Korisničke funkcije – kreiranje vlastitih funkcija

- Za radnje u programu koje često ponavljamo
- Funkcija je dio koda smješten u posebnu imenovanu cjelinu. Izvršava se iz bilo kojeg dijela programskog koda jednostavnim pozivom-navođenjem tog imena.
- Funkcije ugrađene u JavaScript (alert)
- function pozdrav(){alert('Pozdrav!');}
- //poziva funkciju alert kojoj je proslijeđen parametar u obliku teksta koji želimo ispisati
- <body onload=„pozdrav();">
- Funkcija pozdrav ne prihvaća parametre
- Primjer funkcije koja prihvaća parametre: function zbroji(a,b){…} //kod poziva te funkcije unutar zagrada moramo proslijediti dva parametra, npr. <body onload=„zbroji(2,3);">

Funkcije

- Za razdvajanje dijelova koda koji odrađuju specifični zadatak, odnosno kada postoje složeniji dijelovi koda koji će se koristiti na više mjesta
- Postoje ugrađene funkcije ili metode (write) i korisnički stvorene (vlastite) funkcije
- Definiranje funkcija: općenit dogovor kod imenovanja je da počinje malim slovom, nakon toga je camelCase
- U funkciji su naredbe koje će se izvršiti kod poziva, mogu uključivati jednu ili više naredbi return za povratak u pozivajući kod

```
<script>
function umnozak(a, b)
{
return a*b
}
</script>
```

Polje argumenata

- Član svake funkcije, za određivanje broja i vrste varijabli koje se predaju funkciji
- Polje argumenata (arguments array) daje fleksibilnost rukovanja varijabilnim brojem argumenata:

```
<script>
function displayItems()
{
    for(j=o;j<displayItems.arguments.length;++j)
        document.write(displayItems.arguments[j]+”<br>”)
}
</script>
```

Vraćanje vrijednosti

- Funkcije ispisuju, ali se koriste i za izvođenje izračuna i manipulacija podacima uz odgovarajuće vraćanje rezultata
- Dozvoljeno je ulančavanje metoda, između se stavljaju točke

onerror događaj

- Za „hvatanje" i obradu pogrešaka
- JavaScript može otkriti događaje, svaki element na web stranici ima određene događaje moji mogu „okinuti", odnosno pozvati JavaScript funkcije (npr. onclick na button elementu)
- Primjer:

```
<script>
onerror=errorHandler                //govori error događaju da od tog trenutka koristi novu errorHandler funkciju
document.writ(„Welcome to website")          //namjerna pogreška kako bi ispitali funkciju
function errorHandler(message, url, line)
{
out=„Oprostite, dogodila se pogreska.\n\n";
out+=„Pogreska: „+message+"\n";
out+=„URL: „+url+"\n";
out+=„Linija: „+line+"\n\n";
out+=„Pritisni OK za nastavak. \n\n";
alert(out);
return true;
}
</script>
```

Ključne riječi try-catch

- Standardnije i fleksibilnije od onerror tehnike
- Omogućuju hvatanje pogreška odabranog dijela koda
- Ne hvataju sintaksne pogreške pa je za to potrebno koristiti onerror
- try…catch konstrukt podržan je od svih važnih preglednika
- <script>
- try
- {
- …
- }
- catch(err)
- {
- …
- }
- finally //izvodi se uvijek, neovisno o tome da li se pojavila pogreška u dijelu try
- {
- alert(„….“);
- }
- </script>

Explicit Casting

- Ako je potrebno da vrijednost bude određenog tipa koristi se neka od ugrađenih JavaScript funkcija:
- parseInt()
- Boolean()
- parseFloat()
- String()
- split() //array

JavaScript pogreške

- Rukovanje i prikaz poruka o pogreškama ovisi o pregledniku – potrebno je omogućiti konzolu
- Može se koristiti i Firebug plug-in u Firefoxu i Crome-u

Ugrađeni objekti, svojstva objekata i metode

- JavaScript je OO programski jezik (moguće je kreiranje vlastitih klasa i instanciranje objekata, ali dostupni su i mnogobrojni ugrađeni objekti sa ugrađenim metodama, npr. alert objekta window)
- JavaScript ugrađeni objekti mogu biti:
  - Objekti JavaScript jezika
  - Objekti preglednika
  - DOM objekti

Objekti preglednika

- Nekoliko objekata ugrađeno je u preglednik i daju raznovrsne informacije o pregledniku, postavkama ekrana, povijesti pretraživanja i sl.:
  - window
  - navigator
  - screen
  - history
  - location

- Objekt window opisuje prozor preglednika, daje razne informacije o prozoru preglednika, otvara novi prozor, zatvara prozor, mijenja mu svojstva

