DOM – DOCUMENT OBJECT MODEL

JavaScript je programski jezik koji dolazi do izražaja kada treba da se doda živost statičkim veb stranicama. Programiranjem različitih interakcija sa korisnikom, dodavanjem animacija pojedinim elementima strane, moguće je kreirati popularne dinamičke veb sajtove kao i veb aplikacije.

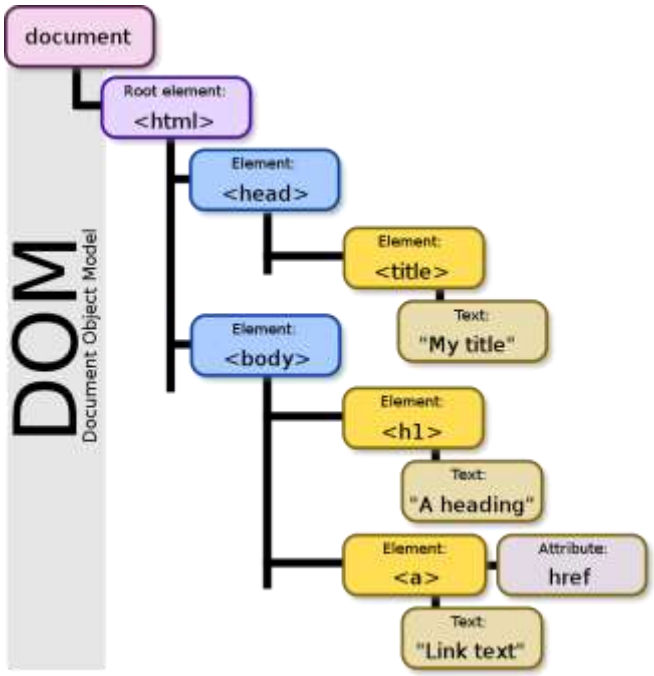
JavaScript kod u kombinaciji sa **DOM** –om omogućava to povezivanje u jednu celinu.

Dokument Object Model predstavlja struktuiran sadržaj na veb strani gde se svaki pojedini element predstavlja objekat. **DOM** je poptuno objektno orijentisan. Omogućava povezivanje elemenata na strani sa programskim jezikom. Odnosi se i na HTML fajlove i na XML fajlove.

DOM predstavlja objektno orijentisanu prezentaciju veb strane, kojoj se može pristupiti i koja može biti modifikovana korišćenjem *JavaScript* jezika.

Kroz JavaScript je moguće pristupiti DOM-u i kreirati različite događaje, menjati sadržaj na strani kao i izgled elemenata. DOM je kreiran tako da je nezavisan od programskog jezika koji mu pristupa.

DOM se može predstaviti kao stablo elemenata koji se nalaze na strani. Može se reći da je sastavljeno iz čvorova (*nodes*) koje predstavljaju elementi strane. Svaki ugnežden tag unutar nekog drugog taga je *child* datog elementa.



Događaji (Events)

Događaji na veb strani predstavljaju situacije koje su posledica delovanja korisnika ili samog sustema. Na primer: klik mišem na neki element, učitavanje strane, prelazak mišem preko nekog elementa, promena dimenzije nekog objekta, pritisak tastera na tastaturi... Programeri prilikom izrade veb sajtova i aplikacija programiraju reakcije na te događaje.

U ovom odeljku se nalaze nabrojani događaji (*on event handlers*) sa kojima ćete se češće sretati prilikom realizacije postavljenih zadataka . O*n event handlers* predstavljaju svojstva DOM objekata i koriste se za inline registrovanje događaja. Više o događajima na sledećem linku: https://developer.mozilla.org/en-US/docs/Web/Events

HTML atribut (on + Događaj)	Događaj se desi ...
`onclick`	Kada korisnik klikne na element
`ondblclick`	Kada korisnik dvoklikne na element
`onmousedown`	Kada korisnik pritisne dugme na mišu dok se nalazi preko elementa
`onmouseup`	Kada korisnik otpusti dugme na mišu dok se nalazi preko elementa
`onmousemove`	Kada se pointer miša pomera dok se nalazi preko elementa
`onmouseover`	Kada pointer miša uđe u granice elementa ili u granice njegovih child elementa
`onmouseout`	Kada pointer miša izađe van granica elementa ili van granica njegovih child elementa
`onkeydown`	Kada korisnik pritisne taster
`onkeypress`	Kada korisnik pritisne i otpusti taster
`onkeyup`	Kada korisnik otpusti taster
`onresize`	Kada se promeni veličina (view) dokumenta
`onscroll`	Kada se skroluje scroll bar na elementu
`onload`	Kada se objekat učita
`onsubmit`	Kada se forma pošalje
`onfocus`	Kada je elemenat u fokusu

onblur	Kada element gubi fokus
onreset	Kada se forma resetuje
onchange	Kada se sadržaj elementa forme menja, promena opcije u select dropdown listi ili promena stanja radio dugmeta i sl.

Registrovanje događaja

Preko HTML atributa – Inline Event Hendler

Ovaj način je jednostavan, ali polako izlazi iz upotrebe. Razlog tome je:

- Nisu odvojeni HTML kod i *JavaScript* kod.
- Ne može se izvršiti više funkcija (*event hendlers*) kao reakcija na jedan događaj
- Pokazivač ***this*** se odnosi na globalni objekat (*window*)
- Ukoliko se desi da je potrebno da se neka funkcija pozove na određeni događaj kod više različitih HTML elemnata, mora se ponavljati pisanje koda (pozivanje funkcija) kod svakog HTML elementa. Na ovaj način se razvija kod koji je teži za održavanje i kosi se sa DRY principom

Primer:

```
<p onclick="ispisiPoruku()"> Klikom na paragraf pozivate funkciju koja
generiše alert</p>
<script>
    function ispisiPoruku () {
        alert('Pozdrav');
    }
</script>
```

Tradicionalno registrovanje događaja

Ovaj način je bolji za korišćenje od *Inline Event Handler*-a jer:

- razdvaja HTML kod od JS koda, što je bitno zboog održavanja koda
- ***This*** pokazivač ukazuje na targetirani objekat

Mana:

- Ne može se izvršiti više funkcija (*event hendlers*) kao reakcija na jedan događaj kao ni kod *Inline Event Handler - a*

Primer:

```
let element = document.getElementById('box'); // targetira se element nad kojiM će se registrovati događaj

            element.onclick = function () { /* telo funkcije koja će se izvršiti kada se klikne na targetirani element */ };
```

Dodavanje Event Listener-a prilikom registrovanje događaja

Ovaj način je trenutno najprihvaćeniji i predstavlja naprednu tehniku registrovanja događaja o čemu će kasnije biti reči u ovom nastavnom materijalu kada se odrade osnove registrovanja događaja.

