STU FIIT
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

VGG
Vision & Graphics Group
Institute of Computer Engineering and Applied Informatics

Základy tvorby interaktívnych aplikácií

Programovanie v JavaScript

Ing. Peter Kapec, PhD.

LS 2020-21

Obsah

- Programovací jazyk JavaScript
  - základné konštrukcie
  - dátové typy
  - polia
- Manipulácia DOM web stránky
- jQuery

JavaScript

- Vznik v rámci vývoja prehliadača Netscape Navigator
- Pôvodne nazvaný Livescript
- Formálne nazývaný ECMAScript (ECMA-262)
- V rámci propagačnej kampaňe nazvaný JavaScript
- Nemá nič spoločné s jazykom Java

Prečo vznikol?

- Hlavným dôvodom boli požiadavky na dynamické správanie sa webových stránok
- Pôvodne išlo len o dočasné riešenie
- Jazyk sa presadil a štandardizoval

Použitie v rámci HTML

- Používame značku **<script>**
- Type **text/javascript**
- Jazyk **javascript**
- Možno vložiť do <head>, <body>
- Možno použiť externý **.js* súbor, alebo URL

```
...
<script language="javascript" type="text/javascript">
   JavaScript code
</script>
...
```

Jednoduchý príklad

- Kód vkladáme do oblasti **<!-- [KÓD] //-->**
- Použijeme globálnu funkciu **document.write()**
- Parametrom funkcie je text ktorý chceme zobraziť

```
<html>
<body>
<script language="javascript" type="text/javascript">
<!--
   document.write("Hello World!")
//-->
</script>
</body>
</html>
```

6

Výstup

- Výstup skriptu môže byť zapísaný do
  - HTML elementu, použitím *innerHTML*
  - HTML výstupu použitím *document.write()*
  - Alert box, použitím *window.alert()*
  - Konzoly browsera, použitím *console.log()*

Klúčové slová

- Break – prerušenie switch-u alebo cyklu
- Continue – pokračovanie cyklu
- Debugger – zastavenie vykonávania skriptu a zavolanie debugovacej funkcie
- do ... while – vykonanie bloku príkazov, pokial je splnená podmienka
- for – cyklus
- function – declarovanie funkcie
- if ... else – vetvenie na základe podmienky
- return – návrat z funkcie
- switch – vetvenie na základe prípadu
- try ... catch – zachytenie chybovýchs stavov
- var – deklarovanie premennej

Základná syntax

- Oddeľovanie príkazov

```
<script language="javascript" type="text/javascript">
<!--
  var1 = 10
  var2 = 20
//-->
</script>
```

```
<script language="javascript" type="text/javascript">
<!--
  var1 = 10; var2 = 20;
//-->
</script>
```

Základná syntax

- Citlivosť na veľkosť písma: **Time != time**
- Znaky medzi *//* a koncom riadka sú komentár
- Znaky medzi */** a ***/** sú komentár i cez viac riadkov
- Kód možno „schovať“ medzi **<!-- kód //-->**

Premenné

- Premenné je nutné* pred použitím deklarovať
- Používa sa na to klúčové slovo **var**

```
<script type="text/javascript">
<!--
var money;  var name;
//-->
</script>
```

Premenné

- Na rozdiel od jaz. C netreba deklarovať **typ**
  - číslo: 42, 12.94
  - reťazec: “text”, ‘text‘
  - boolean: true, false
- Premenná po deklarovaní má hodnotu *undefined*

```
<script type="text/javascript">
<!--
var money;  var name;
//-->
</script>
```

Deklarácia premenných

- Nie je možné použiť kľúčové slová ako názov premennej
- Názov nesmie začínať číslom, musí začínať znakom abecedy alebo znakmi _ $
  premenná môže obsahovať znaky _ $

Deklarácia premenných

- **Rozsah platnosti premenných**
  - **Globálne** premenné - definované mimo tela funkcií, prístupné všade
  - **Lokálne** premenné - definované v tele funkcie, prístupné len v rámci funkcie kde boli definované
  - Premenné platné v ***block*** rozsahu:

```
{
    let x = 10;
    // platnne v rozsahu {...}
}
// tu už x neplatí
```

Základné dátové typy

- Typy s hodnotou
  - string
  - number
  - boolean
  - object
  - function
- Typy ako objekty
  - Object
  - Date
  - Array
  - String
  - Number
  - Boolean
- dátové typy, ktoré nemôžu obsahovať hodnotu:
  - null
  - undefined