Objekt window – metoda confirm

- Pozivom metode confirm objekta window u pregledniku se prikazuje dijaloški okvir za potvrdu. Parametar metode je poruka koja će se ispisati u okviru kad od korisnika želimo dodatnu potvrdu za izvršavanje određene akcije. U okviru za potvrdu korisnik ima gumbe OK i Cancel:

```
var oke=window.confirm('Jeste li sigurni…');
if (oke){alert('Potvrdili ste.');}
else {alert('Odustali ste.');}
```

Objekt window – metoda prompt

- Trebamo li od korisnika zatražiti neki kratki unos preko dijaloškog okvira koji se prikazuje u pregledniku.

```
var ime=window.prompt('Unesite svoje ime: ', 'Ime');
if (ime!=null){alert('Dobar dan '+ime);}
else {alert('Odustali ste.');}
```

- Koriste se dva parametra: poruku i inicijalnu vrijednost polja za unos

Napomene

- Metode alert, confirm i prompt omogućuju u pregledniku prikaz određenog skočnog prozora (pop box)
- Objekt window podrazumijevani je objekt te stoga nije nužno navoditi njegovo ime

Objekt window – metoda open

- Otvara novi prozor preglednika.
- Moguća su četiri parametra:
  - URL (adresa stranice koju otvaramo u novom prozoru ili about:blank kad je prazno)
  - name (određuje atribut target: _target, _self, _parent, _top, odnosno na koji se način novi prozor otvara)
  - specs (lista specifikacija novog prozora, odvajaju se zarezima: fullscreen=yes, height=vrijednost, width=vrijednost, location=no - adresna traka, menubar=1, resizable=0, scrollbars=yes, titlebar=no, toobar=1 i dr.)
  - replace (određuje hoće li adresa prikazana u novom prozoru zamijeniti trenutačni dokument u povijesti pregleda stranica)

Primjer

```
<script type=„text/javascript”>
function otvori(){
window.open(‘http://www.w3.org’,’_blank’,’menubar=0, toolbar=0, resizable=0, width=700, height=400’);
}
</script>

<a href=„#” onclick=„otvori();”>Otvori novi prozor</a>
```

Object window – metoda print

- Pokrećemo dijaloški okvir za ispis stranice (bez parametara)

```
<script type=„text/javascript”>
function ispis(){
window.print();
}
</script>

<a href=„#“ onclick=„ispis();”>Ispisi stranicu</a>
```

Objekt window – metoda setTimeout

- Kad želimo izvršiti neku naredbu (ili funkciju) nakon određenog vremenskog intervala – pokreće ugrađeni timer objekt koji nakon isteka zadanog vremena pokreće zadanu naredbu:

```
<script type=„text/javascript”>
function klik2sec(){
    window.setTimeout(‘alert(„Proslo 2 sekunde”);’,2000);
}
</script>

<input type=„button” value=„2 sekunde” onclick=„klik2sec();”/>
```

- Okvir upozorenja prikazuje se tek nakon 2 sekunde. Prvi argument je naredba, drugi broj milisekundi nakon kojih će se naredba izvršiti.

Objekt window – metoda clearTimeout

- Za opoziv metode setTimeout prije isteka zadanog intervala:

```
<script type=„text/javascript”>

var timer;

function startTimer(){
        timer=window.setTimeout(‘alert(„Proslo 2 sekunde”);’,2000); //varijabli timer dodijeljen poziv metode setTimeout i sačuvana referenca do timer objekta
}

function stopTimer(){
        window.clearTimeout(timer);//referencu prosljeđujemo u metodu clearTimeout i poništavamo timer-objekt
        alert(‘Timer prekinut.’);
}
</script>

<input type=„button” value=„Pokreni timer” onclick=„startTimer();”/>
<input type=„button” value=„Ponisti timer” onclick=„stopTimer();”/>
```

Rekurzivno pozivanje metode setTimeout

- Funkcija navedena kao naredba u metodi setTimeout može sadržavati isti setTimeout poziv te tako zapravo dobijamo odgođene ugniježđene pozive jedne te iste funkcije (rekurziju)

```
<script type=„text/javascript”>

function tick(){
        var vrijeme=new Date();
        window.status=vrijeme.toLocaleTimeString();//dohvaća se i ispisuje trenutačno vrijeme na statusnoj traci preglednika – implementiran digitalni sat unutar statusne trake preglednika
        var timer=setTimeout(‘tick();’, 1000);
}
</script>
<body onload=„tick()”></body>
```

Ako nam statusna traka nije dostupna, ispis vremena se može izvršiti u sadržaj bilo kojeg HTML elementa

