

Web картографија

Предавање 6

Доц. др Милан Килибарда

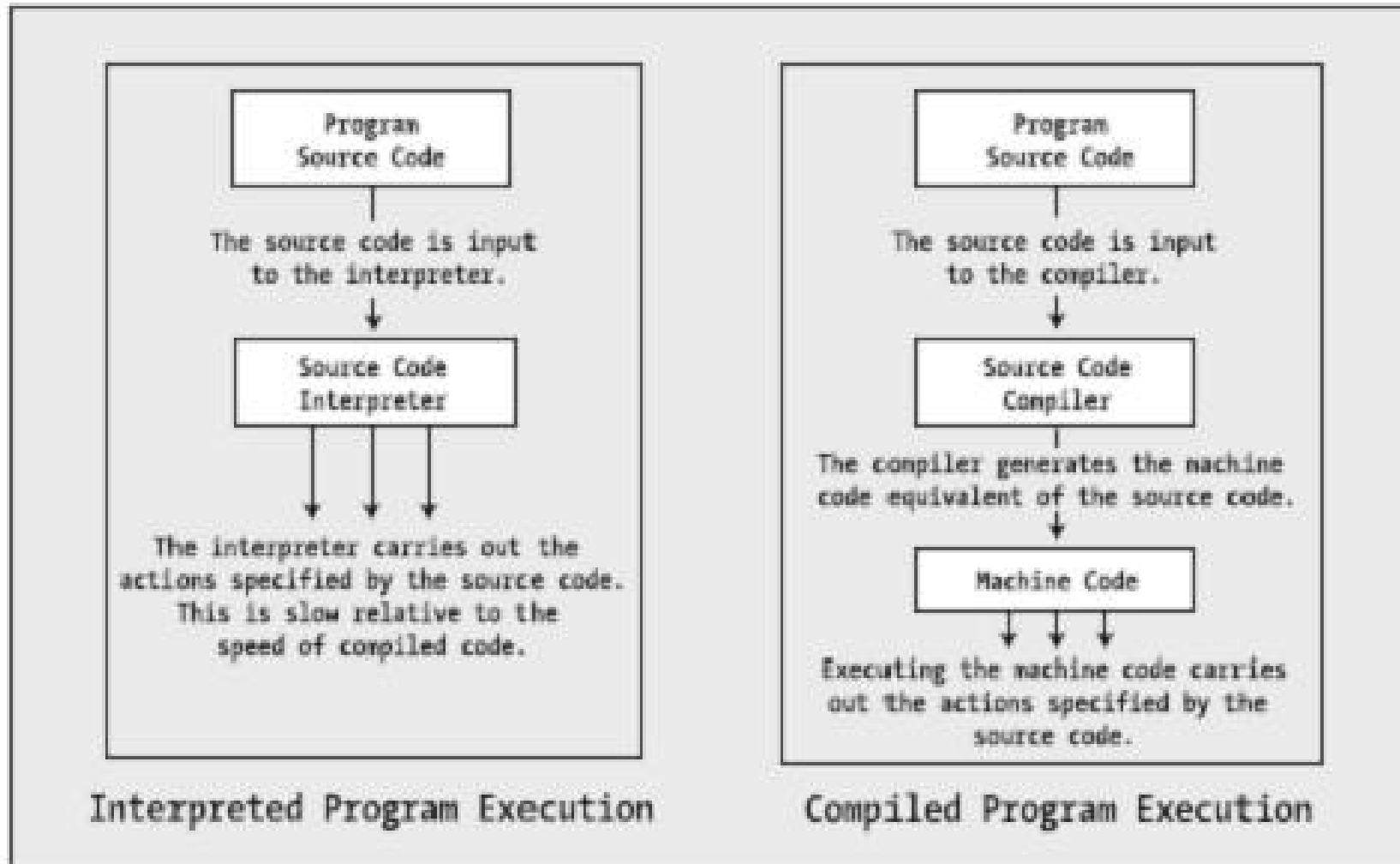
kili@grf.bg.ac.rs

Грађевински факултет,

Одсек за геодезију и геоинформатику,

Универзитет у Београду

Програмирање и скриптовање. Компајлер и интерпретер



Мотивација

- HTML у једном тренутку постаје ограничавајући фактор;
- Потребна је нова технологија за реализацију динамичких делова апликације.

Чему служи

- Омогућава динамички HTML садржај;
- Може да мења вредности HTML елемената и атрибута;
- Користи се за проверу исправности попуњених формулара;
- Реагује на догађаје типа клика, слања формулара итд. ;
- Остварује разне временске функције ...

Историја

- Децембар 1995.год, Netscape и Sun су представили JavaScript 1.0, оригинално назван **LiveScript**;
- Microsoft прати са језиком JScript
- ECMAScript 1.0, касније ISO/IEC 16262 стандард;
- Данас, Netscape и Microsoft стандарди су идентични у преко 95% случајева.

Неке особине

- Платформски неутралан;

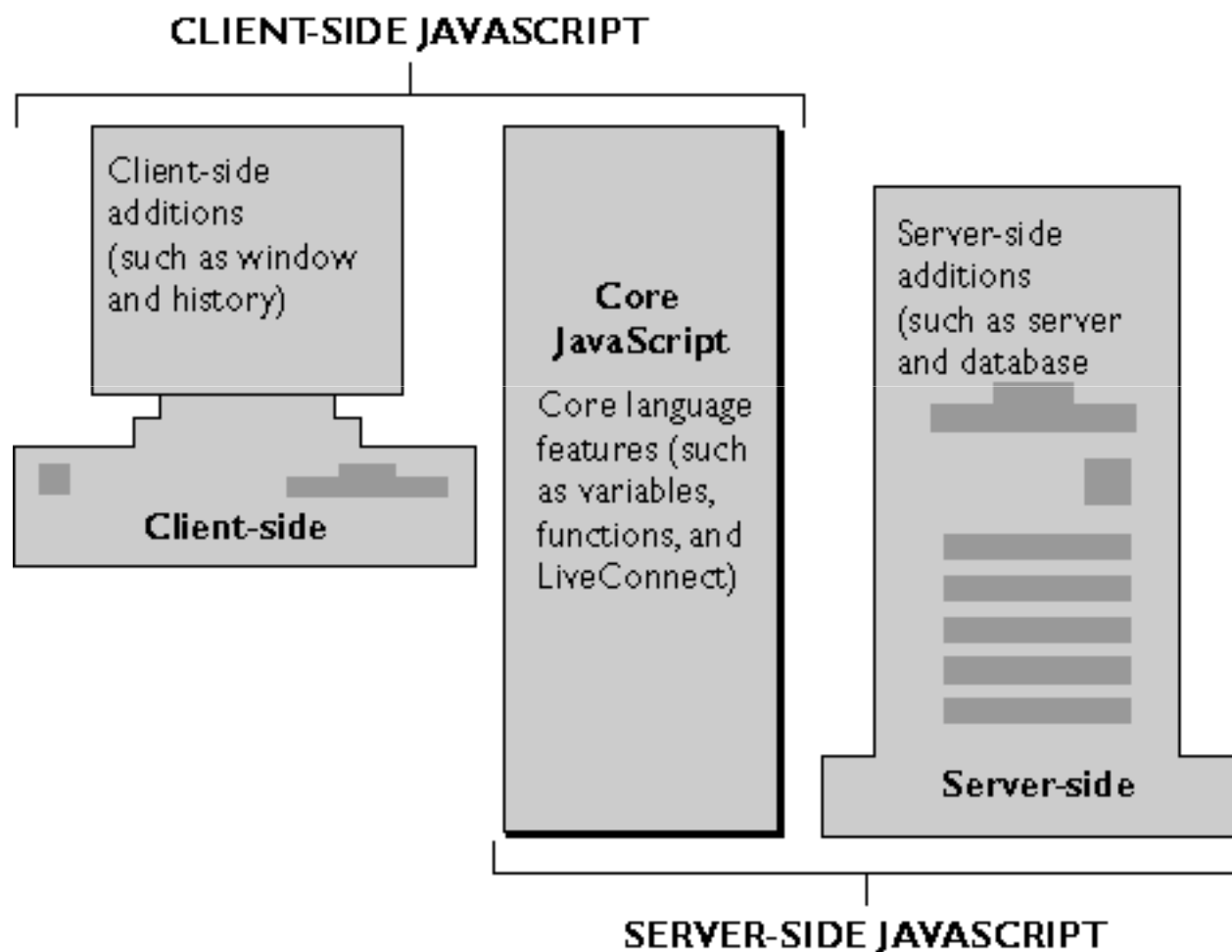
Browser је платформа за рачунање (интерпретер).

