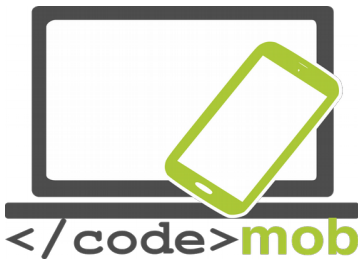
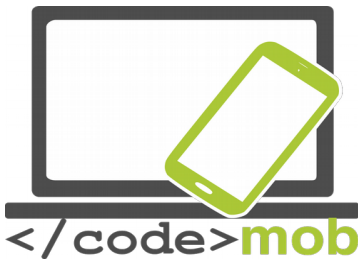
Programiranje

CodeMob: Programiranje
Listopad 2017. http://codemob.eu/**. Izradio:**

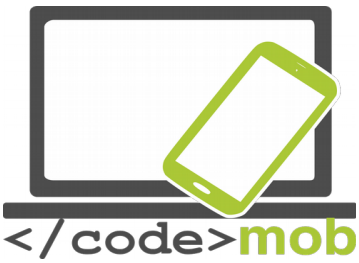
Co-funded by the
Erasmus+ Programme
of the European Union

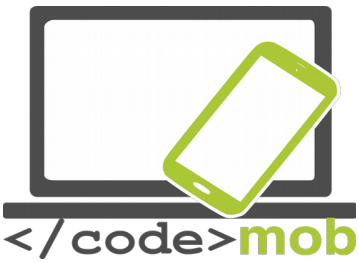


This publication has been co-funded by the European Commission’s Ersamus+ Programme.



Sadržaj





Dobrodošli u tečaj programiranja !

Nakon završetka ovog tečaja biti ćete u stanju :

- napraviti igru (npr.memory) u Javascriptu.
- Napraviti GUI (grafičko korisničko sučelje) sa HTML 5 I CSS 3.
- Razumijeti i objasniti koncepte osnove programiranja.

Prije svega imajte na umu da postoji na stotine resursnih materijala online. Bilo bi gotovo nemoguće unijeti ih sve u ovom priručniku.
Kako god u programiranju sve odgovore i rješenja možete pronaći na webu.

Zamislite ovaj trening kao uvod,korak po korak u Javascript.Uživajte!

Zašto učiti JavaScript?

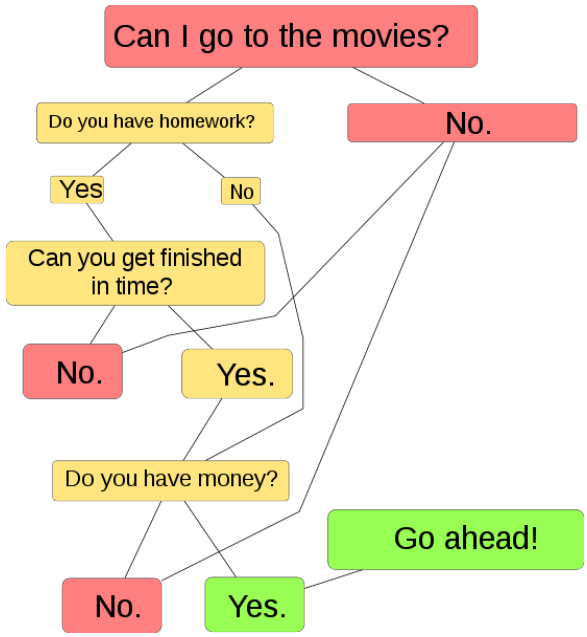
Da se ne bi zbunili s Javom,Javascript nam omogućuje izradu interaktivnih web stranica. Uz HTML i CSS, Javascript je postao osnova web tehnologija i upravo zbog toga većina tražilica radi kroz Javascript. Javascript moramo naučiti kako bismo se ozbiljno bavili razvojem web stranica ili ako planiramo karijeru graditi kao front end developer ili jednostavno koristiti Javascript kao back end platformu.

Javascript se u današnje vrijeme proširila se na razvoj mobilnih aplikacija ,desktop aplikacije I razvoj igara. Sve u svemu opće je popularan i jedna od poželjnih vještina.

Izvor: *http://www.bestprogramminglanguagefor.me/why-learn-javascript*

+Algoritmi

U matematici I informatičkoj znanosti algoritam je samo održivi red korak po korak operacija koja će se izvršiti. Algoritmi izvode kalkulacije procesiranje podataka i/ ili automatsko izvođenje zadataka.



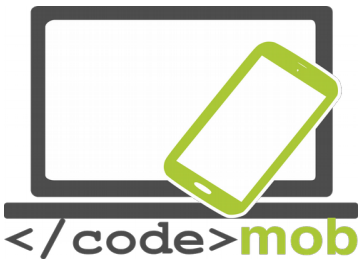
Ovdje je primjer što algoritam može biti :

Koja je poveznica između algoritma I programa?

Na algoritam možemo gledati kao na radnu formulu. Program je niz instrukcija za računalo koje koristi algoritme da bi izvršilo željeni zadatak.

Pokušajmo razumijeti to kroz primjer :

Jednostavni algoritam površine trokuta A=dužina x dubina
Program za izvršenje izračuna površine bio bi
Korisnik unosi
dužinu Korisnik
unosi dubinu
Koristi algoritam za površinu za izračun površine
Prikaži rješenje



Uvod u HTML i CSS

Kako bi načinili web stranicu, moramo ustupiti određene podatke računalu. Nije dovoljno samo unijeti tekst koji će biti prikazan na stranicama, potrebno je znati kako pozicionirati tekst,umetnuti sliku,napraviti poveznicu I sl. Da bismo objasnili računalu što točno želimo potrebno je koristiti jezik koji će računalo prepoznati.

Postoje programski jezici koji služe za programiranje kao što su C++ ili Java. Ovi programski jezici su dosta komplicirani I zahtjevaju puno više vještina I znanja.

HTML I CSS se koriste kao alati za izradu web stranica, I načinjeni su kako bi se što jednostavnije koristili. Svaki od ova dva jezika koriste se za specifične namjene i savršeno se nadopunjuju kako bi finalizirali našu web stranicu.

- HTML : je kratica Hyper Text Markup Language a prvi puta se pojavio 1991 kada je lansiran Web. Uloga HTML-a je uređenje i organizacija sadržaja. Zato ćemo u HTML-u pisati sve što želimo da bude prikazano na stranici :

tekst,linkovi,slike… pa ćemo reći npr: ovo je moj naslov,ovo je moj meni,ovo je glavni tekst moje stranice….

Ova predavanja bazirati će se na zadnjoj verziji HTML-a (HTML 5) koji je već danas jezik budućnosti koji bi svatko od nas trebao koristiti.

• CSS : je kratica od Cascading Style Sheets. Ovaj jezik koristimo kako

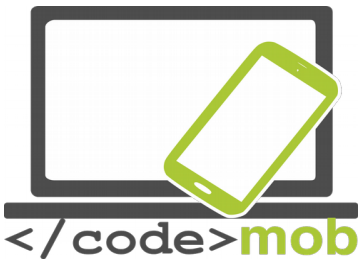
bimo uredili web stanice. U CSS-u ćemo naprimjer reći: “ moji naslovi su crveni I podvučeni, moj tekst je arial font,moje ime je centrirano, moj izbornik ima bijelu pozadinu. …” itd…

S ovim jezikom lako I brzo ćemo kreirati osnovni izgled naše stranice.

Editor

Pitanje koje trebamo postaviti je sljedeće : “ koji program ću koristiti za

izradu web stranice?”



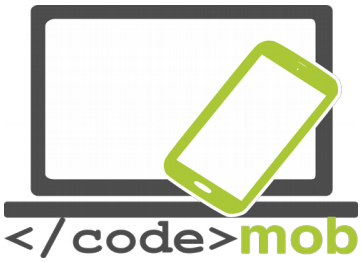
Notepad je dovoljan da bi kreirali web stranicu! No da bismo ubrzali naš posao koristimo editore kao što su Notepad ++ (prednost je da automatski oboji HTML/CSS kod).

Preglednici

Što je preglednik?

Preglednik je program da pregledavamo web stranice.Posao preglednika je da prepozna I pročita HTML/CSS kod koji smo napisali, I da ga pravilno prikaže. Ako CSS kod kaže “naslov je crveni” tada bi preglednik trebao prikazati naslove u crvenoj boji.

Među mnogobrojnim preglednicima evo nekih koji su najprepoznatljivi: Internet Explorer,Mozilla Firefox,Opera,Netscape,Konqueror (za Linux) ,Linx (za Linux) , Apple Safari (za Mac) itd…



Kreirajte svoj prvi HTML document

Što je HTML?

Kao što smo rekli ranije HTML je opisni jezik koji opisuje web document (web stranicu)

- - HTML je kratica od Hyper Text opisni jezik
- - Opisni jezik je set opisnih oznaka (tagova)
- - HTML dokument je opisan HTML oznakama
- - Svaka HTML oznaka označava različiti sadržaj u dokumentu