Objekt window - metode setInterval i clearInterval

- Umjesto rekurzivnog pozivanja iste funkcije u metodi setTimeout, može se koristiti metoda setInterval objekta window
- Nakon poziva metode setInterval js funkcija ili izraz naveden kao prvi parametar u pozivu stalno se izvršava nakon isteka vremenskog intervala određenog drugim parametrom (u ms)
- Metoda setInterval stalno će nastaviti pozivati zadanu funkciju do poziva metode clearInterval ili do zatvaranja prozora preglednika

Primjer

```
<script language=javascript>
var int = window.setInterval(„sat()”, 1000);//pozivom metode setInterval zadano je izvršavanje funkcije sat svake sekunde, varijabli int dodijeljen je identifikator pripadajućeg timer objekta
function sat(){
        var vrijeme=new Date();
        window.status=vrijeme.toLocaleTimeString();//u statusnoj traci preglednika ispisuje trenutačno vrijeme
}
</script>
<body>
        <button onclick=„int=window.clearInterval(int)”>Stop</button>
</body>
```

Klikom na gumb Stop poziva se funkcija clearInterval s proslijeđenim parametrom int koji zaustavlja pripadajući timer-objekt, nakon čega se vrijeme u statusnoj traci prestaje ispisivati.

Objekt navigator

- Sadrži općenite (meta) informacije o pregledniku: ime, verziju i oznaku preglednika, informacije o mogućnosti postavljanja kolačića, upotrebi Jave i dr.
- Svojstva objekta navigator: appCodeName (kodno ime preglednika), appName (ime preglednika), appVersion (informacije o verziji preglednika), cookieEnabled (informacije o mogućnosti upotrebe kolačića), platform (informacije o platformi na kojoj se izvodi preglednik), userAgent (zaglavlje koje preglednik šalje poslužitelju)
- Sadrži metodu javaEnabled koja provjerava omogućava li preglednik upotrebu (izvođenje) Java programa

Primjer upotrebe

<script type=„text/javascript”>

alert(‘Kodno ime preglednika: ‘+navigator.appCodeName+’\r\nIme preglednika: ‘+navigator.appName+’\r\nVerzija preglednika: ‘+navigator.appVersion+’\r\nPlatforma: ‘+navigator.platform);

</script>

- Poznavajući instrukcije za kontrolu toka programa i objekt navigator moguće je provjeriti koji preglednik upotrebljava posjetitelj stranice te potom izvršiti programski kod namijenjen tom pregledniku.

if (navigator.appName==‘Microsoft Internet Explorer’){…}

else {…}

Objekt screen

- Sadrži informacije o korisničkom ekranu: visinu i širinu ekrana bez programske trake operacijskog sustava (availHeight, availWidth), broj bitova za opis boje na ekranu (colorDepth), ukupna visina i širina ekrana (height i width), rezolucija boje ekrana (bitova po pikselu, pixelDepth)

```
<script type=„text/javascript”>
alert(‘Sirina ekrana: ‘+screen.width+’\r\nVisina ekrana: ‘+screen.height);
</script>
```

Objekt history

- Sadrži URL dokumenata otvaranih unutar preglednika: broj URL-ova spremljenih u history listi preglednika (length), vraćanje na prethodno otvoreni URL (back), odlazak na slijedeći URL u history listi (forward), učitavanje određenog URL-a iz history liste (go)
- Dio objekta window te je preko njega dostupan kao njegovo svojstvo (window.history)

Vrati se na prethodnu stranicu

Vrati se za dvije stranice unatrag

Objekt location

- Sadrži informacije o trenutačno učitanom dokumentu, odnosno njegovom URL-u: ime domene, putanja na poslužitelju, port preko kojeg poslužitelj šalje URL...
- Dio objekta window (window.location)
- Metode i svojstva: hash (dohvaća dio URL-a nakon znaka #, tj. vraća sidro iz URL-a), host (dohvaća ime hosta, odnosno domene koja je sadržana u URL-u i port poslužitelja), hostname (dohvaća ime hosta, odnosno domene koja je sadržana u URL-u), href (vraća cijeli URL), pathname (vraća lokaciju dokumenta na poslužitelju), port (vraća broj porta koji upotrebljava poslužitelj), protocol (dohvaća vrstu protokola iz URL-a), search (dohvaća dio URL-a nakon upitnika), assign() (učitava novi dokument), reload() (ponovno učitava trenutačni dokument), replace() (zamjenjuje trenutačni dokument novoučitanim dokumentom)
- Ponovno učitaj dokument

Objekti links i history

- Polje URL-ova: url=document.links[0].href
- Broj linkova u dokumentu: brojlinkova=document.links.length
- Ispis svih linkova: for(j=0;j<document.links.length;++j) document.write(document.links[j].href+'<br>')

- Ispis povijesti preko objekta history: document.write(history.length)
- Objekt history u polju čuva broj web sjedišta
- Može se zamijeniti trenutna stranica sa nekom iz povijesti (ako je poznata pozicija), preko metode go objekta history: history.go(-3)
- Postoje i metode history.back() i history.forward() ili se može trenutni url zamijeniti sa odabranim: document.location.href='http://google.com'

