

Základy tvorby interaktívnych aplikácií

HCI, Interaktívne aplikácie

Ing. Peter Kapec, PhD.

LS 2019-20

Obsah

- Interakcia a druhy rozhraní
- Úvod do jazyka JavaScript
 - Syntax
 - Premenné
 - Jazykové konštrukcie
 - Polia
 - HTML DOM (Document Object Model)

Interakcia a druhy rozhraní

Interakcia

- Nie je len o návrhu softvéru
- Nie je to len návrh aplikácie pre PC
- Neexistuje žiadne najlepšie rozhranie
 - Fyzické prostredie
 - Skúsenosť používateľov
 - Množstvo a druh informácie

Štýl interakcie

- Priama manipulácia
- Menu
- Formulár
- Príkazový riadok
- Prirodzený jazyk

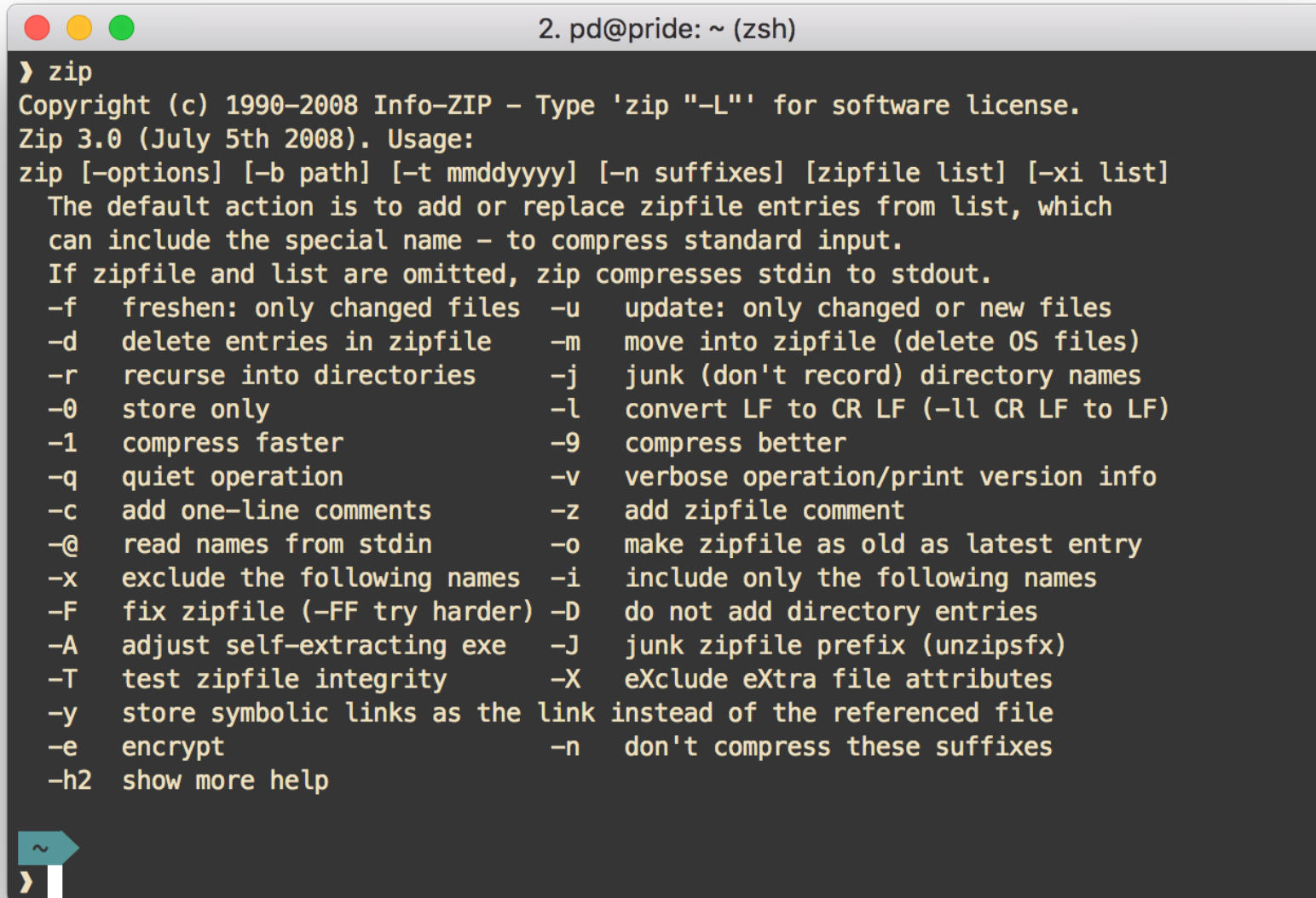
Druh rozhrania

- Automatizované: bez interakcie
- Príkaz: jednorozmerné
- Grafická obrazovka: dvojrozmerné
- Grafické rozhranie: dva a pól rozmerné
- Budúcnosť: troj a viacrozmerné

Druhy rozhraní

- Príkazový riadok (CLI - Command Line Interface)
- Ovládanie cez Menu (MDI -Menu Driven Interface)
- Grafické (GUI - Graphical User Interface)
- Prirodzený jazyk (Natural Language Interface)
- Iné

Príkazový riadok



```
2. pd@pride: ~ (zsh)
> zip
Copyright (c) 1990-2008 Info-ZIP - Type 'zip -L' for software license.
Zip 3.0 (July 5th 2008). Usage:
zip [-options] [-b path] [-t mmdyyy] [-n suffixes] [zipfile list] [-xi list]
The default action is to add or replace zipfile entries from list, which
can include the special name - to compress standard input.
If zipfile and list are omitted, zip compresses stdin to stdout.
