "img" element is available\n';
12    if(document.body){
13        x=document.body.childNodes
14        for (i=0;i<x.length;i++){
15            if (x[i].nodeType == 1)
16                names+=x[i].nodeName+ " , ";
17        }
18        console.log(arg+img_msg+"document.body" contains elements: "+names);
19    }
20    else
21        console.log(arg+img_msg+"document.body" = NULL");
22 }
23 </script>
24 <script>info("Script 1");</script>
25 <script src="a.js"></script>
26 </head>
27 <body onLoad="info('Script 4');">
28 <script>info("Script 2");</script>
29 <script src="b.js"></script>
30 
31 <script>info("Script 3");</script>
32 <script src="c.js"></script>
33 </body>
34 </html>

```

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

a.js, b.js, c.js

Wypisze:

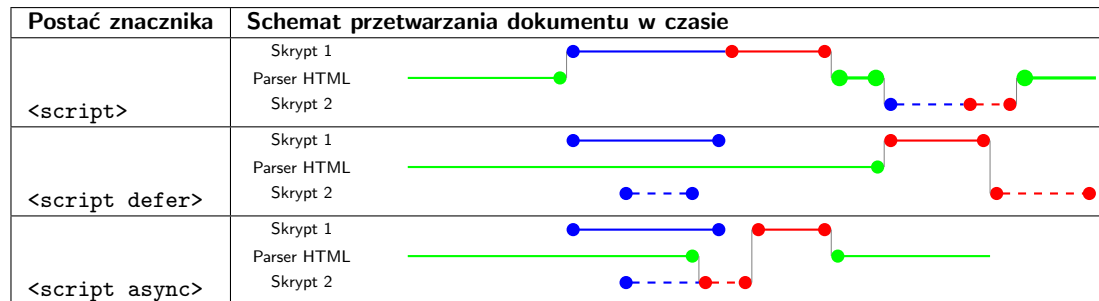
1. Script 1
The 'imagelement is unavailable
'document.body'>NULL
2. a.js
The 'imagelement is unavailable
'document.body'>NULL
3. Script 2
The 'imagelement is unavailable
'document.body' contains elements: SCRIPT,
b.js
4. The 'imagelement is unavailable
'document.body' contains elements: SCRIPT, SCRIPT,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

Notatki



Asynchroniczne lub odroczone wykonywanie skryptów zewnętrznych

Notatki



Parsowanie Pobieranie Wykonywanie

Źródło: <http://peter.sh/experiments/asynchronous-and-deferred-javascript-execution-explained/>

Definiowanie zmiennych

Notatki

```
1 var x=42;
2 //or
3 y = 42; // Not recommended
4 var _y = 42;
5 var $if=42;
6 var r6za = 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
```



Definiowanie stałych

```
1 const PI = 3.1415926;
2 console.log(PI); // 3.1415926
3 console.log(typeof(PI)); // number
4 const PI = 3.14; // redeclaration of const PI
5 PI="3.24" // An attempt to overwrite a constant value
6 console.log(PI); // 3.1415926 that is, the attempt to overwrite failed
7 console.log(typeof(PI)); // number
8
9 var PI = 3.14 // redeclaration of const PI
10 var zmienna=1;
11 const zmienna=1; // redeclaration of var zmienna
```

Notatki



Typy specjalne

Typ „null”

```
1 var empty=null;
2 console.log(typeof(empty)); // object
3
4 var empty1 = NULL; //NULL is not defined
5
6 if(empty)
7   console.log("true");
8 else
9   console.log("false");
10 // false
11
12 console.log(empty-1); // -1
```

Typ „undefined”

```
1 console.log(typeof(abc)); // undefined
2
3 var def;
4 console.log(typeof(def)); // undefined
5
6 function f(arg){
7   console.log("arg="+arg)
8 }
9
10 var result = f() // arg=undefined
11 console.log(result) // undefined
12
13
14 if(def === undefined)
15   console.log("Undefined");
16 else
17   console.log("Defined");
18 // Undefined
19
20 if(def)
21   console.log("true");
22 else
23   console.log("false");
24 // false
25
26 console.log(def-1); // NaN
```

Notatki



Typy proste i ich obiektowe odpowiedniki

Typ prosty	Obiektowy odpowiednik (Prototyp)
boolean	Boolean
number	Number
string	String

Notatki



Typ „boolean”

Notatki



```
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.toFixed(3)) //Implicit conversion to the type (prototype) 'Number' => will write:
    1.99
```

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

Konwersja do typu „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



Typ „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



Konwersja do typu „number”

Notatki



Typ „string”

Notatki



```
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_
```

```
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"
```

Typ „string”

Szablony napisów

Nieoznakowane

```
1 var a = 2;
2 var str = 'Variable 'a' has value ${a}, 2+2=${2+2}\n';
3 console.log(str);
4 /*****/
5 var str = 'Line 1
6 Line 2';
7 console.log(str);
```

Oznakowane (otagowane)

```
1 function tag(strings, val1, val2) {
2   result = "String 1:\t\t'" + strings[0] + "'\n";
3   result += "Raw string 1:\t'" + strings.raw[0] + "'\n";
4   result += "Value 1:\t\t'" + val1 + "'\n";
5   result += "String 2:\t\t'" + strings[1] + "'\n";
6   result += "Raw string 2:\t'" + strings.raw[1] + "'\n";
7   result += "Value 2:\t\t'" + val2 + "'\n";
8   return result;
9 }
10 /*****/
11 var a = 2;
12 var b = 3;
13 str = tag`a+b=\t${a+b}\n, a*b=${a*b}`;
14 console.log(str);
```

Notatki

Typ „string”

Sekwencje specjalne

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

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

Przykład użycia

Notatki

Konwersja do typu „string”

```
1 var dead = true;
2 console.log(typeof(dead)); // boolean
3 var lancuch=dead.toString();
4 console.log(typeof(lancuch)); // string
5 console.log(lancuch); // true
6 var liczba = 0xFF;
7 console.log(0xFF.toString()); // "255"
8 liczba = 11;
9 console.log(liczba.toString()); // "11"
10 liczba = 11.9;
11 console.log(liczba.toString()); // "11.9"
12
13 var liczba=255;
14 console.log(liczba.toString(2)); // 11111111
15 console.log(liczba.toString(4)); // 3333
16 console.log(liczba.toString(8)); // 377
17 console.log(liczba.toString(16)); // ff
18
19 console.log(String(null)); // "null"
20 console.log(String(undefined)); // "undefined"
21 console.log(String(false)); // "false"
22 console.log(String(true)); // "true"
23 console.log(String(255)); // "255"
24 console.log(String(3.14)); // "3.14"___
```

Notatki



Typ „symbol”

```
1 symbol1 = Symbol();
2 symbol2 = Symbol();
3 console.log(typeof(symbol1)); //symbol
4 console.log(symbol1 == symbol2) //false
5
6 symbol3 = Symbol('Symbol description');
7 symbol4 = Symbol('Symbol description');
8 console.log(symbol3); //Symbol(Symbol description)
9 console.log(symbol4); //Symbol(Symbol description)
10 console.log(symbol3 == symbol4); //false
11
12 symbol5 = Symbol.for("symbol3");
13 symbol6 = Symbol.for("symbol3");
14 console.log(symbol5); //Symbol(symbol3)
15 console.log(symbol6); //Symbol(symbol3)
16 console.log(symbol5 == symbol6); //true
17
18 var symbol7 = Symbol.for("uid");
19 console.log(Symbol.keyFor(symbol7)); // "uid"
20 var symbol8 = Symbol.for("uid");
21 console.log(Symbol.keyFor(symbol8)); // "uid"
22 var symbol9 = Symbol("uid");
23 console.log(Symbol.keyFor(symbol9)); // undefined
```

Notatki



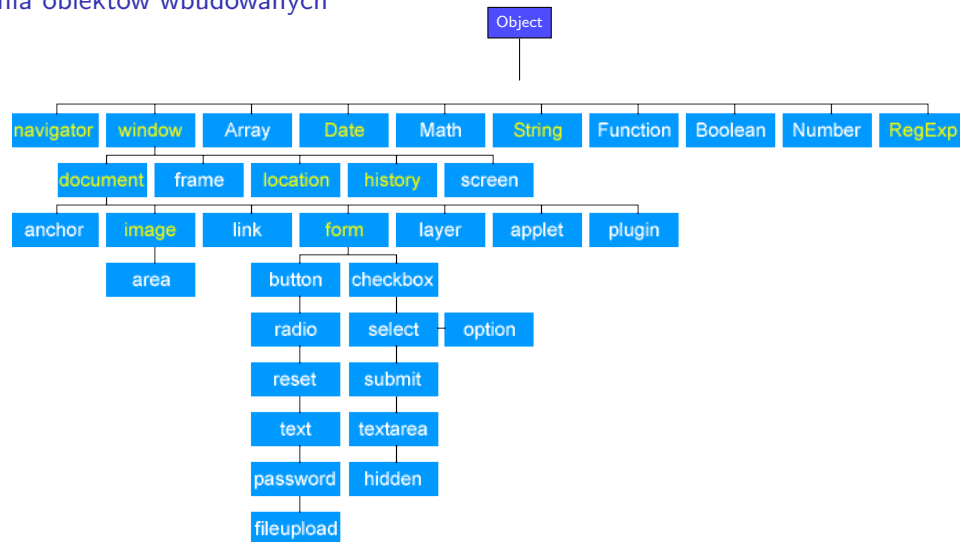
Obiekty

```
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



Hierarchia obiektów wbudowanych



Notatki

Źródło: http://www.visualtech.ca/javascript/javascript_object_model.php



Obiekt „Array”

```
1 var tab1 = new Array(1,2,3); //equivalent to: var tab1=Array(1,2,3)
2 var tab2a = new Array(10); //equivalent to: var tab2a = []; tab2a.length=10;
3 var tab2b = new Array("10");
4 var tab3 = [4,'abc',6];
5
6 console.log(tab1.length); // 3
7 console.log(tab2a.length); // 10
8 console.log(tab2b.length); // 1
9 console.log(tab3.length); // 3
10 console.log(tab1[0]); // 1
11 console.log(tab1.0); //missing ) after argument list
12 console.log(tab2a[0]); // undefined
13 console.log(tab2b[0]); // 10
14 console.log(tab3[1]); // abc
15 console.log(tab3[5]); // undefined
16
17 var tab4 = new Array(new Array(1,null,undefined),new Array('A','B','C'),new Array('x','y','z'));
18 console.log(tab4.length); // 3
19 console.log(tab4[0]); // 1,,
20 console.log(tab4[0][1]); // null
21 var tab1 = new Array(1,2,3);
22 var tab2 = [4,5,6];
23 console.log(tab1); // 1, 2 3
24 console.log(tab1.reverse()); // 3, 2, 1
25
26 var tab3 = tab1.concat(tab2);
27 console.log(tab3); // 1, 2, 3, 4, 5, 6
28 console.log(tab3.splice(1,2)); // 2, 3__
```

Notatki



Obiekt „Map”

```
1 var map = new Map();
2 emptyObject = {};
3 map.set("string","The value associated with the string");
4 map.set(1,{a:10});
5 map.set(emptyObject,1);
6
7 console.log(map); //Map { string: "The value associated with the string", 1: Object, Object: 1 }
8 console.log(map.get(1)); //Object { a: 10 }
9 console.log(map.get(2)); //undefined
10 console.log(map.get("string")); //"The value associated with the string"
11 console.log(map.get({})); //undefined
12 console.log(map.get(emptyObject)); //1
13 console.log(map.size); //3
14 map.delete("string");
15 console.log(map.size); //2
16
17 //Iteration of the hash
18 map.forEach((value,key,map) => {console.log("map["+key+"]="+value)});
19 /*
20 "map[1]=[object Object]"
21 "map[[object Object]]=1"
22 */
23 //Conversion of the array into hash
24 var tab = [{"key1","String"},[{"key2",5}]];
25 map = new Map(tab);
26 console.log(map); //Map { key1: "String", key2: 5 }
```

Notatki



Obiekt „WeakMap”

Notatki

Różnice w stosunku do „Map”