- Модуларно програмирање;
- Интегрисаност са HTML-ом (TV i DVD);
- Нема великих сличности са Јавом;

Слична синтакса са Јавом;

Објектно оријентисан.

Графички приказ организације језика



Извршавање кода

1. У оквиру HTML кода;
 1. У head тагу;
 2. У body тагу .
2. Или у екстерном фајлу;
3. Или некад без `script` тага, кад је у оквиру неког атрибута `html` тага.

Пример 1.1

```
<html>
```

```
<head>
```

```
<title> Java skript je u head tagu </title>
```

```
    <script language="javascript">
```

```
    var sIme=" ";
```

```
    function pozdrav(sIme) {
```

```
        document.write("Dobar dan! Vase ime je: " + sIme + ".  Ucim  Web kartografiju! <br>");
```

```
    }
```

```
    //Ovako se pise komentar u JS
```

```
    sIme=prompt("Upisite Vase cenjeno ime", "Milan Kilibarda");
```

```
    pozdrav(sIme);
```

```
    </script>
```

```
</head>
```

```
<body>
```

```
</body>
```

```
</html>
```

Пример 1.2

```
<html>
```

```
<head>
```

```
<title> Java skript je u BODY tagu </title>
```

```
</head>
```

```
<body>
```

```
    <script language="javascript">
```

```
    var sIme=" ";
```

```
    function pozdrav(sIme){
```

```
        document.write("Dobar dan! Vase ime je: " + sIme + ". Ucim Web kartografiju! <br>");
```

```
    }
```

```
    //Ovako se pise komentar u JS
```

```
    sIme=prompt("Upisite Vase cenjeno ime", "Milan Kilibarda");
```

```
    pozdrav(sIme);
```

```
    </script>
```

```
</body>
```

```
</html>
```

Пример 2

<html>

<head>

<title>JavaScript je u eksternom fajlu</title>

<script language="javascript" type="text/javascript" src="eksterni.js">

</script>

</head>

<body>

</body>

</html>

Пример 3

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>Inline JavaScript</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<A HREF="javascript:history.back()">back to previous page</a>
```

```
</BODY>
```

```
</HTML>
```

Променљиве

- Садрже вредности које програм смешта у оквиру меморије рачунара;
- Да би се користиле у оквиру програма:
 - Написати наредбу која креира променљиву;
 - Доделити јој жељено име.
- На крају вредност треба да се додели променљивој.

Имена променљивих

- Име које се додељује променљивој се назива и идентификатор;
- Правила и конвенције при именовању променљивих:
 - Идентификатор мора почињати словом, знаком долар (\$), или доњом цртом (_);
 - У оквиру имена се могу користити бројеви, али не као први карактер;
 - Не могу се користити празна места у оквиру имена;
 - Не могу се користити резервисане речи за идентификаторе;

var

- У оквиру JavaScript језика, језика, резервисана реч се користи за креирање променљивих (објекта и функција);
- Креирање променљиве се назива и декларација променљиве ;
- Да би се доделила одређена вредност или да би се иницијализовала променљива користи се синтакса:
var imePromenljive = vrednost;
- Вредност додељена променљивој може бити знаковна или нумеричка вредност;

Резервисане речи

**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**

Примери за var

```
<SCRIPT LANGUAGE="JavaScript">  
  
<!--  
  
    var goodvar1 = "goodvar1";  
  
    var Goodvar2 = "Goodvar2";  
  
    var Good_var3 = "Good_var3";  
  
    var _Goodvar4 = "_Goodvar4";  
  
  
    // var 1badvar = "1badvar";  
  
    // var bad var2 = "bad var2";  
  
    // var var = "bad_var3";  
  
//-->  
  
</SCRIPT>
```

Типови података који се додјељују променљивој

- Специфична категорија информација које се садрже у оквиру променљиве;
- Помаже да рачунар одреди колико је простора потребно да би се сместила одређена променљива;
- Тип података дефинише и врсту операција које се могу извршити са датом променљивом;

Декларација типова

- Виши програмски језици ("ЈАКО ПРОГРАМИРАЊЕ"):
 - Захтевају да се декларише тип података променљиве;
 - Статички тип података;
 - Тип података се не мења након саме декларације.
- Једноставнији програмски језици- JavaScript ("СЛАБО ПРОГРАМИРАЊЕ")
 - Не захтевају да се декларише тип података променљиве;
 - Динамички тип података:
 - Тип података се може мењати након саме декларације;

У суштини само два типа података за променљиву, ако изузмемо објекте

Нумерчки (у JavaScript-у све су рационални бројеви double precision -исто као у Матлабу.):

- целобројни бројеви,
- рационални бројеви,
- хексадецимални
- логички тип(boolean).

Тектуални- string тип.

Целобројни тип

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE="JavaScript">

    <!--

      var number_1 = 999;

      var number_2 = 1;

      var sum = number_1 + number_2;

      document.write(sum);

    //-->
  </SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Рационални бројеви