-f freshen: only changed files      -u update: only changed or new files
-d delete entries in zipfile        -m move into zipfile (delete OS files)
-r recurse into directories          -j junk (don't record) directory names
-0 store only                        -l convert LF to CR LF (-ll CR LF to LF)
-1 compress faster                  -9 compress better
-q quiet operation                  -v verbose operation/print version info
-c add one-line comments            -z add zipfile comment
-@ read names from stdin            -o make zipfile as old as latest entry
-x exclude the following names      -i include only the following names
-F fix zipfile (-FF try harder)     -D do not add directory entries
-A adjust self-extracting exe       -J junk zipfile prefix (unzipsfx)
-T test zipfile integrity            -X eXclude eXtra file attributes
-y store symbolic links as the link instead of the referenced file
-e encrypt                          -n don't compress these suffixes
-h2 show more help
```


Príkazový riadok

- Typické aplikácie
 - Administrácia systémov (Unix/Linux)
 - Inžinierske aplikácie
 - Vedecké aplikácie
 - Ideálne pre nevidiacich a zrakovo obmedz.

Príkazový riadok

- Výhody
 - Efektívne využitie prepínačov
 - Vhodné pre expertov
 - Nízke požiadavky na zdroje systému
- Nevýhody
 - Potreba naučiť sa príkazy
 - Neviditeľné možnosti
 - Nevhodné pre začiatočníkov