- ▶ Przechowuje słabe odniesienia do klucza
- ▶ Kluczami mogą być tylko i wyłącznie obiekty
- ▶ Klucze nie są przeliczalne
- ▶ Nie można iterować tego obiektu — patrz linia 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
```



Obiekty „Set” oraz „WeakSet”

Notatki

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+"]="+
8   value)});
9
10 /*
11  "set[napis]=napis"
12  "set[[object Object]]=[object Object]"
13  */
14 set.delete("string")
15 console.log(set.size); //1
16 set.forEach((value,kucz,set) => {console.log("set["+key+"]="+
17   +value)});
18
19 /*
20  "set [[object Object]]=[object Object]"
21  */
22 //*****
23 set = new WeakSet();
24 obj1 = {};
25 obj2 = obj1;
26 set.add(obj1);
27 set.add(obj2);
28 console.log(set.has(obj1));
29 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
7   from our Weakset
8 element.parentNode.removeChild(element);
9
10 //Delete the local reference
11 element = null;
12 console.log(set.has(element)); // false
```



Obiekt „RegExp”

```
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
```

Sprawdzanie poprawności formatu numeru telefonu

Notatki



Obiekt „Function”

```
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



Obiekt „Math”

```
1 ...
2 <canvas id="canvas" width="400" height="100">
3 Your browser does not support the "canvas" element
4 </canvas>
5
6 <script>
7 var canvas = document.getElementById("canvas");
8 if (canvas.getContext) {
9   var ctx = canvas.getContext('2d');
10  var ox = 0, oy = 50;
11  var t_min = 0, t_max = 10*Math.PI;
12  var scale = 20, step = 200, inc = t_max/step;
13
14  ctx.beginPath();
15  for (var t=t_min; t<=t_max; t+=inc){
16    y = scale * Math.sin(t);
17    x = (t / t_max) * canvas.width;
18    ctx.lineTo(ox+x, oy-y);
19  }
20  ctx.stroke();
21 </script>
22 ...
```

Rysowanie wykresu funkcji $y = \sin(x)$ dla $x \in [0, 10\pi]$



Notatki

Obiekt „Date”

```
1 function JSClock() {
2   var time = new Date()
3   var hour = time.getHours()
4   var minute = time.getMinutes()
5   var second = time.getSeconds()
6   var temp = "" + ((hour > 12) ? hour - 12 : hour)
7   if (hour == 0)
8     temp = "12";
9   temp += ((minute < 10) ? ":0" : ":") + minute
10  temp += ((second < 10) ? ":0" : ":") + second
11  temp += (hour >= 12) ? " P.M." : " A.M."
12  return temp
13 }
14 document.write(JSClock()); // 2:21:47 P.M. ↵
```

Wypisanie aktualnego czasu w formacie 12-godzinnym

Notatki

Obiekt „navigator”

```
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")
```

Rozpoznawanie wersji przeglądarki IE

Notatki

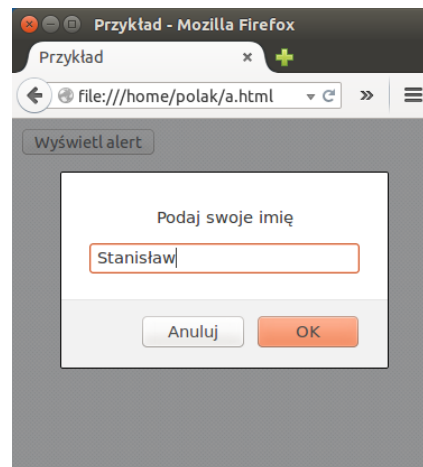


Obiekt „window”

Metody prompt() oraz alert()

```
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>
```

Otwieranie okna wprowadzania / wyprowadzania danych



Notatki



Obiekt „window”

Metody `setTimeout()` oraz `clearTimeout()`

```
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>_
```

Po upływie 2 sekund wyświetla się okienko alertu

Notatki



Obiekt „window”

Metody `setInterval()` oraz `The clearInterval()`

```
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>_
```

Co sekundę wypisywany jest komunikat na konsoli

Notatki



Obiekt „window”

Metody `requestAnimationFrame()` oraz `cancelAnimationFrame()`

```
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     requestID=window.requestAnimationFrame(display);
8 }
9
10 function display(){
11     console.log("Hello World!");
12     requestID=window.requestAnimationFrame(display);
13 }
14
15 function stop_cyclic_execution(){
16     window.cancelAnimationFrame(requestID);
17 }
18 </script>
19 </head>
20 <body>
21 <form>
22 <button type="button" onClick="start_cyclic_execution()">Start</button><br>
23 <button type="button" onClick="stop_cyclic_execution()">Stop</button>
24 </form>
25 </body>
26 </html>_
```

Cyklicznie, podczas odświeżania ekranu, wypisywany jest komunikat

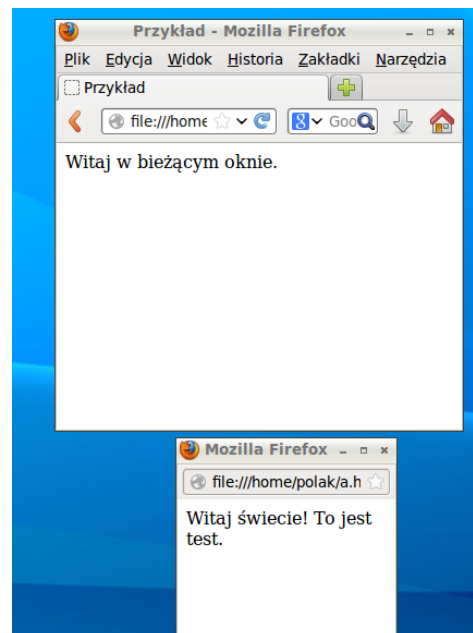


Notatki

Obiekt „document”

```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html;
4     charset=UTF-8">
5 <title>Przykład</title>
6 </head>
7 <body>
8 Witaj
9 <script>
10 var newWindow=window.open('', '', 'toolbar=no,
11     scrollbars=no, width=200, height=150');
12 newWindow.document.open("text/html", "replace");
13 newWindow.document.writeln("Witaj świecie!");
14 newWindow.document.write("To jest test.");
15 newWindow.document.close();
16 document.write(" w bieżącym oknie.");
17 </script>
18 </body>
19 </html>_
```

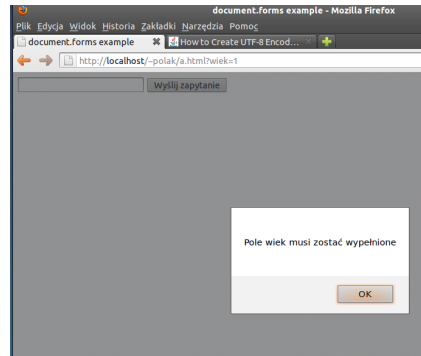
Wypisanie tekstu w nowym oknie



Notatki

Obiekt „form”

```
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>
```



Wstrzymanie wysyłki formularza

Notatki

Obiekt „Image”

```
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 </a>
18 </body>
19 </html>_
```

Wstępne ładowanie dużego obrazka

Notatki

Obiekt „location”

Notatki



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

Przekierowanie do bezpiecznego protokołu HTTPS

Obiekt „history”

Notatki



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

Obiekt „screen”

Notatki

```
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 }~
```

Operatory „odziedziczone” z języka Java

Notatki

Dokładnie takie same jak w Java

- ▶ Arytmetyczne (+, -, *, /, ++, --, %)
- ▶ Warunkowy (?:)
- ▶ Bitowe (&, |, ^, ~, <<, >>, >>>)
- ▶ Logiczne (&&, ||, !)
- ▶ Łańcuchowy (+)
- ▶ Przypisania (=, *=, /=, %=, +=, -=, <<=, >>=, >>>=, &=, ^=, |=)
- ▶ Typu (instanceof)
- ▶ new oraz this

Prawie takie same jak w Java

- ▶ Porównania — dodatkowe: === oraz !==

```
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
```

Dodatkowe operatory porównania

Przecinkowy

```
1 a=1 , b=2;
2 document.write(a); // 1
3 document.write(b); // 2
4 for (var i = 0, j = 9; i <= 9; i++, j--)
5     document.writeln("a[" + i + "][" + j + "]); /* Output:
6 a[0][9]
7 a[1][8]
8 a[2][7]
9 a[3][6]
10 a[4][5]
11 a[5][4]
12 a[6][3]
13 a[7][2]
14 a[8][1]
15 a[9][0]
16 */
```

Notatki



The in operator

```
1 var trees = new Array("redwood", "cedar", "oak", "maple");
2 0 in trees; // true
3 3 in trees; // true
4 6 in trees; // false
5 "maple" in trees; // false (the index number should be given, not the value after the given index)
6 "length" in trees; // true ('length' is the property of the object 'Array')
7
8 //Predefined objects
9 "PI" in Math; // true
10 var myString = new String("coral");
11 "length" in myString; // true
12
13 // User objects
14 var car = {brand: "Honda", model: "Accord", year: 1998};
15 "brand" in car; // true
16 "model" in car; // true
```

Notatki



The delete operator

```
1 x = 42;
2 var y = 43;
3 obj = new Number();
4 obj.h = 4;
5 delete x;           // returns true (you can delete a variable if it is implicitly defined)
6 delete y;           // returns false (the variable can not be deleted if it is defined using 'var')
7 delete Math.PI;      // returns false (you can not delete predefined properties)
8 delete obj.h;        // returns true (you can delete user-defined properties)
9 delete obj;          // returns true (you can delete a variable if it is implicitly defined)
10
11 var trees = new Array("redwood", "cedar", "oak", "maple");
12 document.write(trees.length); // 4
13 document.write(trees[2]);      // oak
14 delete trees[2];
15 document.write(trees.length); // 4
16 document.write(trees[2]);      // undefined
17
18 trees[3] = undefined;
19 if (2 in trees)
20     document.write("The element with index 2 exists in the table");
21 if (3 in trees)
22     document.write("The element with index 3 exists in the table");
23 //Output: The element with index 3 exists in the table__
```

Notatki



The typeof operator

```
1 var myFun = new Function("5 + 2");
2 var shape = "round";
3 var size = 1;
4 var today = new Date();
5
6 typeof myFun;           // returns "function"
7 typeof(shape);          // returns "string"
8 typeof size;            // returns "number"
9 typeof today;           // returns "object"
10 typeof does_not_exists; // returns "undefined"
11 typeof true;            // returns "boolean"
12 typeof null;            // returns "object"
13 typeof 62;              // returns "number"
14 typeof 'Hello world';   // returns "string"
15 typeof Math;            // returns "object"
16 typeof Array;           // returns "function"
```

Notatki



The void operator

Notatki



Dr inż. Stanisław Polak

52

Podstawy języka JavaScript

Instrukcje

Instrukcja for ... in

Notatki

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

Iterowanie obiektu

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

Iterowanie tablicy — niewskazane



Dr inż. Stanisław Polak

54

Instrukcja for ... of

```
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



Instrukcja with

Notatki



Obsługa wyjątków

Instrukcje throw, try, catch oraz finally

```
1 //Creating an object representing an exception
2 function Exception (message) {
3     this.message = message;
4     this.name     = "Negative";
5 }
6
7 try{
8     var age = -1;
9     if(age <0){
10         var exception=new Exception("Age can not be a negative number");
11         throw exception;
12     }
13     console.log("It will not be written anymore");
14 }
15 catch (e if e.name == "Negative") { console.log(e.message);}
16 catch (e) { /* handling exceptions that were not previously captured */}
17 finally{ /*instructions to be always done */}
```

Notatki



Dr inż. Stanisław Polak

57

Podstawy języka JavaScript

Funkcje

Funkcje

- ▶ Definiowanie (inaczej niż w Java) — za pomocą słowa kluczowego function; wywoływanie funkcji oraz zwracanie wartości przez funkcję — tak samo jak w Java
- ▶ Pozwalają definiować prototyp obiektów

```
1 function multiply(a, b=1){
2     var c = a * b;
3     a = 0;
4     b = 0;
5     return c;
6 }
7 function f() { return [1, 2, 3] }
8
9 a = 2;
10 b = 2;
11 var result = multiply(a,b);
12 console.log(result); // 4
13 var result = multiply(a);
14 console.log(result); // 2
15
16 var x, y, z;
17 [x, y, z] = f(); // Returning many values
18 [x, y, z] = (function f() { return [1, 2, 3] })(); //
19 // Simultaneous defining and calling functions
20 console.log(x); // 1
21 //*****
22 const constant = 1;
23 var variable = 2;
24 function constant(){} //Redeclaration of const constant
25 function variable(){}
26 variable(); //variable is not a function
```