Prozori i događaji

Alert

Upotreba: Alert dijalog prozori se najčešće koriste za obaveštavanje korisnika. Recimo ukoliko se desi da korisnik popunjava formu i ne popuni neko input polje koje je bitno za validaciju, može se iskoristiti alert prozor kojim će se korisnik obavestiti da je polje obavezno i da treba da bude popunjeno.

Sintaksa: `window.alert(message);`

- *message* je opcioni string koji će biti prikazan u dijalogu

Confirm

Upotreba: Confirm dijalog se koristi kada je potrebno da korisnik nešto potvrdi. Prozor sadrži dva dugmeta: **OK** i **Cancel**.

Sintaksa: `result = window.confirm(message);`

Ako korisnik klikne na **OK** dugme **confirm()** metod će da vrati *true*. Ako korisnik klikne na **Cancel confirm()** vraća *false*.

```
result = window.confirm(message);
```

- *message* je opcioni string koji će biti prikazan u dijalogu
- *result* confirm metode je boolean. Ako je selektovano u prozoru OK (true) ili Cancel (*false*).

Prompt

Upotreba: Koristi se kada je potrebno da korisnik otkuca neku informaciju. Ima u sebi jedno input polje i dva dugmeta OK i CANCEL

Sintaksa: *result* = window.prompt(*message*, *default*);

- *message* je opcioni parameter i predstavlja poruku koja će se ispisati korisniku. Ukoliko se navede otvoriće se prazan prompt prozor.
- *default* je takođe opcioni parameter i predstavlja vrednost koja će biti u text box polju kada se prozor kreira.
- *result* je povratna vrednost koju je korisnik otkucao ili *null* ukoliko korisnik nije ništa uneo (Cancel)

Primeri:

1. Ispisivanje poruke kada se strana učita

```
<body onload="window.alert('Web strana je učitana!');">
```

2. Generisanje ALERT prozora klikom na dugme na veb strani

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Upotreba alert prozora</title>
   </head>
   <body>
      <p>Klikom na dugme pokrenite funkciju koja će da da generiše alert prozor</p>
      <form>
         <input type="button" value="Pokreni alert" onclick="alertMsg();">
      </form>
      <script>
        function alertMsg() {
            alert ('Krećemo sa zanimljivim gradivom!');
            document.write ('Ovo je alert prozor koji se koristi kada korisnika hoćemo da obavestimo porukom');
        }
      </script>
   </body>
```

```
</html>
```

3. Upotreba CONFIRM prozora

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Upotreba confirm prozora</title>
    </head>
    <body>
        <p onclick="potvrda();">Informator ETŠ "Mihajlo Pupin" </p>
         <script>
            function potvrda() {
                if (confirm('Upišite našu školu!')) {
                    window.open('http://www.etspupin.edu.rs');
                }}
         </script>
    </body>
</html>
```

4. PROMPT dijalog

```
<!DOCTYPE html>
<html>
    <head>
        <title>Prompt dijalog</title>
    </head>
    <body>
        <p>Kliknite na dugme da biste isprobali kako se upotrebljava PROMPT
dijalog prozor</p>
        <button onclick="otvori_prompt()">Otvori PROMPT dijalog</button>
        <p id="ispis"></p>
        <script>
            function otvori_prompt() {
                let ime = prompt('Unesite vaše ime', ' Ime ');
                if (ime != null) {
                    document.getElementById('ispis').innerHTML ='Pozdrav ' +
ime + '! Hvala na poseti!';
                }
            }
        </script>
    </body>
</html>
```

5. Metoda open()

Metoda open kada je potrebno otvoriti prozor za prosleđen url

Sintaksa: `let window = window.open(url, windowName, [windowFeatures]);`

- *url* – adresa resursa na vebu koja će se otvoriti u novom prozoru. Ukoliko se ne navede (prazan string “”) otvoriće se prazna strana
- *windowName* - naziv prozora, *iframe*- a ili tab-a u okviru koga će se prozor otvoriti. Ukoliko se ne navede dati url će se otvoriti u novom prozoru. Ukoliko se navede naziv koji ne postoji u datom kontekstu kreira se novi prozor sa tim imenom. Umesto name-a takođe se mogu navesti sledeće solucije: **_blank, _parent, _self, _top**
- *windowFeatures* – omogućava definisanje nekih specifičnosti novog prozora, kao što su dimenzija, skrolovanje i slično. Navode se svojstvo=vrednost.Primer: width, height, top, left, scrollbars, resizable,…

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Korišćenje open() metode</title>
    </head>
    <body>
        <p>Otletanje školskog sajta u novom prozoru i ponovno zatletanje
</p>
        <button onclick="otvoriNoviProzor()">Otvori novi prozor</button>
         <script>
            function otvoriNoviProzor() {
                window.open('http://www.etspupin.edu.rs', 'ETSPupin',
'width=300, height=400');
            }
         </script>
    </body>
</html>
```

6. Metoda close() zatleta prozor za zadati url

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Korišćenje open() i close() metode</title>
    </head>
    <body>
        <p>Otletanje školskog sajta u novom prozoru </p>
        <input type="submit" onclick="otvoriNoviProzor()" value="Otvori novi
prozor">
        <input type="submit" onclick="zatvoriNoviProzor()" value="Zatvori
novi prozor">
         <script>
            let noviProzor;
            function otvoriNoviProzor() {
                noviProzor = window.open('http://www.etspupin.edu.rs',
'ETSPupin', 'width=300, height=400');
            }
             function zatvoriNoviProzor() {
```

```
                noviProzor.close();
            }
        </script>
    </body>
</html>
```