Menu



Menu

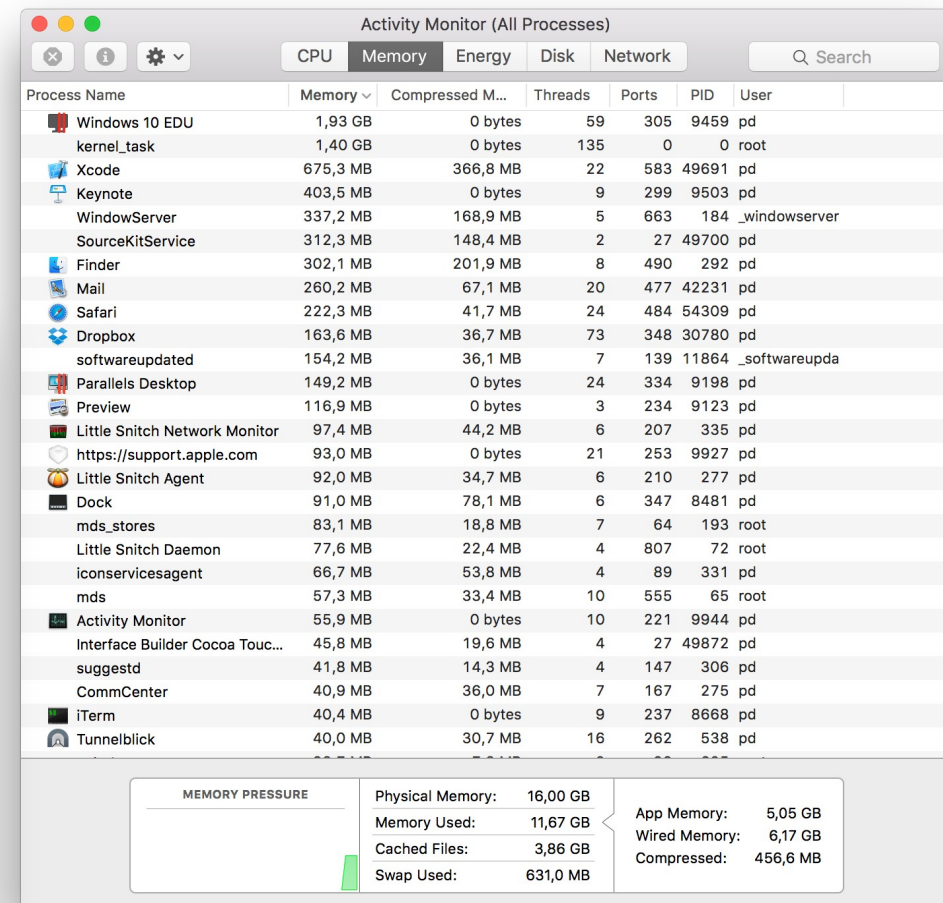
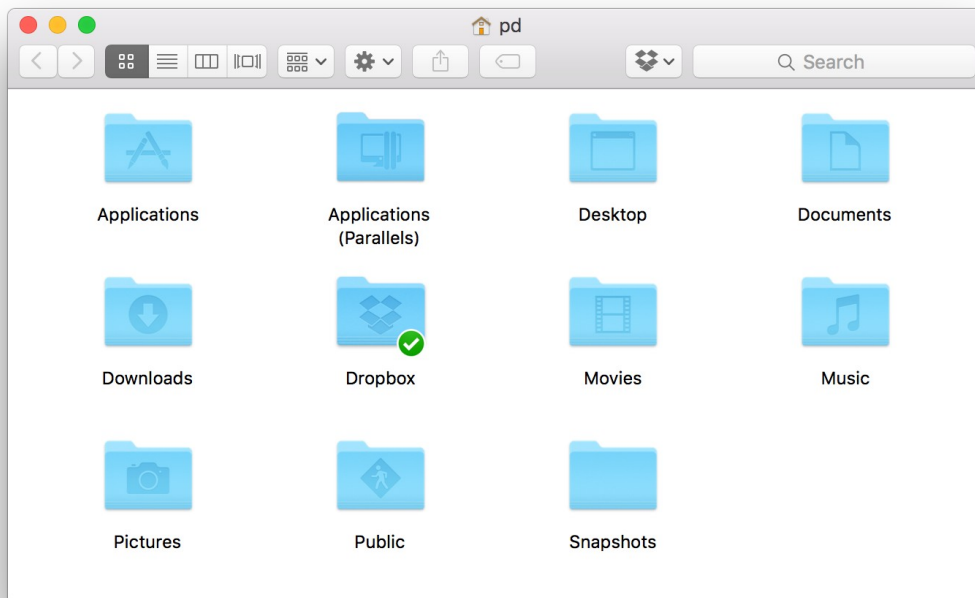
- Typické aplikácie
 - Bankomat
 - Klasické mobilné telefóny
 - Domáce spotrebiče
 - TV
 - Navigačné systémy