Základné dátové typy

- operátor *typeof*
  - typeof "John" // vráti "string"
  - typeof 3.14 // vráti "number"
  - typeof NaN // vráti "number"
  - typeof false // vráti "boolean"
  - typeof [1,2,3,4] // vráti "object"
  - typeof {name:'John', age:34} // vráti "object"
  - typeof new Date() // vráti "object"
  - typeof function () {} // vráti "function"
  - typeof myCar // vráti "undefined"
  - typeof null // vráti "object"

Aritmetické operácie

- JavaScript podporuje nasled. arit. operácie

```
<script type="text/javascript">
<!--
var A = 10; var B = 20
var C = A + B // C je 30
var D = A - B // D je -10
var E = A * B // E je 200
var F = B / A // F je 2
var G = ++A // G je 11
var H = --A // H je 9
//-->
</script>
```

modulo %, mocnina **

Operácie porovnania

- Podobne ako v jazyku C

```
<script type="text/javascript">
<!--
var A = 10; var B = 20
var C = ( A == B ) // C je false
var D = ( A != B ) // D je true
var E = ( A > B ) // E je false
var F = ( A < B ) // F je true
var G = ( A >= B ) // G je false
var H = ( A <= B ) // H je true
//-->
</script>
```

porovnanie hodnoty a zároveň typu: ===, !==

Logické operácie

- 0 je ekvivalentná false
- Nenulové čísla sú true

```
<script type="text/javascript">
<!--
var A = 10; var B = 20
var C = ( A && B ) // C je true
var D = ( A || B ) // D je true
var E = !( A && B ) // E je false
//-->
</script>
```

Vetvenie, if

- Syntax:

```
if ( /* výraz */ ){
   // KÓD ak je výraz true
}

if ( /* výraz */ ){
   // KÓD ak je výraz true
} else {
   // KÓD ak je výraz false
}
```

Vetvenie, else if

- Možno použiť pri komplexnejších podmienkach

```
if ( /* výraz1 */ ) {
   // KÓD ak je výraz true
} else if ( /* výraz2 */ ) {
   // KÓD ak je výraz2 ture
} else {
   // KÓD ak je výraz1 i výraz2 false
}
```

Vetvenie, switch

- Podobne ako u jazyka C

```
switch (/* výraz */)
{
  case podmienka1: //KÓD
                   break;
   case podmienka2: //KÓD
                   break;
   ...
  case podmienkaN: //KÓD
                   break;
  default: //KÓD
}
```

Cyklus, while a do-while

- Stále sa príliš nelíši od C

```
while (/* výraz */){
    // KÓD
}
```

```
do {
   // KÓD
} while (/* výraz */);
```

Cyklus, for

- Konštrukcia for je podobná ako v C

```
for (/* inicializácia */; /* podmienka */; /* iteráčný
 krok */)
{
     // KÓD
}
```

- avšak možno ho zapísať aj nasledovne

```
for (/* premenná */ in /* objekt */){
     // KÓD
}
```

Funkcie

- Deklarujeme klúčovým slovom function

```
function nazov_funkcie(/* zoznam-parametrov */)
{
  // KÓD
}
```

- Volanie funkcie

```
nazov_funkcie(parameter1, parameter2, …, parameterN)
```

Funkcie

- Príklad použitia funkcie **alert** a **prompt**

```
<head>
<script language="javascript" type="text/javascript">
<!--
   alert("Hello World!");
//-->
</script>
</head>
```

Funkcie

- Príklad použitia funkcie **alert** a **prompt**

```
<head>
<script type="text/javascript">
<!--
   var meno = prompt("Zadaj meno : ", "sem napíš meno");
   alert("Zadal si : " +  meno );
//-->
</script>
</head>
```

Funkcie a premenné

- Funkcie možno ukladať do premenných

```
function mojafunkcia(a)
{
 return a + 1;
};

// Možno zapísať i ako
var mojafunkcia = function(a)
{
 return a + 1;
};

// Dá sa priradiť i k inej premennej
 var dalsiafunkcia = mojafunkcia
```

Polia

- Polia možno deklarovať priamo pomocou []
- Vytvorí sa zoznam hodnôt od indexu 0

```
var pole = [ "dom", "strom", "ulica" ]
pole[3] = "auto"

pole.push("lietadlo")


for (i = 0; i < pole.length; i++)
{
    console.log(pole[i])
}
```

Polia

- Polia možno deklarovať aj nasled. notáciou

```
var cars=new Array();
cars[0]="Saab";
cars[1]="Volvo";
cars[2]="BMW";
```

```
var cars=new Array("Saab","Volvo","BMW");
```