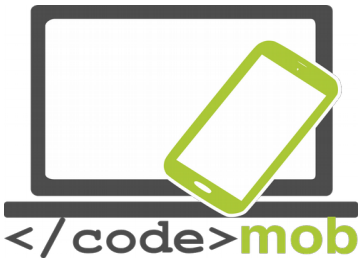
Mali HTML dolument

Primjer

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

 <h1>Moj prvi naslov</h1>
 <p>Moj prvi paragraf</p>

</body>
</html>
```

Objašnjenje primjera

- The <!DOCTYPE html> definiramo da je ovo HTML5 dokument
- The text between <html> and </html>opisujemo HTML dokument
- The text between <head> and </head> informacija o dokumentu
- The text between <title> and </title> naslov dokumenta
- The text between <body> and </body> vidljivi sadržaj dokumenta



- The text between <h1> and </h1> postavljanje naslova
- The text between and postavljanje paragrafa

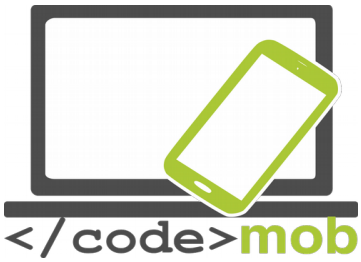
Koristeći ove opise web preglednik će prikazati dokument sa naslovom I zaglavljem.

Za više informacija potražite na sljedećem linku w3schools.

HMTL elementi koji su vam potrebni

U ovoj lekciji vaš zadatak će biti kreirati jednostavan HTML dokument koji sadrži sljedeće:

- Naslov
- Tekst paragrafa



- Ftotgrafija
- Poveznica

Ne zaboravite koristiti komentare

Komentar (comment) je **HTML tag** sa posebnim izgledom.

<!-- ovo je komentar -->

Komentare možete smjestiti bilo gdje u vašem kodu: nema utjecaj na vašu stranicu ali možete koristiti komentare kako bi objasnili vaš kod,koji vam može pomoći ako kasnije želite mjenjati nešto.

Ovo može biti korisno ako imate puno koda.

Upozorenje! Svatko može vidjeti vaš HTML kod nakon što ste postavili stranicu na web. Potrebno je samo kliknuti desni klik na stranicu I odabrati “ Show the source code of the page”. Zbog toga ne ostavljajte osjetljive infromacije kao što su zaporke u komentarima I pazite na svoj kod.

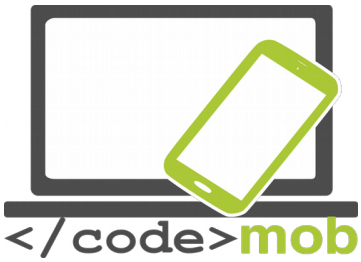
Kreirajte CSS dokument I koristite ga kako bi uredili vaš HTML dokument

Što je CSS?

- CSS znači Cascading Style Sheets
- CSS opisuje kako će HTML elementi biti prikazani na ekranu,papiru ili drugom mediju
- CSS nam štedi puno posla . Može kontrolirati slojeve više web stranica odjednom.
- Vanjski stilovi pospremaju se u zasebnom CSS dokumentu

Zašto koristiti CSS?

CSS koristimo kako bi definirali stil stranice,uključujući dizajn,slojeve I varijacije u izgledu za razlilite uređaje I različite veličine ekrana



CSS rješava veliki problem

Namjena HTML-a nikada nije bila sadržavati oznake za formatiranje web stranica!HTML je načinjen za opis sadržaja web stranice , kao :

<h1>Ovo je naslov </h1>

Ovo je paragrapf

Gdje oznaka <font/> I oznake boje su dodane u HTML 3 2, tada je počela noćna mora web kreatora.

Dizajneri velikih web stranica gdje se informacija o fontovima I bojama morala dodavati na svakoj stranici postala je dug I skup proces.

Kako bi rješili ovaj problem WWW konzorcij (w3c) načinio je CSS. CSS je ukinuo stilove formatiranja za HTML stranice!

CSS štedi puno vremena!

Stilovi koji su definirani obično se pospremaju u vanjskom CSS dokumentu.

Sa vanjskim opisnim dokumentom možete mjenjati izgled cijele web stranice mjenjajući pritom samo jedan dokument.

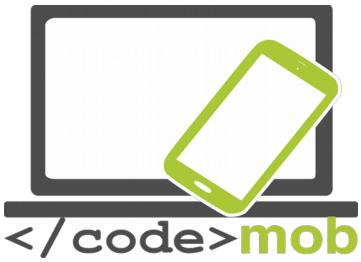
Za više informacija posjetite w3schools.

Vanjski dokumenti stilova

Kada preglednik pročita opisni dokument urediti će HTML dokument prema informacijama opisnog dokumenta (style sheet) .

Tri načina unosa CSS-a

Postoje tri načina na koja možemo unjeti opisne obrasce (style sheet)



1. Vanjski opisni obrazac
2. Unutarnji opisni obrazac
3. Opisni obrazac unutar linije

U ovoj lekciji koristiti ćemo vanjski opisni dokument (style sheet)

Sa vanjskim style sheet-om možete mjenjati kompletan izgled web stranice mjenjajući samo jedan dokument

Svaka stranica mora sadržavati poveznicu na vanjski dokument unutar link elementa <link/>element je sadržan unutar <head/> dijela:

Primjer:

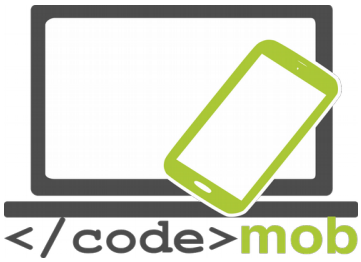
```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Vanjski CSS dokument može biti napisan u text editor.Dokument nesmije sadržavati HTML oznake. CSS dokument mora biti pospremljen sa .css ekstenzijom.

Evo kako “my Style css” izgleda:

```
body {
    background-color: lightblue;
}

h1 {
    color: navy;
    margin-left: 20px;
}
```

Napomena: Nemojte dodavati razmak između vrijednosti I jedinica (kao što su margina,left: 20 px;)točan način je : margin – left: 20 px;



Programiranje JS

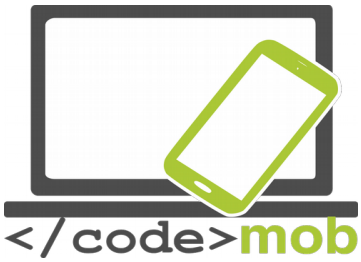
Povijest Javascripta

Programski jezik JavaScript nastao je 1995 a stvorio ga je Brendan Eich (Fundacija Mozilla) od strane Netscape-a. Trenutna verzija jezika 1.8.5 je treća verzija ECMA262 standarda.

- - U prosincu 1995, Sun I Netscape najavljuju izdanje JavaScript-a. Odgovor Microsofta bi je u obliku JScrip-ta (isti jezik različitog naziva kako bi se izbjegla autorska prava)
- 1996 Netscape šalje JavaScript Ecma-u na potvrdu međunarodnog standardiziranja
- **DHTML** (1996), težak I nepotrban start
- **Ajax** (2000), novi interes zahvaljujući novim mogućnostima
- **JSon:** kao alternativa JavaScript baziranom standard XML-a
- **JavaScript** frameworks, jednostava, produktivan I standardiziran ali težak izbor
- **Node.js**; Eventdriven i Serversided
  - NodeWebkit i Cordova

Što je JavaScript?

- Skriptni jezik (visoko standardizirani jezik)
  - Preveden (kao suprotnost compiled jeziku)
- Korisnički orjentirani jezik (program se provodi na računalu korisnika a ne na poslužiteljima)
  - Node.js se pojavljuje 2009 I mijenja neke uvijete
- Mogućnost mijenjanja DOM-a:
  - Kreirati, mijenjati, brisati HTML elemente, njihove attribute ili CSS



- Kreirati, pratiti I brisati događanja

- Nema poveznice sa Sun-ovim Java jezikom ;)

Primjer - http://www.piskelapp.com/ & the Polyfills

Instalacija

Skripta može biti postavljena u bilo kojoj HTML stranici u oznaci (tag-u) <head> ili oznaci <body> važno je skriptu umetnuti prije zatvaranja oznake </body> skripta može biti unutar HTML-a ili vanjska (zasebni dokument). Atribut vrste je izboran (JavaScript je jedini jezik prihvatljiv unutar HTML-a).

```
<script type="text/javascript">
//<![CDATA[
...(Your JS code)
//]]>
</script>
```

```
<!•• ako postoji vanjski dokument oznaka mora biti prazna    ••>
<script src="./js/script.js"></script>
```

JavaScript primjer

Pokušajmo napraviti naš prvi JavaScript dokument. U ovome primjeru napravit će mo jednostava poruku upozorenja. Koristit ć emo vanjski .js dokument koji će biti povezan sa našim .html dokumentom.

Prvi korak je otvoriti mapu sa vašim .html I .css dokumentima. Napravite novi .js dokument I pospremite ga u istu mapu npr. mojaskripta.js.

Sada dodajte sljedeću skriptu u vaš .js dokument.

```
alert("I am an alert box!");
```

poveznica myscript.js na vaš .html dokument:

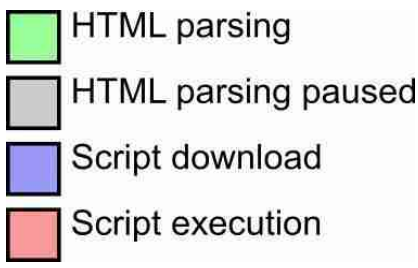
```
<!DOCTYPE html>
<html>
<body>
 <script src="mojaSkripta.js"></
 script> </body>
</html>
```

Možete nadodati eksterni dokument unutar oznake <head> ili <body> po želji. Skripta će se ponašati upravo gdije je postavljena <script> oznaka.

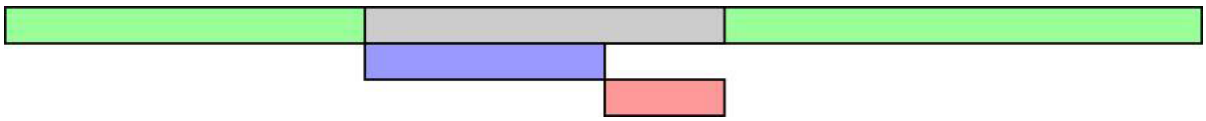
Redoslijed izvršenja skripti - Defer i async

Vrijeme izvršenja skripte ovisi o njenom položaju u HTML kodu sttranice i odmakom te async atributima.

Atributi defer I async vrijede samo za **vanjske (eksterne)** skripte.

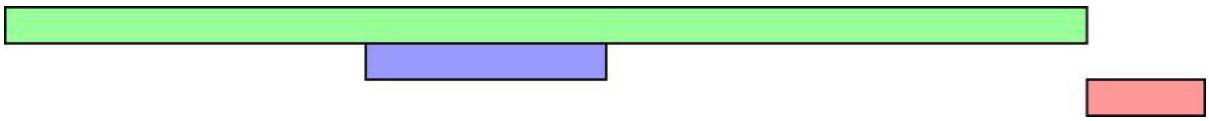


Normalno izvođenje



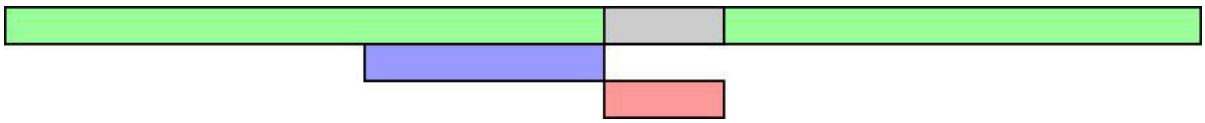
Zajedničko korištenje: ako dvije skripte slijede jedna drugu, druga skripta neće se izvoditi prije kraja prve jer je redosiljed 'blokira'.

Defer



Nikada ne koristite dokument.write() ili ekvivalent (sa odgodom – defer.) Obratite pozornost na izvršavanje skripti jednakih prioriteta, potrebno je poštivati redoslijed zbog grešaka I problema u IE-4-9.

Async (HTML 5)

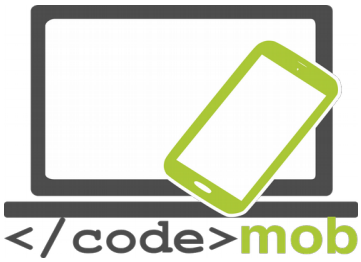


Idealno za skripte čije vrijeme izvršenja nije važno (npr: Google Annalytics), ali nije zajamčen redoslijed izvršavanja skripte!

Nikada ne koristite document.write() ili ekvivalent (sa async)..

"Uništavanje" slojeva

Kada JavaScript izvršava ili piše promjene DOM-a redoslijed slojeva je uključen I mora biti ponovno uzet u obziriza budućnost. Preglednik pokušava pričekati do završetka sličice ali ako moraju napraviti izvršenje prije može znatno utjecati na karakteristike izvođenja.



Ukratko, pokušavajte raditi minimalne preinake I manipulacije DOM-a ili u najgorem sulučaju grupirajte ih koliko je to moguće.

Ukratko, pokušavajte raditi minimalne preinake I manipulacije DOM-a ili u najgorem sulučaju grupirajte ih koliko je to moguće.