Menu

- Výhody
 - Nie je nutné učiť sa komplexné príkazy/jazyk
 - Ľahké použitie i pre nováčikov
 - Ideálne pokiaľ je množstvo volieb obmedzené
- Nevýhody
 - Pre skúsených používateľov zdĺhavé
 - napr. ak sa príkaz nachádza až v 5 podúrovni
 - Obmedzené rozmerom obrazovky a max. počtom položiek

Grafické rozhrania

- Vhodné pre začiatočníkov a pokročilých ale ...
- Vyššie systémové nároky
- Zložitejšia navigácia
- ...



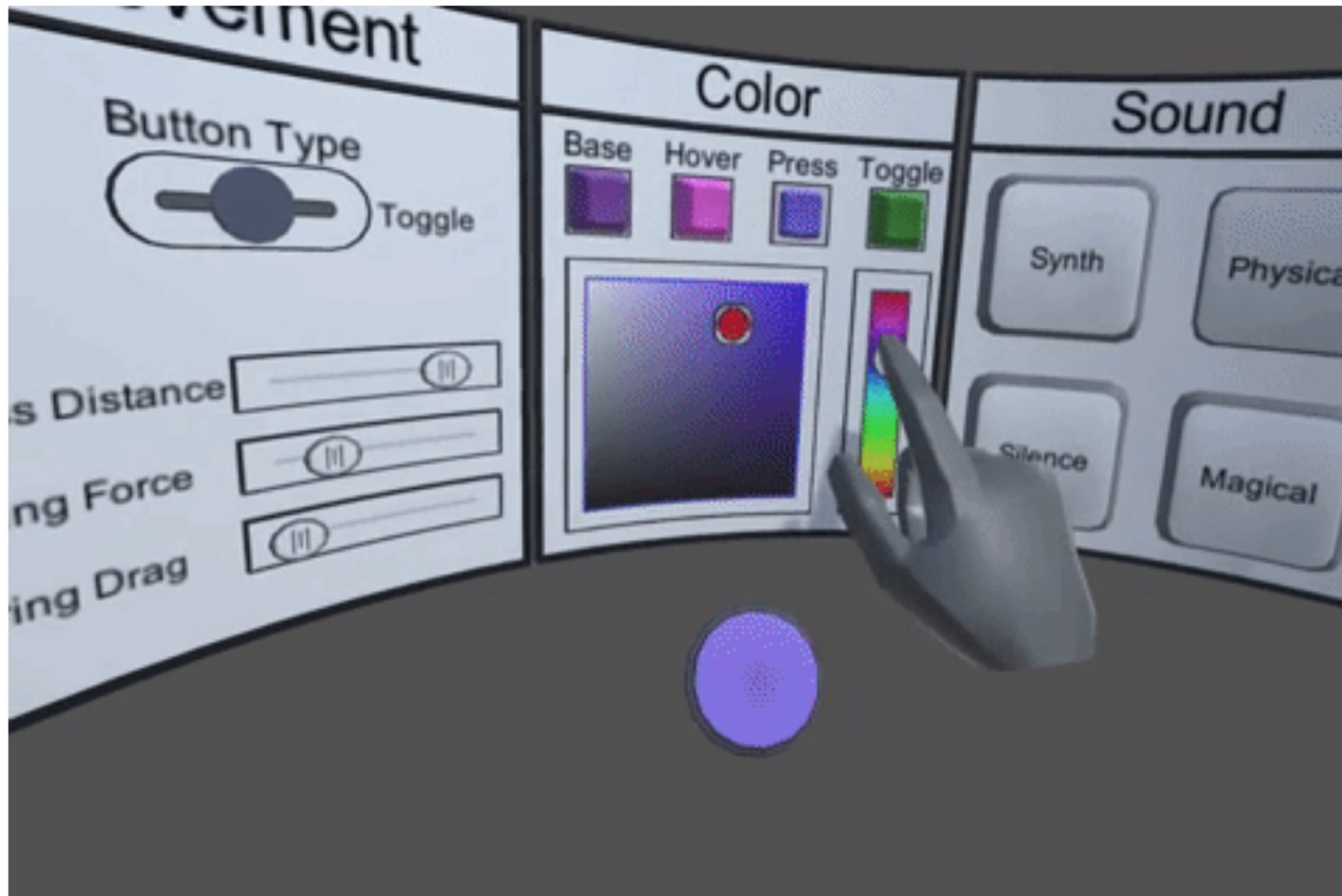
Grafické rozhrania