Przykładowe funkcje

```
1 function change(x,object1,object2){
2     x = 2;
3     object1.brand = "Fiat";
4     object2 = {brand: "Skoda"};
5 }
6
7 var car1 = {brand: "Ferrari"};
8 var car2 = {brand: "Ferrari"};
9 var variable = 1;
10 console.log(variable); // 1
11 console.log(car1.brand); // Ferrari
12 console.log(car2.brand); // Ferrari
13 change(variable, car1, car2);
14 console.log(variable); // 1
15 console.log(car1.brand); // Fiat
16 console.log(car2.brand); // Ferrari
```

Przekazywanie typów prostych oraz złożonych

Notatki



Dr inż. Stanisław Polak

58

Funkcje anonimowe

```
1 //Procedural function
2 function hello1(who) {
3     return 'Hello '+who;
4 }
5 /*****/
6 console.log(hello1('world'));// "Hello world"
7 /*****/
8 /*****/
9 // Function as a variable
10 var hello2 = function (who) {return 'Hello '+who;};
11 // or
12 var hello2 = (who) => {return 'Hello '+who;};
13 // or
14 var hello2 = (who) => 'Hello '+who;
15 /*****/
16 var hello3 = function() {
17     console.log('Hello');
18     console.log('World');
19 }
20 // or
21 var hello3 = () => {
22     console.log('Hello');
23     console.log('World');
24 }
25 console.log(hello2('world'));// "Hello world"
26 hello3(); // "Hello"
27           // "World"
```

```
1 function Person() {
2     // The Person () constructor defines 'this' as an
3     // instance of itself
4     this.age = 0;
5     this.salary = 0;
6
7     setInterval(function () {
8         //Here 'this'  $\longleftrightarrow$  object 'window' that is, it is
9         //different from 'this' defined in the Person
10        //constructor
11        this.age++;
12        console.log("Age="+this.age);
13    }, 1000);
14
15    setInterval(() => {
16        this.salary++;//Here 'this' is a Person object
17        console.log("Salary="+this.salary);
18    }, 1000);
19 }
20 var person = new Person();
```

Leksykalne this

Notatki



Domknięcia

```
1 function init() {
2     var name = "Polak";
3
4     function displayName() {
5         console.log(name);
6     }
7     displayName();
8 }
9 init(); // Polak
```

```
1 function createFunction() {
2     var name = "Polak";
3
4     function displayName() {
5         console.log(name);
6     }
7     return displayName;
8 }
9
10 var myFunction = createFunction();
11 myFunction(); // Polak
```

```
1 function multiply_by(x) {
2     return function(y) {
3         /* the function uses two variables:
4         y - available to the user
5         x - defined only inside the 'multiply_by()' function
6         */
7         return x * y;
8     };
9 }
10
11 var product_5_by = multiply_by(5); //the parameter 'x' is assigned the value 5
12
13 console.log(product_5_by(12)); // will be written 5 * 12 or 60
```

Przykład użycia domknięcia

Notatki

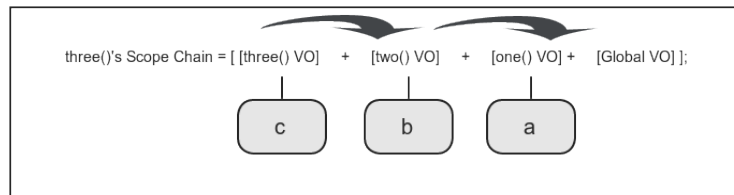


Łańcuch zasięgu

```

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();

```



Źródło: <http://davidshariff.com/blog/javascript-scope-chain-and-closures/>



Zmienne w funkcji

```

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

```

Wyrażenie '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'
7               ' block
8     var b = 11; // the same variable 'b' as above. Range - interior
9               of the 'fun' function
10    console.log(a); // 10
11    console.log(b); // 11
12  }
13  console.log(a); // 3
14  console.log(b); // 11
15 }
16 fun();

```

'let' kontra 'var'



Funkcje ze zmienną liczbą argumentów

```

1 function write() {
2   // go after all arguments
3   for (var i=0; i<arguments.length; i++)
4     console.log(arguments[i]);
5 }
6
7 write("A","B","C");

```

```

1 function write(a, ...others) {
2   console.log(a);
3   console.log(Array.isArray(others));
4   for(counter in others)
5     console.log("others["+counter+"]="+others[
6       counter]);
7   console.log(arguments.length); //SyntaxError: '
8     arguments' object may not be used in
9     conjunction with a rest parameter
10 }
11 write("A","B","C");

```

Notatki

Document Object Model

Ogólna charakterystyka

- ▶ Document Object Model (DOM) — obiektowy model dokumentu
- ▶ Dokument — drzewo obiektów
- ▶ Interfejs programowy (API) dla dokumentów HTML i XML
- ▶ Zbiór własności i metod do manipulacji w/w dokumentami

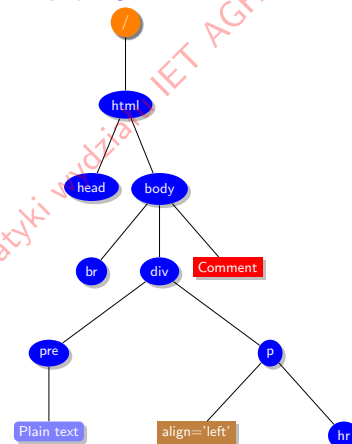
```

1 <html>
2 <head></head>
3 <body>
4 <br>
5 <div>
6 <pre>Plain text</pre>
7 <p align='left'>
8 <hr/>
9 </p>
10 </div>
11 <!-- Comment -->
12 </body>_
13 </html>

```

Dokument HTML

Drzewo DOM



Notatki

Podstawowe własności węzłów

Typ węzła	nodeType	nodeName	nodeValue
Element	Node.ELEMENT_NODE (1)	Nazwa znacznika dużymi literami	null
Atrybut	Node.ATTRIBUTE_NODE (2)	Nazwa atrybutu	Wartość atrybutu
Text	Node.TEXT_NODE (3)	#text	Tekst stanowiący jego treść
Komentarz	Node.COMMENT_NODE (8)	#comment	Treść komentarza
Dokument	Node.DOCUMENT_NODE (9)	#document	null

Notatki



Wyświetlenie informacji o pojedynczym węźle element

Metody getElementById() / querySelector()

Przykłady

Notatki



```
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>_
```

Wyświetlenie informacji o kilku węzłach element

Metody `getElementsByName()` / `querySelectorAll()`

Notatki

```
1 <html>
2   <head>
3     <title>An example</title>
4     <script>
5   function start(){
6     var elements = document.getElementsByTagName("td");
7     # or
8     var elements = document.querySelectorAll("tr td");
9     window.alert(elements);           // [object HTMLCollection]
10    window.alert(elements.length);     // 4
11    window.alert(elements[0]);         // [object HTMLTableCellElement]
12    window.alert(elements[0].nodeType); // 1
13    window.alert(elements[0].nodeName); // TD
14    window.alert(elements[0].nodeValue); // null
15    window.alert(elements[1]);         // [object HTMLTableCellElement]
16    window.alert(elements[1].nodeType); // 1
17    window.alert(elements[1].nodeName); // TD
18    window.alert(elements[1].nodeValue); // null
19  }
20  </script>
21 </head>
22 <body onLoad="start();">
23   <table>
24     <tr><td>a</td><td>b</td></tr>
25     <tr><td>c</td><td>d</td></tr>
26   </table>
27 </body>
28 </html>_
```



Pobieranie potomków węzłów element oraz tekst

Własność `childNodes`

Notatki

```
1 <html>
2   <head>
3     <title>An example</title>
4     <script>
5   function start(){
6     var elements = document.getElementsByTagName("tr");
7     for(var i=0 ; i<elements.length ; i++){
8       var descendants = elements[i].childNodes; // the type of 'descendants' object is "NodeList"
9       for (var j=0;j<descendants.length;j++){
10        var string = descendants[j].nodeName + ": " + descendants[j].childNodes[0].nodeValue;
11        window.alert(string);
12      }
13    }
14  }
15  /* Output:
16  TD: a
17  TD: b
18  TD: c
19  TD: d
20  */
21  </script>
22 </head>
23 <body onLoad="start();">
24   <table border="1">
25     <tr><td>a</td><td>b</td></tr>
26     <tr><td>c</td><td>d</td></tr>
27   </table>
28 </body>
29 </html>_
```



Obsługa atrybutów

Własność `attributes` oraz metody `setAttribute()` i `removeAttribute()`

```

1 <html>
2 <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4   <title>An examples</title>
5   <script>
6     function change(thickness){
7       var element = document.getElementById("elem1"); // "HTMLTableElement" type object
8       window.alert(element.getAttribute('border')); // 1
9       window.alert(element.getAttribute('id')); // elem1
10      element.setAttribute('border',thickness);
11    }
12    //You can do it anyway
13    var attributes = element.attributes; //obiekt typu "NamedNodeMap" type object
14    window.alert(attributes.border.value); // 1
15    window.alert(attributes.id.value); // elem1
16    attributes.border.value = thickness;
17  }
18
19  function delete(){
20    var element = document.getElementById("elem1");
21    element.removeAttribute('border');
22  }
23  </script>
24 </head>
25 <body>
26   <table border="1" id="elem1">
27     <tr><td>a</td><td>b</td></tr>
28     <tr><td>c</td><td>d</td></tr>
29   </table>
30   <form>
31     <input type="button" value="Change the thickness" onClick="change(2);">
32     <input type="button" value="Delete" onClick="delete();">
33   </form>
34 </body>
35 </html>

```



Notatki

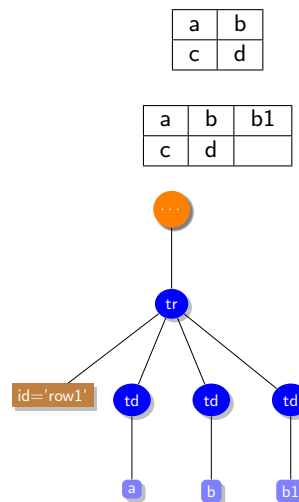
Wstawienie nowej komórki tabeli na końcu wiersza

Metody `createElement()`, `createTextNode()` oraz `appendChild()`

```

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      element.appendChild(newTD);
11    }
12  </script>
13 </head>
14 <body>
15   <table border="1">
16     <tr id="row1"><td>a</td><td>b</td></tr>
17     <tr><td>c</td><td>d</td></tr>
18   </table>
19   <form>
20     <input type="button" value="Insert" onClick="insert();">
21   </form>
22 </body>
23 </html>

```

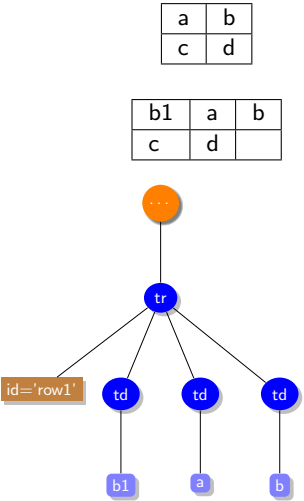


Notatki

Wstawienie nowej komórki tabeli na początku wiersza

Metoda insertBefore()

```
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>_
```



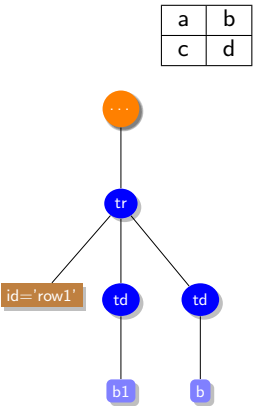
Notatki



Zastąpienie komórki tabeli

Metoda replaceChild()

```
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     >
24   </form>
25 </body>
26 </html>_
```



Notatki



Dostęp do stylów CSS

Obiekt style

Notatki

- Cecha CSS → własność obiektu style
 - background-color → style.backgroundColor
 - border-top-width → style.borderTopWidth
 - Wyjątki:
 - 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>