Više informacija možete pronaći na (https://github.com/wilsonpage/fastdom).

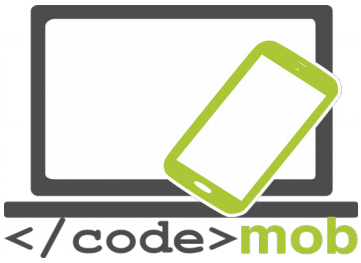
Konzola JavaScript-a

Vrlo je lako izgubiti se pišući JavaScript kod ne koristeći debugger. Debugger je process testiranja, pronalaženja I smanjenja problema tj. pogrešaka (errors) u izvođenju programa.

Ako skripta ne radi (bug) rijetko kada vidimo problem I zbog toga moramo prikaz konzole Javascript-a podeiti da vidi **poruke pogrešaka (**dokumenti I linije koda, boja koda, skripta u tjeku).

- **Firefox**: F12> Konzola (JavaScript)
- **Chrome**: Ctrl + Shift + I> Konzola

Također možete koristiti console.log(); postavke parametara koji trebaju biti prikazani. Moguće je I dodati break point (točke prijeloma) ili koristiti debugger funkciju..



Metoda console.log()

Ako vaš preglednik podržava debugging, možete koristiti console.log() kako bi prikazali vrijednosti JavaScripta u debugger prozoru.

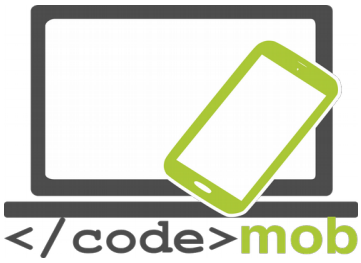
Primjer:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>

<script>
a = 5;
b = 6;
c = a + b;
console.log(c);
</script>

</body>
</html>
```

Postavljanje točaka prijeloma (Breakpionts)

Unutar prozora debugger-a možete postaviti točke prijeloma u JavaScript kodu. Na

svakoj točki prijeloma (breakpoint), JavaScript zaustavlja izvršenje I daje mogućnost provjere vrijednosti.



Nakon provjere možete nastaviti izvršenje koda (najčešće sa play gumbom).

Degugger – keywords (ključne riječi)

Debugger keywords zaustavljauju izvršenje koda JavaScripta I poziva (ako postoje) funkcije debugging-a.

Gore navedeno ima istu funkciju kao I korištenje breakpointa (točke prijeloma) u debugger-u. Ako ne postoji debugger vrijednost debugger nema efekta.

Sa uključenim debuggerom slijedeći kod zaustavit će izvršenje nakon trećeg reda.

```
var x = 15 * 5;
debugger;
document.getElementbyId("demo").innerHTML = x;
```

Imajte na umu da objekti konzole nisu dio standardnih događaja čak I ako su dobro uključeni u Firefox I Chrome preglednike.

Primjer - • API Console Firefox

Komentari

JavaScript komentari koriste se kako bi **objasnili pojedine dijelove koda** I učinili ih preglednijima.

JavaScript komentari mogu se koristiti za **prekid izvršenja** kada testiramo alternativni kod.

Komentari jedne linije

Komentari jedne linje započinju sa: //

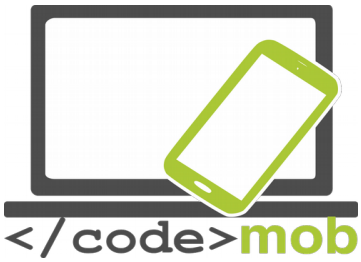
Tekst između // I kraja reda (linije) neće biti izvršeni od strane JavaScript-a.

Ovaj primjer koristi single-line komentare prije svakog reda koda:

```
// Change heading:
document.getElementById("myH").innerHTML = "My First Page";
// Change paragraph:
document.getElementById("myP").innerHTML = "My first paragraph.";
```

Ovaj primjer koristi jednu liniju komentara na kraju svake linije kako bi objasnili kod:

```
var x = 5;       // Declare x, give it the value of 5
var y = x + 2;   // Declare y, give it the value of x + 2
```



Višelinijski Komentari

Multi-line komentari počinju sa / * i završavaju s * /.

Bilo koji tekst između / * i * / JavaScript će ignorirati.

Ovaj primjer koristi multi-line komentar (a comment blok) kako bi objasnili kod:

```
/*
The code below will change
the heading with id = "myH"
and the paragraph with id = "myP"
in my web page:
*/
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
```

Korištenje Komentari za sprečavanje izvršenja

Korištenje komentarea kako bi se spriječilo izvršenje koda je pogodan za testiranje koda.

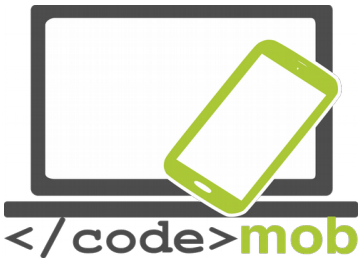
Dodavanje // ispred linija koda mijenja kod linije iz izvršne linije u liniju za komentar.

Ovaj primjer koristi // kako bi se spriječilo izvršenje jednog reda koda:

Example

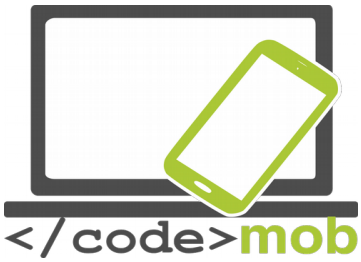
```
//document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
```

Primjer komentara koji spriječava nastavak koda




```
/*
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
*/
```

Komentari na jednoj liniji koji počinju sa // i onih na više linija okruženi /*...*/. U nastavku je malo složeniji slučaj pravomoćng umetanja JavaScript u XHTML dokumentu.



Uvijek koristite komentare komentirati kako bi **komentirali vašu logiku**, parametri očekivanja funkcije , koja se vraća ili deaktivira dio koda bez brisanja.

```
<scripttype="text/javascript">
//<![CDATA[
varmyFirstMessage ='Hello World';

console.log(myFirstMessage);

 functionmyFirstFunction(){
        vardemo =document.getElementById("demo"); demo.
        style.color ="#990000";
}

myFirstFunction(); //]]>
 </script>
```

Ne raditi: Eval & javascript:

Gotovo nikad ne koristite **eval ()**, jer omogućuje izvršavanje proizvoljnog koda

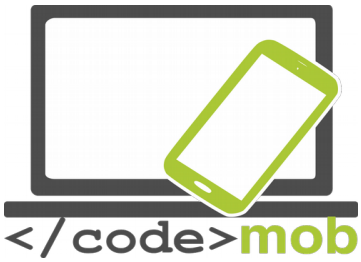
- predstavlja sigurnosni rizik.

Setinterval () funkcija koja zahtijeva svoj prvi parametar nakon određenog vremena koristi **eval ()** ako primi string kao parametar. Dakle, radije koristite neku funkciju kao argument.

```
setInterval( function(){myFunction(param1,param2);},5000);
```

Držite razmak između HTML-a i JavaScript-a (kao i sa CSS-om iz razloga održavanja slijeda I logike) izbjegavajući document.write (...) i odvajanje JavaScript od izgleda (layout-a).

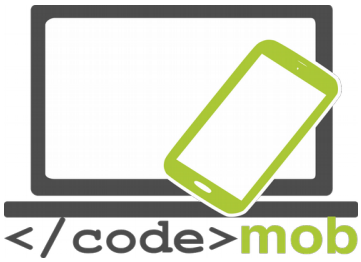
```
<! Nikada ne koristite sljedeće ... >
<ahref="javascript:myFunction();">....</a>

<! ...pokušajte izbjeći >
<atitle="Click to do some JS stuff" href="enablejs.html" onclick=
```



"MyFunction();return false;">link</**a**>

Matematičke operacije

- *, /, +, (redoslijed zagrada)
- % (Modulo)
- +=, =, *= & /= (x = x + 5;)
  - x++ & x (Post prirast ili umanjenje : x = x + 1)
  - ++x & x (preincrement ili umanjivanje)



Konstante

Slično varijablama osim što nakon što se izjave njihova vrijednost se ne može mijenjati. Često se koriste kao unaprijed definiranii parametri. Po dogovoru i na gotovo svim jezicima, imena konstante se pišu u velikim slova odvajanjem podcrtot. Pažnja, IE10 ne podržava konstante.

```
// Constant
constMY_FAVORITE_NUMBER =9;

// 'False' constant only by convention
var MY_LUCKY_NUMBER =7;
```

Varijeble I vrste varijabli

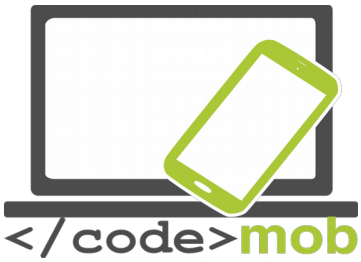
Varijable JavaScripta su **dinamičke vrste** kao što je PHP, koriste pointed sintaksu I osjetljive sun a velika I mala slova stoga pripazite I obratite pozornost zbog mogućih promjena prilikom pokretanja.

Kako bi koristili Lower Camel Case kao ime varijable (svako slovo svake riječi osim prvoga velikog slova, bez razmaka) smatra se kao praksa. JavaScript sadrži

“sakupljaš smeća” – Garbage Collector koji uništava sve variable koje se ne koriste (ili su postavljene na 0 vrijednost).

Ne kristite sintaksu: var a = new Array(); ili crtice u imenima varijabli (na primjer: var causeAnError =5;)

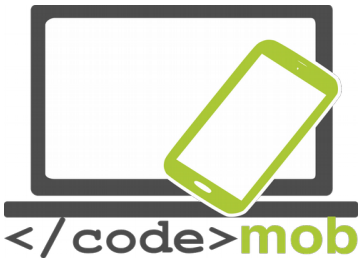
```
 varuserIdNumber; // varijabla proglašena var tag-om
     userIdNumber =5; // učitanje iste varijable variable

varuserIdNumber =5; // Simultaneous declaration and initialization

varvariable1 ="Hello World!";
varvariable2 =42;

varvariable1 ="Hello World!",
        variable2 =42;          // višestruko proglašenje
```



Vrste podataka

- Primitivne
  - **Strings** (znakovne veze) : “Hello World”
  - **Numbers** (brojevi, puni I decimalni brojevi): 3,14 ili 42
  - **Booleans - petlje** (logički nizovi, vrijednosti) : važeće ili ne važeće
  - Undefined (nedefinirani – varijabla koja još nije potvrđena ili nema vrstu)
  - Null (prazna varijabla, varijabla sa doznaćenom vrijednosti nula)

- **Objekti**
  - **Funkcije**
  - **Nizovi**
  - **Datumi, matematičke funkcije**
  - **RegExp (izrazi)**



Dodjela

Moguća je promjena vrsta varijabli (Dodjela – casting) uz pomoć **Number(…)**, String(…) i **Boolean(…)** ili metodama kao **.split()** ili **.join()**.