- Grafické rozhrania často využívajú spoločné prvky
 - Rozmiestnenie ovládacích prvkov
 - Názvy operácií
 - Ikony
 - Poradie a obsah menu
 - Operácie myšou
 - ...

Grafické rozhrania

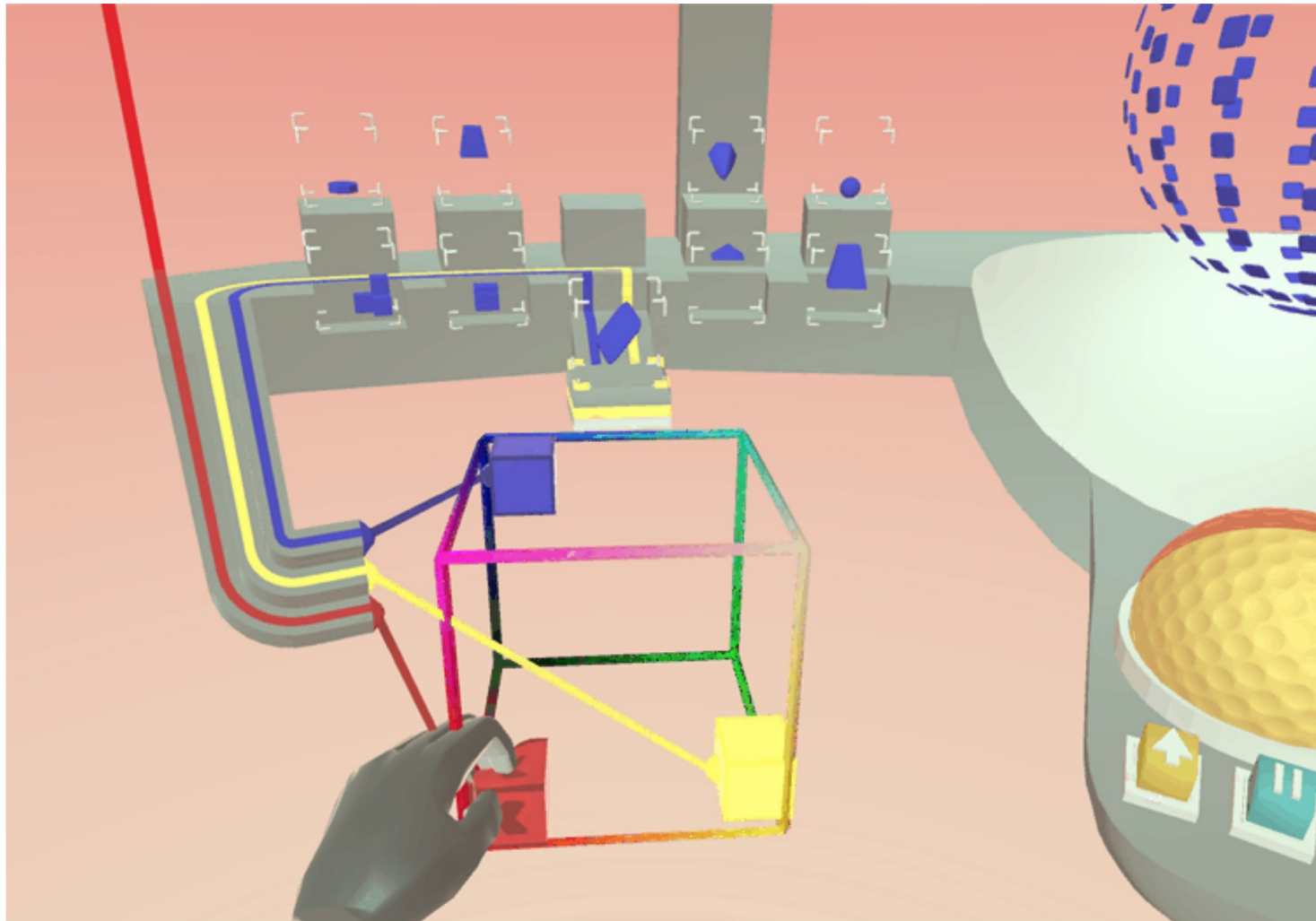
- Štandardizácia GUI ma za dôsledok
 - Zrýchlenie učenia sa používateľov
 - Jednoduchosť použitia aplikácií
 - Rozšírenie množstva riešiteľných problémov
 - Napomáha sebavedomiu nových používateľov
 - Väčšie množstvo užitoč. aplikácií pre používateľa