```



Zmiana koloru tła po kliknięciu komórki tabeli

Obsługa zdarzeń

Notatki

```

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>

```



Komponenty webowe

Elementy składowe

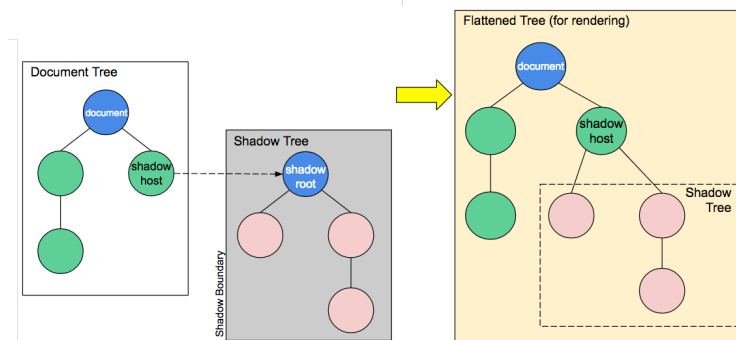
- ▶ Shadow DOM
- ▶ Własne elementy
- ▶ Szablony

Etapy budowania własnego komponentu

1. Stworzenie szablonu
2. Stworzenie klasy komponentu
3. Dodanie drzewa Shadow DOM
4. Rejestracja nowego elementu
5. Rozbudowa logiki klasy

Notatki

Shadow DOM



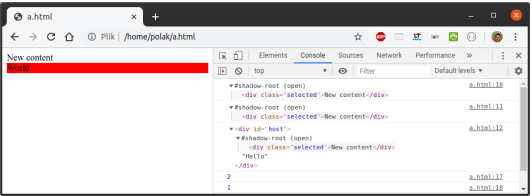
Źródło: https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM

Notatki

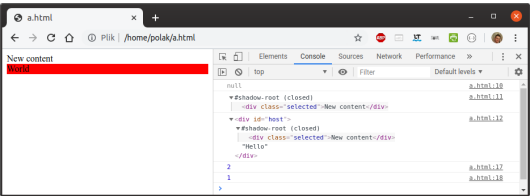
Shadow DOM

Przykład

```
1 <html>
2   <head>
3     <style>
4       .selected {background-color: red }
5     </style>
6   </head>
7   <body>
8     <div id='host'>Hello</div>
9     <div class='selected'>World</div>
10    <script>
11      var host = document.querySelector('#host');
12      var root = host.attachShadow({mode: 'closed'});
13      console.log(host.shadowRoot);
14      console.log(root);
15      console.log(root.host);
16      var div = document.createElement('div');
17      div.textContent = "New content";
18      div.className = "selected";
19      root.appendChild(div);
20      console.log(document.querySelectorAll('div').length)
21      ;
22      console.log(root.querySelectorAll('div').length);
23    </script>
24  </body>
25 </html>
```



Rysunek: Wynik wykonania skryptu



Rysunek: Wynik wykonania skryptu



Notatki

Tworzenie własnych elementów

Samoistne

```
1 ...
2 <colored-text color="red"> Hello </colored-text>
3 <colored-text color="green"> Hello </colored-text>
4
5 <script>
6   class ColoredText extends HTMLElement {
7     constructor() {
8       super();
9       const color = this.getAttribute('color');
10      // Create a shadow root
11      const shadow = this.attachShadow({mode: 'open'
12      });
13      const p = document.createElement('p');
14      p.textContent = this.textContent;
15      // Create some CSS to apply to the shadow dom
16      const style = document.createElement('style');
17      style.textContent = 'p { color: ${color} }';
18      // attach the created elements to the shadow
19      dom
20      shadow.appendChild(style);
21      shadow.appendChild(p);
22    }
23  }
24  customElements.define('colored-text', ColoredText);
25 </script>
26 ...
```

To wyświetli przeglądarka

Hello

Hello

Dostosowanie już istniejących elementów

```
1 ...
2 <p is="colored-text" color="blue">Hello</p>
3
4 <script>
5   class ColoredText extends HTMLParagraphElement {
6     constructor() {
7       super();
8       const color = this.getAttribute('color');
9       // Create a shadow root
10      const shadow = this.attachShadow({mode: 'open'
11      });
12      const p = document.createElement('p');
13      p.textContent = this.textContent;
14      // Create some CSS to apply to the shadow dom
15      const style = document.createElement('style');
16      style.textContent = 'p { color: ${color} }';
17      // attach the created elements to the shadow
18      dom
19      shadow.appendChild(style);
20      shadow.appendChild(p);
21    }
22  }
23  customElements.define('colored-text', ColoredText, extends:
24    "p");
25 </script>
26 ...
```

To wyświetli przeglądarka

Hello



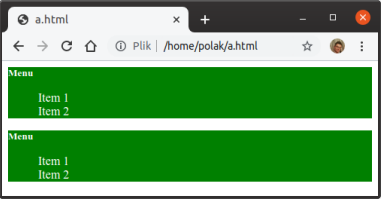
Notatki

Szablony

Element „template”

```
1 <html>
2   <body>
3     <template>
4       <style>
5         nav {
6           color: white;
7           background-color: green;
8         }
9
10        h1 {
11          font-size: smaller;
12        }
13
14        menuitem {
15          display: block;
16        }
17      </style>
18      <nav>
19        <h1>Menu</h1>
20        <menu type="list">
21          <menuitem>Item 1</menuitem>
22          <menuitem>Item 2</menuitem>
23        </menu>
24      </nav>
25    </template>
26
27    <my-menu> </my-menu>
28    <my-menu> </my-menu>
```

```
29 <script>
30   customElements.define('my-menu',
31     class extends HTMLElement {
32       constructor() {
33         super();
34         let templateContent = document.
35           querySelector('template').
36             content;
37         const shadowRoot = this.
38           attachShadow({ mode: 'open'
39             }).appendChild(
40               templateContent.cloneNode(
41                 true));
42       }
43     }
44   </script>
45 </body>
46 </html>
```



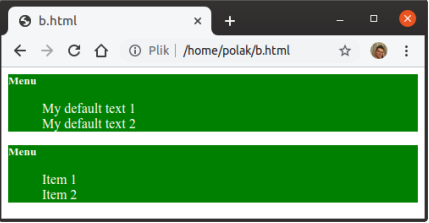
Notatki

Szablony

Element „slot”

```
1 ...
2 <template>
3   <style>
4     ...
5   </style>
6   <nav>
7     <h1>Menu</h1>
8     <menu type="list">
9       <menuitem>
10        <slot name="item1">My default text 1</slot>
11      </menuitem>
12      <menuitem>
13        <slot name="item2">My default text 2</slot>
14      </menuitem>
15    </menu>
16  </nav>
17 </template>
18
19 <my-menu> </my-menu>
20
21 <my-menu>
22   <span slot="item1">Item 1</span>
23   <span slot="item2">Item 2</span>
24 </my-menu>
```

```
25 <script>
26 ...
27 // Ta sama zawartość co na poprzednim slajdzie
28   / The same content as on the previous
29   slide
30 </script>
31 ...
```



Notatki

Model klient-serwer

Usługa WWW

Notatki



Dr inż. Stanisław Polak

81

Podstawowe zagadnienia związane z usługą WWW

Wybrane komendy (metody) protokołu HTTP

Komenda „GET”

URL żądania:

http://www.icsr.agh.edu.pl/index.html

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

Żądanie

Komenda „POST”

URL żądania: 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
```

Żądanie

```
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>
```

Odpowiedź

```
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>
```

Odpowiedź

Notatki



Dr inż. Stanisław Polak

82

Wysyłanie danych z formularza HTML

Notatki



Dr inż. Stanisław Polak

83

Podstawowe zagadnienia związane z usługą WWW

Kodowanie „application/x-www-form-urlencoded”

Przykład

Notatki



Dr inż. Stanisław Polak

84

Zatwierdzanie danych \mapsto kodowanie \mapsto wysyłanie do serwera WWW

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

- ▶ GET
- ▶ POST

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

Dokument HTML

```
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>
```

Login: Jan
Password: Kowalski (Nowak)

Zakodowane dane

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


Kodowanie „multipart/form-data”

Przykład

```
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>
```

```
1 POST /skrypt HTTP/1.0
2 Content-Length: 775
3 Content-Type: multipart/form-data; boundary=-----8152765018186645991017906692
4
5 -----8152765018186645991017906692
6 Content-Disposition: form-data; name="login"
7
8 Jan
9 -----8152765018186645991017906692
10 Content-Disposition: form-data; name="password"
11
12 Kowalski (Nowak)
13 -----8152765018186645991017906692
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 -----8152765018186645991017906692
```

Jan
Kowalski (Nowak)
image.jpg

Notatki

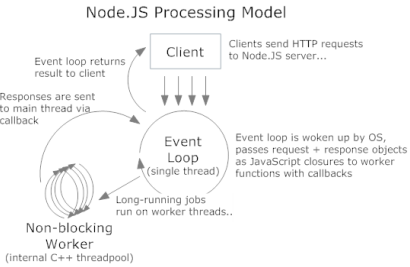


Node.js

Ogólna charakterystyka

- ▶ Udostępnia JavaScript po stronie serwera
- ▶ Używa V8 JavaScript Engine
- ▶ System do tworzenia serwisów sieciowych z asynchronicznym WE/WY
- ▶ Wykorzystuje paradygmat programowania sterowanego zdarzeniami
- ▶ Nadaje się dobrze do pisania aplikacji, które wymagają komunikacji w czasie rzeczywistym pomiędzy przeglądarką a serwerem
- ▶ Pojedyncza instancja Node.js działa jako pojedynczy wątek

Pętla zdarzeń — podmiot, który obsługuje / przetwarza zdarzenia zewnętrzne i konwertuje je na wywołania funkcji zwrotnych



Rysunek: Schemat działania pętli zdarzeń w Node.js

Notatki



Przykład „Hello World”

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

hello.js

Notatki



Wprowadzanie i wyprowadzanie danych

```
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



Dostęp do zmiennych środowiskowych, obsługa linii komend

Notatki

```
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

Dr inż. Stanisław Polak

89



Moduł

Tworzenie

Notatki

```

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 }
13 console.log(myModule()); //Error

```

script.js

```
1 $ export NODE_PATH='dir1:dir2:...:dirN'
```

```

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 |   __proto__: Module {} */
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:
25 |   [Function]
26 | }
27 | console.log(exports); //{ variable2: 2, variable4: 4, fun1:
28 |   [Function]
29 | }
30 | console.log(module.exports == exports); //true

```

node_modules/myModule.js



Obsługa pakietów

```
$ npm install nazwa_pakietu
# Moduły ↳ ./node_modules/
# Pliki wykonywalne ↳ ./node_modules/.bin/
# Manuale ↳ nie są instalowane

$ npm install --global nazwa_pakietu
# Moduły ↳ {prefix}/lib/node_modules/
# Pliki wykonywalne ↳ {prefix}/bin/
# Manuale ↳ {prefix}/share/man/
# {prefix} = np. /usr

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



Obsługa plików

```
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,
6 data) {
7 if (error) throw error;
8 console.log("Read value: "+data);
9 });
10 fs.writeFile('file.txt', '2', function (error) {
11 if (error) throw error;
12 console.log('The value 2 has been saved');
13 });
```

Wersja nieprawidłowa

```
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,
6 data) {
7 if (error) throw error;
8 console.log("Read value: "+data);
9
10 fs.writeFile('file.txt', '2', function (error) {
11 if (error) throw error;
12 console.log('The value 2 has been saved');
13 });
14 });
```

Wersja prawidłowa



Obsługa bazy danych SQLite 3

```

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



Tworzenie dodatków (addons) w C++

Przykład „Hello World”

```

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"));
12   //Specify what function returns
13 }
14
15 void init(Local<Object> exports) {
16   NODE_SET_METHOD(exports, "hello", Method); //Associate the name '
17   hello' with the above C++ method and export it
18 }
19
20 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

Instalacja programu „node-gyp”

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

Kompilacja i uruchomienie

```

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

```

```

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

```

hello.js



Obsługa protokołu HTTP

Szkielet skryptu

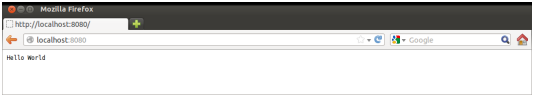
```
1 var http = require("http");
2
3 function requestListener(request, response) {
4   console.log("A request from the client has appeared");
5   response.writeHead(200, {"Content-Type": "text/plain"});
6   response.write("Hello World");
7   response.end();
8 }
9 var server = http.createServer(requestListener);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

Wersja alternatywna

Sprawdzenie działania



Rysunek: Z poziomu przeglądarki WWW



Obsługa parametrów URL

```
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
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

Na wyjściu



Obsługa formularzy

Obsługa kodowania „application/x-www-form-urlencoded”