```
var number_1 = 999;
```

```
var number_2 = 1.00385;
```

```
var sum = number_1 + number_2;
```

```
document.write(sum);
```

Tip string

```
<SCRIPT LANGUAGE="JavaScript">
```

```
<!--
```

```
var string_1 = "<pre>for my next astounding trick, \nI will make  
this string \ndisplay on \nfour different lines.</pre>";
```

```
document.write(string_1);
```

```
// PRE element contains preformatted text -->
```

```
</SCRIPT>
```

Null вредност - антивредност

- Вредност **null** је:
 - тип података/вредност која се може доделити променљивој;
 - Показује да променљива не садржи никакву корисну вредност (АЛИ ЈОЈ КАСНИЈЕ МОЖЕ БИТИ ДОДЕЉЕНА !!!);
 - Променљива са вредношћу "null" има такву додељену вредност
 - null је у ствари "без вредности"

null

```
var novalue=null;
```

```
document.write(novalue);
```

Оператори

- Аритметички;
- На нивоу бита;
- Релационални;
- Логички;

Аритметички оператори

Оператор	Опис	Оператор	Опис
+	сабирање	+=	сабирање додела
-	одузимање	-=	одузимање додела
*	множење	*=	множење додела
/	дељење	/=	дељење додела
%	модуо	%=	Модуо додела
++	инкремент	--	декремент

Пример аритметички оператори

```
var a=1+1;  
var b=a * 3;  
var c=b/4;  
var d=5/a;  
var e = -d;
```

```
document.write("a, b, c, d, e, su", " ",a,  
";",b,";",c,";",d,";",e);
```

Стринг сабирање

```
var he_said="He said \"Will you Marry me?\"";
```

```
var she_said = "She said \"Not even if you were the last  
man on earth!\"";
```

```
var proposal = he_said + she_said;
```

```
document.write(proposal);
```

На нивоу бита

Operator	Upotreba	Opis
Logičko I (AND)	$a \& b$	Rezultatni bit je 1, jedino ako su oba bita 1 u ostalim slučajevima rezultat je 0.
Logičko ILI (OR)	$a b$	Rezultatni bit je 0, jedino ako su oba bita 0 u ostalim slučajevima rezultat je 1.
Logičko ekskluzivno ILI (XOR)	$a \wedge b$	Rezultatni bit je 1, ako biti imaju različite vrednosti, u suprotnom je 0.
Logičko NE (NOT)	$\sim a$	Komplementira bitove operanda a.
Pomeranje ulevo	$a \ll b$	Pomera binarni sadržaj operanda za b mesta ulevo. Prazna mesta popunjava sa vrednošću 0.
Pomeranje udesno sa znakom	$a \gg b$	Pomera binarni sadržaj operanda za b mesta ulevo. Prazna mesta popunjava sa vrednošću najstarijeg bita.
Pomeranje udesno sa nulama	$a \ggg b$	Pomera binarni sadržaj operanda za b mesta udesno. Prazna mesta popunjava sa vrednošću 0.

На нивоу бита

- $15 \& 9$ резултат 9 ($1111 \& 1001 = 1001$)
- $15 | 9$ резултат 15 ($1111 | 1001 = 1111$)
- $15 \wedge 9$ резултат 6 ($1111 \wedge 1001 = 0110$)

Логички оператори

Operator	Upotreba	Opis
I (&&)	expr1 && expr2	Rezultat je true, jedino ako su oba operanda true, u ostalim slučajevima rezultat je false.
ILI ()	expr1 expr2	Rezultat se dobija false, jedino ako su oba operanda false, u ostalim slučajevima rezultat je true.
NE (!)	!expr	Rezultat se dobija komplement od vrednosti operanada. Ako je operand true, rezultat je false, ako je operand false, rezultat je true

Логички

```
a = true;
b = false;
c = a || b;
d = a && b;
f = (!a && b) || (a && !b);
g = !a;
document.write( " a = " + a + "<br />" );
document.write ( " b = " + b + "<br />" );
document.write ( " c = " + c + "<br />" );
document.write ( " d = " + d + "<br />" );
document.write ( " f = " + f + "<br />" );
document.write ( " g = " + g);
```

- Rezultat izvršavanja prethodnog primera je:

```
a = true
b = false
c = true
d = false
f = true
g = false
```

Оператори поређења

Operator	Upotreba	Opis
Jednakost (==) (===)	Rezultat je true ako su operandi jednaki	$x == y$ rezultat je true ako su x i y jednaki.
Nejednakost (!=) (!==)	Rezultat je true ako su operandi različiti.	$x != y$ rezultat je true ako su x i y različiti.
Veće (>)	Rezultat je true ako je levi operand veći od desnog operanda.	$x > y$ ako je x veće od y .
Veće ili jednako (>=)	Rezultat je true ako je levi operand veći ili jednak desnom operandu	$x >= y$ ako je x veće ili jednako y .
Manje (<)	Rezultat je true ako je levi operand manji od desnog operanda	$x < y$ rezultat je true ako je x manje od y .
Manje ili jednako (<=)	Rezultat je true ako je levi operand manji ili jednak desnom operandu	$x <= y$ rezultat je true ako je x manje ili jednako y .
Jednako bez konverzije tipova (===)	Rezultat je true ako su operandi jednaki bez konverzije podataka	$x === y$ rezultat je true ako su x i y jednaki bez konverzije podataka
Različito bez konverzije tipova (!==)	Rezultat je true ako su operandi različiti bez konverzije podataka	$x !== y$ rezultat je true ako su x i y različiti bez konverzije podataka

Оператори поређења - битни за контролу тока

```
var a = 4;
```

```
var b = 1;
```

```
var c = a < b;
```

```
var d = a == b;
```

```
document.write( " c = " + c + "<br />" );
```

```
document.write ( " d = " + d );
```

//Резултат извршавања претходног примера
је:

```
c = false
```

```
d = false
```

Промнљива може бити низ

- Листа сличних променљивих;
- Користи се исти тип података у једном низу (не важи за JS);
- Нпр: вектор бројева или листа са именима студената итд;
- Низ је индексиран:
 - $A[0]$ - је први елемент низа A ;
 - $A[1]$ - други елемент низа A ;

Низ као променљива

```
var favorite_cities = ["Rome",  
    "Zurich", "Venice"];
```

```
document.write(favorite_cities);
```

Објекти као вредност променљиве

- Листа различитих променљивих;
- Организовање података на уређен и структуриран начин;
- Различити типови података у једном објекту;

Објект као променљива

```
var flight_reservation = {airline: "JAT", number: 342,  
    seat: "34f"};
```

```
document.writeln(flight_reservation.number);
```

```
document.writeln(flight_reservation["airline"]);
```

Команде у JavaScript-у

- Унос података;
- Излаз;
- Контрола тока;
- Петље;
- Функције.

Ulaz izlaz - primer

```
<SCRIPT LANGUAGE="JavaScript">
```

```
<!--
```

```
var name=" ";
```

```
name=prompt("What is your name?");
```

```
alert("Hello " + name);
```

```
//-->
```

```
</SCRIPT>
```

Контрола тока - if

- Синтакса:

```
if (logIzraz) {izraz1;  
                };  
else {izraz2;  
        };
```

Пример за if

```
var secret_password="jasmine";

var entered_password=prompt("enter the secret code to
continue: ", " ");

if (entered_password != secret_password){

    document.write("Nice try, but that is not the
secret password!");
};

else {
    document.write("Yes! That is the correct
password.");
};
```

Пример 2 за if

```
var username="user";
var password="karta";
var entered_username=prompt("enter username: ", " ");
var entered_password=prompt("enter password: ", " ");

if ((entered_username == username) && (entered_password == password)){
    document.write("You have been Logged in!")
};

else{
    document.write("Sorry, either the username or password is incorrect.");
};
```

Пример 3 за if

```
if (mesec == 1)
    ime_meseca = "Januar";
else if (mesec == 2)
    ime_meseca = "Februar";
else if (mesec == 3)
    ime_meseca = "Mart";
else if (mesec == 4)
    ime_meseca = "Maj";
elseif
    ....
else if (mesec == 12)
    ime_meseca = "Decembar";
```

Команда switch

```
switch (izraz) {  
    case vr1: naredbe1; [break;]  
    ...  
    case vrN: naredbeN; [break;]  
    [default: naredbeDef;  
}
```

Пример команды switch

```
var password_administrator="ice";
var password_developer="snow";
var password_user="water";

var entered_password=prompt("enter the secret code to continue: ", " ");

switch (entered_password) {

    case "ice":

        alert("you have administrator access");

        break

    case "snow":

        alert("you have developer access");

        break

    case "water":

        alert("you have user access");

        break

    default:

        alert("password incorrect!");

        break

}
```

Петље - while

- Синтакса:

```
while(logIzraz){  
    naredbe;  
}
```

Пример 1 while

```
i=1  
while(i<=10){  
    document.write(i+ "<br />");  
    i=i+1;  
};
```

Пример 2 while

```
var requested_month=0;
var MonthArray=["", "January", "February", "March", "April", "May",
"June", "July", "August", "September", "October", "November",
"December"]

document.write("Enter a number between 1 and 12, and I will tell you
the name of the month.<br>");
requested_month=prompt("Enter a number between 1 and 12 (or 0 to
exit)", 0);

while (requested_month != "0"){
    document.write(MonthArray[requested_month] + "<BR>");
    requested_month=prompt("Enter a number between 1 and 12 (or 0
to exit)", 0);
}
```

Петље- do - while

```
do {  
    naredbe;  
} while (logIzraz);
```

- Пример:

```
i=1;  
do {  
    document.write(i+ "<br>");  
    i=i+1;  
} while(i<=10)
```

Петље - for

Синтакса:

```
for(inicijalizacija, logIzraz, iteracija){  
    naredbe;  
}
```

Пример:

```
favorite_cities = ["Rome", "Zurich", "Venice"];  
var i;  
  
for(i=0; i<=2; i++){  
    document.write(i+ ":  "+ favorite_cities[i]+ "<br />");  
}
```

Петље - for in

- Синтакса:

```
for (imePromenljive in imeSkupa) {  
    naredbe;  
}
```

Primer za for in

```
favorite_cities = ["Rome", "Zurich", "Venice"];  
var i;
```

```
for(i in favorite_cities) {  
    document.write(i + ": " + favorite_cities[i]  
        + "<BR>");  
}
```

Функције

- Функције представљају групу наредби која се може извршити као појединачна јединица;
- Функције су корисне јер чине програм прегледнијим и лакшим за ажурирање;
- Функција се састоји из декларације функције и њеног позива.

Декларација функција

- Декларације функције се може наћи у заглављу и телу HTML koda, али препоручиво је да се смести у заглавље HTML кода.

- Синтакса за дефинисање функције је :

```
function imeFunkcije(parametri) {  
    naredbe;  
}
```

Пример за функцију без аргумената

```
function greet_the_user() {
```

```
    document.write("Zdravo !<br>");  
};
```

```
// Funkcija se ne izvrsava dok je ne pozovemo
```

```
greet_the_user();
```

```
greet_the_user();
```

```
greet_the_user();
```

Пример за функцију са аргументима

```
var name=" ";
```

```
function greet_the_user(name){
```

```
    document.write("Zdravo ! " + name + " je moje  
ime <br>");
```

```
}
```

```
greet_the_user("Milan");
```

Домен променљиве

```
var x = 10;  
function f()  
{  
    var y = 15;  
  
    function g()  
    {  
        var z = 25;  
        alert(x+y+z);  
    }  
    g();  
}  
  
f(); // this displays 50
```

The diagram illustrates the scope resolution process for the provided code. It shows three nested scopes, each represented by a red bracket and a variable in a red circle:

- global scope:** Contains the variable `x`.
- f scope:** Contains the variable `y`.
- g scope:** Contains the variable `z`.

Blue arrows indicate the lookup path for the `alert(x+y+z)` call within the `g` function:

- The variable `z` is found in the `g` scope.
- The variable `y` is not found in `g`, so the lookup proceeds to the `f` scope, where it is found.
- The variable `x` is not found in `f`, so the lookup proceeds to the `global` scope, where it is found.

Наредба return

- Служи да би се вратио резултат функције позваној наредби;

```
function kub(a){  
    d = a*a*a;  
    return d;  
}
```

Пример за return

```
var broj=0;  
var brojna3=0;
```

```
function kub(a){  
    d = a*a*a  
    return d;  
}
```

```
broj=prompt("Unesi neki broj",0)  
brojna3=kub(broj)  
alert("Kub broja:" + broj + " je :" + brojna3 )
```

Уградјене функције

- [escape](#)
- [eval](#)
- [isFinite](#)
- [isNaN](#)
- [number](#)
- [parseFloat](#)
- [parseInt](#)
- [string](#)
- [taint](#)
- [unescape](#)
- [untaint](#)
- [encodeURIComponent](#)
- [decodeURI](#)
- [decodeURIComponent](#)
- [alert](#)
- [getAttribute](#)
- [setAtt](#)

Пример за parseInt и parseFloat

