

Programska koda na odjemalcu

JavaScript

Programska koda na odjemalcu

- Programska koda, ki je vgrajena v samo spletno stran – HTML datoteko
- Izbiro programskega jezika določa izbira internetnega brskalnika
 - VBScript (Microsoft IE)
 - JavaScript (ECMA-262)
 - JScript (Microsoft): podpora na strežniku in odjemalcu
 - skriptni jezik (interpretirani jezik)
- Dopolnilo za atraktivnejši izgled spletne strani
 - Ključna funkcionalnost naj deluje tudi brez skript!

Osnove JavaScript jezika

- ❑ Prvi brskalnik s podporo za JavaScript je bil Netscape 2.0
- ❑ Sintaksa jezike je osnovana na programskem jeziku Java (Sun Microsystem)
- ❑ Opozorilo o napakah v kodi je na voljo šele med zagonom programa
- ❑ Pogoji za poganjanje JavaScript programov
 - Interpreter (scripting engine)
 - Gostujoče okolje (hosting environment)

- Najpogostejše naloge JavaScript-a
 - Dinamična vsebina spletnih strani
 - Odziv na akcije uporabnika
 - Spreminjanje vsebine HTML elementov
 - Pregledovanje in kontrola vnesenih podatkov pred pošiljanjem na strežnik
 - Delo s piškotki (cookies)

Osnove JavaScript jezika

- Vključevanje JS programov v HTML dokument
 - V glavo dokumenta: ponavadi skripte, ki se izvedejo, ko jih pokličemo

```
<head>  
  <script type="text/javascript">  
    ...  
  </script>  
</head>
```
 - V telo dokumenta: skripte se izvedejo ob nalaganju dokumenta

```
<body>  
  <script type="text/javascript">  
    ...  
  </script>  
</body>
```

Osnove JavaScript jezika

- Vključevanje JS programčkov v HTML dokument

- V glavo in telo hkrati
- Kot zunanjo datoteko

<head>

<script src="moji_programi.js"></script>

</head>

- Osnovna sintaktična pravila
 - Sintaksa je podobna C-jevskim jezikom
 - Občutljivost na male in velike črke
 - Podpičje na koncu vrstic je opcijsko
 - Posamezne bloke kode lahko združimo s pomočjo oklepajev

```
{  
...  
}
```
 - Komentarji v kodi: `//` ali `/*...*/`
 - Izpis v HTML: `document.write (..);`

□ Spremenljivke

- Spremenljivke nimajo tipov
- Opcijska uporaba besede **var** za najavo spr.

var x=5;

var oseba="Janez";

x=5;

var zaposlen=false;

- Najava konstant

const Pi = 3.14;

- Vrednost spremenljivke pripada enemu od tipov
 - Boolean, Number, String, object, undefined, null
 - Z ukazom **typeof** lahko ugotovimo tip spremenljivke

□ Tipi spremenljivk

■ Boolean

- Dve možni vrednosti: true ali false
- false: 0, -0, null, undefined, NaN, prazen string, false

■ Number

- Vsa števila so tipa float
- Cela števila (integer) so tipa float, a brez decimalne vejice
- Različni načini zapisov
 - cifra = 314.54; (decmalno)
 - cifra = 2.14e+5; (eksponentno)
 - cifra = 0xFE; (šestnajstiško)
 - cifra = 0377; (osmiško)

□ Tipi spremenljivk

■ String

- Dolžina je eden ali več znakov
- Ni tipa "char"
- Posebni znaki
 - \b brisanje (backspace)
 - \n nova vrstica (new line)
 - \" narekovaj
 - \\ leva poševnica (backslash)
 - ...

■ Null in Undefined

- Avtomatska pretvorba tipov spremenljivk
- Globalne in lokalne spremenljivke!!

□ Operatorji

■ Osnovne rač. operacije:

□ `+` `-` `*` `/`

□ `++` `--` (pozor: `a++` ali `++a`)!!

■ Prirejanje

□ `=` `+=` `-=` `*=` `/=`

■ Pogojno prirejanje: `x=(pogoj)?vr1:vr2;`

■ Znak `+` lahko uporabimo tudi za združevanje besed (tipov String)

■ Če združimo String in Number vedno dobimo String

■ Rezultat (napaka) nelogičnih operacij je `NaN`

- Primerjanje spremenljivk
 - Enakost: ==
 - Stroga enakost (tip in vrednost): ===
 - Neenakost: !=
 - < > >= <=
 - Logični operatorji
 - In (AND): &&
 - Ali (OR): ||
 - Ne (NOT): !