Notatki



Wolnostojący serwer WWW

Notatki



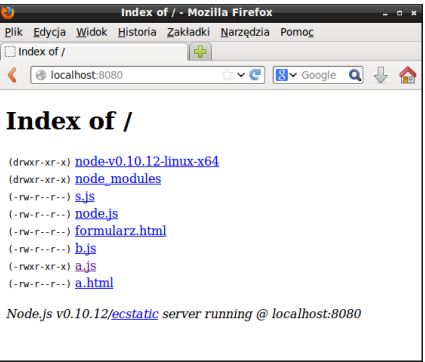
Instalacja i uruchamianie

```
$ 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
```



Tworzenie szkieletu aplikacji

```
1 $ mkdir MySite
2 $ cd MySite
3 $ vi package.json
4 $ npm install #installing dependencies
```

```
1 {
2   "name": "MySite",
3   "version": "0.0.1",
4   "private": "true",
5   "dependencies": {
6     "express": "*",
7     "pug": "*",
8     "morgan": "*"
9   }
10 }
```

package.json

Notatki



Główny plik aplikacji

```
1 var express = require('express'),
2     logger = require('morgan');
3
4 var app = express();
5 var router = express.Router();
6 app.set('views', __dirname + '/views');
7 app.set('view engine', 'pug');
8 app.use(logger('dev'));
9 app.use(express.static(__dirname + '/public'));
10 router.get('/', function (req, res) {
11   res.render('index',
12     { title : 'Przykład' }
13   );
14 });
15 app.use('/', router);
16 app.listen(3000);
```

app.js

Notatki



Pliki Pug

Notatki

```
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



Plik CSS

Notatki

```
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 }
```

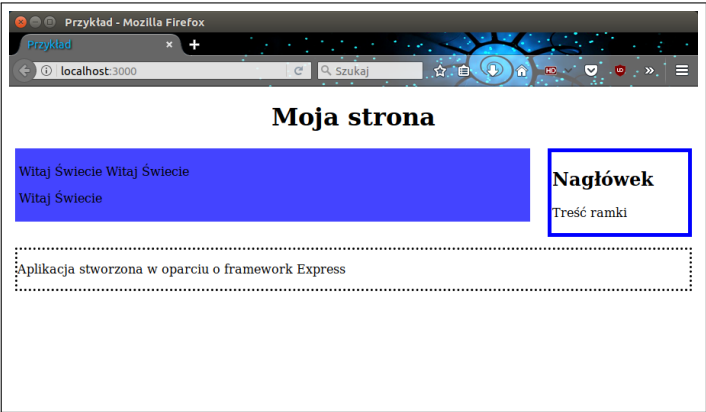
public/stylesheets/style.css



Uruchamianie aplikacji

Notatki

```
1 $ node app
2 GET / 304 847.588 ms - -
3 GET /stylesheets/style.css 304 4.478 ms - -
```



Polecenie express

Notatki

```
1 $ npm install --global express-generator
2 $ express --help
3 Usage: express [options] [dir]
4
5 Options:
6
7   -h, --help            output usage information
8   -v, --version          output the version number
9   -e, --ejs              add ejs engine support
10  --pug                  add pug engine support
11  --hbs                  add handlebars engine support
12  -H, --hogan             add hogan.js engine support
13  -v, --view <engine>    add view <engine> support (ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)
14  -c, --css <engine>     add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)
15  --git                  add .gitignore
16  -f, --force             force on non-empty directory
```



Generowanie aplikacji „Hello World”

Użycie polecenia `express`

```
1 $ express --view=pug MySite
2   create : MySite
3   create : MySite/package.json
4   create : MySite/app.js
5   create : MySite/public
6   create : MySite/routes
7   create : MySite/routes/index.js
8   create : MySite/routes/users.js
9   create : MySite/views
10  create : MySite/views/index.pug
11  create : MySite/views/layout.pug
12  create : MySite/views/error.pug
13  create : MySite/bin
14  create : MySite/bin/www
15  create : MySite/public/javascripts
16  create : MySite/public/images
17  create : MySite/public/stylesheets
18  create : MySite/public/stylesheets/style.css
19
20  install dependencies:
21    $ cd MySite && npm install
22
23  run the app:
24    $ DEBUG=mysite:* npm start
```

Notatki



Plik 'package.json'

```
1 {
2   "name": "mysite",
3   "version": "0.0.0",
4   "private": true,
5   "scripts": {
6     "start": "node ./bin/www"
7   },
8   "dependencies": {
9     "body-parser": "~1.16.0",
10    "cookie-parser": "~1.4.3",
11    "debug": "~2.6.0",
12    "express": "~4.14.1",
13    "morgan": "~1.7.0",
14    "pug": "~2.0.0-beta10",
15    "serve-favicon": "~2.3.2"
16  }
17 }
```

Notatki



Główny plik aplikacji (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



Pliki z definicją tras

```
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



Pozostałe wygenerowane pliki

```
1 extends layout.pug
2
3 block content
4   hi= title
5   p Welcome to #{title}
```

views/index.pug

```
1 doctype html
2 html
3   head
4     title= title
5     link(rel='stylesheet', href='/stylesheets/style.
      css')
6   body
7     block content
```

views/layout.pug

Notatki

```
1 body {
2   padding: 50px;
3   font: 14px "Lucida Grande", Helvetica, Arial, sans-serif;
4 }
5
6 a {
7   color: #00B7FF;
8 }
```

public/stylesheets/style.css

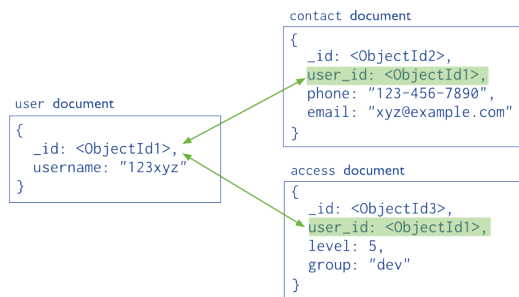
```
1 $ cd MySite && npm install
2 $ npm start
```

Instalowanie zależności i uruchamianie aplikacji



Model danych w MongoDB

Referencje



Osadzone dokumenty



Notatki

Instalowanie zależności

```
1 $ vi package.json
2 $ npm install # installing dependencies
```

```
1 ...
2 "dependencies": {
3   ...
4   "monk": "*"
5 }
6 ...
```

package.json

Notatki



Tworzenie zawartości bazy danych

```
1 $ mongo
2 MongoDB shell version: 2.6.3
3 connecting to: test
4 > use testbase1
5 switched to db testbase1
6 > db.datacollection.insert({"title":"Express & Mongo"})
7 > db.datacollection.insert({"title":"Express"})
8 > db.datacollection.find()
9 { "_id" : ObjectId("52f8bb757bade7e2c4741741"), "title" : "Express & Mongo" }
10 { "_id" : ObjectId("52f8bd407bade7e2c4741742"), "title" : "Express" }
11 > db.datacollection.find({"title":"Express"})
12 { "_id" : ObjectId("52f8bd407bade7e2c4741742"), "title" : "Express" }
```

Notatki



Modyfikacje

```

1 ...
2 var bodyParser = require('body-parser');
3 // New code
4 var monk = require('monk');
5 var db = monk('localhost:27017/testbase1');
6 ...
7 //new code
8 app.use(function(req,res,next){
9   req.db = db;
10  next();
11 });
12 //existing code
13 app.use('/', index);
14 app.use('/users', users);
15 ...

```

app.js

```

1 extends layout.pug
2
3 block content
4   h1 Tytuły
5   ul
6     each element in titlelist
7       li #{element.title}

```

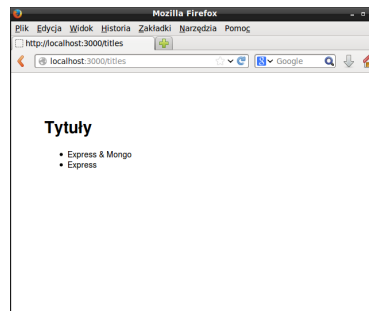
views/titlelist.pug

```

1 ...
2 router.get('/titles', function(req, res) {
3   var db = req.db;
4   var collection = db.get('datacollection');
5   collection.find({}, {}, function(e, docs){
6     res.render('titlelist', {
7       "titlelist" : docs
8     });
9   });
10 });
11
12 module.exports = router;

```

routes/index.js



Notatki

Dr inż. Stanisław Polak

117

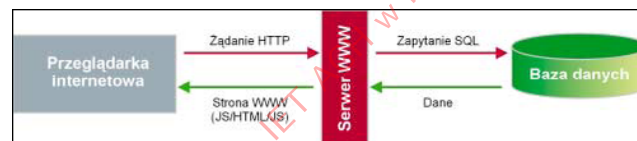
Asynchroniczne zapytania

AJAX

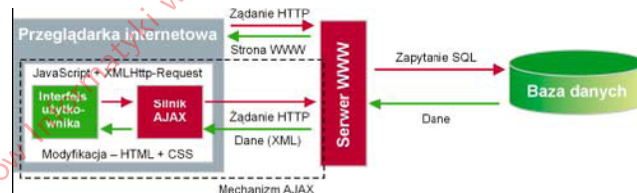
AJAX

Notatki

- ▶ **AJAX (Asynchroniczny JavaScript i XML)**
- ▶ **AJAX = HTML + CSS + DOM + XMLHttpRequest + XML + JavaScript**
- ▶ **Możliwości:**
 - ▶ Wysyłanie zapytania do serwera bez przeładowywania strony
 - ▶ Aplikacja może dokonywać szybkich, przyrostowych aktualizacji w interfejsie użytkownika bez potrzeby przeładowywania całej strony w przeglądarce
 - ▶ Parsowanie i praca z dokumentami XML
- ▶ Czy warto go zawsze stosować?



Rysunek: Schemat działania typowej witryny internetowej



Rysunek: Schemat działania witryny internetowej opartej o AJAX

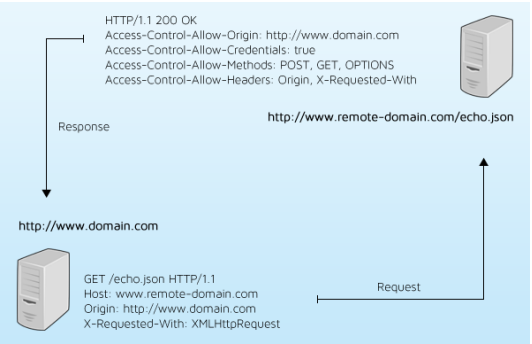
Dr inż. Stanisław Polak

118

Zapytanie

```
1 var xhr;  
2  
3 xhr = new XMLHttpRequest();  
4 if (!xhr) {  
5   alert('I can not create an XMLHttpRequest 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



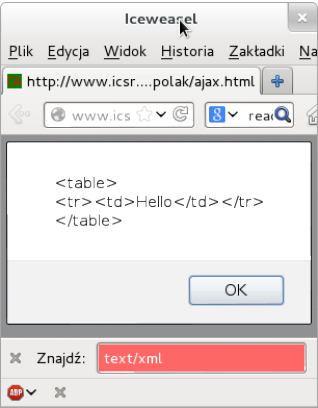
Źródło: <https://zinoui.com/blog/cross-domain-ajax-request>

Notatki



Obsługa odpowiedzi

```
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



Obietnice

- ▶ Reprezentują / Przechowują wyniki operacji asynchronicznej¹, np. zapytania AJAX
- ▶ Wyniki operacji mogą nie być jeszcze dostępne, ale kiedyś będą
- ▶ Obietnice można ze sobą składać

Stany obietnic

- ▶ W oczekiwaniu (ang. pending)
- ▶ Spełniona (ang. fulfilled)
- ▶ Odrzucona (ang. rejected)
- ▶ Rozstrzygnięta (ang. settled)

¹Wartość zwracaną (w przypadku sukcesu) lub błąd (w razie porażki)

Notatki

Obietnice

Przykład użycia

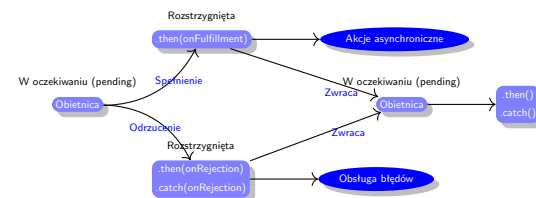
```
1  /** Creating a promise */
2  function createPromise() {
3    return new Promise(function(resolve, reject) {
4      // Calculations performed asynchronously
5      setTimeout(function() {
6        var divider = Math.floor(Math.random() * 3);
7        if(divider !== 0)
8          resolve(10/divider); // 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
18      created
19    .then(function(result) {
20      console.log('Division result:', result);
21    })
22    .catch(function(error) {
23      console.log('An error occurred!', error);
24    });
25  }
26  usePromise();
```