```
<script language="javascript">
number1=prompt("Koliko godina imas?", 0);
number2=prompt("Koliko ima tvoj mlađji brat", 0);
document.write("Zajedno imate:",parseInt(number1) +
    parseInt(number2));
```

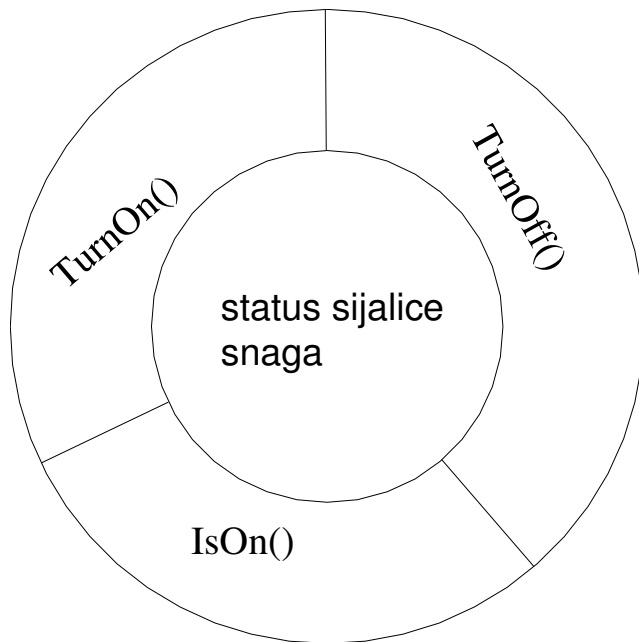
```
document.write("<hr>Rad sa decimalnim i parseFloat<br>");
number1=prompt("Koliko kosta \"Milka\" ?", 0.0);
number2=prompt("Koliko kosta \"Simka\" ?", 0.0);
document.write("Milka i Simka su zajedno: ",parseFloat(number1) +
    parseFloat(number2),"dinara");
</script>
```

Објекти

- JavaScript подржава објекте као структуре података;
- За сваки објекат су везане особине и методе;
- Методе су функције у оквиру објектата којима могу да се мењају њихове особине.

PODACI + FUNKCIJE vezane za te podatke

Енкапсулација



OSNOVNA KLASA JE SIJALICA

Подаци (особине –
Properties):

Status sijalice : boolean

Snaga : integer

Финкције (методе - **Methods**):

IsOn()

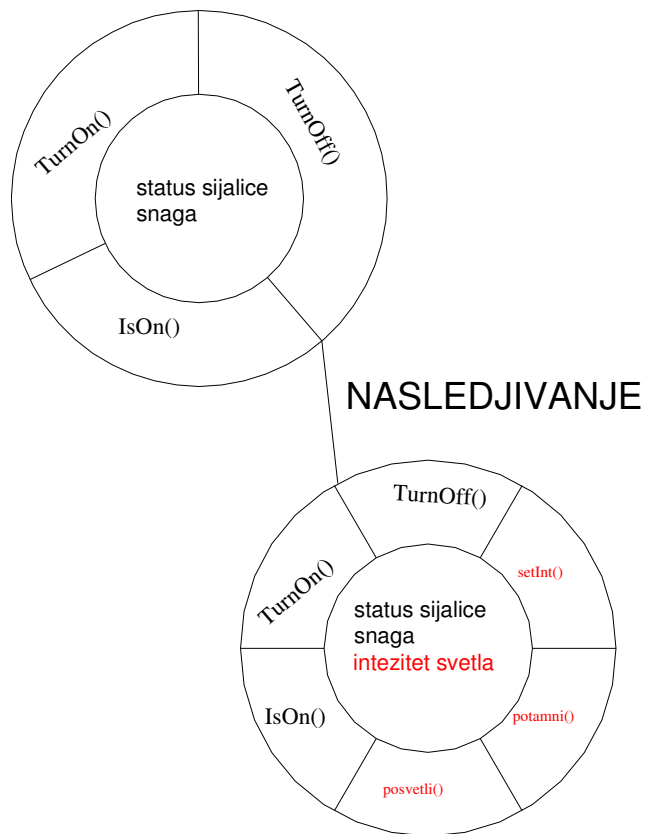
TurnOff()

TurnOff()

Енкапсулација

- Везују се подаци и функције, прави се чаура или капсула;
- Робустност се постиже скривањем података и детаља;
- Добија се један ефикасан систем са мало кода, можемо да користимо предефинисане објекте и методе, а да не знамо како су те методе испрограмиране, али да знамо чему служе;
- **Klasa je prototip objekta = model podataka + funkcije;**
= osobine objekta + metode;
- У Javascript можемо директно мењати особине објекта што није случај са другим језицима.

Nasledjivanje



Наслеђена класа

има све што и претходна + додатне особине (податке) + додатне методе (функције)

изворни објекат;

дериватни (проширени).

• КОД JAVASRIPТ ЈЕЗИКА
НАСЛЕЂИВАЊЕ СЕ ОДНОСИ
НА ОБЈЕКТЕ НЕ НА КЛАСЕ.

Полиморфизам

- Кад имамо више објеката са истом особином и методом;
- Можемо да применимо заједничке функције.

`niz_sijalica=[s1,s2,s3,si1,si2]`

Свакој сијалици у низу подеси да је угашена, користимо заједничку функцију `TurnOff()` и за објекте обичне сијалице и сијалице за које се подешава интензитет.

Објекти у Javascript-у

- На десетине предефинисаних објеката;
- Можемо да их користимо, а не морамо да знамо како су направљени;
- Неопходно је да имамо добар *reference guide*.

Уграђени објекти

- Array - низ
- RegExp - узорци за претраживање
- String - знаковни низ
- Date - време
- Event - објекти које генеришу догађаји
- Boolean - логичка променљива
- Math - математички објект
- DOM - HTML и JavaScript објекти

Особине и методи уграђених објеката

- Могу се наћи на
 - <http://www.w3.org/TR/DOM-Level-2-HTML/html.html>
 - <http://www.w3.org/TR/DOM-Level-2-HTML/ecma-script-binding.html>
 - https://developer.mozilla.org/en/Gecko_DOM_Reference
 - http://www.devguru.com/Technologies/ecmascript/quickref/javascript_index.html
 - <http://www.javascriptkit.com/domref/>

Пример коришћења објекта Date

Особине (**подаци**):

godina, mesec, dan, sat, minut, sekund

Методи (**функције**)

Преко 40 метода које манипулишу са датумом.

Инстанца објекта Date

Конструктор објекта датум је **Date**, те се време инстанцира на следеће начине:

```
imeDatuma = new Date();
```

Дакле сад имамо објекат који се зове *imeDatuma* и са њим можемо да манипулишемо.

Пример 1

```
tempDate = new Date()
```

```
document.write("The current date and time  
is: " + tempDate.toLocaleString())
```

Пример 2

```
tempDate = new Date(1997, 11, 25, 12, 15, 30)
```

```
document.write(tempDate + "<br>")
```

```
document.write("Date: " + tempDate.getDate() + "<br>")
```

```
document.write("Month: " + tempDate.getMonth() + "<br>")
```

```
document.write("Year: " + tempDate.getFullYear() + "<br>")
```

```
document.write("Day of the week: " + tempDate.getDay() + "<br>")
```

Пример 3

```
tempDate1 = new Date(1997, 11, 25, 12, 15, 30)
Sad = new Date()
```

```
function stampajDatum(tempDate){
document.write(tempDate + "<br>")
```

```
document.write("Date: " + tempDate.getDate() + "<br>")
```

```
document.write("Month: " + tempDate.getMonth()+1 + "<br>")
```

```
document.write("Year: " + tempDate.getFullYear() + "<br>")
```

```
document.write("Day of the week: " + tempDate.getDay() + "<hr>")
}
```

```
stampajDatum(tempDate1)
```

```
stampajDatum(Sad)
```

Креирање објекта са МЕТОДОМ

Креирамо модел података за нову класу објекта коју ћемо назвати Круг.