□ Pogojni stavki

if (*pogoj*)

{

...

}

else

{

...

}

if (x == 1)

{

x += 2;

}

Osnove JavaScript jezika

□ Odločitveni stavki

switch (***spremenljiva***)

```
{  
case 1:  
    ...  
    break;  
case 2:  
    ...  
    break;  
default:  
    ...  
}
```

switch(oseba)

```
{  
case "Janez":  
    document.write("Zdravo Janez");  
    break;  
case "Miha"  
    document.write("Zdravo Miha");  
    break;  
default:  
    document.write("Zdravo ???");  
}
```

Osnove JavaScript jezika

□ Ukazi za ponavljanje

- Omogočajo, da se določen kos kode ponovi poljubno število ciklov
- Ukaza
 - For: ponavljanje določeno število ciklov
 - While: ponavljanje dokler ni izpolnjen določen pogoj

```
ime: for (i=1; i<=10; i++)  
{  
  ...  
  break ime;  
}
```

```
i=1;  
while (i <=10)  
{  
  ...  
  i++;  
}
```

Osnove JavaScript jezika

□ Funkcije

- Funkcija loči oz. omeji določen del kode in se ta izvede le ob klicu
- Lahko je definirana v glavi ali telesu HTML dokumenta
- Lahko vrne vrednost ali pa tudi ne (lahko vrne več različnih vrednosti)
- Parameter funkcije je lahko funkcija
- Spremenljivke, določene v funkcijah so lokalne

```
function vsota(param1, param2)
{
    return param1+param2;
}
```

```
function izpis(besedilo)
{
    document.write(besedilo);
}
```


□ Funkcije

■ Privzete oz. vgrajene funkcije

- parseInt(), parseFloat()

Pretvorba texta (string) v številko (number)

- isNaN(), isFinite()

Preverjanje napačnih rezultatov mat. operacij

- encodeURIComponent(), decodeURIComponent()

Kodiranje oz. dekodiranje URL zapisa –
odstranjevanje nedovoljenih znakov v URL zapisu

`www.blabla.com/Moji dokumenti/pomembno!.doc`

`www.blabla.com/Moji%20dokumenti/pomembno%21.doc`

REZERVIRANE BESEDE

abstract
boolean **break** byte
case catch char class const **continue**
debugger **default delete do** double
else enum export extends
false final **finally** float **for function**
goto
if implements import **in instanceof** int
interface
long
native **new null**
package private protected public
return
short static super **switch** synchronized
this throw throws transient **true try typeof**
var volatile **void**
while with

□ Objekti

- JavaScript je v svoji osnovi objektni jezik
- Objekt vsebuje različno število lastnosti (**property**) in metod (**method**)
- Sintaksa za dostop do lastnosti in metod:
`ime_objekta.lastnost; ime_objekta.metoda();`
- Napredne lastnosti objektov
 - Dedovanje, prototipi, konstruktorji...

Osnove JavaScript jezika

- Definicija objekta

```
var moj_objekt = new Object();
```

```
var moj_objekt = {a:2; ime:"Jaka"; b:null};
```

- Lastnosti objekta

```
moj_objekt.a = ...
```

ali

```
moj_objekt["a"] = ...
```

- Metode objekta

```
function metoda(){  
    ...  
}
```

```
var objekt = new (Object);  
objekt.a = 1;  
objekt.nekaj = metoda;
```

- Metode objekta

```
var objekt = new (Object);  
objekt.a = 1;  
objekt.nekaj = function () {  
    ...  
};
```

Osnove JavaScript jezika

□ Definicija objekta avto

```
avto={};  
avto.barva="rdeča";  
avto.letnik="2001";  
avto.prebarvaj = function (barva) {  
    this.barva=barva;  
}
```

□ Primer objekta avto

```
avto.znamka="Ferrari";  
avto.barva="rdeča";  
avto.letnik="2001";  
avto.vozi();  
avto.prebarvaj("zelena");
```