Na wyjściu

'Division result:' 5

Na wyjściu

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



Obietnice

Składnia `async` / `await`

Notatki

Tak się używało obietnic dotychczas

```
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();
```

Użycie `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();
```

Fetch API

Ogólna charakterystyka

Notatki

- ▶ Interfejs JS do uzyskiwania dostępu oraz manipulowania częściami potoku HTTP, takimi jak: żądanie oraz odpowiedź
- ▶ Różnice w stosunku do AJAX-a:
 - ▶ Korzysta z obietnic, a nie funkcji zwrotnych
 - ▶ Obietnica będąca rezultatem wykonania zapytania nie otrzyma stanu 'Odrzucona', nawet wtedy status odpowiedzi HTTP to 404 lub 500.
 - ▶ Domyślnie funkcja do wykonywania zapytań nie wysyła ani nie odbiera żadnych plików cookie z serwera

Fetch API

Przykład użycia

```
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



Podstawy

```
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.
9     min.js"></script>
10   <script >
11     //$(document).ready(function(){
12     jQuery(document).ready(function(){
13       $("a").click(function(event){
14         alert("Jak widzisz, ten odsyłacz nie
15           przeniesie cię już do jquery.com");
16         //As you can see, this link will not
17         take you to jquery.com anymore
18         event.preventDefault();
19       }); //$(a).ready(function()
20     }); //jQuery(document).ready(function()
21   </script>
22 </body>
23 </html>_
```



Notatki



Notatki

Pobieranie treści tekstowej

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       p { color:blue; margin:8px; }
6       b { color:red; }
8     </style>
9     <script src="http://code.jquery.com/jquery-3.4.1.min.js"></script>
10   </head>
11   <body>
12     <p><b>Test</b> Paragraph.</p>
13
14     <p></p>
15     <script>
16       var str = $("p:first").text(); //The variable 'str' contains the string 'Test Paragraph'
17       $("p:last").html(str);
18     </script>
19   </body>
20 </html>

```

Test Paragraph.

Test Paragraph.

Notatki



Dostęp do stylów CSS

Odpowiednik przykładu DOM

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5     <title>An example</title>
6     ...
7   </head>
8   <body>
9     <table border="1">
10      <tr><td id="cell1">a</td><td>b</td></tr>
11      <tr><td>c</td><td>d</td></tr>
12    </table>
13    <form>
14      <input type="button" value="Red" onClick="color('#FF0000');">
15      <input type="button" value="Green" onClick="color('#00FF00');">
16    </form>
17  </body>
18 </html>_

```

a	b
c	d

```

1 ...
2 <script>
3   function color(value){
4     $('#cell1').css("backgroundColor",value);
5   }
6 </script>

```

Użycie metod jQuery

```

1 <script>
2   function color(value){
3     var element = document.getElementById("cell1");
4     element.style.backgroundColor = value;
5   }
6 </script>

```

Użycie metod DOM

Notatki



Dodawanie nowego elementu do zbioru dopasowanych elementów

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src="http://code.jquery.com/jquery-3.4.1.min.js"></script>
5   </head>
6   <body>
7     <p>Hello</p><span>Hello Again</span>
8     <script>
9       $("p").add("span").css("background", "yellow");
10    </script>
11  </body>
12 </html>

```

Hello

Hello Again

Notatki



Wstawienie nowej komórki tabeli na końcu wiersza

Odpowiednik przykładu DOM

```

1 <html>
2   <head>
3     <title>An example</title>
4     ...
5   </head>
6   <body>
7     <table border="1">
8       <tr id="row1"><td>a</td><td>b</td></tr>
9       <tr><td>c</td><td>d</td></tr>
10    </table>
11    <form>
12      <input type="button" value="Insert" onClick="insert();">
13    </form>
14  </body>
15 </html>

```

a	b
c	d

a	b	b1
c	d	

```

1 ...
2 <script>
3   function insert(){
4     $('#row1').append($('<td>').text('b1'));
5   }
6 </script>
7 ...

```

Użycie metod jQuery

```

1 ...
2 <script>
3   function insert(){
4     var newTD = document.createElement("td");
5     var newTextNode = document.createTextNode("b1");
6     newTD.appendChild(newTextNode);
7     var element = document.getElementById("row1");
8     element.appendChild(newTD);
9   }
10 </script>
11 ...

```

Użycie metod DOM

Notatki



Animacja tekstu

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       div {
6         background-color:#bca;
7         width:100px;
8         border:1px solid green;
9       }
10    </style>
11    <script src="http://code.jquery.com/jquery-3.4.1.min.js"></script>
12  </head>
13  <body>
14    <button id="go">&raquo; Run</button>
15
16    <div id="block">Hello!</div>
17    <script>
18      /* Using many types of units in one animation. */
19
20      $("#go").click(function(){
21        $("#block").animate({
22          width: "70%",
23          opacity: 0.4,
24          marginLeft: "0.6in",
25          fontSize: "3em",
26          borderWidth: "10px"
27        }, 1500 );
28      });
29    </script>
30  </body>
31 </html>_

```

» Run
Hello!

Rysunek: Pierwsza klatka animacji

» Run

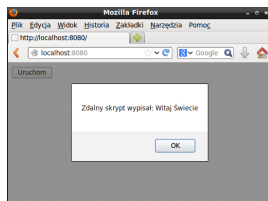
Hello!

Rysunek: Ostatnia klatka animacji

Notatki



Obsługa technologii AJAX



```

1 ...
2 <script>
3 $(document).ready(function(){
4   $('#button1').click(function(){
5     $.ajax({
6       type: "GET",
7       url: "http://localhost:8080/hello",
8       success: function(msg){
9         alert( "Zdalny skrypt wypisał: " + msg ); //The
          remote script was written:
10       }
11     });
12   });
13 });
14 </script>
15 ...
16 <form action="#">
17   <input type="button" value="Uruchom" id="button1" />
18 </form>
19 ...

```

form.html

```

1 var http = require("http");
2 var url = require("url");
3 var fs = require('fs');
4
5 function requestListener(request, response) {
6   var url_parts = url.parse(request.url,true);
7
8   if(url_parts.pathname == '/hello'){
9     response.writeHead(200, {"Content-Type": "text/plain
10     });
11     response.write("Witaj Świecie");
12     response.end();
13   }
14   else {
15     var filename="form.html"
16     var stat = fs.statSync(filename);
17     response.writeHead(200, {
18       'Content-Type': 'text/html',
19       'Content-Length': stat.size
20     });
21     var readStream = fs.createReadStream(filename);
22     readStream.pipe(response);
23   }
24 }
25 var server = http.createServer(requestListener)
26 server.listen(8080);

```

node.js

Notatki



Przykład „Hello World”

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

script.ts

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

script.js

Notatki



Obsługa błędów

Notatki



Błąd typowania

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

script.ts

Błąd składniowy

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

script.ts

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

script.js

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

script.js

Deklarowanie typu zmiennej

Notatki

```

1 let alive:boolean = true;
2 let age:number = 48;
3 let name:string = 'Kowalski';
4
5 let names:string[] = ['Jan','Jerzy'];
6 let children_age:Array<number> = [1, 20, 3];
7 let parents_age:Array<number> = [40, "forty
   one"]; //... Type 'string' is not
   assignable to type 'number'.
8
9 let tuple:[string,number,boolean];
10 tuple = ['1',2,true];
11 console.log(tuple[2]); // true
12 tuple = ['1',2,true,4]; // OK
13 tuple = ['1',2]; //... Type '[string, number]'
   is not assignable to type '[string,
   number, boolean]'. ...
14 tuple = [1,2,3]; //... Type '[number, number,
   number]' is not assignable to type '[
   string, number, boolean]'.
15
16 enum Eyes {Blue, Green = 4, Grey, Brown};
17 let eye_color = Eyes.Blue;
18 console.log(Eyes[0]); // Blue
19 console.log(Eyes[4]); // Green
20 console.log(Eyes[5]); // Grey

```

```

21 let anything:any = 4;
22 anything = "String"; //OK
23 anything = true; //OK
24 anything.x; //OK
25 anything(); //OK
26 new anything(); //OK
27 let string: string = anything; //OK
28
29 let something: unknown = 4;
30 something = "String"; //OK
31 something = true; //OK
32 something.x; //... Object is of type 'unknown'
33 something(); //... Object is of type 'unknown'
34 new something(); //... Object is of type '
   unknown'
35 let string2: string = something; // ... Type '
   unknown' is not assignable to type '
   string'.
36
37 function print(message): void {
38     console.log(message);
39 }
40
41 function exception(message): never {
42     throw new Error(message);
43 }
44
45 function loop(): never {
46     while(true){}
47 }

```



Dr inż. Stanisław Polak

137

Podstawy języka TypeScript

Typowanie

Notatki

```

1 let string:string;
2 string = 1; //Error, you can not assign a number to
   a string type variable
3
4 let number = 1;
5 number = '2'; //Error, you can not assign a string
   to a numeric type variable

```

script.ts

```

1 let string:string;
2 string = <any> 1; //Now it is OK
3 //or
4 string = 1 as any;
5 let number = 1;
6 number = <any> '2'; //Now it is OK
7 //or
8 number = '2' as any; //Now it is OK

```

script.ts



Dr inż. Stanisław Polak

138

Interfejsy

Tworzenie typu złożonego

```
1 interface Person {
2   first_name: string; //mandatory
3   last_name: string; //mandatory
4   age?: number; //optional
5 }
6 /*****/
7 let user: Person;
8 /*****/
9 user = {
10   first_name: 'Jan',
11   last_name: 'Kowalski'
12 }
13 /*****/
14 user = {
15   first_name: 'Jan',
16   last_name: 'Kowalski',
17   age: '40'//Error: wrong value type
18 }
19 /*****/
20 user = {
21   // Error: 'last_name' is not specified
22   first_name: 'Jan'
23 }
24 /*****/
25 user = {
26   first_name: 'Jan',
27   last_name: 345, //Error: wrong value type
28   age: 'teenager'
29 }
```

script.ts

Notatki



Interfejsy

Określanie typów funkcji

```
1 interface stringFunction {
2   (param1:string, param2:string): string;
3 }
4 /*****/
5 let addStrings: stringFunction;
6 let addNumbers: stringFunction;
7 /*****/
8 addStrings = function(string1: string,string2: string) {
9   return string1+string2;
10 } //OK
11
12 addNumbers = function(number1: number, number2: number) {
13   return number1 + number2;
14 } /*...
15 error TS2322: Type '(number1: number, number2: number) => number' is not assignable to type 'stringFunction'.
16   Types of parameters 'number1' and 'param1' are incompatible.
17   Type 'number' is not assignable to type 'string'.
18
19 */
20 /*****/
21 addStrings('Jan','Kowalski'); //OK
22 addStrings('Kowalski'); //... error TS2346: Supplied parameters do not match any signature of call
23   target.
24 addStrings(1,2); //... error TS2345: Argument of type 'number' is not assignable to parameter of type 'string'.
```

script.ts

Notatki



Interfejsy

Typ indeksowany

```
1 interface hashString {  
2     [index: string]: string;  
3 }  
4 let parameters: hashString;  
5  
6 parameters['server'] = 'HP'; // OK  
7 let bar:number = parameters['server']; //... error TS2322: Type 'string' is not assignable to type '  
    number'.  
8 parameters['server'] = 234; //... error TS2322: Type 'number' is not assignable to type 'string'.
```

script.ts

Notatki



Interfejsy

Typy klasowe

```
1 class Person {  
2     id:number;  
3 }  
4  
5 interface Validator{  
6     checkIdUniqueness(s: number): boolean;  
7 }  
8  
9 class Employee extends Person implements Validator {}
```

script.ts

Notatki



Klasy

Modyfikatory

```

1 class Person {
2     protected _id:number;
3     readonly first_name:string;
4     readonly last_name:string = "Anonym";
5
6     get id(): number {
7         return this._id;
8     }
9
10    set id(value: number) {
11        this._id = value
12    }
13
14    constructor(first_name:string,name:string) {
15        this.first_name = first_name;
16        this.last_name = last_name;
17    }
18 }

```

```

19 class Employee extends Person {
20     constructor(id:number, first_name:string,
21         last_name:string) {
22         super(first_name,last_name);
23         this.id=id;
24     }
25 }
26
27 let Employee = new Employee(1,"Stanisław","
28     Polak");
29 console.log(Employee.id); //1
30 console.log(Employee._id); //Property '
31     _id' is protected and only accessible
32     within class 'Person' and its subclasses.
33 console.log(Employee.first_name); //Stanis
34     ław
35 console.log(Employee.last_name); //Polak
36 Employee.first_name = "Jan"; //...Left-hand
37     side of assignment expression cannot be a
38     constant or a read-only property.