```
varnbr2Boo =Boolean(0);                 // false
varstr2Nbr =Number('12');               // 12
varboo2Str =String(true);               // "true"
```

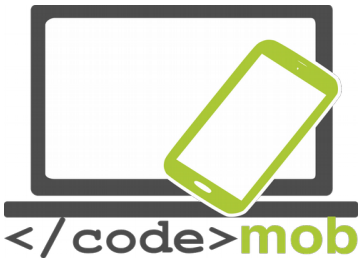
Veze (strings)

- Povezivanje veza izvodi se **“+”**
    - Veze (Strings) su definirane ili dvostrukim navodnim znacima ili jednostrukim navodnim znakovima.
        - Za razliku od PHP-a ne postoji razlika između navedenih linija.
        - Posebni znakovi mogu se koristiti uz \(escape znak)
            - \ ", \ n (povratak na liniju, \\, ...
            - \ na svaki povratak linije
- Duljina je dostupan putem **.length** svojstva
- Veze imaju više metoda;

...

- .charAt(); indexOf(), substr, substring(), toLowerCase(), toUpperCase(),
- Ne pristupajte vezama kao da su polja (Array).

Primijer: - http://www.w3schools.com/js/js_strings.asp



Brojevi

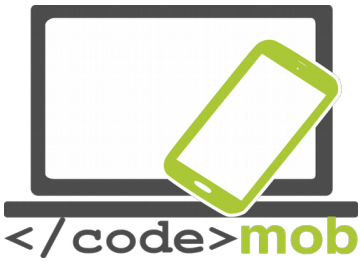
- Samo jedna vrsta (Integer & Float) pohranjen pod 64bit s pomičnim zarezom
- nfinity & -Infinity konstante očito predstavljaju pozitivnu beskonačnost i negativnu beskonačnost.
- NaN (nije broj) prikazat će se kao matematička nemogućnost kao dijeljenje string-a npr.
- **toString()** metoda pretvaranja u stringove.
    - .toString(2): Pretvaranje u binarni
    - .toString(16): Pretvaranje u heksadecimalni
    - Objekt **Math** nudi mnoge opcije:
    - Math.random(): Vraća slučajan broj između 0 (uključen) i 1 (isključen)
    - Math.min(…,…)& Math.max(…,…): Minimum & Maximum
    - Math.round(), Math.ceil() & Math.floor(): Zaokružene, zaokružene prema gor I dolje
    - Main.sin(), Math.cos(), Math.PI, …

Primjer: http://www.w3schools.com/js/js_number_methods.asp

Vježba: Slučajno odabrani broj 0 ili 1 u konzoli svaki puta kada se učita stranica (f5 – osvježi).

Niz (Array)

- Vježba: Slučajno odabrani broj 0 ili 1 u konzoli svaki puta kada se učita stranica (f5 – osvježi).



- **var**myArray =[];// Novi prazni niz

- Indeks počinje kao u gotovo svim računalnim jezicima na 0.
  - myArray[0]=42;

- Moguće je naći dužinu niza pomoću **.length** svojstva
  - Nikada ne mjenjajte niz u kojemu se prikazuje.

- Oprez, ovo nisu asocioni nizovi

- Niz ima metodu gotovo za sve; .push(), .pop(), .splice(….),…
  - Moguće je sortiranje niza znakova u nizu koristeći metodu .sort()

.sort()

- and digits via .sort() koristite funkciju argument za funkciju usporedbe

.sort(function(a, b) {return a - b})

Primjeri: W3Schools& www.w3schools.com/js/js_array_methods.asp

Vježba: Prikažite posljednji element u nizu čiju dužinu ne poznajte.

Testiranje vrsta varijabli

Mnogo je načina na koji možemo testirati varijable.

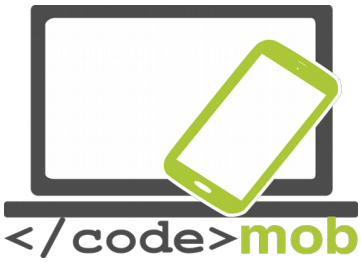
- **typeof** myVariable
    - **typeof**myVar ===**'undefined'**
    - The NaN data type is 'broj'
    - The data type of an array is 'objekt'
    - The data type of the Date object is 'objekt'
    - The data type of null is 'objekt'
    - The data type of an indefinite variable is 'nedefinirano'

Primjer

```
typeof "John"                 // Returns "string"
typeof 3.14                   // Returns "number"
typeof false                  // Returns "boolean"
typeof [1,2,3,4]              // Returns "object" (not "array", see note
below)
typeof {name:'John', age:34} // Returns "object"
```

The typeof operator returns "object" for arrays because in JavaScript arrays are objects.

- isNaN(…)
- isFinite(…)
- Array.isArray(…);
- …



Var ili ne var?

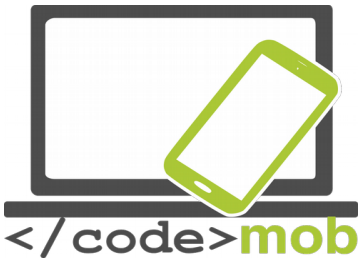
U funkciji, izjava varijable sa var je lokalna u protivnome je globalna.

Savjetuje se da uvijek deklarirate varijable (izbjegnite preopterećenje, ECMA6,

…)

> Trajenje im je drugačije: lokalne varijable prestaju na kraju funkcija koje ih sadržavaju (trenutna izvršenja), globalne varijable nakon zatvaranja HTML stranice.

> Tu postoji puno više ali imajete na umu da moramo koristiti var kako bi proglasili varijablu bez da je dodjelimo I da samo ne proglašene varijable možemo brisati (što se ne savjetuje).



Djelokrug

U JavaScriptu objekit I funkcije također su varijable.

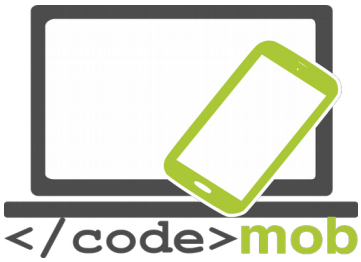
Djelokrug je set varijabli, funkcija I objekata kojima trebate pristupiti u određeno vrijeme. Zbog toga se mijenjauju unutar same funkcije (djelokrug funkcije).

Djelokrug uvijek ide u smjeru od lokalnog do generalnog ako JavaScript ne pronađe varijable u djelokrugu tada ih traži u glabalnim funkcijama I na kraju generira error ako ništa nije pronađeno.

Postavljanje / Hoisting

JavaScript uvijek proglašava varijablu I funkciju prije nego što pokrene kod: za početak pomiče ih na početak bloga gdje im je I mjesto.

Proporučuje se da se funkcije I varijable proglase na početku djelokruga.



Uvjeti

Uvjetne izjave koriste se kako bi se izvršile **pojedine akcije s obzirom na različite uvjete.**

Uvjetne izjave

Vrlo često prilikom pisanja koda želite načiniti različite akcije za različite odluke.

Uvjetne izjave u kodu možete koristiti za sljedeće.

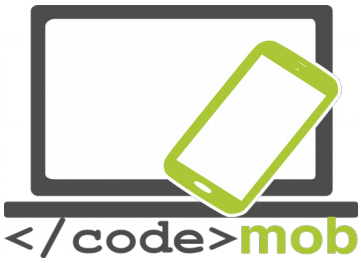
U JavaScript-u postoje sljedeće uvjetne izjave:

- Koristite **if** kako bi ste izvršili pojedini dio bloka koda, ako je uvjet točan
- Koristite **else** kako bi ste izvršili pojedini dio bloka koda ako je isti uvjet netočan
- Koristite **else if** kako bi specificirali novi uvjet za testiranje ako je prvi uvjet netočan
- Koristite **switch** kako bi specificirali puno alternativnih blokova koda I time ih pokrenuli

If ... then ... else ...

Koristite razmake između else i if (ne koristiti kao u PHP-u npr. Elseif)..

```
if(money <0){
        status ="I'm broke..."; }
 elseif(money <10000){
        status ="Could be worse...";
```




```
}else{

        status ="I'm rich!";
}
```

Trostruke operacije

Kompaktno korištenje if…then, korisno prilikom korištenja izjava varijabli npr.

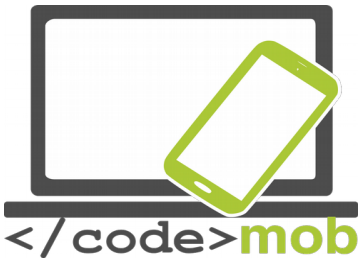
```
varadmissibleAtI3 =(sex =="F")?true : false;
```

Switch

Jednostavan način za pregled mnogih izbora.

Swich nam omogućava višestruke točke u istome kodu koristeći **stric comparison (===).**

```
switch (newDate().getDay()){

        case1:


        case2:    case3:
        default:

                text ="Looking forward to the Weekend

                ";
                break;

         case4: case5:

                text ="I'm tired and bored";

                break;

        case0: case6:
```

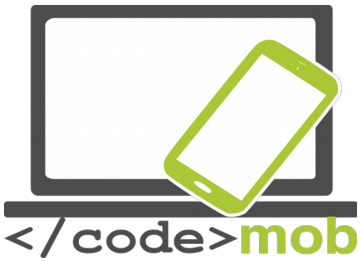



```
            text ="Time to play : )";

}
```

Logične operacije

- == (different from =), !=
- ===, !== : comparison of value and type
- >, >=, <, <=
- && (And)
- || (Or)
- ! (Not)
    - toggle = !toggle // true becomes false and false becomes true.

Java Script kao I svi programski jezici zaustavlja se na prvom netočnom djelu koda ili nepotpunoj izjavi ili uvijetu što nam omogućava prvo ispitivanje I testiranje postojanja varijabli I zatim rad bez upozorenja error ili poruke da ne postoje.



JavaScript ne posjeduje XOR (isključivo ili) ali je zamjenjiva funkcijama.

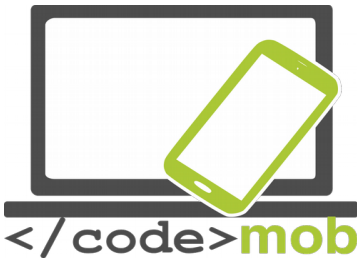
	&& (AND)	|| (OR)	XOR
True & True	True	True	False
True & False	False	True	True
False & True	False	True	True
False & False	False	False	False

Vježba: Napravite uvjet sa dva parametra koji zamjenjuje XOR.

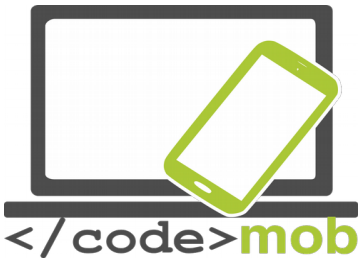
```
vara =true; // Test 4 times avec true, true, false, false
varb =true;                 //                          true, false, false, true

if(...){
        console.log('XOR is true');

}else{
        console.log('XOR is false');

};
```

Petlje

```
// petlja FOR

for(vari =0;i <10;i++){...};
// Optimized loop FOR

for(vari =0,len =myArray.length;i <len;i++){...};
// Loop FOR IN To iterate the properties of an object

for(prop inobj){
        //    prop
```




```
        //                  obj[prop]

};

// Loop

vari =0;

while(i <10){
        // ...