Polia

- V poli možno uchovať rôzne dáta

```
var arr = [ 10, "hello", function(){}, [1, 2, 3]]
for (i=0; i<arr.length; i++)
{
   console.log(typeof(arr[i]) + " - " + arr[i])
}
```

```
number - 10
string - hello
function - function (){}
object - 1,2,3
```

Manipulácia DOM HTML stránky

HTML DOM (Document Object Model)

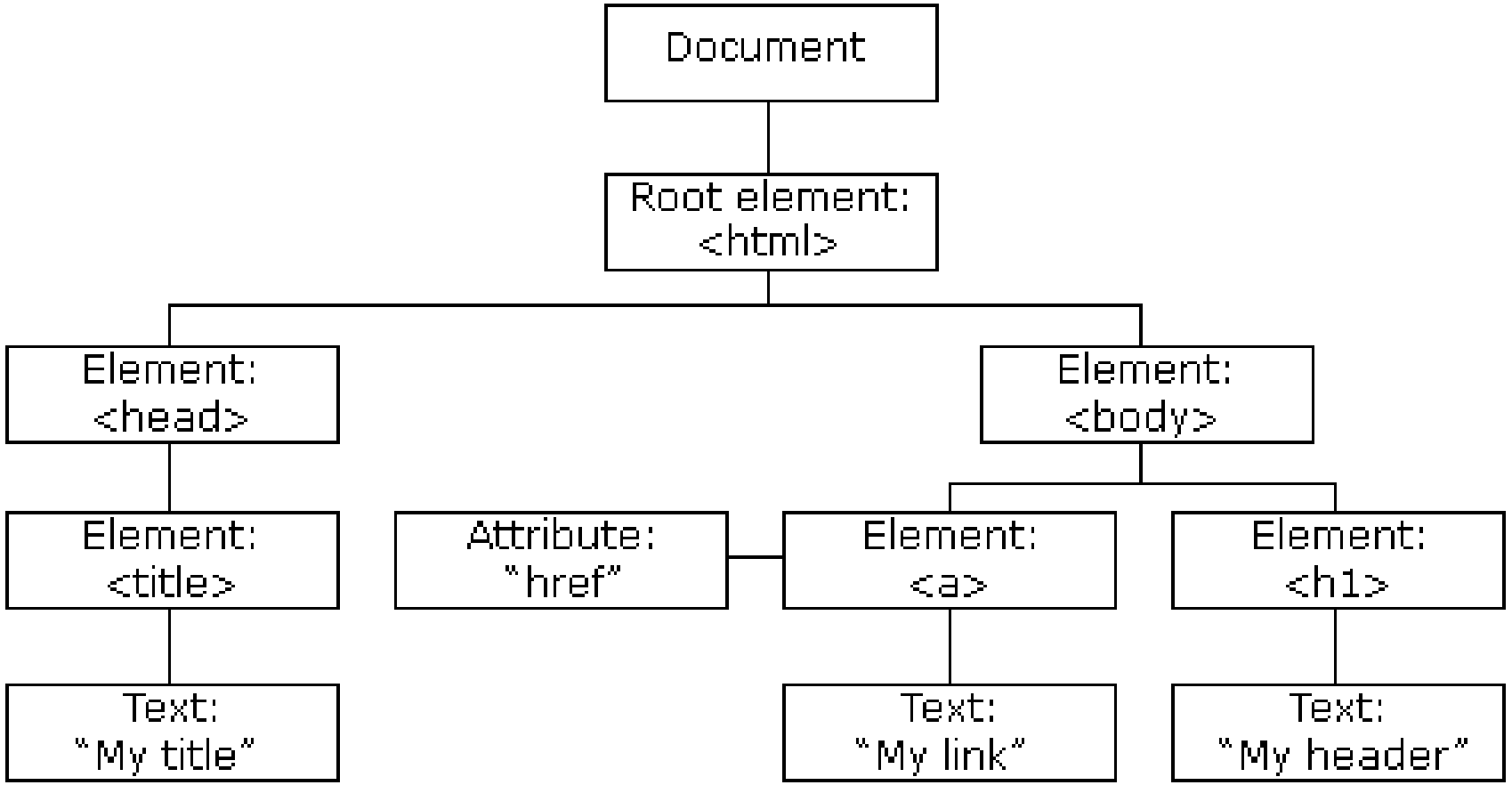
- HTML DOM je štandardný objektový model a programovacie rozhranie pre HTML
- Definuje:
  - Prvky HTML ako objekty
  - Vlastnosti všetkých prvkov HTML
  - Metódy prístupu ku všetkým prvkom HTML
  - Udalosti pre všetky prvky HTML
- HTML DOM je štandardom pre získavanie, zmenu, pridávanie alebo odstraňovanie prvkov HTML