DOM - Tipovi podataka

Najčešći tipovi podataka sa kojima ćete se sretati prilikom povezivanja JavaScript koda sa DOM-om je

- **document** - je objekat koji je koren svih objekata na veb strani
- **element** - se može odnositi na bilo koji objekat na strani (bilo koji node)
- **nodeList –** predstavlja listu elemenata. Npr metoda `document.getElementsByTagName()` vraća *node list* – sve elemente u okviru strane koji imaju određeni tag
- **attributes –** su objekti koji se odnose na atribute elemenata koji čine DOM

```
let list = document.getElementsByTagName('p'); // lista paragrafa u dokumentu
list[0] // klasično pristupanje elementu preko indeksa
list.item(0) // pristupanje elementima preko metode item() koja je definisana
u nodeList objektu
```

DOM - Metode

Neko od metoda koje se često koriste prilikom rada sa DOM objektima, a odnose se i na HTML stranu kao i na XML fajlove.

`document.getElementById(id)`

```
let e = document.getElementById(id)
```

Vraća referencu na element čiji id je prosledjen metodi getElementById()

id je *case-sensitive* string

7. Promena boje pozadine i teksta paragrafa.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Primer upotrebe metode getElementById</title>
    </head>
    <body>
        <p id="text">Upotreba getElementById metode</p>
        <button onclick="changeColor('white', 'gray');">Bela boja
teksta</button>
        <button onclick="changeColor('red', 'black');">Crvena boja
teksta</button>

        <script>
            function changeColor(textColor, bgColor) {
                let e = document.getElementById('text');
                e.style.color = textColor;
                e.style.backgroundColor = bgColor;
                e.style.padding = 10;
                e.style.width = 500;
            }
        </script>
    </body>
</html>
```

document.getElementsByTagName(name)

Vraća HTMLColection (kolekciju HTML elemenata) za zadati naziv taga. Elementima u kolekciji se pristupa kao u nizu.

8. Primer upotrebe metode getElementsByTagName

```
<!DOCTYPE html>
<html>
    <head>
        <title>Primer upotrebe metode getElementsByTagName</title>
    </head>
    <body>
        <p>Upotreba getElementsByTagName metode</p>
        <p>Upotreba getElementsByTagName metode</p>
        <p>Upotreba getElementsByTagName metode</p>
        <p>Upotreba getElementsByTagName metode</p>
        <p>Upotreba getElementsByTagName metode</p>
        <button onclick="changeParagrafBackgroundColor();">Oboji paragrafe
različitim bojama</button>
```

```
        <script>
            function changeParagrafBackgroundColor() {
                let colors = ['#fff000', '#00ffff', '#c70210', '#f5f5f5',
'#333333']
                let elements = document.getElementsByTagName('p');
                for( let i = 0; i < elements.length; i++) {
                elements[i].style.backgroundColor = colors[i];
                }
            }
        </script>
    </body>
</html>

document.querySelector(selectors)
```

Objekat **document** se odnosi na page (html stranu). Preko njega možemo da pristupimo svim ostalim objektima na strani. Selektovanje elemenata na strani se može izvršiti na različite načine.

```
let el = baseElement.querySelector(selectors);
            // selectors je string koji može da sadrži jedan ili više
selektora razdvojenih zarezom
```

Vraća prvi element na strani na koji naiđe za zadati selektor

```
// Selektovanje elementa preko naziva taga
let el = document.querySelector('p');

// Selektovanje elementa preko naziva klase
let el = document.querySelector('.className');

// Selektovanje elementa preko naziva identifikatora
let el = document.querySelector('#idName');
```

Ukoliko se **querySelector**- **u** prosledi grupa selektora razdvojenih zarezom, vratiće prvi element na koji naiđe u dokumentu iz grupe selektora.

```
let el = document.querySelector("#nav, #nav-bar");
// el će u ovom slučaju da sadrži prvi element na koji naiđe u dokumentu a da
mu je id nav ili nav-bar. Ukoliko ne naiđe na takav element el će biti null
```

Redosled selektora u grupi ne utiče na samu pretragu u okviru dokumenta. Rezultat bi bio isti i da smo izraz kreirali na sledeći način:

```
let el = document.querySelector("#nav-bar, #nav");
```

Rezultat isključivo zavisi od redosleda selektora u samom dokumentu.

`document.getElementsByClassName(name)`

Vraća **HTMLColection** (niz HTML elemenata) za zadati naziv klase. Pretražuje sve elemente za zadatu klasu koji se nalaze u okviru root elementa)(*element*), uključujući i njega . Elementima u kolekciji se pristupa kao u nizu.

```
let elements = element.getElementsByClassName(names);
```

elements – predstavlja HTML kolekciju

element - *root* element

```
let elements = document.getElementsByClassName(names); // Vraća sve elemente za zadatu klasu u okviru celog dokumenta
```

primer:

Linija koda kojom se može dobiti kolekcija elementata koji imaju klasu sa nazivom **red**, a nalaze se u okviru elementa koji ima id **box**.