3D rozhrania vo VR a AR



<http://bit.ly/39nlclC>

3D rozhrania vo VR a AR



<http://bit.ly/2lekIZy>

Prirodzený jazyk

- Výhody
 - Nie je nutné učiť sa spôsob interakcie
 - Častokrát rýchlejšie než klávesnica
 - Nie je nutné použiť ruky
 - Ľahko použiteľné i pre ľudí s obmedzením

Prirodzený jazyk

- Nevýhody
 - Častokrát nedostatočná spoľahlivosť
 - Problém s nárečím a s niektorými jazyk. konštrukciami
 - Umelé jazyky a obmedzenia sú často presnejšie

Úvod do jazyka JavaScript

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 zobrazit'

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

Výstup

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

Základná syntax

- Oddelovanie 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 deklarovat'
- Používa sa na to kľúčové 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;  
    // platné 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>
```

Kľúčové slová

- *break* – prerušenie switch-u alebo cyklu
- *continue* – pokračovanie cyklu
- *debugger* – zastavenie vykonávania skriptu a zavolanie debugovacej funkcie
- *do ... while* – vykonanie bloku príkazov, pokiaľ je splnená podmienka
- *for* – cyklus
- *function* – deklarovanie funkcie
- *if ... else* – vetvenie na základe podmienky
- *return* – návrat z funkcie
- *switch* – vetvenie na základe prípadu
- *try ... catch* – zachytenie chybových stavov
- *var* – deklarovanie premennej

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 */; /* iteráčny  
    krok */)
{
    // KÓD
}
```