```
function Krug(x,y,r) {  
  this.xcoord = x;  
  this.ycoord = y;  
  this.radius = r;  
}
```

Креирање објекта са МЕТОДОМ

Креирамо методе за класу објекта (објекат)
Круг.

```
Krug.prototype.retPov = function () {  
    return ( Math.PI * this.radius * this.radius );  
};
```

```
Krug.prototype.retObim = function () {  
    return ( Math.PI * this.radius * 2 );  
};
```

Користимо наш објекат и методе

```
Novi_krug= new Krug(2,2,1.35)
```

```
document.write("Povrsina objekta koji se zove  
Novi_krug je :" + Novi_krug.retPov()+"<br>")
```

```
document.write("Obim objekta koji se zove  
Novi_krug je :" + Novi_krug.retObim()  
+"<br>")
```

Креирање објекта са методом2

```
function Krug(x,y,r) {  
    this.xcoord = x;  
    this.ycoord = y;  
    this.radius = r;  
  
    this.retPov = function () {  
        return ( Math.PI * this.radius * this.radius );  
    }  
  
    this.retObim = function () {  
        return ( Math.PI * this.radius * 2 );  
    }  
  
}
```

DOM објекти

Document Object Model слично као BOM
BOM зависи од браузера.

DOM – је стандардан;

Служе за организације веб странице;

Основни објекти могу бити: слика,
формулар , поље.

Пример : <http://paletteman.com/>

DOM објекти

- DOM објекти су
 - **window** - активни прозор, највиши ниво
 - **navigator** - даје инфо о brauzeru
 - **history** - историја браузовања
 - **screen** - подаци о екрану
 - **location** - рад са URL-ом
 - **document** - активни документ је HTML DOM објекат, унутар кога можемо да приступамо појединачним елементима.

document.body.innerHTML - access to html under body

- сви HTML елементи су HTML DOM објекти

Anchor

Table - `var xCell=document.getElementById('myTable').rows[0].cells[0];`

TableRow

Form A special object for many different tags

Anchor

Table - `var xCell=document.getElementById('myTable').rows[0].cells[0];`

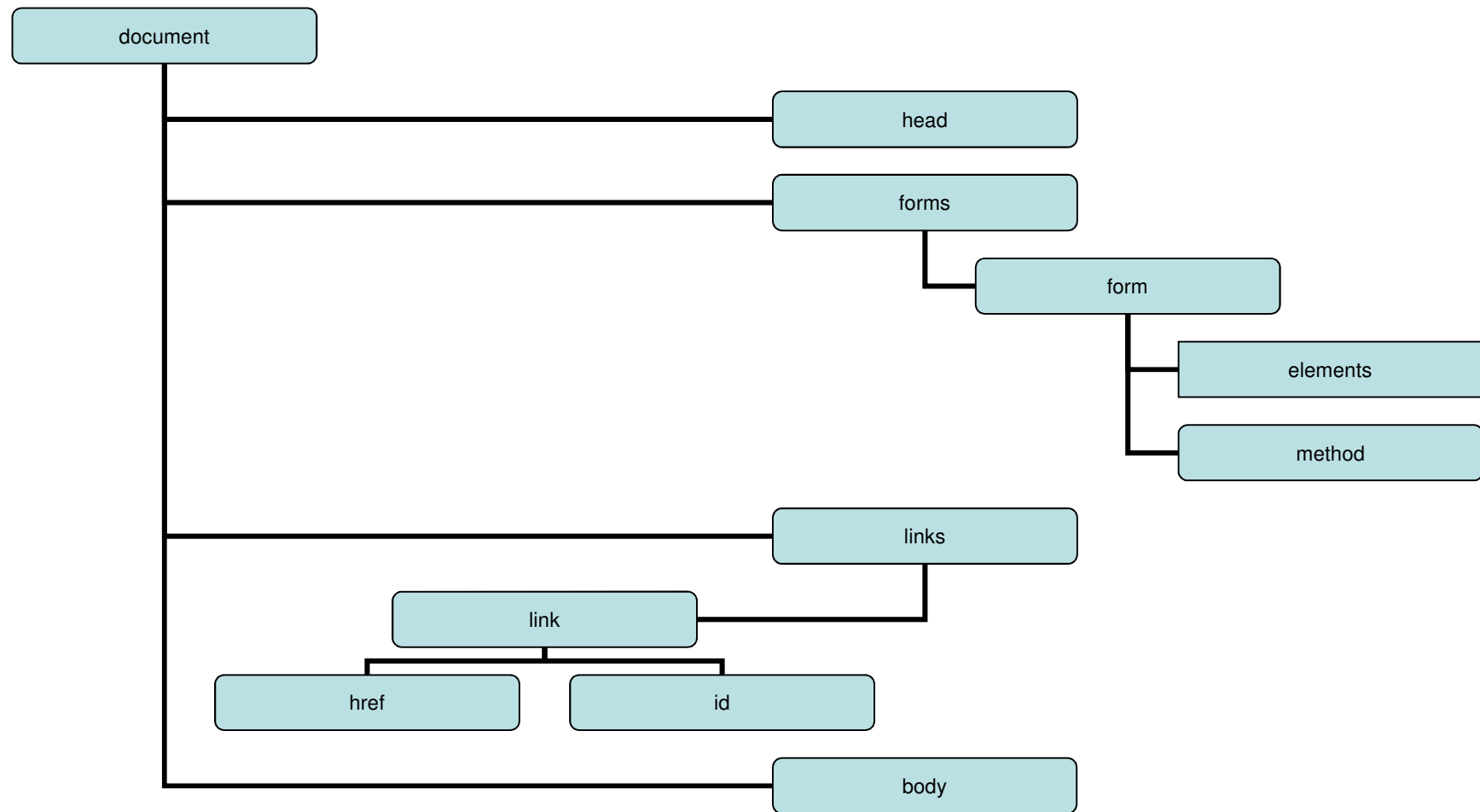
TableRow

Form

A standard way of getting a reference
to an object using its ID

For example in html we had `<table ID="myTable">...`

DOM објекти



Заједничке особине DOM објектата

- *attributes[]* - низ атрибута елемента
- *childNodes[]* - низ деце елемента
- *className* - име класе елемента
- *id* - id елемента
- *innerHTML* - садржај елемента
- *nodeName* - име ентитета
- *nodeValue* - вредност ентитета
- *parentNode* - родитељ ентитета
- *style* - стил елемента
- ...

Заједничке методе DOM објектата

`appendChild(element)`

`getAttribute(imeAtributa)`

`getElementsByTagName(imeTaga)`

`hasAttribute(imeAtributa)`

`insertBefore(noviElement, targetElement)`

`removeAttribute(imeAtributa)`

`removeChild(imeDeteta)`

`replaceChild(noviDete, staroDete)`

...

Пристап DOM објектима

- DOM објектима се приступа помоћу ID-а и методе *getElementsById* али и на друге начине

- Синтакса:

e/Nivo0.e/Nivo1...e/NivoK.imeAtributa

e/Nivo0.e/Nivo1...e/NivoK.innerHTML

- Где је:

e/NivoX=getElementsById(ID) ili

e/NivoX=getElementsByTagName(tag) ili

e/NivoX=imeAtributa ili

e/NivoX=firstChild...

- Уколико је спецификација елемента иста за више елемената онда као резултат даје низ тих елемената

Window opener