         i++; // Without this increment, the loop is infinite

};
```




```
// Loop executed at least once
do{
// ...
i++;
} while(i <10);
```

Svi preglednici imaju maksimalno vrijeme izvršenja za JavaScript

Prekid, nastavak i oznake

Prekidi i nastavci ddaju kontrolu nad radom petlji.

Oznake upućuju na mjesta u skripti na koje je moguće usmjeriti. Moraju biti izjavljene prije prekida I nastavka.

- **break**: izlazi iz petlje
  - break label: : može ukazivati na bilo koji dio bloka koda (višestruke petlje)

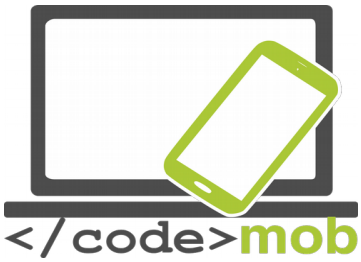
- **continuous**: nastavlje trajanje petlje

Vježba: Kreirajte popis sa oznakama koje prolaze kroz sadržaj reda tri puta (sa kratkim opisom kroz getElementByld i innerHTML).

Pukušajte uhvatiti (Try catch)

Rezervna riječ throw uvijek izbaciti eror (pogrešku) – uključujući I vašu vrstu pogreške.

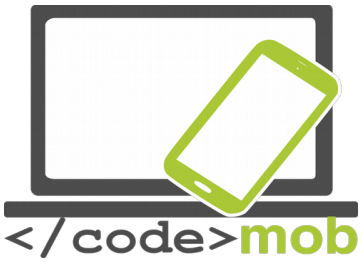
```
try{
// dio koda koji može uzrokovati error. }
```




```
catch(e){
// dio koda zadužen za mogući error e. }

finally{
// Block of code that will be executed independently of the result of the
// try catch (return or other throw exception)
}
```

Rezervna riječ throw uvijek izbaciti eror (pogrešku) – uključujući I vašu vrstu pogreške.

```
throw404;
thrownewError("Invalid
age");
```



Funkcije

Funkcije imaju sljedeće prednosti:

- Grupiranje koda
- Poboljšanje čitljivosti
- Ponovno korištenje
    - Generalizirano (isti dio koda, različite vrijednosti)

```
functionsum(arg1,arg2){returnarg1 +arg2 };
```

Pokušajte sistematizirano izvršavati funkcije koje uvjek vraćaju vrijednost.

Višak argumenti se zanemaruju kao I nenavedenih argumenta "nedefinirano" nakon vrijednosti; napravite svoju provjeru unutar funkcije kroz **argumente** (arguments object) koji je globalno sličan reu. (.length i indeks kroz […]).

Sintaksa funkcija u JavaScript-u

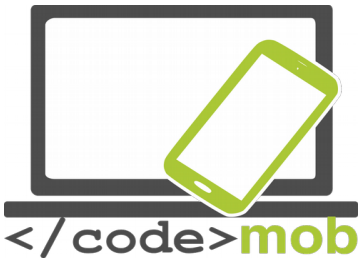
Funkcija unutar JavaScripta je definirana ključnom riječi funkcije nakon koje sljedi ime I zatim zagrade ().
Nazivi funkcija može sadržavati slova, brojeve, podvlake I znakove dolara (jednako pravilo kao I za varijable).
Zagrede mogu uključivati imena parametara odvojenih zarezima: **parametar1, parametar2**, …)

Kod koji će se izvršiti nalazi se unutar vitiščastih zagrada: { }

```
Ime funkcije(parametar1, parametar2,
  parametar3){ Kod za izvršenje
}
```

Funkcije **parametric** su **imena** definirana unutar funkcije.

Argumenti funkcije su realne vrijednosti **funkcije** u trenutku poziva.



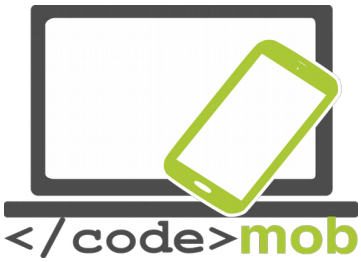
Unutar funkcije argumenti (parametric) ponašaju se kao lokalne varijable.

Dozivanje funkcija i povratak

Dozivanje funkcija

Kod unutar funkcije izvršit će se ukoliko nošto pozove (inicira) funkciju:

- Kada se nešto dogodi (kada korisnik pritisne gumb)
- Kada se pozove iz koda JavaScript-a
- Automatski (samopozivom)



Funkcija povrata

Kada JavaScript pristupi povratnoj izjavi (**return statement**) funkcija će se zaustaviti. Ako je funkcija pozvana od strane izjave, JavaScript se "vraća" izvršenju koda nakon pozvane izjave. Funkcije često izračunaju povratnu vrijednost. Povratna vrijednost se "vraća" natrag na "pozivatelja":

Primjer: Izračunajte vrijednost dvaju brojeva I prikažite rezultat:

```
var x = myFunction(4, 3);         // Function is called, return value will end up in
x function myFunction(a, b) {
    return a * b;                 // Function returns the product of a and b
}
```

Rezultat u x će biti: 12

Poziv i referenca funkcija

Varijabla može uz pomoć imena pozvati se na funkciju. Poziv se vrši sa zagradama () i eventualnim argumentima.

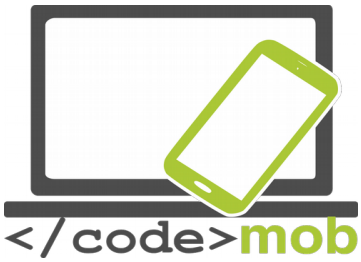
Funkcije višeg reda su funkcije koje objedinjavaju jednu ili više funkcija kao zagrade ili povratak drugih funkcija.

Anonimne funkcije

```
varmyFunction =function(message){ alert(message); }; myFunction('This is a test'); //
Displays : This is a test
```

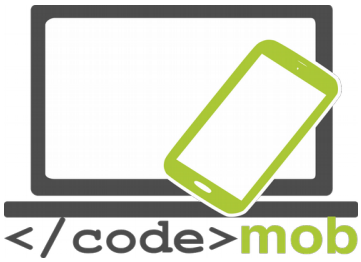
Vježba: Kreirajte faktorsku funkciju (n){…} koja vraća factor broja n.

Zatvaranja



Pojam je pre specifičan za JS I preširok ali pokušajte zapamtiti da kada vidit riječ unutar druge funkcije unutarnja funkcija ima pristup varijablama vanjske funkcije (kao i “običan” djelokrug unutar local I global funkcija).

Primjer:
http://www.w3schools.com/js/js_function_closures.asp



Objekti

Za razliku od String-a, brojeva, ili logičkih varijabli, objekti sadržavaju višestruke vrijednosti kao paris name: value.

```
varx1 ={};// New object

varperson ={firstName: "David", lastName: "

                    Collignon", age: 39
                };

console.log(person.firstName); console.

log(person['firstName']);
```

Anonimni objekti

Objekti baš kao I druge vrste podataka nemaju obavezu da ih se imenuje. Ovo je česti sučaj kod objekata za konfiguraciju koji se koriste kao parametric klasa.

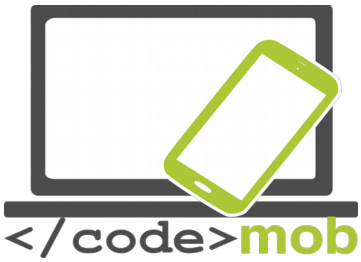
```
$('.bxslider').bxSlider({mode:'fade', captions:true});
```

Zajednički objekti

Mnogi korisni objekti direktno su dostupni: dokumenti, prozori, matematika I sl.

- Objekt: eg document
  - Vrijednost: na primjer .innerHTML ili .textContent
  - Metoda: npr. .getElementByld()
  - Ključne riječi: ovaj (this)

```
document.getElementById("demo").innerHTML ="Hello World!";
```



Ključna riječ ovaj vraća object koji “posjeduje” dio koda tako da je object unutar objekta. O ovome više u detaljima JQuery-a.

Raspored po vrijednosti i adresi

Kompleksn vrste podataka u JavaScript-u dodjeljene su kao reference a ne kao vrijednosti. Prema vašim potrebama morat će te izraditi skriptu kako bi kopirali tablicu ili objekt. (Deep copy).

```
// Short examples of Deep copy JSON.parse(JSON.stringify(obj))// only if there is no fn
varnewObject =jQuery.extend(true,{},oldObject); varnewArray =jQuery. extend(true,
[],oldArray);
```

DOM

Što je DOM?

DOM je W3C (World Wide Web Consortium) standard DOM je

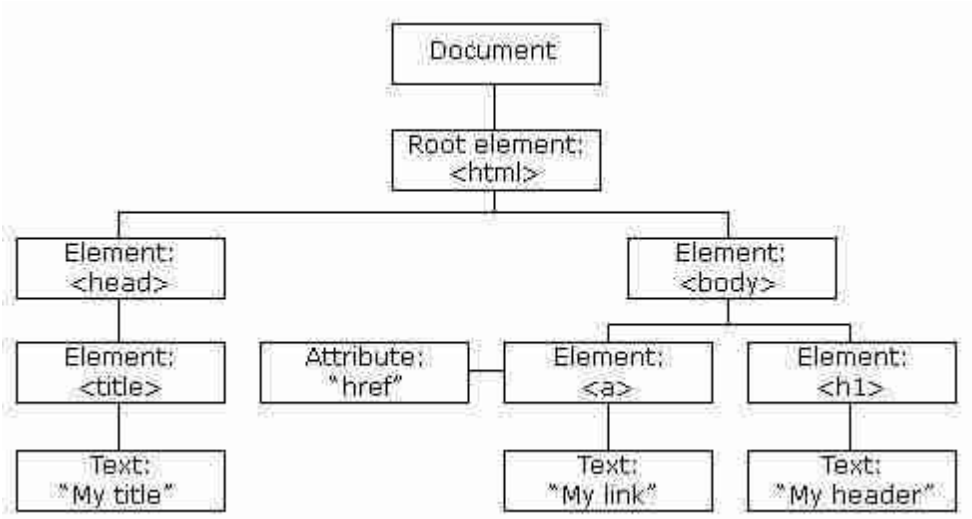
standard koji koristimo za pristup dokumentima:

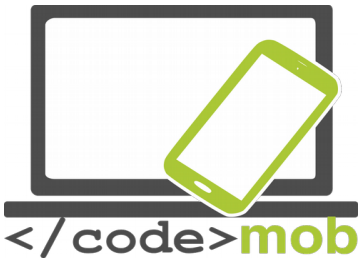
"W3C Document Object Model (DOM) je platforma i jezik neutralnog sučelja koje omogućuje programima i skriptama dinamički pristup i ažuriranje sadržaja, struktura i stil dokumenta."

DOM standar podjeljen je u tri dijela:

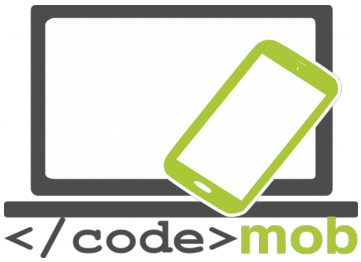
- Core DOM – standardni model za sve vrste dokumenta
- XML DOM – standardni model za XML dokumente
- HTML DOM – standardni model za HTML dokumente

JavaScript omogučuje vam da pregledavate, čitate I mjenjate Document Object Model (DOM)





Vježba: Pretvorite gore navedeni dijagram u HTML stranicu.



Objekt dokumenta I odabir dijela HTML-a

Objekt dokumenta predstavlja cijeli HTML dokument (root) kada je učitan u prozor preglednika. Ponuđene su nam neke metode za prepoznavanje pojedinog dijela kao na primjer:

```
varnode1 =document.getElementById("demo"); // #demo varnode2
=document.querySelector("p"); // The first p found
```

GetElementByID and querySelector respectively return the corresponding id and only the first occurrence found or **null** in the opposite cases.

