

PRAKTIČNI JAVASCRIPT

1. BOLJI JS

- Uvijek DEKLARIŠI variablu prije upotrebe
- NE-DEKLARIŠE VARIABLE SU GLOBALNE PO DEFAULTU
- UVJEK UPOTREBLJAVAT ; za kraj reda
- NE upotrebjavat GLOBALNE VARIABLE
- upotrebjavat HUNGARIAN metodu
- isbjegavat eval()

* HUNGARIAN METODA

- ispred naziva stavljač 1-2 slova koja označavaju i čemu se radi
 - b - Boolean (true, false)
 - c, n - count of some number of items
 - a - array
 - s - string
 - i - index u nekoj nizu
- funkcija
 - cElements
 - aNames
 - sName
 - aNames[iName]

* UPOTR.

- upotrebljavanje native funkcije - built in

Array.sort() funkcija - bez parametara sortira niz abecedno

↳ NOVE I S PARAMETRIMA

- continue statement za FOR PETLU → IDE NA SVJEDOČU
VRIJEDNOST C-a

* FOR/IN KONSTRUKT

```
for (stock in portfolio) {  
    total += portfolio[stock];  
}
```

→ EFIKASNJI NACIN
LOOP-OMA KROZ SVE
ELEMENTE U OBJEKTU
PORTFOLIO

* GOTCHAS

- OBJEKTI SU ASOCIATIVNI NIZovi

Obj.property je isto kao Obj.[property]"

NIZovi SE MOGU TRETIRAT KAO OBJEKTI, A
OBJEKTI KAO NIZOV

- Obj[3+4] je var kao Obj[7]
- BROJEVI i BOOLEANI + VARIABLE VRIJEDE SMO U FUNKCIJAMA LOKALNO

PR.

```

function f(x) { x+=5; }    ↗ OVDE LOKALNO JE X=10
function h() { var x=5;
}
f();
} → NAKON "ODRAGENE" JE X=5

```

- OBJEKTI i MIZOVI SE MOGU PROMIJENIT U DRUGOJ FUNKCII

PR.

```

function f(obj) { obj.d = new Date('now'); }
function h() {
  var x = new Object();
  x.d = new Date();
}
f();
} → NAKON OVOGA JE X.d PROMIJENJEN

```

#2. EXCEPTION HANDLING

* NE UPOTREBLJAVAT "WITH" PJU

* setInterval i setTimeout - poziv ih u funkcijama
a ne stringovima

pr.

```
setTimeout ( function () { moveElementBy(x);  
    x+=5;  
});
```

UMJESTO

```
setTimeout ("moveElementBy(x); x+=5;", 1000);
```

ovo nije efikasno jer
se kod mora interpretirati

* CONCATENACIJA STRINGOVA. UPOTREBLJAVAT += UMJESTO +

ovo NE:

```
str += "str x" + "str y" + "str z" ...; → NE-EFIKASNO JE  
POGOTOVO NAKON  
PUNO "ZBKIJANJA"
```

ovo DA:

```
str += "str x";  
str += "str y";  
str += "str z";
```

* KAO UPOTREBE VIŠE HTML MARKUP TEKTA

UPOTREBA VAT innerHTML U MJESTO DOM METODA.

innerHTML browser BRŽE OBRAO

* NE RADIT PUNO MANJIH DOM IZMJENA, NEGO JEDNU
VEĆU, ZBOG DOCUMENT REFLOW-a.

- KAO RJEŠENJE, MANJE IZMJENE RADIT NA TEMPORARY OBJEKTU
I KAO JE GOTOVU ZALJEPIT U DOKUMENT

MODULI

- namespaces

com_browsotic - otvoreni globalni objekt
- reverse domain names za ime

com_browsotic. NAME = "The module";

com_browsotic. VERSION = "1.0";

PROPERTY1

com_browsotic.method1 = function{

...radij fje...

dalje stvari
nije ostale

STVARI I FJE

"NA" GLOBALNI
OBJEKT

com_browsotic.property1 = [];

:

Exception handling

2. EXCEPTION HANDLING

BROWSER RAISES ERROR
EXCEPTION → SKRIPTA → EXPLICITNO BACI EXCEPTION (neke)

- try, catch, finally

* try {

hod se izvrsava normalno, ali u slučaju exceptiona izvršava se nekolicin catch blok hoda

}

→ INFORMACIJE → EXCEPTIONU

catch (e) {

ovaj kod se izvršava samo ako se u try rečenici dogodi exception

}

finally {

ova se vrsta izvršava, i ako nema exceptiona u try rečenici

}

catch : finally su:

OPCIJALNOST, ali ako postoji try mora biti bar jedna od catch i finally

* THROW EXCEPTION-a

- throw "ovo je greška";
- throw 12345;

- throw new Error ("Dogodila se greška!");

TO JE OVO

→ OVO JE
ERROR OBJEKT, IMAMO

PROPERTY :