Primjer:

```
<html>
        <head>
        <title> Link test </title>
        </head>
        <body>
                <a id=„mylink” href=„http://mysite.com”>Click me</a><br>
                <script>
                url=document.links.mylink.href      ili url=mylink.href *
                document.write (‘URL je ‘+url)
                </script>
        </body>
</html>
* Ili url=document.getElementById(‘mylink’).href
```

JavaScript objekti

- Ugrađeni u programski jezik
- Olakšavaju rad s raznim tipovima podataka
- Array – polje, sprema više vrijednosti u jednu varijablu (objekt), različitim vrijednostima pristupa se preko indeksa (rednih brojeva lokacija)
- Date – rad s datumom i vremenom
- Math – vrijednosti nekih matematičkih konstanti (pi, prirodni logaritam i sl.), metode za potenciranje, korjenovanje i sl.
- String – manipulacija tekstualnim podacima

Objekt Array

- Deklaracija – pozivom konstruktora Array(): var auti=new Array();
- Elemente se može navesti i u pozivu konstruktora ili pri deklaraciji varijable inicijalizacijom: var auti=[‘audi’,’bmw’,…];
- Pristup elementima: alert(auti[1]);
- Dohvaćanje elemenata polja pomoću petlje for..in

```
<html>
        <body>
            <script type=„text/javascript”>
                var auti=[‘audi’,’bmw’,’opel’];
                for(var a in auti){ document.write(auti[a]+’<br/>’}
            </script>
        </body>
</html>
```

Metode objekta Array (najčešće)

- length – svojstvo koje sadrži broj elemenata polja, npr. auti.length
- sort – metoda koja abecedno sortira elemente polja, npr. auti.sort()
- concat – metoda koja spaja više polja u jedno, npr. var sviauti=auti.concat(auti2); alert(sviauti);
- join – metoda koja spaja elemente polja u podatak tipa string u kojem će biti odvojeni nekim separatorom, npr. alert(auti.join('/'));

Objekt Date

- Za rad s vremenom i datumima na klijentskoj strani
- Deklaracija – pomoću konstruktora Date(), npr. var vrijeme=new Date() //trenutačni datum i vrijeme sistemskog sata klijentskog računala
- Nova varijabla može se stvoriti i sa new Date(0);//u varijabli vrijeme dobiva se vrijednost za 01.01.1970. – brojčani parametar proslijeđen u konstruktor objekta tipa Date predstavlja broj ms
- Konstruktor može primiti i tekstualni parametar: new Date('July 15, 2001 12:00:00'); a moguće je i proslijediti više cjelobrojnih parametara: godinu, mjesec, dan, sat, minute i sekunde, npr. new Date(2011, 7, 15, 11, 30, 0);//vrijeme se može i izostaviti, mjeseci se broje od 0

Ugrađene metode varijable tipa Date

- alert(vrijeme);//neformatirani oblik ispisa datuma i vremena ili alert(vrijeme.toLocaleString());
- getDate() - vraća dan u mjesecu
- getDay() – vraća brojčani oblik dana u tjednu (0 za nedjelju)
- getFullYear(), getHours(), getMinutes(), getSeconds(), getMonth()
- getTime() – vraća broj milisekundi od 1.1.1970.
- toLocaleString() – vraća datum i vrijeme prema regionalnim postavkama klijentskog računala
- toLocaleDateString() – vraća datum prema regionalnim postavkama
- toLocaleTimeString() - vraća vrijeme prema regionalnim postavkama

Metode za postavljanje komponenti objekta tipa Date

- setDate(dan) – postavlja dan u pozivajućem objektu tipa Date
- setMonth(mjesec) – postavlja mjesec
- setFullYear(godina) – postavlja godinu
- setHours(sati) – postavlja sate
- setMinutes(minute) – postavlja minute
- setSeconds(sekunde) - postavlja sekunde
- setMilliSeconds(milisekunde) – postavlja milisekunde

Objekt Math

- Sadrži svojstva i metode za rad s naprednim matematičkim operacijama i izrazima
- Ne sadrži konstruktor, svim metodama i svojstvima pristupa se izravno pozivom objekta Math
- var broj_PI=Math.PI;
- Svojstva objekta Math sadrže vrijednosti matematičkih konstanti: E (Eulerov broj, 2.718), LN2, LN10, LOG2E, LOG10E, PI, SQRT2
- Metode objekta Math omogućuju izvođenje nekih složenijih matematičkih operacija: abs(broj), cos(broj), tan(broj), sin(broj), round(broj) – najbliži cijeli broj, ceil(broj) – najbliži veći cijeli broj, floor(broj) – najbliži manji cijeli broj, log(broj), max(broj1, broj2, ...), min(broj1, broj2,...), pow(broj1, broj2), random(), sqrt(broj)...