```
varnodeList1   =document.getElementsByClassName("demo");  //          .demo
varnodeList2   = document.getElementsByTagName("p");// All the p varnodeList3
=document.querySelectorAll( "p"); // All the p
```

Ove metode vračaju popis čvorišta (Node list) koje sadržavaju .length svojstva čiji su indeksi dostupni kroz zagrade [] pa se mogu preusmjeriti u for loop petlje. Treba imati na umu da to ipak nije polje.

Imajte na umu da za kompleksne dijelove CSS-a .querySelector metoda još uvijek nije dovoljno dobra kao definiranje baza kao što su JQuery.

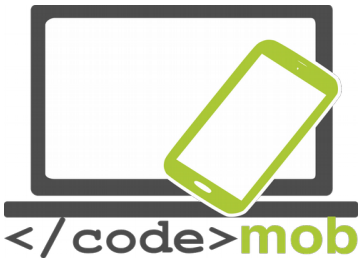
Čitanje i mjenjanje HTML-a

Što je HTML DOM?

Evo odličnog objašnjenja: https://css-tricks.com/dom/

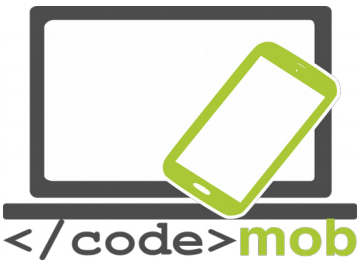
Možete brzo promijeniti sadržaj i atribute HTML oznaka s .innerHTML (u čitati i pisati) i dobavljač / postavljač za atributa.

```
// Get & Set

varhref  =document.getElementById("myLink").getAttribute("href");
document.getElementById("
```




```
myLink").setAttribute("href",newValue); // Tests the existence

varhasH =document.getElementById("myLink").hasAttribute("href"); //
Deletion document. getElementById("myLink").removeAttribute("href");

// Inject HTML document.getElementById("demo").innerHTML ="New
text"; // Add text content document.getElementById("demo").textContent
="New text";
```

Uočite razliku između .innerHTML i .textContent, prvi će se raščlaniti i tako prihvaća HTML dok drugi ne. TextContent će biti brži i sigurniji.



Promjene CSS-a

Sve vrijednosti CSS-a dostupne su kroz sintaksu svojstva stilova kroz sljedeće značajke:

- Svojstva su pisana u “Lower Camel Case” (veličinafonta – fontSize)
- Vrijednosti su uvijek stringovi (npr. Ne 250 već “250px”)
- Nova vrijednost se dodaje u inline stilu!
- Pozicija kroz .offsetTop I .offsetLeft (ne desno ili dolje)
- Ukupna širina I visina (padding, border) kroz .offsetWidth I .offsetHeigth

```
varelement =document.getElementById("demo"); element.style.backgroundColor ="
green";// camel Cased element.style["background•color"]="green"; // standard CSS
```

```
varelement =document.getElementById("demo"); element.className
=element.className
+"otherclass";// space
```

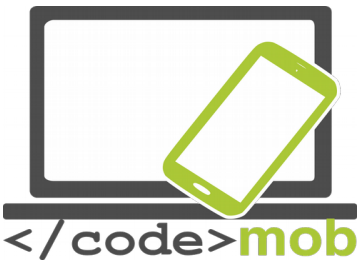
Obično JavaScript dohvača unutarni stil. Ako želimo dohvatiti stil iz vanjskog CSS dokumenta trebamo koristiti <style> tag, koristiti .getComputedStyle():

```
varstyle =window.getComputedStyle(document.querySelector('nav'),null); var
bgc =style.getPropertyValue('background•color');//rgb(255, 191, 0)
```

Promjena DOM-a

Neke od metoda za izradu I umetanje HTML oznaka u DOM.

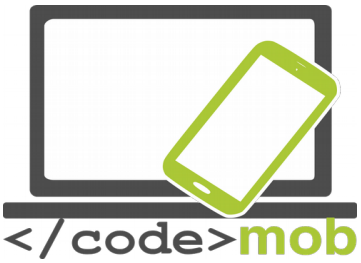
- **document.createElement("htmlTag")**• kreira HTML oznaku
- **document.createTextNode("Hello world")** •Kreira tekstualni sadržaj



- **.removeChild(childToBeRemoved)•** uklanja element “child”
- **.appendChild(newContent)•** Dodavanje novoga elemeta na kraj parenta “roditelja”
- **.insertBefore(newElement, currentElement)•** Dodavanje novoga elementa
- **.replaceChild(newElement, currentElement)•** Zamjena elementa

Neke od korisnih metoda za pregled DOM-a

- **.parentNode•** Fokus na čvorište drugog čvorišta
- **.children[index]•** Fokus na “children” kroz startni indeks koji je jednak 0
- **.nextElementSibling•** Fokus na “sister” čvorište (čvorište sa jednakim početkom)
  - Ne treba zamijeniti sa .nextSibling što također vrača vrijednost čvorišta I komentare
- **.previousElementSibling•** Pronalazi prethodno čvorište “sestre” – sister
- **…**



Pokušajte sami!

JavaScript HTML DOM elementi:
http://www.w3schools.com/js/js_htmldom_elements.asp

Promjena HTML sadržaja: http://JavaScript HTML DOM - Changing HTML

Promjena HTML sadržaja: http://www.w3schools.com/js/js_htmldom_css.asp

Potpunu listu svojstava I metoda možete naći : http://www.w3schools.com/jsref/dom_obj_all.asp **Primjer:** W3Schools [1][2][3][4]

Objekti zaslona

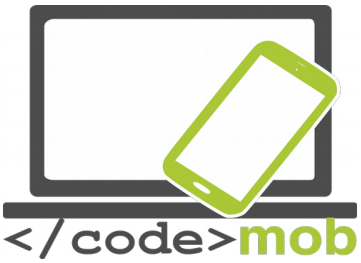
Objekti zaslona pružaju podatke o zaslonu korisnika. Ova informacija nije dostupna preko skripti poslužitelja.

- **screen.width**& **screen.height**
  - **screen.availHeight**& **screen.availWidth**(Windows taskbar less)
- **screen.colorDepth**

Objekti navigatora

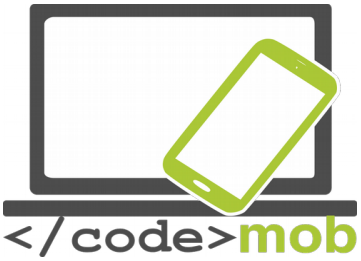
Objekti navigator pružaju informaciju vezanu za korisnikov preglednik.

- **navigator.userAgent**• Puni naziv preglednika
- **navigator.platform** Korisnikov operativni sustav
- **navigator.onLine**• Dali je korisnik na mreži ili ne
- **navigator.language**• Jezik sučelja preglednika



- **navigator.geolocation**• Geopozicioniranje nakon potvrde od strane korisnika
- **navigator.cookieEnabled**• Dali korisnik prihvaća “kolačiće”
- ...

Primjer •http://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_nav_geolocation



Događaji

Moguće je povezati bilo koji događaj (klik, dupli klik, prelazak preko, error, promjenu dimenzija…) sa elementom DOM-a korz sljedeću sintaksu:
element.addEventListener(event, function, useCapture);

```
document.getElementById("myBtn").addEventListener("click", doStuff);

function doStuff(ev){...}
```

Naziv događaja parametra je “string” koja ignorira postojanje HTML atributa

(primjer: <a href="" onClick="…" /> → click).

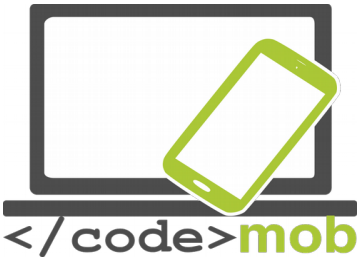
Kompletan popis na: W3Schools: http://www.w3schools.com/jsref/dom_obj_event.asp

HTML događaji su “stvari” koje se događaju HTML elementima.

Kada se JavaScrip koristi unutar HTML stranice, JavaScript može reagirati na događanja. Događaji su akcije ili zbivanja koji se događaju unutar Sistema koji programirate na koje vas upućuje system kako bi mogli utjecati na njih. Na primjer, ako korisn na stranic pritisne tipkom miša na gumb možete odgovoriti akcijom koja prikazuje prozor sa porukom.

U slučaju Web-a događajise odvijaju unutar prozora preglednika I imaju funkciju povezivanja specifičnih elemenata. TO mogu biti pojedinačni elementi, aktivirajući elementi, HTML dokument učitan u aktivnoj kartici ili cijeli prozor preglednika. Postoji pregršt **različitih događanja** koja se mogu odvijati npr. :

- Korisnik klikne mišem na određeni element ili pređe preko njega mišem.
- Korisnik pritisne tipku na tipkovnici



- Korisnik promjeni veličinu prozora preglednika ili zatvori preglednik.
- Web straniza završi učitavanje
- Obrazac se pošalje
- Video se prikazuje ili je pauziran/ završio
- Pojavila se pogreška (error)

Svaki pojedini događa ima svoj obraza događaja koji je sačinjen od bloka koda koji će biti pokrenut svaki puta kada se događaj pokrene.

Kada se definira kao blok koda kako bi se pokrenuo na odgovor, a na početak događaja, kažemo da se registrira “event handler”. Imajte na umu da provoditelj događaja ponekad

naziv “Slušatelji događaja” – oni su izmijenjivi prema našim zahtijevima iako rade zajedno. “Slušatelj” prati događanja a “rukovoditelj” je kod koji je zadužen za događaj.

Jednostavan primjer:

U sljedećem primjeru imamo jedan gumb <button> koji nakon pritiska mijenja pozadinsku boju slučajnim odabirom.

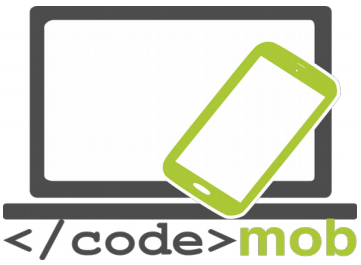
```
<button>Change color</button>
```

The JavaScript looks like so:

```
var btn = document.querySelector('button');

function random(number) {
  return Math.floor(Math.random()*number);
}

btn.onclick = function() {
  var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255)
  document.body.style.backgroundColor = rndCol;
}
```

U ovome kodu sadržane su reference gumba unutar varijable pod nazivom btn koristeći Funkciju Document.querySelector().

Također definiramo funkciju koja vraća slučajnu vrijednost broja. Treći dio koda je zadužen za događaj. Varijabla btn usmjerava na <button> element. Ovi objekit imaju broj zadužen za količinu događanja. Svakim klikom (pokretanjem) slušamo događaj onclick koji je jednak anonimnoj funkciji koja je zadužena za nasumični odabir spektra RGB I postavlja <body> kao dio pozadine.



Kod će se pokrenuti svaki puta kada nastupi događaj “klik” na element <button>

Kod i rezultati: https://codepen.io/pen/

Svojstva “rukovoditelja” događanja

Povratkom na prethodni primjer:

U ovom primjeru svojstvo onclick je “rukovoditelj” svojstva koje se koristi u ovoj situaciji.
To je jedno od osnovnih svojstava baš ako i svojstvo gumba (Button) ali je posebne vrste – ako postavite da je jednako kodu taj kod će se pokrenuti kada se pokrene i događaj na gumbu.

Upravljanje događajima

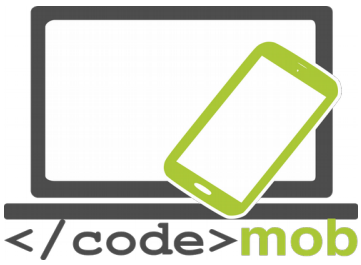
Objekt događanja predstavlja DOM događaj. Možemo ga koristiti unutar naše funkcije koja je zadužena za događaj.

Sadrži mogo svojstava I metoda….

- **event.target•** Objekt koji je focus događaja
- **event.currentTarget•** Vraćanje element na dio gdije je događaj dodan
- **event.preventDefault()•**- Poništavanje normalnog ponašanja
- **event.stopPropagation()•** Poništava multipliciranje događaja

Zadržavanje i propuštanje

Zadržavanje je silazna faza (iz vanjskog elementa prema unutarnjem) a propuštanje je uzlazna faza (poput ribolovca koji zaranja I izranja)

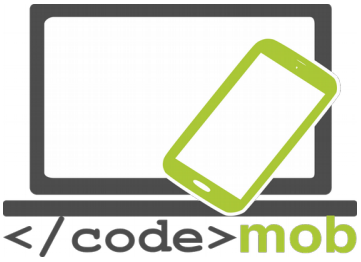


Zadana virjednost useCapture je negativna (false) a IE9 podržava samo metodu propuštanja.

Obrazac za uklanjanje događanja

Koristite samo .removeEventListener() ali nije moguće ukolniti anonimne funkcije. U protivnome , svaki “rukovoditelj” događanja mora biti uklonjenj pojedinačno jedan događaj kroz propuštanje I jedan događaj korz zadržavanje u istome događaju kroz dva različita enentListener-a.

```
vare =document.getElementById("myDIV")
e.addEventListener(" mousemove",myFunction);
e.removeEventListener("mousemove", myFunction);
```



Kviz

U nastavku je mali kviz koji možete riješiti.

Rezultati se nalaze na kraju kviza. Pokušajte ne varati.

Pitanja

Uvod

1. *Kad god vaš program ne radi, prva stvar koju trebate učiniti je uvijek gledati na poruke o pogreškama. Kako otvoriti web konzolu u oba Firefox i Chrome?*

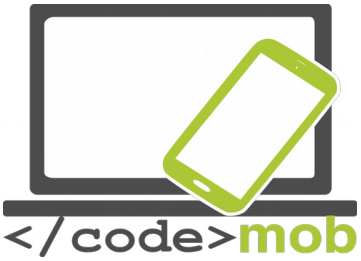
   - F12 (Chrome and Firefox)
   - Open Menu > Developer > Web Console (Firefox)
     Open Menu > More Tools > Developer Tools > Console Tab (Chrome)
   - Ctrl + Shift + K (Firefox)
     Ctrl + Shift + I (Chrome)
   - Ctrl + C (Chrome and Firefox)

2. *Možete odredite varijablu sa i bez var ključnu riječ. Na koji način je trebate koristiti?*

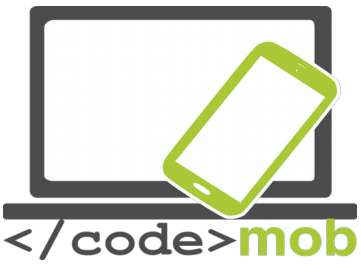
   - Oba načina, ne postoji razlika
   - - Varijabla proglašena sa “var” ključna riječ pripada mjesnim okvirima, a drugi pripada globalnom okviru. Uvijek koristite var kako bi se izbjeglo zagađivanje globalni doseg.
   - O varijablu proglasio s 'var' ključna riječ pripada globalnom okviru, a drugi pripada lokalnom okviru.

3. *Zašto treba uvijek pisati komentare?*

   - Ključno je za objašnjavanje pojedinog dila koda. Bilo da se radi o vama kao nastavak rada na starom projektu ili rad u timu.



- Komentari su gubitak vremena nitko ih ne čita
- Jednostavna je metoda poništavanja dijela koda bez da ga izgubimo



Uvijeti

1. *Postoji li logična razlika između structure " if else" I structure "if" ukoliko postoje suprotni uvijeti?*

   - Ne postoji ali "if else" je puno učinkovit
   - Da, nisu isti.

2. *Kako možete napraviti petlju kroz svaki predmet a da ne poznajete dužinu?*

   - Kroz korištenje niz vrijednosti .length.
   - Kroz korištenje niz vrijednosti metodom .length()
   - Kroz korištenje funkcije count().

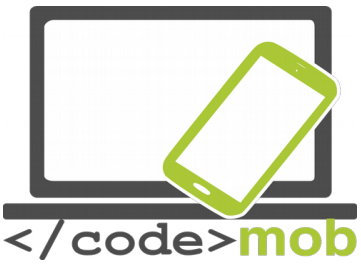
3. *Što može unaprijediti navedeni kod?*
for (var i = 0, len = a.length; i < len, i++) { /* ... */ }
brings compared to
for (var i = 0; i < a.length, i++) { /* ... */ }
?

   - Ništa
   - Varijabla "len" provjerava se samo jednom umjesto jednom po ponavljanju.
   - Broj ponavljanja je drugačiji

Funkcije

1. *Dali je raspored argumenata funkcije naziva "divide" (kaoja vraća vrijednost podijele prvog argumenta kroz drugi) važna?*

   - Da, mjenja rezultat



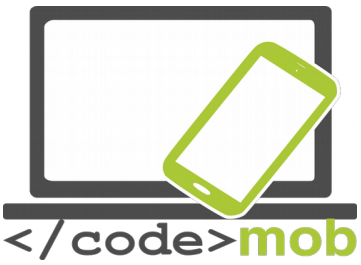
- Ne

2. Koja je razlika između “function call” I “function reference”?

- A 'Poziv funkcije' (njeno ime slijedi zagradi) zatražiti izvršenje funkcija koda pravo u trenutku poziva. A 'funkcija reference "(u povratni poziv na primjer) jednostavno pohraniti naziv funkcije za buduću upotrebu.
- Ne postoji razlika ako je funkcija nije nikakav argument.

3. Koji su pridržane ključne riječi koj se koriste za određivanje koje se vraća vrijednost po pozivu funkcije?

- Return
- Pass
- Exit



DOM

1. Koja od navedenih metoda se ne referira na prvi ili samo pronađeni događaj?

- document.querySelector()
- document.querySelectorAll()
- document.getElementById()
- document.getElementsByTagName()
- document.getElementsByClassName
- document.getElementsByName()

2. U sljedećem kodu, koji je argument prima povratni poziv?
document.getElementById('myID').addEventListener('click', myCallBack);

- Događaj koji je aktivirao callback
- Argument nije prošao kroz pozi (callback)

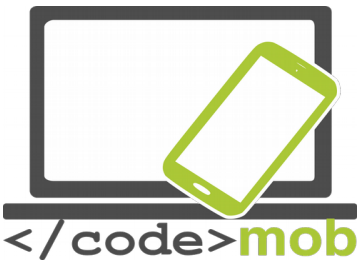
3. Kako možemo dobiti vrijednost kontrole nad HTML-om (UI) npr. Input tag?

- Kroz svojstvo .value koja vraća sadržaj “value” atributa
- Kroz metodu .getValue() koja vraća sadržaj atributa “value”
- Kroz metodu ..val() koja vraća sadržaj atributa “value”

CSS

1. Koji CSS selector omogućava pronalazak paragrafa koji je direct “child” div-a ?

- Div > p
- Div p
- Div, p

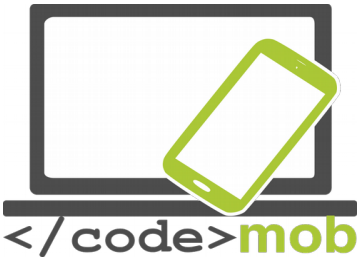


2. *Koji CSS selector omogućava targetiranje samo prvog paragrafa bez obzira na broj "potomaka" ?*

   - p:nth-of-type(1)
   - p:first-child
   - p:nth-child(1)

3. Ako dva CSS selektora (.red et #blue) ciljaju isti element, koje će biti boje?

   - Plave
   - Crvene
   - Ljubičaste
   - Ovisno o tome koji je proglašen zadnji



Odgovori

Uvod

1. Kad god vaš program ne radi, prva stvar koju trebate učiniti je uvijek gledati na poruke o pogreškama. Kako otvoriti web konzolu u oba Firefox i Chrome?

- F12 (Chrome and Firefox)
- Open Menu > Developer > Web Console (Firefox)

  Open Menu > More Tools > Developer Tools > Console Tab (Chrome)
- Ctrl + Shift + K (Firefox)
  Ctrl + Shift + I (Chrome)

2. Možete odredite varijablu sa i bez var ključnu riječ. Na koji način je trebate koristiti?

- Varijabla proglašena sa “var” ključna riječ pripada mjesnim okvirima, a drugi pripada globalnom okviru. Uvijek koristite var kako bi se izbjeglo zagađivanje globalni doseg. O varijablu proglasio s 'var' ključna riječ pripada globalnom okviru, a drugi pripada lokalnom okviru.

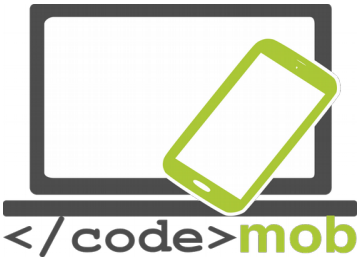
3. Zašto treba uvijek pisati komentare?

- Ključno je za objašnjavanje pojedinog dila koda. Bilo da se radi o vama kao nastavak rada na starom projektu ili rad u timu.
- Jednostavna je metoda poništavanja dijela koda bez da ga izgubimo

Uvijeti

1. Postoji li logična razlika između structure “ if else” I structure “if” ukoliko postoje suprotni uvijeti?

- Ne postoji ali “if else” je puno učinkovit



2. Kako možete napraviti petlju kroz svaki predmet a da ne poznajete dužinu?

- Kroz korištenje niz vrijednosti .length.

3. Što može unaprijediti navedeni kod?
for (var i = 0, len = a.length; i < len, i++) { /* ... */ }
brings compared to

for (var i = 0; i < a.length, i++) { /* ... */ }
?

- Varijabla “len” provjerava se samo jednom umjesto jednom po ponavljanju.

Funkcije

1. Dali je raspored argumenata funkcije naziva “divide” (kaoja vraća vrijednost podijele prvog argumenta kroz drugi) važna?

- Da, mjenja rezultat

2. Koja je razlika između “function call” I “function reference”?

- 'Poziv funkcije' (njeno ime slijedi zagradi) zatražiti izvršenje funkcija koda pravo u trenutku poziva. A 'funkcija reference "(u povratni poziv na primjer) jednostavno pohraniti naziv funkcije za buduću upotrebu.

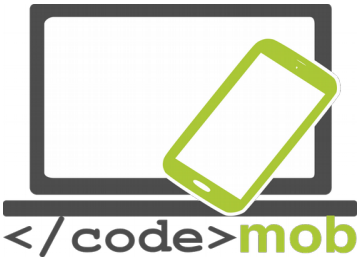
3. Koji su pridržane ključne riječi koj se koriste za određivanje koje se vraća vrijednost po pozivu funkcije?

- Return

DOM

1. Koja od navedenih metoda se ne referira na prvi ili samo pronađeni događaj?

- document.querySelector()



- document.getElementById()

2. U sljedećem kodu, koji je argument prima povratni poziv
document.getElementById('myID').addEventListener('click', myCallBack);

- Događaj koji je aktivirao callback

3. Kako možemo dobiti vrijednost kontrole nad HTML-om (UI) npr. Input tag?

- Kroz svojstvo .value koja vraća sadržaj "value" atributa

CSS

1. Koji CSS selector omogućava pronalazak paragrafa koji je direct "child" div-a ?

- Div > p

2. Koji CSS selector omogućava targetiranje samo prvog paragrafa bez obzira na broj potomaka" ?

- p:nth-of-type(1)

3. Ako dva CSS selektora (.red et #blue) ciljaju isti element, koje će biti boje?

- Plave

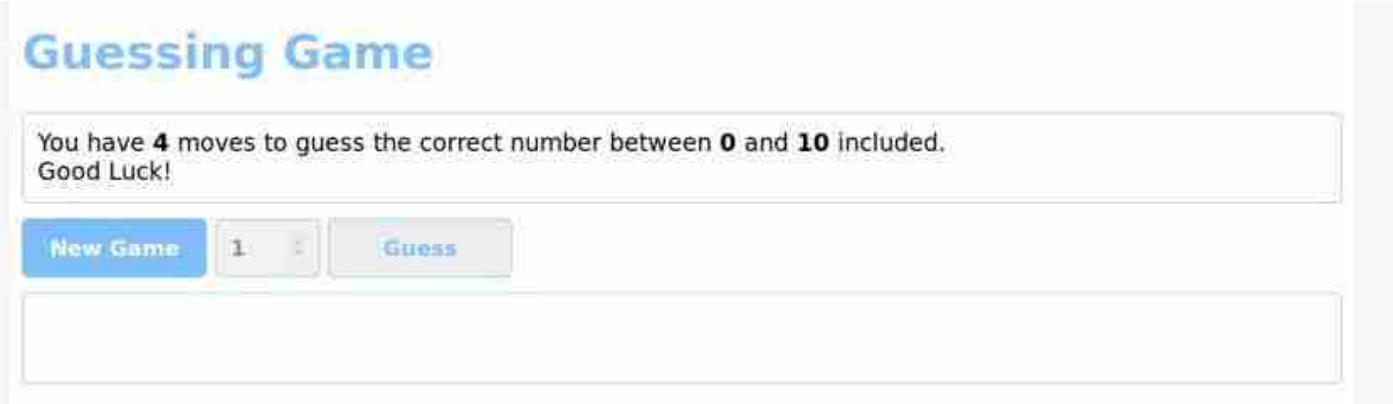
Labaratori programiranja: Igra pogađanja

Sada kada razumijete osnove HTML/CSS/JavaScript-a vrijeme je za zabavu i vašu prvu igru…..

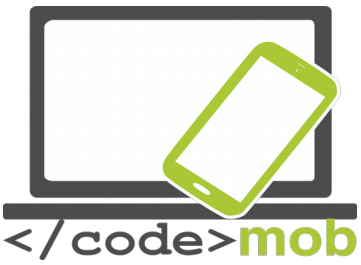
U ovoj lekciji ponovno će te kreirati igru pogađanja. To znači da će te kreirati:

1. HTML dokument u koji će te smjestiti sve elemente potrebne za igru
2. CSS dokument kroz koji će te urediti izgled vase igre
3. JavaScript dokument koji će sadržavati sve funkcije potrebne za igru

Ovako bi igra trebala izgledati:



Na sljedećoj stranici pronaći će te korak po korak upute …ako je potrebno..



Početak

1. Za početak potreban nam je barem jedan numerički brojač koji će testirati rezultat ali I neka vrsta “kontenjera” u koji ć emo unositi rezultate I podatke čak i ako je u početku prazan ili nevidljiv.

Pokušajte razmisliti o tome što je potrebno za gum “nova igra” postavite se u ulogu igrača I programera I odgonetnite što je potrebno a što nije. Nakon završetka vježbe možete se vratiti u početak koda I mijenjati kod prema vašim idejama.

2. Pokušajte zamisliti koji su vam sve koraci potrebni kako bi programirali ovu mini igru.

   Ovo će biti početak vašeg bloka koda. Ako se pojavi problem jeednostavno ga razlomite na manje dijelove.

Ako se pokaže potreba da morate višestruko koristi isti dio koda pravi put za to je koristiti funkcije posebno ako se početni parametric mogu mijenjati.

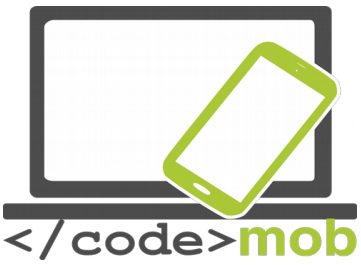
Uvijek koristite varijable umjesto više teških linija koda I vrijednosti kako bi jednostavnije mjenjali parameter igre kao što su broj pokušaja. Takav nači poboljšava preglednost I čitkost koda, održavanje i kod čini jednostavnijim.

Ostavljate komentare u kodu (objašnjavajući čemu služi linija koda ili koja je akcija iza navedenog koda) od koristi je vama I vašem timu.

3. getRandomIntegerBetween() je jednostavan primjer funkcije koji poboljšava čitanje programa I pretvara dio koda u višestruko iskoristivi kod.

Uzmite u obzir da nasumične funkcije nije nešto što je bazično u računalnoj znanosti. Računala su deterministička što znači ako postavljate isto pitanje dobit će te isti odgovor svaki puta. Naravno postoje računala posebnih namjena načinjena samo za slučajne odabire. Pogledajte pseudo random u tražilici Google.

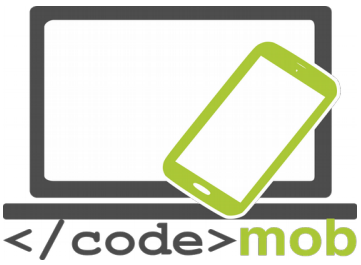
http://engineering.mit.edu/ask/can-computer-generate-truly-random-number



U checkResponse(), vidimo da povratak može biti I izlaz iz funkcije bez da dobijemo povratak premošćivanjem “završetak igre” testiranja nakon što je igra već dobivena.

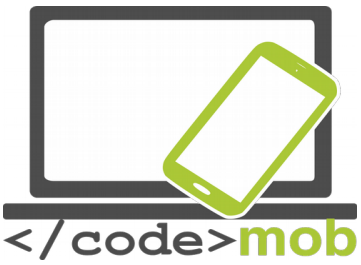
Na kraju vidimo da web tehnologije poput JavaScripta-a svode se na poboljšanja HTML-a I DOM-a sa funkcijama poput .getElementByld(“korisnikPogađa”) I result.innerHTML ili CSS. Svaka od ovih relativno jednostavnih tehnologija radi u paru.

Sljedeća stranica, natuknice za strukturu I funkcije…

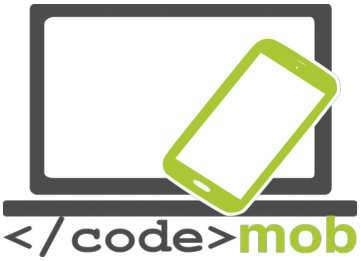


STRUKTURA I FUNKCIJE