- Nizi (arrayi)

```
var niz1 = new Array();
```

```
var niz2 = new Array(1, "Jaka", true);
```

```
var niz3 = [1, "Jaka", true];
```

```
niz1[0]=...
```


- Nizi (arrayi)

```
niz4 = [[ "A", "B", "C"],  
        [ "a", "b", "c"],  
        [1, 2 ,3]];
```

- Lastnost **length**

```
var dolzina = niz.length;  
niz.length = 2;
```

- Nizi (arrayi)
 - Metode objekta Array
 - **toString()**: vrne niz kot string, elementi so ločeni z vejico
 - **push(nov_element)**: doda element
 - **pop()**: vrne zadnji element in ga odstrani
 - **shift()**: vrne prvi element in ga odstrani

□ Vgrajeni JavaScript objekti

□ window

□ String

- Lastnosti: length...
- Metode: big(), bold(), fontcolor(), indexOf(), toUpperCase(), charAt(cifra), concat(beseda), ...

□ Date

- Metode: Date() – današnji datum in ura
getDate(), getHours() – dan, ura, ...
setDate(), setHours() – nastavljanje dni, ur, ...

□ Boolean

□ Array

□ Math

- Lastnosti: E, LN2, PI, SQRT2, ...
- Metode: abs(), sin(), cos(), sqrt(), round(), random(), ...

- Kontrola nad morebitnimi napakami v kodi
 - `try ... catch` stavek (`throw`)

```
try
{
    ...
}
catch(napaka)
{
    ...
    ... napaka ...
}
```

```
try
{
    throw("Moja napaka");
}
catch (npek)
{
    ... npek ...
}
```

- Kontrola nad morebitnimi napakami v kodi
 - **onerror** dogodek

```
onerror=ObdelajNapako;
```

```
...
```

```
...
```

```
function ObdelajNapako(opis, naslov, vrstica)
```

```
{
```

```
...
```

```
... opis ...
```

```
}
```

- Opozorilna okna oz. okna za komunikacijo z uporabnikom
 - Opozorilo
`alert("Besedilo opozorila");`
 - Potrditev (
`confirm("Vprašanje uporabniku");`
 - Vrne **true** ali **false**
 - Vnosno polje
`prompt("Spremno besedilo", "Privzeti vnos");`
 - Vrne vneseno besedilo ali null

- Dogodki za klicanje programčkov
 - Akcije uporabnika, ki jih zazna brskalnik in pokliče predviden JavaScript program
 - Dogodke vključimo v posamezne HTML elemente
``
 - Nabor dogodkov
 - onload, onunload
 - nalaganje in zapiranje spletne strani
 - onfocus, onblur
 - Ko nek element pridobi ali izgubi "focus"
 - onchange, onscroll
 - Ko se spremeni vsebina nekega polja

- Dogodki za klicanje programčkov
 - Nabor dogodkov
 - onclick, ondblclick, onmousedown, onmouseup, onmousemove, onmouseout, onmouseover
 - Različne akcije uporabnika s pomočjo miške
 - onkeydown, onkeyup, onkeypress
 - Različne akcije uporabnika s pomočjo tipkovnice
 - onresize
 - Ko je spremenjena velikost okna
 - onselect
 - Ko je izbran besedilo nekega elementa
 - onsubmit, onreset
 - Ob pritisku gumba submit ali reset