```
let elements = document.getElementById('box').getElementsByClassName('red');
```

HTMLColection

- **elements.length** – svojstvo **length** vraća broj elemenata u kolekciji
- **elements.item()** - metoda **item()** vraća element za zadati indeks...npr **elements.item(0)** što je identično **elements[0]**

```
document.forms[0] === document.forms.item(0) // true
```

- **elements.namedItem(id)** – vraća element za dati id... npr **elements.nameditem(box)** što je identično **elements.box**

Elementu HTML kolekcije se može pristupiti i direktno preko naziva identifikatora elementa
`elements.box`

Forme. Validacija forme

Formi i njenim elementima možemo pristupiti preko naziva (*name* atributa) ili preko *id*-a kao jedinstvenog identifikatora.

Primer ako se na veb strani nalazi forma čiji je id= "login" ili name="login", može joj se preko kolekcije `forms` pristupiti na dva načina:

```
document.forms.namedItem('login'); // korišćenjem metode namedItem
            document.forms.login; // direktno preko naziva ili id-a
```

- `forms` je svojstvo `document` objekta koji vraća kolekciju (sve forme u okviru `document objekta`)
- `namedItem` metod vraća element čiji je name ili id prosleđen kao parametar. Ukoliko takav element ne postoji vraća `null` objekat
- `length` svojstvo vraća dužinu kolekcije
- `elements` svojstvo vraća sve elemente za datu formu
- `value`

Ostalim elementima forme se takođe pristupa preko naziva ili namedItem metode.

Document.

9. Primer Forme i elementi forme:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Forma i elementi forme</title>
    </head>
    <body>
        <p id="msg-box">Form info</p>
        <hr width="50%" align="left">
        <form name="login" method="post">
            <label for="username">Username: </label>
            <input type="text" name="username"><br>
            <label for="password">Password: </label>
            <input type="password" name="password"><br>
            <input type="submit" onclick="loginMsg()" value="Login">
        </form>
        <hr width="50%" align="left">
```

```
        <button onclick="formInfo()">Prikaži info o formi</button>
        <button onclick="printFormToConsole()">Prikaži form objekat u
konzoli</button>
         <script>
            function loginMsg() {
                alert('Uspešno ste se prijavili na sistem!')
            }
            function formInfo() {
                let username = document.login.username.value;
                let password = document.login.password.value;
                document.getElementById('msg-box').innerHTML = 'Form info
<br> username:'+ username + '<br> password:' + password ;
            }
             function printFormToConsole () {
                let forma = document.forms.login
                console.log(forma);
                console.log(forma.length);
                console.log(forma.elements);
                alert('Pogledajte konzolu!');
             }
         </script>
    </body>
</html>
```

10. Primer Forme i elementi forme:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Rad sa formom</title>
    </head>
    <body>
        <form action="" method="POST">
            <input type="button" value="Predji mišem preko"
onmouseover="mouseMoveCounter()">
            <input type="button" value="Kraj" onclick="message()">
        </form>
    <script>
        let counter = 0;
        function message() {
            alert('Gotovo je!');
        }
        function mouseMoveCounter() {
            counter++;
            if (counter == 1) {
                alert("Prvi prelazak preko dugmeta!");             }
            else {
                alert("Ukupno " + counter + " prelazaka preko dugmeta!");
            }
        }
     </script>
    </body>
</html>
```

11. Primer Forme i elementi forme:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Rad sa formom</title>
    </head>
    <body>
        <form action="" method="post" name="mojaforma">
            Upišite prvi broj (0-10): <input type="text" name="broj1"
size=5><br>
            Upišite drugi broj (0-10): <input type="text" name="broj2"
size=5><br><br>
            <input type="button" value="saberi" name="dugme"
onclick="saberi()"><br><br>
            Zbir ta dva broja iznosi: <input type="text" name="zbir"
size=5><br>
            Tekstualni podatak o rezultatu: <input type="text" name="tekst"
size=30>
        </form>
        <script>
            function saberi() {
                let br1 = document.mojaforma.broj1.value - 0;
                let br2 = document.mojaforma.broj2.value - 0;
                let zbir = br1 + br2;
                let poruka = "";
                if (zbir <= 0)
                    poruka = "nula ili negativan!";
                else if (zbir > 10)
                    poruka = "veći od deset!";
                    else poruka = "između 1 i 10!";
                    document.mojaforma.zbir.value = zbir;
                    document.mojaforma.tekst.value = "zbir je " + poruka;
            }
        </script>
    </body>
</html>
```

Date

Date objekat je ugrađen JavaScript objekat, koji se koristi prilikom programiranja aplikacija, kada su potrebne informacije o datumu i vremenu. Kako je u aplikacijama česta potreba za korišćenjem kalendara bitno je savladati tehniku rada sa ovim objektom i nj ovim metodama.

Kreiranje instance Date predstavlja samo jedan momenat u vremenu. Predstavlja vreme u milisekundama, a računa se od referentnog vremena - od ponoći **1. Januara 1970** po UTC vremenu.

Instanciranje objekta Date se vrši na više načina:

```
new Date();
```

```
new Date(value);
```
// kreira se novi Date objekat na osnovu *value*, vremena u milisekundama koje se računa od 1.1.1970.

```
new Date(dateString);
```
// kreira se novi Date objekat na osnovu *stringa* (formata datuma)

```
new Date(year, monthIndex [, day [, hours [, minutes [, seconds [, milliseconds]]]]]);
```
// kreira se novi Date objekat na osnovu parametara za: godinu, mesec, sate, minute, sekunde i milisekunde

```
// (monthIndex) indeks meseca  za januar je 0, a za December = 11.
```

Formati datuma i vremena

Godina:
YYYY (2018)
Godina i mesec:
YYYY-MM (2018-04)
Kompletan datum:
YYYY-MM-DD (2018-04-16)
Kompletan datum plus sati i minuti:
YYYY-MM-DDThh:mmTZD (2018-04-16T19:20+01:00)
Kompletan datum plus sati, minuti i sekunde:
YYYY-MM-DDThh:mm:ssTZD (2018-04-16T19:20:30+01:00)
Kompletan datum plus sati, minuti, sekunde i decimalni deo sekunde:
YYYY-MM-DDThh:mm:ss.sTZD (2018-04-16T19:20:30.45+01:00)

Gde je:

YYYY = četiri cifre za godinu
MM = dve cifre za mesec (01=January, etc.)
DD = dve cifre za dan u mesecu (01 through 31)
hh = dve cifre za sat (od 00 do 23) (am/pm nije dozvoljeno)
mm = dve cifre za minute (od 00 do 59)
ss = dve cifre za sekunde (od 00 do 59)
s = jedna ili više cifara predstavljaju decimalni deo sekunde
TZD = odrednica vremenske zone (Z or +hh:mm or -hh:mm)

Često korišćene metode:

Metode	Opis	Povratna vrednost
getdate() getUTCDate()	Vraća dan u mesecu za specificiran datum	1-31

`getDay()` `getUTCDay()`	Vraća dan u nedelji (integer)	0-6, Nedelja 0, Ponedeljak 1,…
`getFullYear()` `getUTCFullYear()`	Vraća godinu (sa četiri cifre)	1900+
`getHours()` `getUTCHours()`	Vraća sate u toku dana (integer)	0-23
`getMilliseconds()` `getUTCMilliseconds()`	Vraća milisekunde (od poslednje sekunde)	0-999
`getMinutes()` `getUTCMinutes()`	Vraća minute (od poslednjeg sata)	0-59
`getMonth()` `getUTCMonth()`	Vraća mesece	0-11
`getSeconds()` `getUTCSeconds()`	Vraća sekunde (od poslednjeg minuta)	0-59
`getTime()`	Vraća milisekunde od 1 January 1970	
`getYear()`	Vraća godine	0-99 za godine između 1900-1999 Četiri cifre za godine 2000+
`parse()`	Vraća broj milisekundi od ponoći 1. januara 1970. Za dati datum i vreme	
`setDate()` `setUTCDate()`	Setuje dan na dan u mesecu 1-31	Vreme u miliskundama
`setFullYear()` `setUTCFullYear()`	Setuje godinu za date četiri cifre	Vreme u miliskundama
`setHours()` `setUTCHours()`	Setuje sate na dati broj između 0-23	Vreme u miliskundama
`setMilliseconds()` `setUTCMilliseconds()`	Setuje milisekunde na dati broj	Vreme u miliskundama
`setMinutes()` `setUTCMinutes()`	Setuje minute na dati broj između 0-59	Vreme u miliskundama
`setMonth()` `setUTCMonth()`	Setuje mesece na dati broj između 0-11	Vreme u miliskundama
`setSeconds()` `setUTCSeconds()`	Setuje sekunde na dati broj između 0-59	Vreme u miliskundama
`setTime()`	Setuje datum za dati broj milisekundi od 1. januara 1970	Vreme u miliskundama
`setYear()`	Setuje godinu za dve ili četiri cifre	Vreme u miliskundama

`toString()`	Vraća lokalni datum i vreme kao string	Zavisi od operativnog sistema i brauzera

Primeri:

1. **getDate()**

```
let birthday = new Date('August 14, 1976 13:15:30');
let date = birthday.getDate();
console.log(date); // 14
```

2. **getDay()**

```
let birthday = new Date('August 14, 1976 13:15:30');
let day = birthday.getDay();
console.log(day); // 6
```

3. **getFullYear()**

```
let datum = new Date('August 14, 76 13:15:30');
console.log(datum.getFullYear()); // 1976
```

4. **getHours()**

```
let datum = new Date('August 14, 76 13:15:30');
console.log(datum.getHours()); // 13
```

5. **getMilliseconds()**

```
let today = new Date();
let milliseconds = today.getMilliseconds();
console.log(milliseconds);
```

6. **getMinutes()**

```
let today = new Date();
let minutes= today.getMinutes();
console.log(minutes);
```

7. **getMonth()**

```
let today = new Date();
let month= today.getMonth();
console.log(month);
```

8. **getSeconds()**

```
let today = new Date();
let seconds= today.getSeconds();
console.log(seconds);
```

9. getTime()

```
let today = new Date();
let time= today.getTime();
console.log(time);
```

10. getYear()

```
let today = new Date();
let year = today.getYear();
console.log(year); // problem za godine 2000 pa na dalje vraća broj
godina + 100...zato je u upotrebi getFullYear
```

11. parse()

```
let time1 = Date.parse('01 Jan 1970 00:00:00 GMT');
let time2 = Date.parse('04 Dec 1995 00:12:00 GMT');
console.log(time1); // 0
console.log(time2); // 818035920000
```

12. setDate()

```
let event = new Date('August 14, 76 13:15:30');
event.setDate(24);
console.log(event);
```

…

Zadaci za vežbu

1. U okviru html strane dodati sledeći skript

```
<script>
    let currentDate = new Date(),
    let day = currentDate.getDate(),
    let month = currentDate.getMonth() + 1,
    let year = currentDate.getFullYear();
    document.write(day + "/" + month + "/" + year)
</script>
```

2. U okviru html strane dodati sledeći skript

```
<script>
    let currentTime = new Date(),
    let hours = currentTime.getHours(),
    let minutes = currentTime.getMinutes();

    if (minutes < 10) {
        minutes = "0" + minutes;
    }
    document.write(hours + ":" + minutes)
</script>
```

3. U okviru html strane dodati sledeći skript

```
<script>
    let currentTime = new Date(),
    let hours = currentTime.getHours(),
    let minutes = currentTime.getMinutes();
    if (minutes < 10) {
        minutes = "0" + minutes;
    }

    let suffix = "AM";
    if (hours >= 12) {
        suffix = "PM";
        hours = hours - 12;
    }
    if (hours == 0) {
        hours = 12;
    }
    document.write(hours + ":" + minutes + " " + suffix)
</script>
```

12. Ispisati na veb strani u okviru h1 informaciju o današnjem datumu:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Rad sa datumom</title>
    </head>
    <body onload="ispis_datuma()">
        <p></p>
        <script>
            //funkcija za ispis danasnjeg datuma
            function ispis_datuma() {
                let d = new Date(); // novi objekat date, danasnji datum i
vreme
                let naziv_dana_u_nedelji = new Array('nedelja', 'ponedeljak',
'utorak',  'sreda', 'četvrtak', 'petak', 'subota');
                let i = d.getDay(); //metod getDay vraća redni broj dana u
nedelji
                let dan_u_n = naziv_dana_u_nedelji[i]; //naziv dana u sedmici
                let dan_u_m = d.getDate(); // getDate metod za dobijanje
rednog broja dana u mesecu
                let m = d.getMonth() + 1; //metod za dobijanje rednog broja
meseca
                let g = d.getFullYear(); //metod za dobijanje godine
                document.getElementsByTagName('p')[0].innerHTML = '<h1>' +
dan_u_n + ",  " + dan_u_m + "." + m + ". " + g + ".</h1>" ;
            }
```