```
// Return an integer between min and max included
        function getRandomIntegerBetween(min, max)

// Activate User Interface for new game
        function activateUI()

 // Game over: Deactivate User Interface
        function deactivateUI()

        function init(){
            // Reset to maximum
            // Clear logs
            // Get a new random number
            // Cheet ;)
            activateUI();
        }

// check the answer
function checkResponse()

// Number of available try for the player before game over

        // Random number to be guessed by the player

        // Minimum and maximum values (included) to choose from
```




```
// Display the rules

// Player log

// UI

// Can you guess the optimal way to always win this game ?
```



Reference i korištena literatura

JAVASCRIPT

[EN] JavaScript

http://www.w3schools.com/js/default.asp

http://www.w3schools.com/js/js_performance.asp

http://eloquentjavascript.net

https://developer.mozilla.org/en- US/en %C2%ADUS/docs/Web/JavaScript

https://developer.mozilla.org/en/docs/Web/API/Even t *https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript*

[EN] JavaScript Style Guide

http://www.w3schools.com/js/js_conventions.asp

JavaScript for the Total Non-Programmer

http://www.webteacher.com/javascript/

Defer & async [EN]

http://www.growingwiththeweb.com/2014/02/async•vs•defer•attributes.html

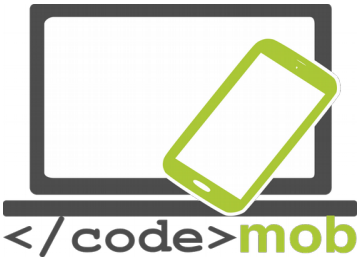
Modernizr

http://modernizr.com/

What is the function of the var keyword and when to use it ?

http://stackoverflow.com/questions/1470488/what-is-the-purpose-of-the-var-keyword- and-when-to-use-it-or-omit-it

[EN] The infamous 'this' keyword

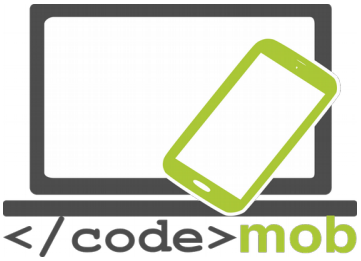


https://www.sitepoint.com/mastering-javascripts-this-keyword/

[EN] ECMA 6 • The future of JavaScript

http://es6-features.org/#Constants

The JavaScript Source is an excellent JavaScript resource with tons of "cut and paste" **JavaScript examples for your Web pages**. All for free!
http://www.javascriptsource.com/



HTML & CSS

HTML5 Tutorial
http://www.w3schools.com/html/default.asp

CSS3 Tutorial
http://www.w3schools.com/css/default.asp

And so much more on Google, YouTube and the whole web!

Thanks to David Collignon, for the precious help and course notes.