□ Piškotki

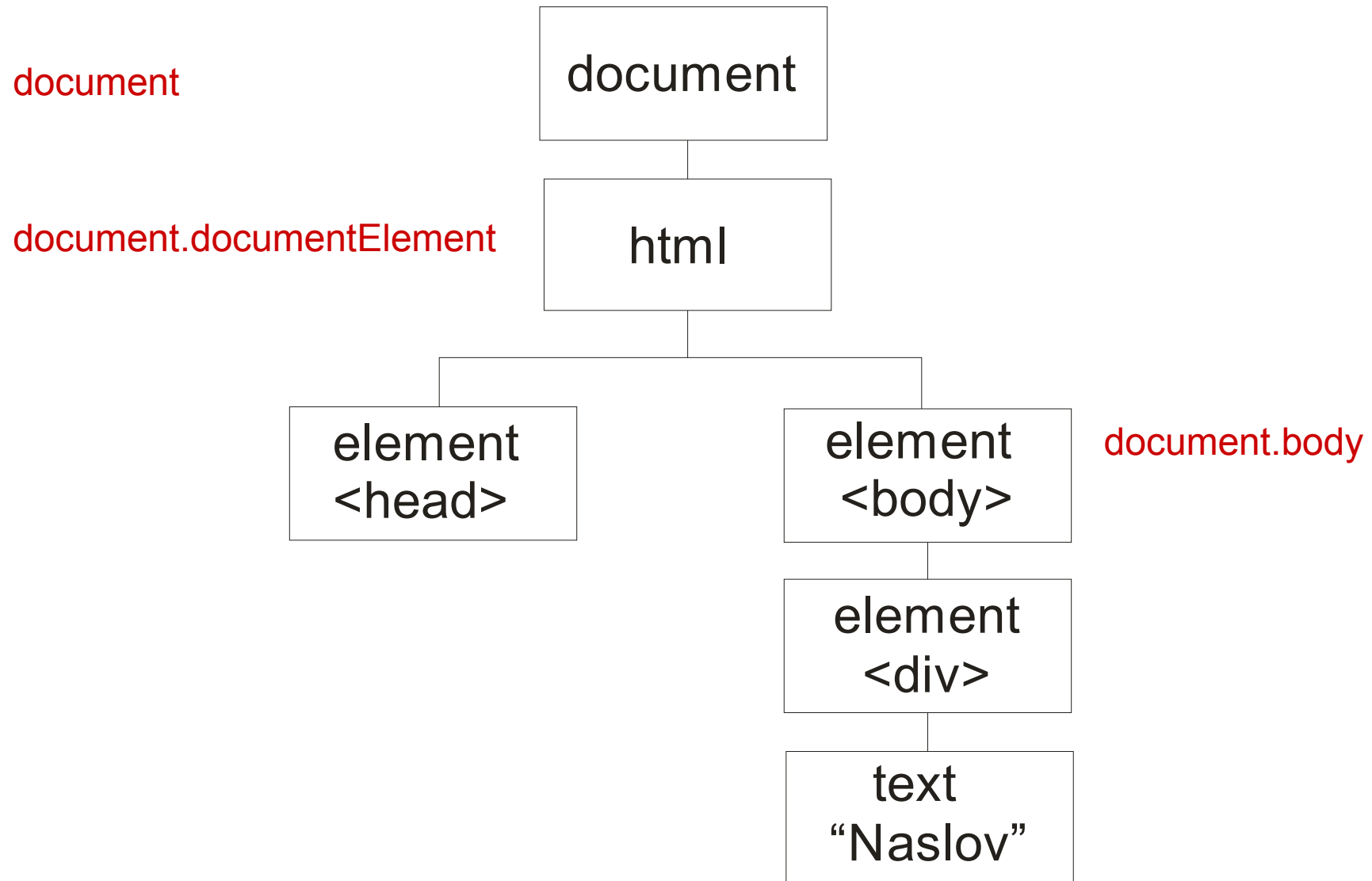
```
document.cookie = 'moj_piskotek=vrednost;  
expires=Thu, 2 Aug 2001 20:47:11 UTC;  
path=/'
```

Document Object Model - DOM

Spreminjanje HTML dokumenta s pomočjo JavaScript programov

- HTML Document Object Model – HTML DOM
 - Dostop in spreminjanje strukture in vsebine HTML dokumenta
 - Povezava med JavaScript programi in HTML dokumentom, ki je trenutno prikazan v brskalniku
 - HTML dokument je predstavljen kot objektna drevesna struktura (elementi, lastnost, besedilo)
 - W3C DOM – poenoten model v vseh brskalnikih
 - Osnovni gradnik DOM-a je objekt **document**, ki predstavlja koren drevesa in je ustvarjen s strani brskalnika (podobno kot alert in prompt)

HTML DOM



- Povezave med vozlišči v drevesu
 - Izhodiščno vozlišče oz. koren drevesa se imenuje root – **document**
 - Vsako vozlišče razen korena ima natanko enega starša (**parent**)
 - Vsako vozlišče ime lahko poljubno potomcev (**children**)
 - Vozlišče brez potomcev imenujemo list (leaf)
 - Vsa vozlišča, ki imajo istega starša (bratje) se imenujejo **siblings**