```
var newWindow
```

```
function makeNewWindow(){
```

```
    newWindow=window.open("", "", "HEIGHT=300,WIDTH  
    =300");
```

```
}
```

```
makeNewWindow()
```

Локација објекта

```
var window1
```

```
var window2
```

```
function makeWindows(){  
    window1=window.open("", "", "HEIGHT=300,WIDTH=300");  
    window2=window.open("", "", "HEIGHT=300,WIDTH=300");}
```

```
function loadPages(){  
    window1.location.href = "http://www.grf.bg.ac.rs/";  
    window2.location.href = "http://www.volcaniclab.com/";  
}
```

```
makeWindows();
```

```
if (window.confirm("Load Pages into Windows?")){
```

```
    loadPages();
```

```
}
```

Особине објекта

```
document.write("host = " + location.host + "<br>")
```

```
document.write("hostname = " + location.hostname + "<br>")
```

```
document.write("href = " + location.href + "<br>")
```

```
document.write("pathname = " + location.pathname + "<br>")
```

```
document.write("port = " + location.port + "<br>")
```

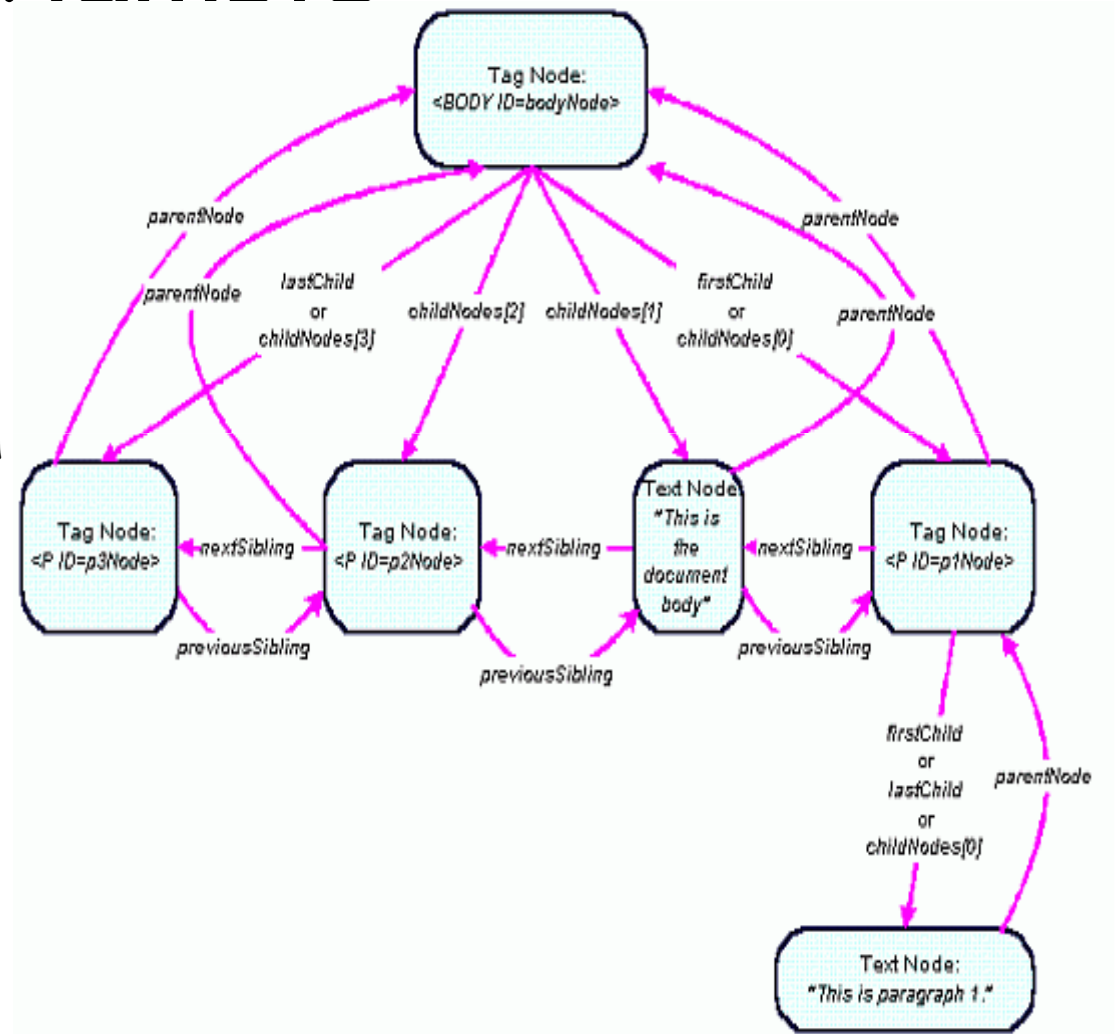
```
document.write("protocol = " + location.protocol + "<br>")
```

Особине и методе DOM објекта

- Осим заједничких сваки објект има своје особине и методе;
- Специфичне особине HTML елемената као објекта су једнаке њиховим атрибутима;
- Детекција догађаја су такође методи одређених HTML елемената.

A simple Tree Structure Example

```
<HTML>
  <HEAD>
    <TITLE> Simple DOM Demo
    </TITLE>
  </HEAD>
  <BODY ID="bodyNode">
    <P ID = "p1Node">This is paragraph
      1.
    </P>
    This is the document body
    <P ID = "p2Node"> </P>
    <P ID = "p3Node"></P>
  </BODY>
</HTML>
```



Onload - dogadjaj

```
function display_greeting(){  
  
    document.write("Hi there!<br>");  
  
    tempDate = new Date();  
  
    document.write("The current date and time is: " +  
tempDate.toLocaleString());  
  
    document.write("The time will not change if you  
refresh this page, try it!");  
  
}
```

Догађаји

Click

Dupli klik

Mouse over

Mouse out ...

Vidi primer

Пример везивања догадјаја и функције

```
var newWindow;  
function openNewWindow()  
{  
    newWindow=window.open("", "", "HEIGHT=300,WIDTH=300");  
}  
</SCRIPT>  
</HEAD>  
<BODY>  
  