Primjer

```
<script type=„text/javascript”>
        var broj1=prompt(‘Unesi prvi broj: ‘);
        var broj2=prompt(‘Unesi drugi broj: ‘);
        alert(‘Rezultat potenciranja je ‘+Math.pow(broj1, broj2));
</script>
```

- Metoda random vraća slučajno odabrani decimalni broj iz intervala [0,1]: Math.random();
- Ako trebamo broj iz intervala cijelih brojeva, npr. slučajno odabrana vrijednost komponente boje treba biti iz cjelobrojnog intervala [0,255]: alert(Math.floor(Math.random()*255));

Objekt String

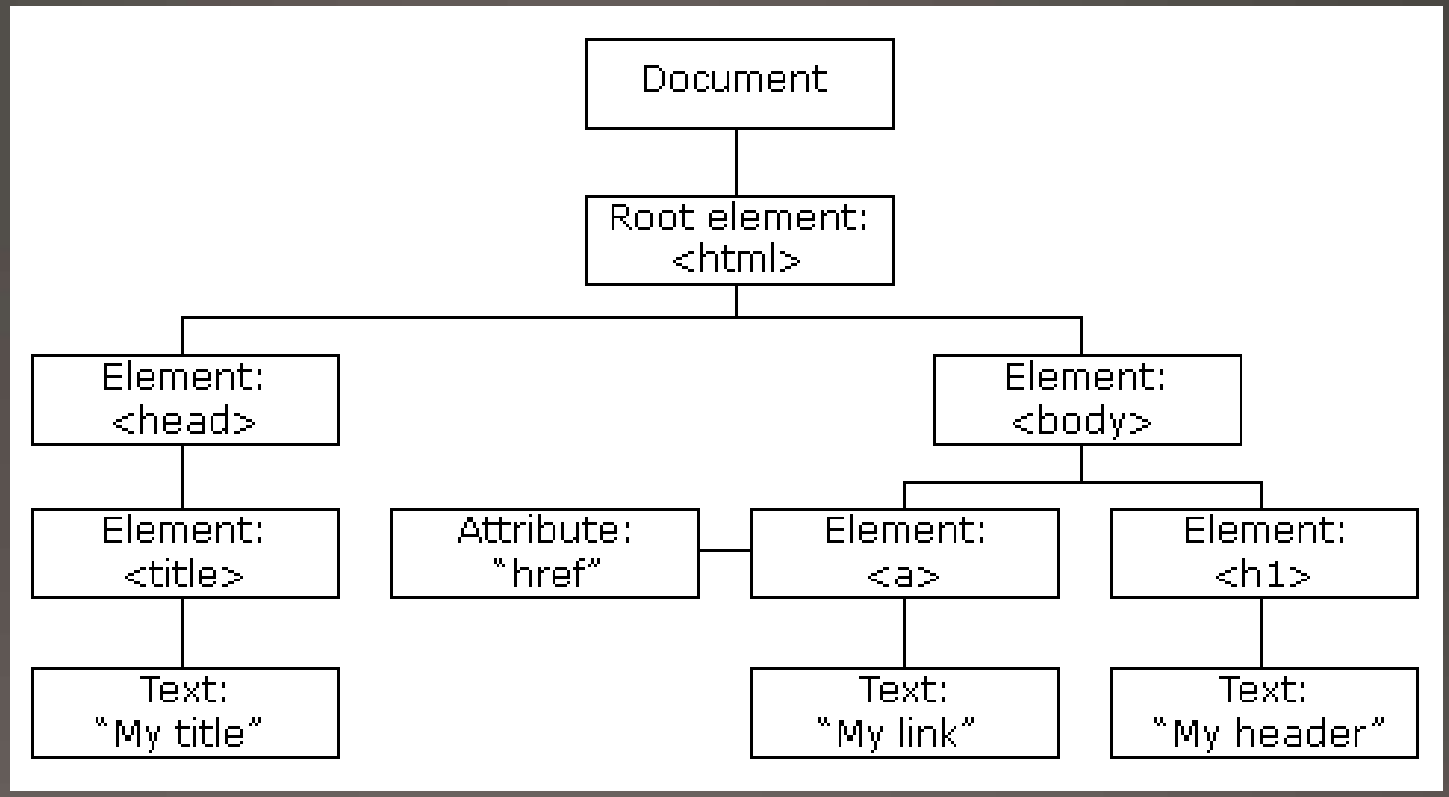
- Tekstualni tip podatka
- Deklaracija se obavlja pomoću konstruktora: var tekst=new String('neki tekst'); ili var tekst='neki tekst';
- Najčešće upotrebljavana svojstva i metode: tekst.length - vraća broj znakova, tekst.charAt(n) – znak koji se nalazi na poziciji n, tekst.indexOf('i') – indeks na kojem se pojavljuje znak, tekst.lastIndexOf('t') - indeks posljednjeg pojavljivanja znaka, tekst.toLowerCase() - postavlja sve znakove u mala slova, tekst.toUpperCase() - postavlja sve znakove u velika slova, tekst.concat(s1, s2) – spaja redom u novi string, tekst.slice(poc, kraj) – vraća podniz znakova, tekst.substring(poc, kraj) – slično osim što za jedan negativan argument vraća cijeli string, tekst.substr(poc, dulj) – vraća podniz od dulj znakova, tekst.replace('n', 'N') – mijenja podniz, tekst.split(sep) – dijeli string i vraća polje elemenata…