HTML DOM

□ Primer DOM drevesa za HTML datoteko

```
<html>
  <head>
    <title>Moja primer</title>  <!-- tole je naslov -->
  </head>
  <body>
    <h1>Drevesna struktura HTML dokumenta</h1>
    <p>
      <b> Tole je pomembna vaja! </b> <br />
      <a href="www.najdi.si"> Link na NAJDI </a>
    </p>
  </body>
</html>
```

- Osnovni tipi vozlišč v DOM drevesu
 - Element (**element**) – 1
 - Node.ELEMENT_NODE
 - Lastnost (**attribute**) – 2
 - Node.ATTRIBUTE_NODE
 - Besedilo (**text**) – 3
 - Node.TEXT_NODE
 - Komentar (**comment**) – 8
 - Node.COMMENT_NODE
 - Dokument (**document**) - 9
- Vsa besedila v HTML datoteki so predstavljena kot vozlišča tipa **text**

- Dostop do vozlišč v drevesu
 - `document.getElementById(id_vozlisca)`
 - Vrne objekt vozlišča, ki ustreza kriteriju
 - `vozlisce.getElementsByTagName(ime_vozlisca)`
 - Vrne polje (array) vozlišč, ki ustrezajo kriteriju
 - `document.getElementsByTagName(HTML_znacka)`
 - Vrne polje (array) vozlišč, ki ustrezajo kriteriju
 - Klicanje metod na podlagi relacij vozlišč v drevesu
 - `firstChild`, `lastChild`, `parentNode`, `previousSibling`, `nextSibling`
 - Metode vrnejo objekt vozlišča
 - `childNodes`
 - Vrne polje(array) vozlišč – otrok
 - `document`, `document.documentElement` in `document.body`

- Uporaba DOM elementov v programski kodi
 - Vsak DOM element je predstavljen kot objekt
 - Nabor lastnosti (property) in metod (method)
 - Lastnosti
 - `nodeType`: tip vozlišča (1, 2, 3 .. 9)
 - `nodeName`: HTML značka
 - `nodeValue`: besedilo (samo v text elementih)
 - `innerHTML`: vsa HTML vsebina znotraj elementa
 - `parentNode`: starš
 - `childNodes`: set otrok
 - `previousSibling`: naslednji brat
 - `nextSibling`: prejšnji brat
 - `attributes`: lastnosti znotraj HTML značke

- Informacije o posameznem vozlišču
 - **nodeName**
 - Lastnost lahko le preberemo (read-only)
 - Tri možne vrednosti
 - HTML značka vozlišča (FORM, DIV, P, itd.)
 - **#text**: kadar gre za vozlišče tipa text
 - **#document**: kadar gre za vozlišče tipa document
 - **nodeValue**
 - Lastnost lahko beremo ali pišemo
 - Dve možni vrednosti
 - Besedilo (kadar gre za vozlišče tipa text)
 - Lastnost (kadar gre za vozlišče tipa attribute)

- Informacije o posameznem vozlišču
 - **nodeType**
 - Številka, ki ustreza tipu vozlišča: 1, 2, 3, 8, 9, ??
 - **innerHTML**
 - Celotna HTML koda, ki se nahaja znotraj značk, ki določata neko vozlišče

- Uporaba DOM elementov v programski kodi
 - Metode
 - getElementById(id)
 - getElementsByName(ime)
 - getElementsByTagName(ime)
 - hasAttributes()
 - hasChildNodes()
 - appendChild(vozlisce)
 - removeChild(vozlisce)
 - insertBefore(vozlisce1, vozlisce2)
 - replaceChild(vozlisce1, vozlisce2)
 - setAttribute(ime, vrednost)

- Ustvarjanje novih vozlišč

```
document.createElement(znacka);
```

```
document.createTextNode(text);
```

```
node.cloneNode();
```

```
node.cloneNode(true);
```

- klonira tudi vse otroke

- Posebni DOM objekti
 - window
 - Predstavlja okno brskalnika in s tem koren HTML drevesa
 - Glavne lastnosti
 - document, location, status, itd.
 - Glavne metode
 - alert(), blur(), close(), focus(), scrollTo(), itd.
 - http://www.w3schools.com/jsref/obj_window.asp