- NAME - TIP EXCEPTIONA, OBICNO "Error"
- MESSAGE - STRING koji će se prikazati

* PRIMJER

```
function testException() {
```

```
try {  
    nekaFja();  
}
```

```
catch (e) {  
    alert (e.name + ": " + e.message);  
}
```

```
}
```

→ AKO SE U OVOJ FJI
DOGODI ERROR / EXCEPTION
↓
IZVRŠIT ĆE SE
OVAJ ALERT BOX
←

3. EVENT MODEL

- bolji event handling
- DOM level 2
- eventi mogu biti chainani ukupu i neovisno dodani i mjenuti
- faze: capture, at target, bubbling
- PRIMJERI MODERNOG (I KOMPLICIRANOG) EVENT HANDLINGA

#4. UNOBTRUSIVE JS

- JS u externim fileovima
- - no inline JS
- no JS u tagovima <script> </script>

} konvencija o tome
gdje se nalazi JS

#5. OBJECT-ORIENTED JS

- JS omogućava da mi napravim objekte
- objekti mogu bit standalone (svosi tip i property) ili mogu imati propertye od drugih objekata
- deklaracija vlastitog objekta je podnomo \rightarrow sa new
- extendanje propertya drugih objekata se radi s prototype propertyom
 - kad instanciramo objekt, on ima nove propertye
- deklaracija objekta

function Rectangle () {};

- napravio tip s istim imenom kao objekt

→ ZADAO JE OVO
EMPTY OBJECT

- stvaranje instance objekta

var myRect = new Rectangle();

- upotrebila "new"
- možemo napraviti u neku varijablu

- PROPERTY-*i* METHODS su definirane kod je object napravljena this keywordima:

pp:

function Rectangle (w, h) {

→ OVO SE ZOVE OBJECT KONSTRUKTOR.

this.width = w;

this.height = h;

}

- METODE SU FUNKCIJE KOD OBJEKATA. DODAVAJU SE UPOTREBOM PROTOTYPE OBJEKTA

NATIV FJE/METOD

*2.

Rectangle.prototype.area = function () {

return this.width * this.height;

}

Rectangle.prototype.perimeter = function () {

return (2 * this.width) + (2 * this.height);

}

- kod ne instance Rectangle s mijenjama, on ima oblik property w, h, area i perimeter

- class level property i metode

Rectangle.EMPTY = new Rectangle(0, 0);

→ NISU POVEZANI S INSTACAMA

→ pozivamo ih pr.

Rectangle.Max = Function (a, b) {

if (myRect.area > Rectangle.EMPTY.area)

- vraca jedan ili dva imena
već poznata

}

* VOBICAJENE METODE NA OBJEKTI, MA

• `toString()`

- vraca string prikaz objekta

`Rectangle.prototype.toString = function () {`

`return "Rectangle of width:" + this.width + "& height:" + this.height;`

`}`

- po defaultu je vraca: [object class]

→ OVAJE IDE KLAST
OBJEKTA

• `valueOf()`

- vraca "primitivni" vrijednost koja je
može upotrijebiti u numeričkim izračunima

`Rectangle.prototype.valueOf = function () {`

↳

`return this.area();`

`}`

• `equals()`

`Rectangle.prototype.equals = function (oRect) {`

→ Rectangle object
instance

`return (this.width == oRect.width & & this.height == oRect.height);`

`}`

↳ ovaj nam treba još ne možemo uporabiti
objekte (myRect1 == myRect2)

PRIKAZ:

`myRect.equals(myRect2);`

• compareTo()

Rectangle.prototype.compareTo = function(oRect) {
 ↗ Rectangle object
 ↗ instance

return (this.area() - oRect.area());

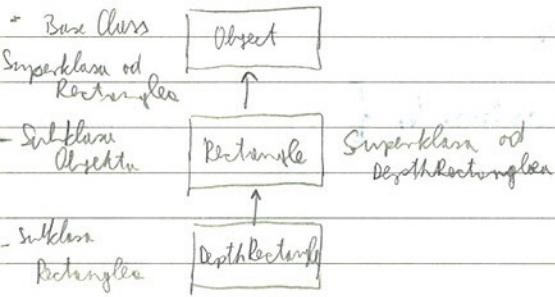
}

- funkcija marge od tko je objet reci, a
 vrise od tko je objet manji. vraca 0 ako su isti

• instanceof operator

myObject instanceof myClass
 ↗ pravoznjava da li myObject je objekat myClass
 - vraca true ili false

* OBJECT INHERITANCE



• 1. DEFINIRANJE KLASE (KONSTRUKTOR)

function DepthRectangle(w, h, d) {

Rectangle.call(this, w, h);

this.depth = d;

}

- subklasa ima iste elemente kao i superklase (w i h) i još dodatno d

→ poziva konstruktor od Rectangle da možemo upotrebjavati sve metode od toga

2. PROTOTIP OBJEKTA SUBKLASE JE SUPERKLASA

DepthRectangle.prototype =
 new Rectangle();