```
        </script>
    </body>
</html>
```

13. Izračunajte koliko je ostalo dana do kraja kalendarske godine

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Rad sa datumom</title>
    </head>
    <body>
        <script>
            let danas = new Date(); // danasnje vreme i datum
            let kraj_god = new Date(); // nova promenljiva za kraj godine
            kraj_god.setMonth(11); // 11 je decembar
            kraj_god.setDate(31); // 31. dana u mesecu
            // Koliko ima milisekundi izmedju 31.12. i danasnjeg dana
            let razlika = kraj_god.getTime() - danas.getTime();//
Potrebno je izraziti u danima
            razlika = Math.floor( razlika/ (1000*60*60*24) );
            document.write("<p> Do kraja godine je ostalo " + razlika + "
dana!!!</p>"); </script>
        </script>
    </body>
</html>
```

14. Napisati JavaScript kod kojim će se omogućiti da bude prikazana slika na strani samo do određenog datuma

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Rad sa datumom</title>
    </head>
    <body>
        <h4>primer slike koja je vidljiva samo do odredjenog
datuma...</h4>

        <script>
            let ponuda = '<h1>Popust</h1>';
            let datum = new Date(); /*danasnji datum*/
            function proveraisteka(datumisteka) {
                let datisteka = new Date(datumisteka); /* datum isteka */
                if (datum.getTime() <= datisteka.getTime()) {
```

```
                    document.write(ponuda);
                    document.write('<img src="sl1.jpg"><hr>Ponuda vazi do
dana:<br>' + datisteka);
                } else {
                    document.write("<hr>Ponuda je vazila do dana:<br>" +
datisteka);
                }
            }
            proveraisteka("November 30, 2019")
        </script>
    </body>
</html>
```

15. **Kreirati digitalni časovnik koji će vrednost vremena ispisivati u okviru forme (u input text box-u), u obliku HH:MM:SS**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Digitalni časovnik</title>
    </head>
    <body onload= "digitalni_sat()">
        <form name= "sat"><input name="vreme"></form>
        <script>
            function digitalni_sat() {
                let danas = new Date();
                let sat = danas.getHours();
                let minut = danas.getMinutes();
                let sekunda = danas.getSeconds();
                let temp = sat;
                temp += ((minut < 10) ? ":0" : ":") + minut;
                temp += ((sekunda < 10) ? ":0" : ":") + sekunda;//u
formi sat u njeno polje vreme postavi vrednost temp
                document.sat.vreme.value = temp;
                setTimeout("digitalni_sat()", 1000);  //novi poziv f-je
digitalni_sat() nakon 1000ms
            }
        </script>
    </body>
</html>
```

16. **Napisati JavaScript kod kojim će se ispisati tekst Dobrodošli u okviru paragrafa. Posle 5 sekundi potrebno je da se tekst obriše i da se korisniku ispiše poruka obaveštenja, da je vreme od 5 sekundi, od učitavanja strane isteklo**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Ispis i brisanje teksta u okviru forme</title>
    </head>
    <body onload= "ispis()">
        <p id="demo"></p>
        <script>
            function ispis() {
```

```
                document.getElementById('demo').innerHTML =
"Dobrodošli";
                setTimeout('brisanje()', 5000);  //novi poziv f-je
digitalni_sat()nakon 1000ms
            }
            function brisanje() {
                document.getElementById('demo').innerHTML = "";
                alert('Prošlo je 5 sekundi od učitavanja strane!');
            }
        </script>
    </body>
</html>
```

17. Napisati skript koji će da menja boju pozadina u zavisnosti od doba dana (od 20h uveče do 5 ujutru crna, od 5 do 12 žuta, od 12 do 16 plava i od 16 do 20 narandžasta. Za izradu skripte koristiti onload događaj u okviru body taga (koji će pozvati funkciju promena_boje())

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Menjanja pozadine</title>
    </head>
    <body onload="promena_boje()">
        <p>Promena boje pozadine u zavisnosti od doba dana</p>
        <script>
            function promena_boje() {
                let datum = new Date(); /* prihvatanje tekuceg datuma u
promenljivu */
                let sati = datum.getHours(); /* smestanje tekuceg časa
u promenljivu */
                if (sati > 21 || sati < 6)
                    document.bgColor= "#000";
                else if (sati > 6 && sati < 12)
                    document.bgColor="#FFE97F";
                else if (sati > 12 && sati < 16)
                    document.bgColor="#14D3FF";
                else if (sati > 16 && sati < 20)
                    document.bgColor="#FF8432";
            }
        </script>
    </body>
</html>
```

18. Kreirati skript koji omogućava učitavanje različite pozadine (slučajan izbor) iz niza pozadina. Izabrati tri pozadinske slike.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Menjanja pozadine - Slučajan izbor slike</title>
```

```
    </head>
    <body onload="bojaPozadine()">
    <p>Promena pozadinske slike slučajnim izborom <p/>
    <script>
        function bojaPozadine() {
            let background1 = "img1.jpg";
            let background2 = "img2.jpg";
            let background3 = "img3.jpg";
            let background;
            let sl_broj = Math.round(5 * Math.random()) // slučajni
broj između 0 i 5, round zaokruzuje na blizi ceo broj
            if (sl_broj < 2) // izbor pozadine zavisi od slučajnog
broja
                background = background1;
            else if (sl_broj < 4)
                background = background2;
            else  background = background3;
            document.body.background = background;
        }
        </script>
    </body>
</html>
```

19. Upotreba svojstva innerHTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>innerHTML - Get</title>
    </head>
    <body>
        <div id='box'>
        <div> Sadržaj u okviru div-a čiji je id box </div>
        <p>Paragraf teksta</p>
        </div>

        <script>
            document.body.onload = innerContent; //Pozivanje metode na
događaj učitavanja strane
            function innerContent () {
                console.log(document.getElementById('box').innerHTML);
            }
        </script>
    </body>
</html>
```

20. Promena sadržaja u paragrafu

```
<!DOCTYPE html>
<html>
    <head>
```

```
        <title>innerHTML - Set</title>
    </head>
    <body>
        <div id='box'>
        <div> Sadržaj u okviru div-a čiji je id box </div>
        <p id="par">Paragraf teksta</p>
        <button onclick="setContentToParagraf();">set</button>
        </div>

        <script>
            document.body.onload = innerContent; //Pozivanje metode na
događaj učitavanja strane
            function innerContent () {
                console.log(document.getElementById('box').innerHTML);
            }
            function setContentToParagraf(){
                document.getElementById('par').innerHTML = '<a
href="#">Umesto paragrafa link</a>';
                console.log(document.getElementById('par').innerHTML);
            }
        </script>
    </body>
</html>
```

element.style

Svojstvo style omogućava dodavanje inline stila datom elementu.