- Posebni DOM objekti
 - Navigator
 - Informacije o brskalniku
 - Glavne lastnosti
 - `appName`, `appVersion`, `userAgent`, itd.
 - Screen
 - Informacije o zaslonu odjemalca
 - Glavne lastnosti
 - `availHeight`, `availWidth`, `colorDepth`, itd.
 - History
 - Omogoča prikaz zgodovine in navigacijo na strani iz zgodovine brskalnika
 - `back()`, `forward()`, `go()`
 - Location

- Objekti za nastavljjanje lastnosti vseh vrst HTML elementov
 - Anchor, Body, Button, Form, Image, Select, itd.
 - Vsak objekt ima nabor specifičnih lastnosti in metod
 - Referenca: w3schools

- Spreminjanje stila (CSS)
 - Vsakemu objektu lahko nastavljamo in spreminjamo lastnosti stila preko lastnosti **style**

```
document.getElementById("moj_element").style.backgroundColor="blue";
```

```
document.getElementById("moj_element").style.zIndex="2";
```

- Referenca: w3schools

Asinhroni JavaScript in XML AJAX

AJAX

- Osnovne značilnosti
 - Dopolnjuje klasični način komunikacije brskalnika s strežnikom (preko HTML obrazcev - POST in GET)
 - Komunikacija brskalnika s strežnikom z uporabo JavaScript programov
 - Asinhroni način – brez ponovnega nalaganja strani
 - Pogoji za delovanje AJAX tehnologije je podpora s strani brskalnika
 - Ni samostojni standard, temelji na:
 - JavaScript
 - XML
 - HTML
 - CSS

- Objekt XMLHttpRequestObject
 - Posebni JavaScript objekt, ki omogoča asinhrono komunikacijo s strežnikom
 - Podprt v brskalnikih
 - Internet Explorer 5.5+
 - Safari 1.2
 - Mozilla 1.0+
 - Opera 8+
 - Vsak brskalnik uporablja omenjeni objekt na svoj način
 - Pri pisanju aplikacije moramo zagotoviti podporo za vse vrste brskalnikov

AJAX

- Inicializacija XMLHttpRequest objekta

```
function ajaxFunction() {  
    var xmlhttp;  
    if (window.XMLHttpRequest)  
    {  
        // podpora za IE7+, Firefox, Chrome, Opera, Safari  
        xmlhttp=new XMLHttpRequest();  
    } else {  
        // podpora za IE6, IE5  
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  
    }  
}
```

- Objekt XMLHttpRequestObject
 - Lastnost ***onreadystatechange***
 - Lastnost mora določati funkcijo, ki se pokliče, ko pridejo zahtevani podatki iz strežnika

```
xmlHttp.onreadystatechange=function()  
{  
  // Programska koda, ki obdela podatke  
}
```

□ Objekt XMLHttpRequestObject

■ Lastnost **readyState**

- Vsebuje informacijo o statusu odziva strežnika
 - Možne vrednosti so: 0,1,2,3,4 (4 pomeni končan odziv strežnika)
- Vedno se spremeni istočasno kot onreadystatechange

```
xmlHttp.onreadystatechange=function()  
{  
    if (xmlHttp.readyState==4 && xmlHttp.status==200){  
        // Odziv strežnika končan, lahko beremo podatke  
    }  
}
```

- Objekt XMLHttpRequestObject
 - Lastnost **responseText**
 - Vsebuje dejanski odziv strežnika (informacijo, ki jo vrne skripta na strežniku)

```
xmlHttp.onreadystatechange=function()  
{  
    if (xmlHttp.readyState==4 && xmlHttp.status==200){  
        {  
            izpis = document.getElementById("moj_element");  
            izpis.innerHTML = xmlHttp.responseText;  
        }  
    }  
}
```


- Objekt XMLHttpRequestObject
 - Pošiljanje zahteve na strežnik (metoda GET)
 - Ukaz **open** (*metoda, skripta na strežniku, asinhroni način*)
 - Ukaz **send** (*parametri*)

xmlHttp.open("GET","skripta.asp",true);

xmlHttp.send(null);

xmlHttp.open("GET","skripta.asp?param1=2¶m2=joco",true);

xmlHttp.send(null);

AJAX

- Pošiljanje zahteve na strežnik (metoda POST)

```
xmlhttp.open("POST","ajax_test.asp",true);  
xmlhttp.setRequestHeader("Content-type","application/x-www-form-  
urlencoded");  
xmlhttp.send("fname=Henry&lname=Ford");
```