    <p>JavaScript Links</p>  
    <ul>  
<li>[<A HREF="javascript:openNewWindow()">click to open a  
    window</A>] </li>  
</BODY>
```

Текст поља

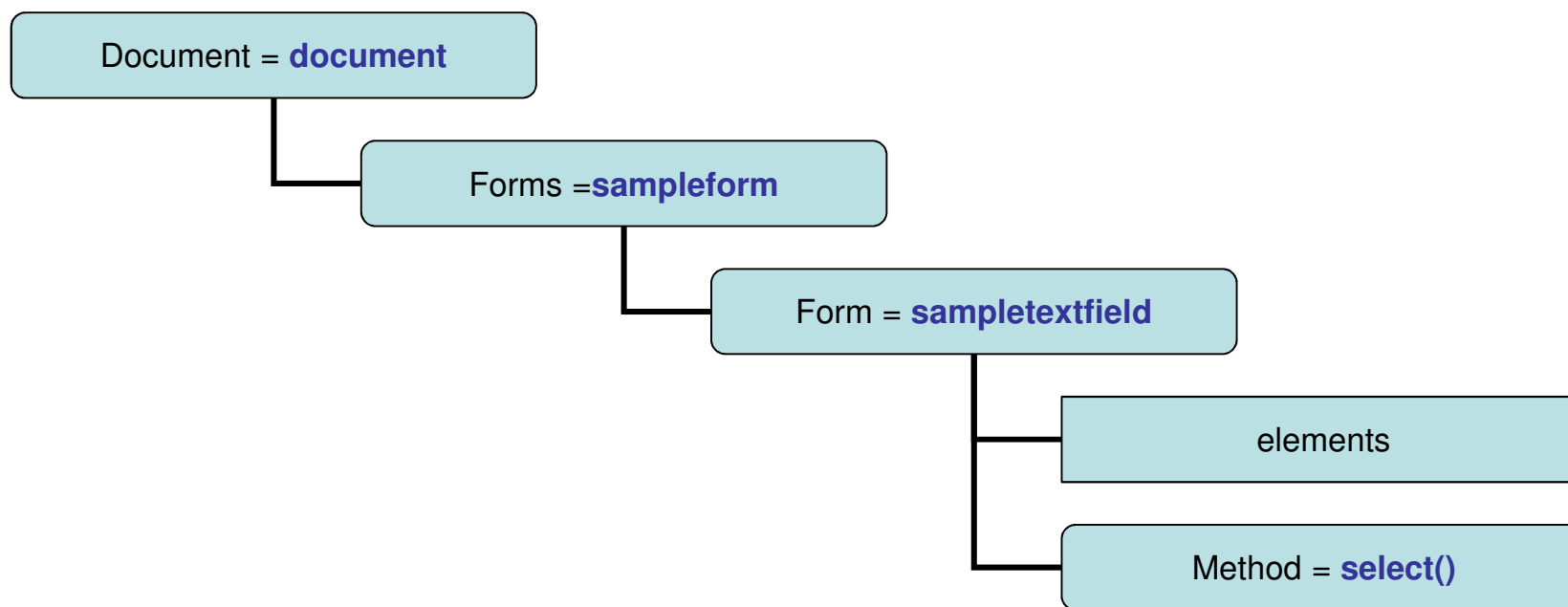
```
<html>
<head>
<script language="javascript">
function blurSampleField()
{document.sampleform.sampletextfield.select()}
</script>
</head>
<body>
  <form name="sampleform">
    <input name="sampletextfield" value="demonstrating text field methods"
      size="30">
    [<a href="javascript:selectSampleField()">Pokreni akciju</a>]

  </body>

</html>
```

Како приступити елементу DOM (Методи над текст пољима)

```
document.sampleform.sampletextfield.select()
```



Атрибути текст поља и манипулација са њима (Како приступити елементу DOM?)

```
<html>
<head>
<script language="javascript">
function SelectHandler(field)

{   window.alert("Select Event. The following text was selected: " + field.value())
</script>
</head>
<body>

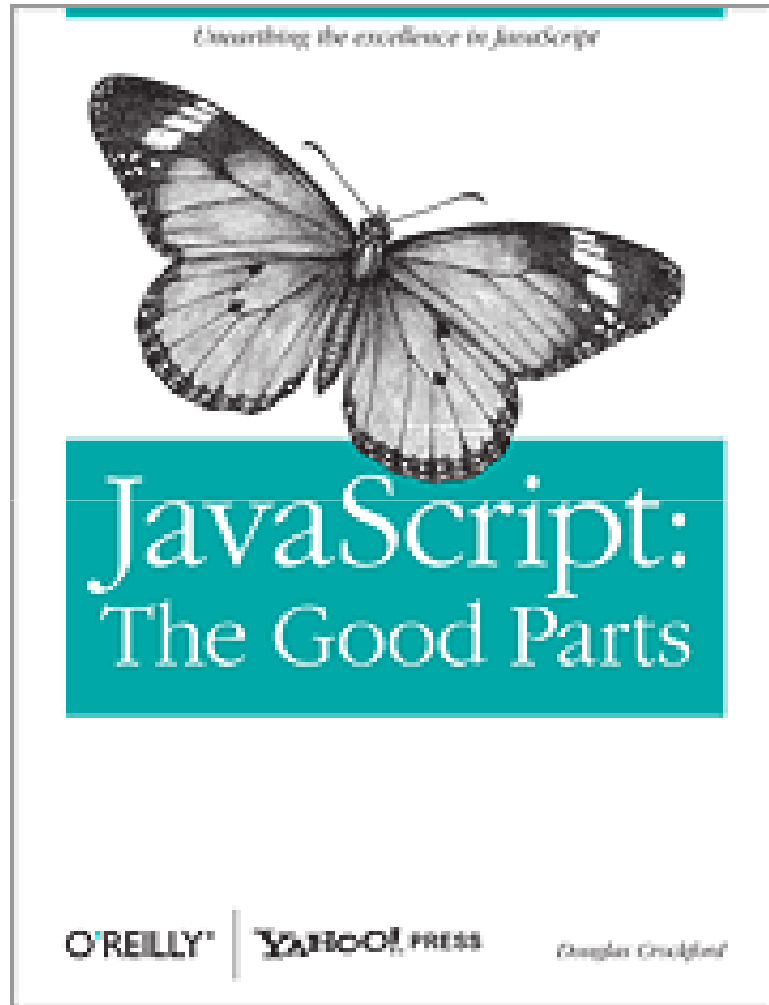
    <form>
<input type="text" value="select event" name="field4" onSelect="SelectHandler(this)" >
        triggers when field value is selected<br>

    </form>

</body>

</html>
```

Литература (рекомендация)



JavaScript: The Good Parts
Unearthing the Excellence in
JavaScript

By [Douglas Crockford](#)

Publisher: O'Reilly Media /

Yahoo Press Released: May

2008 Pages: 176