DOM (Document Object Model) i objekt document

- Prilikom učitavanja HTML dokumenta u preglednik, taj dokument postaje objekt tipa document kojem je bitna karakteristika njegov HIJERARHIJSKI USTROJ u kojem je svaki HTML element u nekakvom „rodbinskom" odnosu s ostalim elementima (obiteljsko stablo)
- Na vrhu hijerarhije je objekt document koji ima jedan element dijete (korijenski-root element html). Taj element sadrži elemente head i body koji opet sadržavaju svoje elemente djecu itd.
- Ovako hijerarhijski ustrojen model naziva se DOM

Primjer

```
<!DOCTYPE html>
<html>
<head>
    <title>Naslov dokumenta</title>
</head>
<body>
<h1>Moj prvi naslov</h1>
<a href=„”>Poveznica</a>
<p>Odlomak</p>
</body>
</html>
```



Objekti

- U DOM modelu definiranom unutar objekta document svi elementi HTML dokumenta predstavljeni su kao objekti kojima možemo manipulirati metodama te čijim svojstvima možemo pristupiti programskim kodom, npr. možemo pristupiti sadržaju odlomka te taj sadržaj spremiti u varijable kako bi njima kasnije mogli manipulirati

JavaScript i DOM

- JavaScript građen oko DOM-a i razbija dijelove HTML dokumenta na diskretne objekte, svaki sa svojim vlastitim svojstvima i metodama
- JavaScript dijeli objekte, svojstva i metode upotrebom točke
- JavaScript uključuje hijerarhiju objekata (DOM)
- Npr. url je dio body sekcije HTML dokumenta i referencira se kao url=document.links.linkname.href

Metode objekta document

- Za manipuliranje HTML dokumentom i HTML elementima:
- getElementById(id) – dohvaća (prvi, jedini) element u strukturi HTML dokumenta s identifikatorom id.
- getElementsByName(ime) – dohvaća sve elemente u HTML strukturi čija je vrijednost atributa name jednaka ime
- getElementsByTagName(tip) – dohvaća sve elemente nekog tipa (npr. sve odlomke ili sve naslove h1)
- write() – ispisuje HTML ili JavaScript izraz u strukturi HTML dokumenta (proslijeđeni parametar)
- writeln() – ispisuje HTML ili JavaScript izraz u strukturi HTML dokumenta sa prebacivanjem u novi red

Svojstva objekta document

- title – vraća ili postavlja naslov dokumenta
- length – vraća broj elemenata na formi
- forms – vraća kolekciju (polje) svih objekata Form u dokumentu (prilikom učitavanja za svaki element form u HTML dokumentu kreira se pripadajući objekt Form. Najbitnije svojstvo objekta Form je svojstvo elements i predstavlja polje elemenata za unos (input) sadržanih u toj formi.)
- Svaki dohvaćeni HTML element dio je hijerarhije DOM-a pripadajućeg HTML dokumenta. Pomoću svojstva childNodes možemo izravno dohvatiti njegove element djecu, pomoću svojstva parentNode njegovog elementa roditelja, pomoću svojstva nextSibling njegovog slijedećeg elementa rođaka (slijedeći element dijete njegovog elementa roditelja) i drgue..

Svojstva i metode

- Dohvaćene HTML elemente mijenjamo mijenjanjem njihovih svojstava i metoda:
- className – dohvaća i postavlja ime klase nekog HTML elementa
- height – dohvaća i postavlja vrijednost visine nekog HTML elementa
- id – dohvaća i postavlja vrijednost identifikatora HTML elementa
- innerHTML – dohvaća i postavlja sadržaj nekog HTML elementa
- style – dohvaća i postavlja vrijednost atributa style nekog HTML elementa
- width – dohvaća i postavlja vrijednost širine dohvaćenog HTML elementa
- Objekt document posjeduje svojstva specifična za svaki HTML element. Ime svojstva jednako je atributu nekog HTML elementa. Poznavajući objekt DOM-a možemo manipulirati svim elementima HTML-a. Za različite interakcije DOM se upotrebljava u kombinaciji sa različitim događajima.