21. Primena style svojstva - Postavljanje boje pozadine i teksta

```
<!DOCTYPE html>
<html>
    <head>
        <title>style svojstvo</title>
    </head>
    <body>
        <p id="par">Primena style svojstva - Postavljanje boje pozadine i
teksta </p>
        <button onclick="setStyleOfParagraf();">set</button>

        <script>
            function setStyleOfParagraf(){
                document.getElementById('par').style = 'background-color:
red; color: white;';
            }
        </script>
    </body>
</html>
```

element.setAttribute()

Metoda **setAttribute()** postavlja određene atribute za dati element.

primer1:

```
document.getElementById('box').setAttribute('class', 'red');
```

22. Primena metode setAttribute

```
<!DOCTYPE html>
<html>
    <head>
        <title>setAttribute</title>
        <style>
            .red {
               background-color: red;
            }
        </style>
    </head>
    <body>
        <p id="par" style="width: 400px; padding: 10px;" >Primena metode
setAttribute</p>
        <button onclick="setAttrToParagraf();">set</button>

        <script>
            function setAttrToParagraf(){
                document.getElementById('par').setAttribute('class', 'red');
                document.getElementById('par').setAttribute('style',
'background-color: green;'); // ovaj način nije dobar jer će prepisati inline
svostvo koje je postojalo
            }
        </script>
    </body>
</html>
```

23. Primena metode setAttribute

```
<!DOCTYPE html>
<html>
    <head>
        <title>setAttribute</title>
        <style>
            .red {
                background-color: red;
            }
        </style>
    </head>
    <body>
        <p id="par" style="width: 400px; padding: 10px;" >Primena metode
setAttribute</p>
        <button onclick="setAttrToParagraf();">set</button>
        <script>
            function setAttrToParagraf(){
                document.getElementById('par').setAttribute('class', 'red');
```

```
                document.getElementById('par').style.backgroundColor =
'green';
            }
        </script>
    </body>
</html>
```

element.getAttribute()

Vraća vrednost za dati atribut elementa. U primeru koji sledi u konzoli će biti ispisan *inline style* za paragraf čiji je **id** par

24. Primena metode getAttribute

```
<!DOCTYPE html>
<html>
    <head>
        <title>getAttribute</title>
    </head>
    <body>
        <p id="par" style="width: 400px; padding: 10px;" >Primena metode
setAttribute</p>
        <button onclick="getAttrToParagraf();">get</button>

        <script>
            function getAttrToParagraf(){

console.log(document.getElementById('par').getAttribute('style'));
            }
        </script>
    </body>
</html>
```

Dodavanje Event Listener-a prilikom registrovanje događaja

element.addEventListener()

Metoda **addEventListener** omogućava da se elementu dodaju događaji, pri čemu mogu biti isti ili različiti.

25. Promena boje pozadine – Slučajan izbor boje. Klikom na pozadinu omogućiti promenu boju pozadine slučajnim izborom boje.

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
            #btn {
                background-color: #fff;
                border: 1px solid #000;
                width: 300px;
            }
        </style>
    </head>
    <body>
        <p id="btn">Promeni boju pozadine</p>
        <script>
            let el = document.getElementById('btn');

            // Math.random() vraća slučajan broj izmedju 0 i 1

            function random(number) {
                return Math.floor( Math.random() * (number+1) ); // da bi smo
dobili // broj od 0 do 255
                // kada prosledimo funkciji random kao parametar 255
            }

            function colorBackground() {

document.getElementsByTagName('body')[0].style.backgroundColor = 'rgb('+
random(255) + ', '+ random(255) + ','+ random(255) + ')';
                // moglo je i jednostavnije
                // document.body.style.backgroundColor = 'rgb('+ random(255)
+ ', '+ //random(255) + ','+ random(255) + ')';
            }
            el.addEventListener('click', colorBackground);

        </script>
    </body>
</html>
```

26. Paragrafu se dodaju na primer dva ista događaja click, na čije dešavanje će da se izvrše dve različite funkcije

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
            #paragraf {
                background-color: #fff;
                border: 1px solid #000;
                width: 300px;
```

```
            }
        </style>
    </head>
    <body>
        <p>Primer upotrebe addEventListener metode koja dodaje paragrafu dva
identična eventa, pri čemu će se izvršiti različite funkcije</p>

        <p id="paragraf">Oboji paragraf na klik i proširi ga</p>

        <script>
            let el = document.getElementById('paragraf');
            el.addEventListener('click', obojiParagraf);
            el.addEventListener('click', prosiriParagraf);

            function obojiParagraf() {
                this.style.backgroundColor = '#C70210';
                this.style.color = '#fff';
                alert('Prvi klik');
            }

            function prosiriParagraf() {
                this.style.width = '500px';
                alert('Drugi klik');
            }
        </script>
    </body>
</html>
```

27. Paragrafu se dodaju na primer tri različita događaja click, mouseover i mouseout, na čije dešavanje će da se izvrše tri različite funkcije

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
            #paragraf {
                background-color: #fff;
                border: 1px solid #000;
                width: 300px;
            }
        </style>
    </head>
    <body>
        <p>Primer upotrebe addEventListener metode koja dodaje tri različita
eventa, pri čemu će se izvršiti tri različite funkcije</p>