- avšak možno ho zapísať i 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 deklarovat' 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 deklarovat' i 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
```

HTML DOM (Document Object Model)

HTML DOM (Document Object Model)

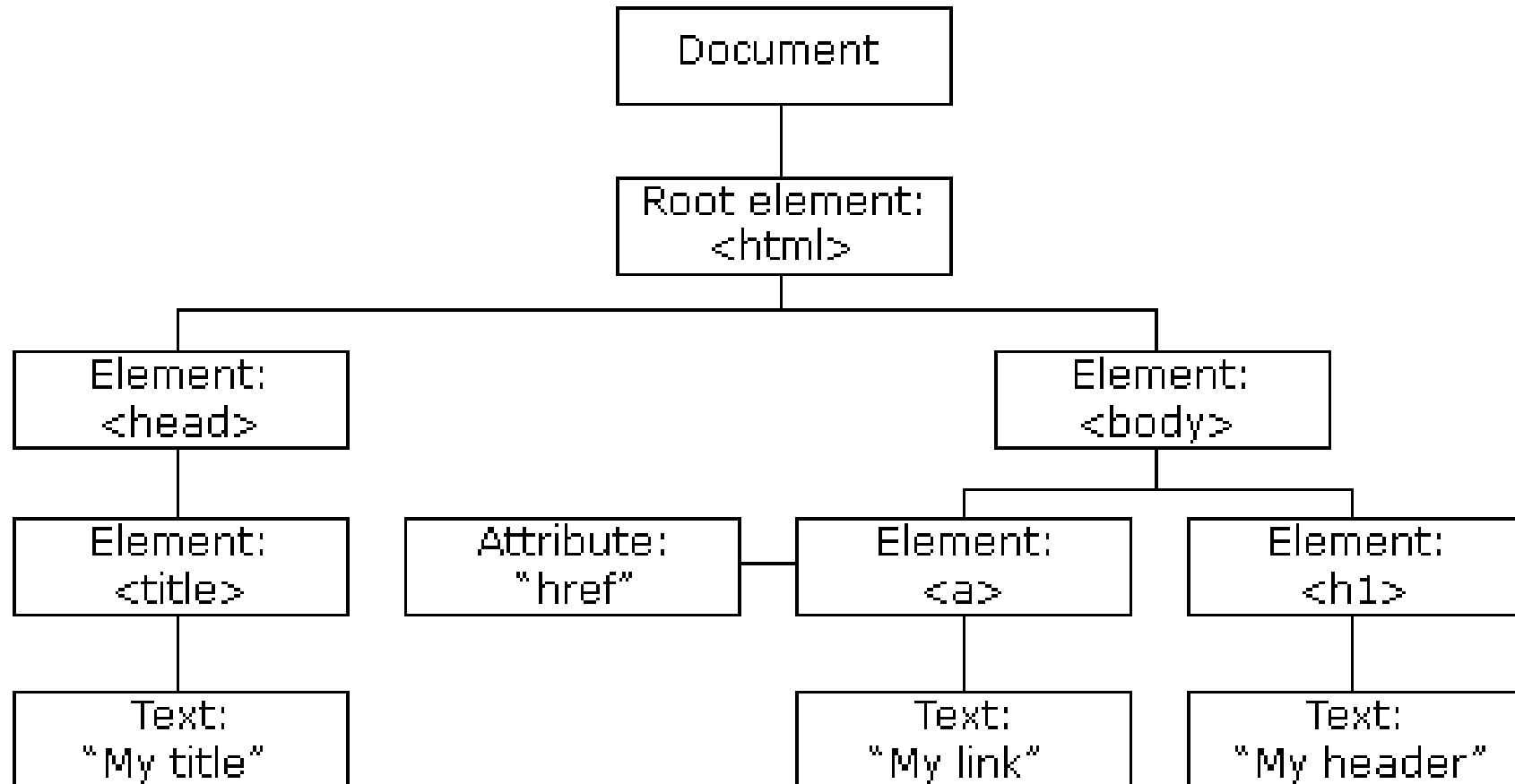
- HTML DOM je štandardný objektový model a programovacie rozhranie pre HTML
- Definuje:
 - Prvky HTML ako objekty
 - Vlastnosti všetkých prvkov HTML
 - Metódy prístupu ku všetkým prvkom HTML
 - Udalosti pre všetky prvky HTML
- HTML DOM je štandardom pre získavanie, zmenu, pridávanie alebo odstraňovanie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

HTML DOM (Document Object Model)

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

- Nájdienie 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>

<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 mazat'

```
<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
- Ak z daného elementu sa vieme dostať k ďalším uzlom stromu cez tieto vlastnosti uzla:
 - *parentNode*
 - *childNodes[nodenumbers]*
 - *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="0oops!";
}
</script>
</head>
<body>
<h1 onclick="changetext(this)">Click on this text!</h1>
</body>
</html>
```

Online Zdroje

- <http://www.w3schools.com/>
- ...

Odovzdanie - KB1

- KB1 - dokumentácia k návrhu hry
 - Návrh všetkých obrazoviek hry a menu
 - Popis základných cieľov a fungovania hry
 - Návrh ovládania hry
- Dokumentáciu odovzdať do AIS podľa pokynov v mieste odovzdania
 - Termín: do AIS do **08.03.2020** do **23:59**

Ďakujem za pozornosť