Dohvaćanje elementa pomoću metode getElementByID

- Određeni element u HTML dokumentu možemo dohvatiti preko vrijednosti atributa id koji prosljeđujemo u poziv metode getElementById objekta document:

```
<html>
<head>
    <title>Naslov dokumenta</title>
</head>
<body>
<p>Moj prvi odlomak</p>
<p id=„p2”>Drugi odlomak</p>
<p>Odlomak</p>
</body>
</html>
```

- Drugi odlomak dohvaćamo sa document.getElementById(‘p2’);

Svojstvo innerHTML

- Omogućuje dohvaćanje i postavljanje unutarnjeg HTML sadržaja zadanog elementa: var sadrzaj=document.getElementById('p2').innerHTML;
- Novi tekst u drugi odlomak možemo postaviti sa: document.getElementById('p2').innerHTML='Novi tekst.');
- Vrijednost svojstva innerHTML može sadržavati i HTML kod.

- *Zadatak: Od korisnika zatražiti unos imena u tekstualno polje te zatim klikom na gumb Obradi ispisati pozdravnu poruku. Napraviti funkciju obradi() koja iz polja ime u odlomak poruka ispisuje pozdravnu poruku. Unijeto ime ispisati podebljano.*

Svojstvo style

- *Zadatak: Na stranici sa tri odlomka, izmjeniti svojstva širine i pozadinske boje elementa odlomka s id-om prvi nakon pritiska na gumb Promjeni svojstva odlomka.*

- Svojstva CSS-a koja mijenjamo JavaScriptom pozivaju se imenima bez crtice (operator oduzimanja), odnosno pišu se spojeno s tim da se druga riječ piše velikim početnim slovom.

Metoda getElementsByTagName

- Kad trebamo u dokumentu postaviti ista svojstva svim elementima istog tipa:

- var elementi_p=document.getElementsByTagName('p');
- for(i=0;i<elementi_p.length;i++){elementi_p[i].style…}

- zadatak

Objekt Form

- Pristup pojedinim elementima forme

<form name=„forma”>...</form>

```
function obradi(){
var f=document.forms[0];
var ispis='';
for(var i=0;i<f.length;i++)
    ispis+=f.elements[i].value+'\n';
alert(ispis);
}
```

Dohvaćanje članova polja forms i elements preko imena

- Članove polja forms i elements kontrolira se preko imena pa se umjesto tekstualnih indeksa u uglatoj zagradi navodi sadržaj atributa name određenog elementa:

```
function obradi(){
        var f=document.forms['forma'];
        alert(f.elements['txtIme'].value);
}
```

Registriranje učitavanja stranice, klika na link ili gumb… Zahtjeva odgovaranje na događaj
Ostvaruje se interakcija, jedna od ključnih uloga JavaScripta

Događaji

Događaji elemenata body i frameset

- onload (učitavanje svih elemenata koje element sadržava)
- onunload (izvršava programski kod nakon što se stranica zatvori), onemogućen u nekim suvremenim preglednicima

Događaji elemenata forme

- Elementi forme mogu registrirati nekoliko posebnih događaja s obzirom na stanje u kojem se nalaze:

onblur	Događaj koji nastaje trenutkom gubljenja fokusa nekog elementa i prebacivanjem fokusa na drugi element forme
onchange	Događaj promjene vrijednosti elementa forme (mijenjanje unosa, odabir stavke…)
onfocus	Suprotan događaju onblur, fokusiranje elementa
onreset	Izvršava programski kod nakon resetiranja forme
onselect	Izvršava programski kod nakon selektiranja elementa
onsubmit	Izvršava programski kod nakon slanja podataka iz forme

Primjer

```
<html>
<body>
<script type="text/javascript">
function upali(e)
{
e.style.backgroundColor='Yellow';
}
function ugasi(e)
{
e.style.backgroundColor='';
}
</script>
<form name="forma">
<p>Ime: <input type="text" name="txtIme" onfocus='upali(this)'
onblur='ugasi(this)' /></p>
<p>Prezime: <input type="text" name="txtPrezime" onfocus='upali(this)'
onblur='ugasi(this)' /></p>
</form>
</body>
</html>
```

Događaji elementa img

- Element img prikazuje grafičku datoteku s adrese navedene kao sadržaj atributa src
- U slučaju pogreške prilikom učitavanja grafike registrira se događaj onabort

Događaji s obzirom na način nastajanja

- Prilikom upotrebe tipkovnice ili miša

Događaji tipkovnice

- Nastaju pritiskom tipke na tipkovnici
- onkeydown - pritiskom
- onkeypress - držanjem
- onkeyup - otpuštanjem