HTML DOM (Document Object Model)

- Umožňuje:
  - zmeniť všetky elementy HTML na stránke
  - zmeniť všetky atribúty HTML na stránke
  - zmeniť všetky štýly CSS na stránke
  - odstrániť existujúce prvky a atribúty HTML
  - pridávať nové elementy a atribúty HTML
  - reagovať na všetky existujúce udalosti HTML na stránke
  - na stránke vytvárať nové udalosti HTML

HTML DOM (Document Object Model)

- Reprezentácia načítaného dokumentu pomocou JavaScript objektov



HTML DOM (Document Object Model)

- **Objekty**
  - V DOM sú všetky prvky HTML definované ako objekty, pričom každý objekt má (programové rozhranie):
    - vlastnosti
    - metódy
- **Vlastnosť** je hodnota, ktorú môžeme získať alebo nastaviť (napríklad zmena obsahu prvku HTML)
- **Metóda** je akcia, ktorú môžeme nad objektom vykonať

HTML DOM (Document Object Model)

- **Objekty** sú reprezentované ako uzly stromovej štruktúry
  - Celý dokument je uzlom dokumentu
  - Každý element HTML je uzol
  - Text vo vnútri HTML elementu je „textový“ uzol
  - Každý atribút HTML elementu je „atribútový“ uzol
  - Všetky komentár je „komentárový“ uzol

HTML DOM

- Nájdenie HTML elementu
  - podľa id-čka elementu:
    document.getElementById(id)
  - podľa názvu tagu:
    document.getElementsByTagName(name)
  - podľa triedy elementu
    document.getElementsByClassName(name)

HTML DOM

- Je možné element uložiť do premennej

```
<!DOCTYPE html>
<html>
<body>
<h1 id="header">Old Header</h1>
<script>
var element=document.getElementById("header");
</script>
</body>
</html>
```

HTML DOM

- Je možné element uložiť do premennej a zmeniť jeho obsah

```
<!DOCTYPE html>
<html>
<body>
<h1 id="header">Old Header</h1>
<script>
var element=document.getElementById("header");
element.innerHTML="New Header";
</script>
</body>
</html>
```

document.getElementById(id).innerHTML = new HTML

HTML DOM

- Do dokumentu možno priamo zapisovať

```
<!DOCTYPE html>
<html>
<body>
    <script>
        document.write(Date());
    </script>
</body>
</html>
```

!!! nepoužívať po načítaní stránky – prepíše obsah !!!

HTML DOM

- Môžeme manipulovať CSS štýl elementu

```
<html>
<body>
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color="blue";
</script>
</body>
</html>
```

document.getElementById(id).style.property = new style

HTML DOM

- Môžeme manipulovať atribút elementu

```
<html>
<body>
<img id="obrazok" src="some.png">
<script>
document.getElementById("obrazok").src="novy_obr.png";
</script>
</body>
</html>
```

document.getElementById(id).attribute = new value

HTML DOM

- Elementy možno vytvárať

```
<!DOCTYPE html>
<html>
<body>
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>
<script>
  var p = document.createElement("p");
  var node = document.createTextNode("This is new.");
  p.appendChild(node);
  var element = document.getElementById("div1");
  element.appendChild(p);
</script>
</body>
</html>
```

HTML DOM

- Elementy možno mazať

```
<div>
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>
<script>
var element = document.getElementById("p1");
element.remove();
</script>
```

HTML DOM

- Navigácia v DOM strome
- Z daného elementu sa vieme dostať k ďalším uzlom stromu cez tieto vlastnosti uzla:
  - *parentNode*
  - *childNodes[nodenumber]*
  - *firstChild*
  - *lastChild*
  - *nextSibling*
  - *previousSibling*

HTML DOM

- HTML udalosti (HTML events)
- Zachytenie a reakcia (vykonanie JavaScript kódu):
  - Po načítaní webovej stránky
  - Po načítaní obrázka
  - Keď sa myš pohybuje nad elementom
  - Keď používateľ klikne na myš
  - Keď sa zmení vstupné pole
  - Po odoslaní formulára HTML
  - Keď používateľ stlačí klávesu

HTML DOM

- Môžeme vykonať kód pri kliknutí myšou

```
<html>
<body>
<h1 id="id1">My Heading 1</h1>
<button type="button"
onclick="document.getElementById('id1').style.color=
'red'">  Click Me!
</button>
</body>
</html>
```