```

script.ts

Notatki

Dr inż. Stanisław Polak

143



Klasy oraz typy generyczne

```

1 class CustomCollection<T> {
2     private itemArray: Array<T>;
3
4     constructor() {
5         this.itemArray = [];
6     }
7
8     Add(item: T) {
9         this.itemArray.push(item);
10    }
11
12    GetFirst(): T {
13        return this.itemArray[0];
14    }
15 }
16
17 /*****/
18 class User {
19     public Name;
20 }
21
22 /*****/
23 class Message {
24     public Message;
25 }

```

```

24 class MyApp {
25     constructor() {
26         let myUsers = new CustomCollection<User>();
27         let myMessages = new CustomCollection<Message>();
28
29         let user: User = myUsers.GetFirst(); // OK
30         let message: Message = myUsers.GetFirst(); // Error because of
31         // the Generic type validation.
32         myUsers.Add(new Message()); // Error because of
33         // the Generic type validation.
34     }
35 }

```

script.ts

Źródło: <https://gist.github.com/abergs/5817818>

Notatki



Dekoratory

```

1 function f() {
2   console.log("f(): evaluated");
3   return function (target, propertyKey: string, descriptor:
4     PropertyDescriptor) {
5     console.log("f(): called");
6     console.log(target);
7     console.log();
8     console.log(propertyKey);
9     console.log();
10    console.log(descriptor);
11  }
12 }
13
14 function g(value) {
15   console.log("g("+value+"): evaluated");
16   return function (target, propertyKey: string, descriptor:
17     PropertyDescriptor) {
18     console.log("g("+value+"): called");
19   }
20 }
21
22 class C {
23   @f()
24   @g('abc')
25   method() {}
26 }

```

script.ts

Źródło: <https://github.com/Microsoft/TypeScript-Handbook/blob/master/pages/Decorators.md>

Dr inż. Stanisław Polak

145

Notatki

[illegible]

Przestrzeń nazewnicza

Jednoplikowa

```

1 namespace A {
2     var a:string = 'abc';
3     export class Twix {
4         constructor() {
5             console.log('Twix');
6         }
7     }
8
9     export class PeanutButterCup {
10         constructor() {
11             console.log('PeanutButterCup');
12         }
13     }
14
15     export class KitKat {
16         constructor() {
17             console.log('KitKat');
18         }
19     }
20 }
21 let o1 = new A.Twix(); // Twix
22 let o2 = new A.PeanutButterCup(); // PeanutButterCup
23 let o3 = new A.KitKat(); // KitKat
24 console.log(A.a); //...error TS2339: Property 'a'
    does not exist on type 'typeof A'.

```

script.ts

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



Notatki

[illegible]

Przestrzeń nazwiczna

Wieloplikowa

```
1 namespace A {
2   export class Twix { ... }
3 }
```

global1.ts

```
1 namespace A {
2   export class PeanutButterCup { ... }
3 }
```

global2.ts

```
1 namespace A {
2   export class KitKat { ... }
3 }
```

global3.ts



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

script.ts

Notatki



Moduły

Eksportowanie

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

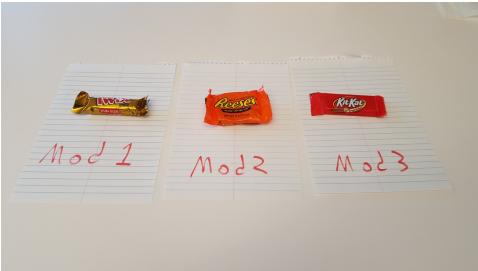
Mod1.ts

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

Mod2.ts

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

Mod3.ts



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

Notatki



Moduły

Importowanie

```
1 export class Twix {...}
2 export {Twix as Raider};
```

Mod1.ts

```
1 class PeanutButterCup {...}
2 export {PeanutButterCup};
```

Mod2.ts

```
1 export class KitKat {...}
```

Mod3.ts

```
1 import {Twix, Raider} from './Mod1';
2 import {PeanutButterCup} from './Mod2';
3 import {KitKat} from './Mod3';
4 import {KitKat as KitKatChunKy} from './Mod3';
5 import * as Mars from './Mod1';
6
7 let o1 = new Twix(); // Twix
8 let o2 = new Raider(); // Twix
9 let o3 = new PeanutButterCup(); // PeanutButterCup
10 let o4 = new KitKat(); // KitKat
11 let o5 = new KitKatChunKy(); // KitKat
12 let o6 = new Mars.Twix(); // Twix
13 let o7 = new Mars.Raider(); // Twix
```

script.ts

Notatki



Przestrzeń nazwiczna w modułach

```
1 export namespace A {
2   export class Twix { ... }
3 }
```

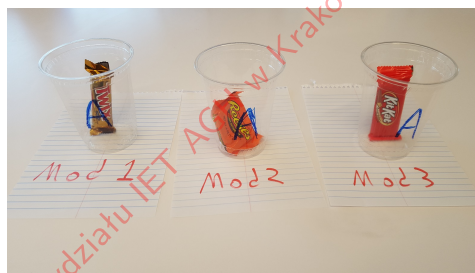
Mod1.ts

```
1 export namespace A {
2   export class PeanutButterCup { ... }
3 }
```

Mod2.ts

```
1 export namespace A {
2   export class KitKat { ... }
3 }
```

Mod3.ts



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

Notatki



Moduły

Importowanie w stylu „NodeJS”

```
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



Tworzenie projektu

```
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



Pliki deklaracji

```

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

```

script.ts

```

1 function area(radius){
2   return Math.PI*Math.pow(radius,2);
3 }
4 module.exports = area;

```

area.js

```

1 declare function area(radius: number) : number
  ;
2 export = area

```

area.d.ts

```

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

```

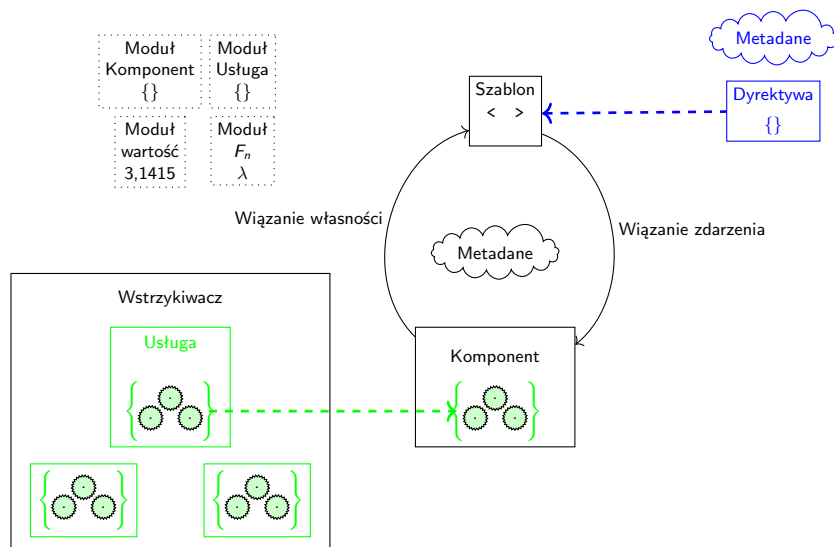
script.js

Notatki



Architektura

Notatki



Generowanie szkieletu aplikacji

```
1 $ npm install --global @angular/cli
2 $ ng new hello
3 $ cd hello
4 $ ng serve
```

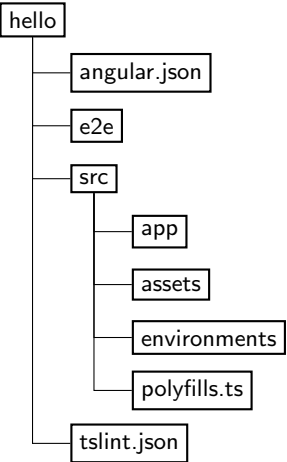


Notatki



Struktura projektu

Ważniejsze pliki oraz katalogi



Notatki



Komponent oraz moduł

Notatki

```
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 ;
3 import { NgModule } from '@angular/core';
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



Plik główny

Notatki

```
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



Szablon

Notatki

```
1 <!--The content below is only a placeholder and can be
   replaced.-->
2 <div style="text-align:center">
3   <h1> Welcome to {{title}}!! </h1>
4   ...
```

src/app/app.component.html

```
1 ...
2 @Component({
3   selector: 'app-root ',
4   ...
5 })
6 export class AppComponent {
7   title = 'app';
8 }
```

src/app/app.component.ts



```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>Hello</title>
6     <base href="/">
7
8     <meta name="viewport" content="width=device-width, initial
9       ~scale=1">
10    <link rel="icon" type="image/x-icon" href="favicon.ico">
11  </head>
12  <body>
13    <app-root> </app-root>
14  </body>
15 </html>
```

src/index.html



Środowisko do uruchamiania testów jednostkowych „Karma”

Konfiguracja

Notatki

```
1 ...
2 module.exports = function (config) {
3   config.set({
4     ...
5     frameworks: ['jasmine', '@angular-devkit/
6       build-angular'],
7     plugins: [
8       require('karma-jasmine'),
9       require('karma-chrome-launcher'),
10    ],
11    ...
12  });
13};
```

src/karma.conf.js

```
1 // This file is required by karma.conf.js and loads
   recursively all the .spec and framework files
2
3 import 'zone.js/dist/zone-testing';
4 import { getTestBed } from '@angular/core/testing';
5 import {
6   BrowserDynamicTestingModule,
7   platformBrowserDynamicTesting
8 } from '@angular/platform-browser-dynamic/testing';
9
10 declare const require: any;
11
12 // First, initialize the Angular testing environment
13 getTestBed().initTestEnvironment(
14   BrowserDynamicTestingModule,
15   platformBrowserDynamicTesting()
16 );
17 // Then we find all the tests.
18 const context = require.context('./', true, /\.spec
19   \.ts$/);
20 // And load the modules.
21 context.keys().map(context);
```

src/test.ts



Środowisko do uruchamiania testów jednostkowych „Karma”

Testy

Notatki

```
1 ...
2 describe('AppComponent', () => {
3   beforeEach(async(() => {
4     TestBed.configureTestingModule({
5       declarations: [
6         AppComponent
7       ],
8     }).compileComponents();
9   }));
10
11   it('should create the app', () => {
12     const fixture = TestBed.createComponent(AppComponent);
13     const app = fixture.debugElement.componentInstance;
14     expect(app).toBeTruthy();
15   });
16
17   it('should have as title \'hello\'', () => {
18     const fixture = TestBed.createComponent(AppComponent);
19     const app = fixture.debugElement.componentInstance;
20     expect(app.title).toEqual('app');
21   });
22
23   it('should render title in a h1 tag', () => {
24     const fixture = TestBed.createComponent(AppComponent);
25     fixture.detectChanges();
26     const compiled = fixture.debugElement.nativeElement;
27     expect(compiled.querySelector('h1').textContent).toContain('Welcome to hello!');
28   });
29 });
```

src/app/app.component.spec.ts

```
1 $ ng test
2 ...
3 24 05 2019 14:09:36.410:INFO [karma-server
4   ]: Karma v4.0.1 server started at
5   http://0.0.0.0:9876/
6 24 05 2019 14:09:36.411:INFO [launcher]:
7   Launching browsers Chrome with
8   concurrency unlimited
9 24 05 2019 14:09:36.431:INFO [launcher]:
10  Starting browser Chrome
11 ...
12 Chrome 74.0.3729 (Linux 0.0.0): Executed 3
13   of 3 SUCCESS (0.255 secs / 0.239 secs)
14
15 TOTAL: 3 SUCCESS
16 TOTAL: 3 SUCCESS
```