        <p id="paragraf">Oboji paragraf na klik i proširi ga</p>

        <script>
            let el = document.getElementById('paragraf');
            el.addEventListener('mouseover', obojiParagraf);
            el.addEventListener('click', prosiriParagraf);
            el.addEventListener('mouseout', promeniBoju);
```

```
            function obojiParagraf() {
                this.style.backgroundColor = '#C70210';
                this.style.color = '#fff';
            }

            function prosiriParagraf() {
                this.style.width = '500px';
            }
            function promeniBoju() {
                this.style.backgroundColor = '#347fe1';
                this.style.color = '#fff';
            }
        </script>
    </body>
</html>
```

Dinamičko dodavanje sadržaja na strani

document.createElement(name)

Ova metoda kreira element u okviru **root** elementa. Kao paramatar prima naziv taga kao obavezni parametar.

28. Dodavanje sadržaja kada se strana učita

```
<!DOCTYPE html>
<html>
    <head>
        <script>
        // funkcija će se izvršiti kada se dokument učita
        window.onload = function() {

            // dodavanje naslova h1 i paragrafa p u praznu stranu kada se
učita u prozoru brauzera
            let naslov = document.createElement('h1');
            let tekst_naslova = document.createTextNode('Naslov!');
            naslov.appendChild(tekst_naslova);
            document.body.appendChild(naslov);
            let paragraf = document.createElement('p');
            let text_paragrafa = document.createTextNode('Tekst koji će biti
ispisan u paragrafu.');
            paragraf.appendChild(text_paragrafa);
            document.body.appendChild(paragraf);
        }
        </script>
    </head>
    <body>
    </body>
</html>
```

```
let newElement = document.createElement('div');
```

29. Upotreba createElement metode

```
<!DOCTYPE html>
<html>
    <head>
        <title>createElement</title>
    </head>
    <body>
        <div id='box'>Tekst iznad ovog div-a se kreira dinamički korišćenjem
createElement metode</div>
```

```
        <script>
            document.body.onload = addElement; //Pozivanje metode na događaj
učitavanja strane
            function addElement () {
                // kreiranje elementa
                let newElement = document.createElement('div');
                // kreiranje sadržaja
                let newContent = document.createTextNode('Dinamički ubačen
tekst u veb stranu!');
                // dodavanje sadržaja u novi div
                newElement.appendChild(newContent);
                let currentElement = document.getElementById('box');
                // ubacivanje novog elementa iznad div-a čiji je id box
                document.body.insertBefore(newElement, currentElement);
            }
        </script>
    </body>
</html>
```

parentNode.appendChild(node)

Ova metoda dodaje novi elemenat (**node**) na kraj liste svih elemenata, koji se nalaze u datom parent elementu (**parentNode**).

30. Upotreba appendChild metode

```
<!DOCTYPE html>
<html>
    <head>
        <title> appendChild </title>
    </head>
    <body>
        <div id='box'>Unutar ovog diva ce se ubaciti novi div</div>

        <script>
            document.body.onload = addElement; //Pozivanje metode na događaj
učitavanja strane
            function addElement () {
                // kreiranje elementa
                let newElement = document.createElement('div');
                // kreiranje sadržaja
                let newContent = document.createTextNode('Novi div');
                // dodavanje sadržaja u novi div
                newElement.appendChild(newContent);
                document.getElementById('box').appendChild(newElement);
            }
        </script>
```

```
    </body>
</html>
```

element.innerHTML

Svojstvo innerHTML vraća HTML sadržaj nekog elementa ili ubacuje novi sadržaj njega. U konzoli će biti prikazane sledeće dve linije koda:

```
<div> Sadržaj u okviru div-a čiji je id box </div>

<p>Paragraf teksta</p>
```

Event object – Naprednija tehnika

U gore navedenom primeru je definisan tip **event**-a na čiji događaj će se izvršiti određeni kod.

Zamislite situaciju da se na formi nalazi više elemenata koji treba da se ponašaju na isti način kada se klikne na njih. U tim situacijama se koristi **event objekat** koji nosi više informacija o objektu nad kojim se događaj desio.

Prosleđivanjem event objekta funkciji koja treba da se desi na određeni event može se markirati element nad kojim se događaj desio korišćenjem **event.target** svojstva. **Target** svojstvo **event** objekta je uvek referenca na objekat, nad kojim se događaj dogodio.

Prilikom prosleđivanje event objekta prosleđuje even objekat (e, evt, event).

31. Popunjavanje body taga sa 12 div elemenata, pri čemu je omogućena sledeća funkcionalnost onclick događaja za svaki div: Kada se klikne na bilo koji od njih boja pozadine mu se menja u neku random boju.

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
            body {
                background-color: #000;
            }
            div{
```

```
                background-color: #000;
                border: 1px solid #000;
                height: 150px;
                width: 33%;
                float: left;
            }
        </style>
    </head>
    <body>
    <script>
        // popunjavanje body taga (crne pozadine) sa 12 div elemenata
        for (let i = 1; i <= 12; i++) {
            let el = document.createElement('div');
            document.body.appendChild(el);
        }
        // funkcija za generisanje slučajnog broja
        // da bi smo dobili broj od 0 do 255, funkciji kao parametar
prosle]ujemo 255
        function randomNumber(number) {
            return Math.floor( Math.random() * (number+1) );
        }
        // funkcija za bojenje pozadine
        function colorBackground() {
            let randomColor = 'rgb('+ randomNumber(255) + ', '+
randomNumber(255) + ','+ randomNumber(255) + ')';
            return randomColor;
        }
        //niz div elemenata
        let divs = document.getElementsByTagName('div');
        // definisanje onclick dogadjaja za svaki div
        // pri čemu se izvršava anonimus funkcija kojoj se prosleđuje event
objekat
        for(let i = 0; i < divs.length; i++) {
                divs[i].onclick = function(e) {
                e.target.style.backgroundColor = colorBackground(); //
event.target predstavlja referencu na div nad kojim se događaj desio
            }
        }
    </script>
    </body>
</html>
```