Primjer

- Forma za unos telefonskog broja

```
<html>
<body onLoad="document.forms['f'].elements['t1'].focus();">

<form name="f">
Telefon:(<input type="text" name="t1" size="3" maxlength="3" onkeyup="provjeriDuljinu(this,'t2');„ />)
<input type="text" name="t2" size="3" maxlength="3" onkeyup="provjeriDuljinu(this,'t3');" /> -
<input type="text" name="t3" size="3" maxlength="4" />
<input type="button" value="ok">
</form>

<script type="text/javascript">

function provjeriDuljinu(x, nextName)
{
if(x.value.length>=x.maxLength){x.form.elements[nextName].focus();}
}
</script>

</body>
</html>
```

Događaji miša

- Nastaju pomicanjem pokazivača miša nad nekim HTML elementom

onclick	Nastaje klikom
ondblclick	Nastaje dvostrukim klikom
onmousedown	Nastaje pritiskom
onmouseup	Nastaje otpuštanjem
onmousemove	Nastaje pomicanjem pokazivača
onmouseout	Nastaje pomicanjem pokazivača izvan elementa
onmouseover	Nastaje pomicanjem pokazivača iznad elementa

Primjeri

```
Datum: <input type=„text” name=„txtDatum” ondblclick=‘this.value=new Date().toLocaleDateString()’ />

<ul>
<li onmouseover=„oznaciElement(this, true)” onmouseout=„oznaciElement(this, false)”>Zagreb</li>
<li onmouseover=„oznaciElement(this, true)” onmouseout=„oznaciElement(this, false)”>Split</li>
<li onmouseover=„oznaciElement(this, true)” onmouseout=„oznaciElement(this, false)”>Rijeka</li>
<li onmouseover=„oznaciElement(this, true)” onmouseout=„oznaciElement(this, false)”>Osijek</li>
<li onmouseover=„oznaciElement(this, true)” onmouseout=„oznaciElement(this, false)”>Varaždin</li>
</ul>

function oznaciElement(el, on){
if(on)
        el.style.backgroundColor=‘Orange’;
else
        el.style.backgroundColor=‘’;
}
```

Svojstva objekta event

- Registriranjem događaja stvara se novi objekt event preko čijih je svojstava moguće dohvatiti druge korisne informacije o tom događaju (parametre miša i tipkovnice)

altKey	Vraća da li je bila pritisnuta tipka ALT
button	Vraća koja je tipka miša bila kliknuta
clientX	Vraća vodoravnu koordinatu pokazivača miša u odnosu na prozor preglednika
clientY	Vraća okomitu koordinatu pokazivača miša u odnosu na prozor preglednika
ctrlKey	Vraća da li je bila pritisnuta tipka CTRL
screenX	Vraća vodoravnu koordinatu pokazivača miša u odnosu na cijeli ekran
screenY	Vraća okomitu koordinatu pokazivača miša u odnosu na cijeli ekran
shiftKey	Vraća da li je bila pritisnuta tipka SHIFT

Primjer

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

function dajSvojstva(e)
{
var str='Svojstva objekta event: \n';
str+='altKey: '+e.altKey+'\n';
str+='button: '+e.button+'\n';
str+='clientX: '+e.clientX+'\n';
str+='clientY: '+e.clientY+'\n';
str+='ctrlKey: '+e.ctrlKey+'\n';
str+='screenX: '+e.screenX+'\n';
str+='screenY: '+e.screenY+'\n';
str+='shiftKey: '+e.shiftKey+'\n';
alert(str);
}
</script>
```

```
<body onmousedown='dajSvojstva(event)'></body>
```

Validacija unosa pomoću JavaScripta

- Provjera unosa podataka unutar elemenata forme kako neželjene vrijednosti ne bi završile na strani poslužitelja, odnosno u bazi

Primjer – forma za registraciju novog korisnika

```
<html>
<body>
<script type="text/javascript">
function provjera()
{
if (document.getElementById('korisnicko_ime').value=='')
                {alert('Niste unijeli korisnicko ime!');location.reload();}
elseif(document.getElementById('lozinka').value='')
                {alert('Niste unijeli lozinku!');location.reload();}
}
</script>

<form id=„form1"name=„form1" method=„post" action=„posalji.php" onsubmit=„provjera();"> //na događaj onsubmit poziva se funkcija provjera()
<p>Korisničko ime: <br /><input type="text" name=„korisnicko_ime„ id=„korisnicko_ime" /></p>
<p>Lozinka: <br /> <input type=„password" name=„lozinka" id=„lozinka"  /></p>
<p><input type=„submit" name=„registracija" id=„registracija" value=„Registriraj se!" /></p> //klikom na gumb šaljemo formu na poslužitelj
</form>

</body>
</html>
```