Uruchomienie testu



Środowisko do uruchamiania testów e2e „Protractor”

Konfiguracja

Notatki

```
1 // Protractor configuration file, see link for more information
2 // https://github.com/angular/protractor/blob/master/lib/config.ts
3
4 /*global jasmine */
5 var SpecReporter = require('jasmine-spec-reporter');
6
7 exports.config = {
8   allScriptsTimeout: 11000,
9   specs: ['./src/**/*.e2e-spec.ts'],
10  capabilities: {'browserName': 'chrome'},
11  directConnect: true,
12  baseUrl: 'http://localhost:4200/',
13  framework: 'jasmine',
14  jasmineNodeOpts: {
15    showColors: true,
16    defaultTimeoutInterval: 30000,
17    print: function() {}
18  },
19  onPrepare() {
20    require('ts-node').register({
21      project: require('path').join(__dirname, './tsconfig.e2e.json')
22    });
23    jasmine.getEnv().addReporter(new SpecReporter({ spec: { displayStacktrace: true } }));
24  }
25};
```

e2e/protractor.conf.js



Środowisko do uruchamiania testów e2e „Protractor”

Test

Notatki

```
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
```

Uruchomienie testu



Koncepcja aplikacji

Notatki

Założenia

- ▶ Aplikacja, na samym początku, wyświetla:
 - ▶ domyślny tytuł („Welcome to app!”)
 - ▶ listę produktów w postaci tabeli HTML
 - ▶ przycisk z napisem „Ustaw tytuł” (Eng. Set title)
- ▶ Za wyświetlanie listy produktów oraz przycisku odpowiada odrębny komponent
- ▶ Treść tytułu oraz nagłówek tabeli określa komponent główny
- ▶ Naciśnięcie przycisku ma spowodować zmianę tytułu z „Welcome to app!” na „Welcome to Lista produktów!”
- ▶ Produkty są przechowywane, w programie, w postaci tablicy obiektów
- ▶ Strona ma być responsywna

Wymagane składniki

- ▶ Moduł główny
- ▶ Komponenty:
 - ▶ główny (nadrzędny)
 - ▶ potomny (podrzędny), wyświetlający listę produktów
- ▶ Klasa reprezentująca produkt
- ▶ Usługa dostarczająca zawartość listy produktów



Notatki

```
1 $ ng new shop
2 $ cd shop
3 $ ng generate class Product # Create a class skeleton
4 CREATE src/app/product.spec.ts (158 bytes)
5 CREATE src/app/product.ts (25 bytes)
6 $ ng generate service Products # Create a service skeleton
7 CREATE src/app/products.service.spec.ts (343 bytes)
8 CREATE src/app/products.service.ts (137 bytes)
9 $ ng generate component ProductsList # Create the component's skeleton
10 CREATE src/app/products-list/products-list.component.css (0 bytes)
11 CREATE src/app/products-list/products-list.component.html (32 bytes)
12 CREATE src/app/products-list/products-list.component.spec.ts (671 bytes)
13 CREATE src/app/products-list/products-list.component.ts (296 bytes)
14 UPDATE src/app/app.module.ts (422 bytes)
```



Notatki

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class ProductsService {
7
8   constructor() { }
9 }
```

```
src/app/products.service.ts
```

```

1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-products-list',
5   templateUrl: './products-list.component.html',
6   styleUrls: ['./products-list.component.css']
7 })
8 export class ProductsListComponent implements OnInit {
9   constructor() {}
10  ngOnInit() {}
11 }

```

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

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

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

```
1 export class Product {
2 }
```

src/app/product.ts

```

1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5 import { ProductsListComponent } from './products-list/
  products-list.component';
6
7 @NgModule({
8   declarations: [
9     AppComponent,
10    ProductsListComponent
11  ],
12  imports: [
13    BrowserModule,
14  ],
15  providers: [],
16  bootstrap: [AppComponent]
17 })
18 export class AppModule { }

```

src/app/app.module.ts



Czynności wstępne

Włączenie responsywności

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Shop</title>
6     <base href="/">
7     <link rel="stylesheet"
8       href="https://stackpath.bootstrapcdn.com/bootstrap/
9       4.3.1/css/bootstrap.min.css">
10    <meta name="viewport" content="width=device-
11      width, initial-scale=1">
12    <link rel="icon" type="image/x-icon" href="
13      favicon.ico">
14  </head>
15  <body class="container">
16    <app-root></app-root>
17    <script src="https://code.jquery.com/jquery-3.3.1.slim.
18      min.js"></script>
19    <script src="https://cdnjs.cloudflare.com/ajax/libs/
20      popper.js/1.14.7/umd/popper.min.js"></script>
21    <script src="https://stackpath.bootstrapcdn.com/
22      bootstrap/4.3.1/js/bootstrap.min.js"></script>
23  </body>
24 </html>
```

src/index.html

Osadzanie komponentu potomnego

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

src/app/app.component.html

```
1 ...
2 @Component({
3   selector: 'app-products-list',
4   ...
5 })
6 ...
```

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

Notatki



Przekazywanie danych pomiędzy komponentami

Wiązanie własności oraz wiązanie zdarzenia

```
1 ...
2 export class AppComponent {
3   productsHeaders : string[] = [ 'Nazwa', 'Opis', 'Cena'
4     ];
5   setTitle (title: string){
6     title = 'shop';
7   }
8 }
```

src/app/app.component.ts

```
1 <h1> {{title}} </h1>
2 <app-products-list
3   [headers]='productsHeaders'
4   <!-- bind-headers='productsHeaders' -->
5   (changeTitle)='setTitle($event)' >
6   <!-- on-changeTitle='setTitle($event)' -->
7 </app-products-list>
```

src/app/app.component.html

```
1 import { ..., Input, Output, EventEmitter }
2   from '@angular/core';
3 export class ProductsListComponent implements
4   OnInit {
5   @Input() headers: string[];
6   @Output() changeTitle: EventEmitter<string> =
7     new EventEmitter<string>();
8   ...
9   onClick() {
10     this.changeTitle.emit ('Lista produktów');
11   }
12 }
```

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

```
1 <ng-template ... [ngForOf]=' headers ">
2   ...
3 </ng-template>
4 ...
5 <button (click)='onClick()' >Ustaw tytuł</button>
6 <!-- ... on-click='onClick()' ... -->
```

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

Notatki



Klasa reprezentująca produkt

Notatki

```
1 export class Product {  
2   id: number;  
3   name: string = '';  
4   description: string = '';  
5   price: number;  
6 }
```

src/app/product.ts



Implementacja usługi dostarczenia listy produktów

Notatki

```
1 import { Injectable } from '@angular/core';  
2 import { Product } from './product';  
3  
4 const PRODUCTS: Product[] = [  
5   {  
6     id: 1,  
7     name: 'spadochron',  
8     description: 'Rewelacyjny spadochron dla nurków! Znakomicie zapobiega zderzeniu się z dnem',  
9     price: 250.99  
10  },  
11  {  
12    id: 2,  
13    name: 'wanna',  
14    description: 'Dzięki wbudowanym drzwiom już więcej nie poślizniesz się wychodząc z wanny',  
15    price: 599.80  
16  },  
17 ];  
18 /*****  
19 @Injectable()  
20 export class ProductsService {  
21   getProducts(): void {  
22     return PRODUCTS;  
23   }  
24 }_
```

src/app/products.service.ts



Wstrzykiwanie usługi

```
1 ...
2 import { ProductsService } from '../products.service';
3 import { Product } from '../product';
4 ...
5 export class ProductsListComponent implements OnInit {
6     products: Product[];
7
8     //Injecting the 'ProductsService' service - after
8     //the injection, the service will be available
8     //using the expression 'this.productsService'
9     constructor(private productsService:
9         ProductsService) {}
10
11     ngOnInit(): Product[] {
12         this.products = this.productsService.getProducts();
13     }
14 }
```

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

```
1 ...
2 import { ProductsService } from '../products.service';
3 ...
4 @NgModule({
5     ...
6     providers: [ProductsService],
7     ...
8 })
9 ...
```

src/app/app.module.ts

Notatki



Wyświetlanie listy produktów



```
1 <table class="table">
2   <tr>
3     <ng-template ngFor let-header [ngForOf]="headers"
4       >
5       <!-- duplicated table cell -->
5       <th>{{header}}</th>
6     </ng-template>
7   </tr>
8   <ng-template ngFor let-product [ngForOf]="products">
9     <!-- duplicated table row -->
10    <tr>
11      <td>{{product.name}}</td>
12      <td>{{product.description}}</td>
13      <td [innerHTML]="product.price"></td>
14    </tr>
15  </ng-template>
16 </table>__
```

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

```
1 <table class="table">
2   <tr>
3     <th *ngFor="let header of headers">{{header
4       }}</th>
5   </tr>
6   <tr *ngFor="let product of products">
7     <td>{{product.name}}</td>
7     <td>{{product.description}}</td>
8     <td [innerHTML]="product.price"></td>
9   </tr>
10 </table>__
```

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

Notatki



Koncepcja rozbudowy aplikacji

Założenia

- ▶ Aplikacja ma wyświetlać listę nazw produktów w postaci listy nienumerowanej HTML
- ▶ Liczbę wyświetlanych produktów (na stronie) można zmieniać za pomocą suwaka
- ▶ Kliknięcie nazwy produktu ma spowodować wyświetlenie szczegółowych informacji o tym produkcie
- ▶ Tytuł strony określa komponent główny, a tytuły nagłówków są zawarte w szablonie — brak komunikacji pomiędzy komponentami

Wymagane (nowe) składniki

- ▶ Komponenty:
 - ▶ wyświetlający szczegóły produktu
- ▶ Usługi:
 - ▶ dostarczająca informację o ilości produktów
 - ▶ dostarczająca informację o wybranym (klikniętym) produkcie

Notatki



Generowanie plików szkieletowych komponentu

```
1 $ ng generate component ProductDetails
2 installing component
3   create src/app/product-details/product-details.component.css
4   create src/app/product-details/product-details.component.html
5   create src/app/product-details/product-details.component.spec.ts
6   create src/app/product-details/product-details.component.ts
7   update src/app/app.module.ts
```

Notatki



Implementacja nowych usług

Notatki



Wyświetlanie listy produktów

Dwukierunkowe wiązanie danych

Notatki



```
1 ...
2 @Injectable()
3 export class ProductsService {
4   ...
5   getProductsLength(): number {
6     return PRODUCTS.length
7   }
8   /*****
9   getProduct(id:number): Product {
10    return PRODUCTS.find((element) => element.id == id);
11  }
12 }
```

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

```
1 ...
2 export class ProductsListComponent implements OnInit {
3   ...
4   productsLength: number; // the size of the product table
5   pNumber: number; // the number of products on the page
6   ...
7   ngOnInit(): void {
8     ...
9     this.productsLength = this.productsService.getProductsLength
10    ();
11    this.pNumber = this.productsLength;
12  }
13 }
```

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



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

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



Wyświetlanie szczegółów produktu

Notatki

```
1 import { ActivatedRoute } from '@angular/router';
2 ...
3 export class ProductDetailsComponent implements
4   OnInit {
5   product: Product;
6
7   constructor(private productService:
8     ProductService, private route:
9     ActivatedRoute) { }
10
11   ngOnInit() {
12     let id = +this.route.snapshot.params['id'];
13     this.product = this.productService.getProduct(id);
14   }
15 }
```

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

```
1 <h2>{{product.name}}</h2>
2 <table class="table">
3   <tr>
4     <th>Opis</th>
5     <th>Cena</th>
6   </tr>
7   <tr>
8     <td>{{product.description}}</td>
9     <td>{{product.price}}</td>
10  </tr>
11 </table>
```

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



Konfiguracja trasowania

Notatki

```
1 ...
2 import { RouterModule } from '@angular/router';
3 ...
4 @NgModule({
5   ...
6   imports: [
7     BrowserModule,
8     FormsModule,
9     HttpClientModule,
10    RouterModule.forRoot([
11      { path: '', component: ProductsListComponent },
12      { path: 'product/:id', component:
13        ProductDetailsComponent }
14    ])
15  ],
16  ...
17 })
18 ...
```

src/app/app.module.ts

```
1 ...
2 export class AppComponent {
3   title = 'Sklep';
4 }_
```

src/app/app.component.ts

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

src/app/app.component.html



Źródła I



Notatki

Notatki