- inicijalizuje se samo subklasa

Objekt base Objekta



3. RE-ASSIGNAT CONSTRUCTOR PROPERTY NA ORIGINALNU KLASU

DepthRectangle.prototype.constructor = DepthRectangle;

4. SAD MOŽEMO DODAVAT METODE

DepthRectangle.prototype.volume = function() {

}

DepthRectangle.prototype.toString() = function() {

}

* EXTENDANJE JS TYPEOVA

- možemo extendati ugrađeni npr. Array

Array.prototype.longestElement = function() {

}

var testArray = new Array(...);

testArray.longestElement();

↓
EXTENDAMO NA
ISTI NACIN KAO
VALASTITE OBJEKTE

Regular expressions

* WRAPPER OBJECTS

IDEJA:

- definiramo objekt i metode
- HTML objekt instancinamo kao definisani objekt
- nad zapisom HTML-om možemo upravljati pomoću definisanih metoda i ostalo...

6. REGULAR EXPRESSIONS

if (nekiString.match(/^ [a-zA-Z0-9]+\$/) != null) { ... }

SLOVA I BROJEVI

POČETAK

KRAJ

- nekiString.match(/^\d{5}\$/)

POČETAK

KRAJ

DIGITS

5x DIGITS

SVE STO JE U ZAGRADU JE
OPRIORITNO I MOže SE
PONOVITI 0 ili 1 PUT

- nekiString.match(/^\d{5}(-\d{4})?\$/)

5x DIGITS

DASH

(ZBOG ??)

* ALTERNATIVES & GROUPING

- za alternative $\Rightarrow |$

- za grupiranje ()

PR:

- "hello" ili "goodbye" /hello|goodbye/

- 4 digits ili 5 characters /\d{4}|\w{5}/

- 5 digits .+ optionalno 4 digits ili "wow" ~~/^(\d{5}(- \d{4})?)? /~~
/^(\d{5}(- \d{4})?)\\$ | wow\\$ /

↳ "wow" na kraju stringa
/^\w+/ → ovde je "hey" mera bit na početku stringa

* SUBEXPRESSIONS

↳ djeleme stvaranje u zagrade pa ne moremo referencirati podje na to

REFERENCA NA
PREVI "IZRAS" U
ZAGRAĐU
/([,])[^,]*\1/
SUF \$TO
NIJE ")" OVO ZNACI BA IZRAS
ISPRED MORE BIT VISE PUTA

* MATCH FLAGS

- case-insensitive search /hello/i \rightarrow "hello", "HELLO", "HeLLo"

- global search /hello/g \rightarrow mi "hello" strings

- global & case-insensitive /hello/gi

* PROCESSING STRINGS

- `string.search()` - trazi pattern i gde je se pojavljuje u stringu
 - ne radi globalni search
 - mica -1 (nema rezultata) i character position
 - ✓ of the match
 - ako nadjeli nesto
- `string.match()` - može izvesti globalni search → array svih nadrijeđenih rezultata
 - svihki u array rezultata
 - SVE matching posao
- `string.replace()`
 - za zamjenu
 - rezultat je cijeli blok tečja izmjena

PR.
`string.replace(regex, replacement)`

→ ovo je Ići TEXT ili
FJA KOJA GENERIRA TEXT ZA
REPLACE

PRIMERI

~~string~~ rezultat = `string.match(nekiRegexp);`

rezultat[0] → ovo je "puna" mijednost između koja je mesta

rezultat[1] → ovo je prva subexpression iz Regexpa

rezultat[2] → druga subexpression iz Regexpa

;

* RegExp OBJEKT

- RegExp()
- inner properties i methods
- RegExp. test () - testira, ali je u posredovanju
- RegExp. exec () - najmočnejši
- RegExp. ignoreCase - vse kar na i. u // nista bri
- RegExp. global - vse kar g u /-/ nista bri
- RegExp. source
- RegExp. lastIndex - pozicija zadnjeg match-a

7. PRAKTIČNI JAVASCRIPT

* DEFINIRANJE OBJEKATA

- var obj = new Object();

je isto kao

- var obj = {};

* ISENJE JE ZA NIZOVE

- var myarray = new Array();

je isto kao

- var myarray = [];

* SINTAKSA DEFINIRANJA PROPERTYA ZA OBJEKTE

- var obj = {};

obj.property = vrijednost ili function () { ... };

↑
EKVIVALENT JEKA JE

- var obj = {

 property : function () { ... };

};

* MJEJENJANJE CSS LAYOUTA S OBIZIROM NA ŠIRINU EKRANA

var clientWidth = document.documentElement.clientWidth; → dobijemo širinu ekrana

8. ALATI ZA DEBUGIRANJE

- YSlow for Firefox
- DevBar for Firefox
- Firebug
- Developer bar for IE