HTML DOM

- Môžeme vykonať kód pri kliknutí myšou

```
<html>
<head>
<script>
function changetext(element){
   element.innerHTML="Ooops!";
}
</script>
</head>
<body>
<h1 onclick="changetext(this)">Click on this text!</h1>
</body>
</html>
```

Knižnica jQuery

jQuery

- jQuery je knižnica určená na rýchlu manipuláciu s DOM pre JavaScript
- Radikálne redukuje čas vývoja web aplikácií
- Umožňuje tvorbu rôznych špeciálnych efektov a animácií
- Poskytuje funkcionalitu pre tvorbu AJAX dopytov a zjednodušuje ich spracovanie

jQuery

- Typicky ukladáme knižnicu k vyvíjanej aplikácii

```
<html>
<head>
<script type="text/javascript" src="js/jquery-1.9.1.min.js">
</script>
....
```

- Môžeme použiť odkaz na google projekt

```
<html>
<head>
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"
></script>
....
```

DOM manipulácia

```
<script type="text/javascript">
$( function(){
  $("#p1").text("Hello World");
  }
);
</script>

<body>
<p id="p1"></p>
</body>
```

```
$(document).ready(function(){
$("#p1").text("Hello World");
});
```

Volanie funkcionality

- Volanie pomocou funkcie jQuery

```
jQuery("#p1");
```

- Volanie skrátenou formou

```
$("#p1");
```

Selektor

- Volanie nad jediným elementom

```
$("#p1");
```

- Nastavenie viacerých elementov

```
<body>
<p id="p1"></p>
<p id="p2"></p>
</body>
```

```
$("p").text("Hello World");
```

Selektor

- Hľadanie na základe parametra

```
<p important="true"></p>
<p id="p2"></p>
```

```
$("p[important=true]").text("This is an important
paragraph.");
```

- Hľadanie na základe tried

```
<p class="c1">Hello World</p>
<p class="c2">Hello Moon</p>
```

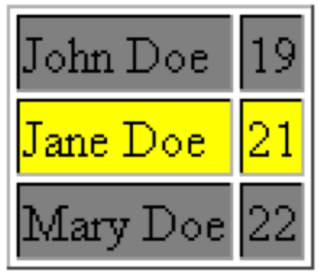
```
$(function(){
  $(".c1").text("You have class c1");
});
```

Selektor

- Pokročilé hľadanie

```
<table border="1">
<tr>
<td>John Doe</td>
<td>19</td>
</tr>
<tr>
<td>Jane Doe</td>
<td>21</td>
</tr>
<tr>
<td>Mary Doe</td>
<td>22</td>
</tr>
</table>
```

```
$(function(){
  $("tr:even").css(
    "background-color", "grey");
  $("tr:odd").css(
    "background-color", "yellow");
});
```



jQuery Objekty

- jQuery neposkytuje rovnaké rozhranie pre manipuláciu s DOM objektami, má vlastné
- Je však možné ľahko previesť akýkoľvek DOM objekt na jQuery objekt

```
$("#p1")

$(document.getElementById("p1"))
```

Udalosti

- Ľahko možno definovať spracovanie akcií nad viacerými elementami

```
<p>
Hello world!
</p>
<p>
Hello moon!
</p>
```

```
$("p").click( function(){ alert($(this).text()); } );
```

Reťazenie

- Ľahko možno definovať spracovanie akcií nad viacerými elementami

```
<p id="p1">jQuery is fun!!</p>
<button>Click me</button>
```

```
$("button").click(function(){
  $("#p1").css("color","red").slideUp(2000).slideDown(2000);
});
```

jQuery a AJAX

- Pohodlné spracovanie zasielania a spracovania udalostí

```
$("button").click(
  function(){
    $.get("test_json.php",function(data,status) {
      var json = JSON.parse(data)
      console.log(json)
    }
  }
);
```

```
<button>get json</button>
```

Online Zdroje

- http://www.w3schools.com/
- …

Zhrnutie

- Klúčové poznatky z prednášky
  - JavaScript
    - Jednoduchá syntax jazykových konštrukcí
    - Základné dátove typy
    - Heterogénne polia
    - Objektové programovanie
  - Pomocou JavaScriptu vieme upravovať DOM web stránky v browseri
  - jQuery zjednodušuje manipuláciu DOM

Nabudúce

- Objektovo-orientované programovanie v jazyku JavaScript
  - Objektovo-orientovaná paradigma
  - Čo sú triedy a objekty (dátové typy)
  - Kompozícia objekdov, dedenie,...
- Objektovo-orientovaný návrh (v kontexte hry)
- HTML Canvas na zobrazovanie poč. grafiky

Ďakujem za pozornosť