Statičke metode

JavaScript takođe daje mogućnost pisanja statičkih metoda.

Statičke metode su metode koje su definisane uz pomoć rezervisane reči static. Ove metode pripadaju klasi isključivo i ne mogu se pozivati nad objektima te klase.

Primer:

```
class Automobil{

    constructor(marka){
        this.proizvodjac=marka;
    }

    static statickaMetoda(){
        console.log("Ovo je staticka metoda klase Automobil!");
}
}

    let punto =  new Automobil("Fiat");

    //staticku metodu pozivamo nad klasom
    Automobil.statickaMetoda();


    //naredna linija koda dovela bi do greske
    punto.statickaMetoda();
```

U primeru iznad definisali smo statičku metodu klase Automobil. Prikazano je kako se vrši pozivanje statičke metode. Ukoliko želimo da naša statička metoda koristi neka svojstva objekta , objekat prosleđujemo kao parametar.

* *Zadatak*

Definisati klasu Osoba koja sadrži svojstva koja identifikuju osobu. Zatim definisati statičku metodu koja prima parametar i ispisuje jmbg objekta.

Nasleđivanje klasa

Klase nam obezbeđuju i nasleđivanje , odnosno mogućnost da jedna klasa nasledi drugu klasu.

Klasa roditelj predstavlja klasu koja se nasleđuje.

Pogodnosti koje dobijamo od nasleđivanja su te da klasa koja nasleđuje može imati metode klase roditelja kao i neke druge karakteristike definisane u toj klasi.

Primer nasleđivanja klase:

```
class Osoba {

        constructor(ime,prezime,jmbg){
            this.ime=ime;
            this.prezime=prezime;
            this.jmbg=jmbg;
    }


        predstaviSe(){
            console.log("Zovem se "+this.ime+" "+this.prezime);
}
}

class Ucenik extends Osoba {

        constructor(ime,prezime,jmbg,skola,razred){
                super(ime,prezime,jmbg);
                this.skola=skola;
                this.razred=razred;
}
}
```

U primeru iznad vidimo da klasa Ucenik nasleđuje klasu Osoba uz pomoć rezervisane reči **extends**.

Klasa Osoba poseduje svojstva ime , prezime i jmbg. Takođe poseduje i konstruktorsku metodu i metodu predstaviSe.
Pošto klasa Ucenik nasleđuje klasu Osoba ona takođe ima ta svojstva i metode. Potrebno je u konstruktoroj metodi klase Ucenik izvršiti pozivanje konstruktorske metode roditelja klase uz pomoć metode super() kojoj se prosleđuju parametri sa kojima radi konstruktorska metoda klase roditelja.

Primer kreiranja objekta roditelja klase i klase deteta:

```
let a = new Osoba("Djordje","Djordjevic",2312993333111);
let b =new  Ucenik("Nikola","Nikolic",3111331333112,"Gimnazija","Treci");

a.predstaviSe();
b.predstaviSe();
```

Nakon što smo kreirali dva objekta , pozvali smo metodu predstaviSe() nad oba objekta.

* *Zadatak*

Definisati klasu Vozilo koja ima svojstva godinaProizvodnje,brojSasije,boja,kubikaža. Takođe potrebno je da ova klasa poseduje metodu koja izračunava starost vozila i ispisuje poruku o tome koliko je vozilo staro.
Zatim napraviti klasu Automobil koja nasleđuje klasu Vozilo i ima svojstva marka, model , pogon , emisionaKlasa, brojBrzina, brojVrata, gorivo. Takođe potrebno je da nova klasa poseduje metodu koja ispisuje sve parove svojstvo vrednost.
Nakon što je napravljena klasa Automobil , treba napraviti klasu Kamion koja nasleđuje klasu Vozilo i poseduje svojstva marka,model,brojOsovina,maksimalnoOpterecenje.
Nakon što su napisane klase napraviti objekat za svaku klasu i pozvati sve metode koje se mogu pozvati nad svakim objektom.

JS HTML DOM

Kada je neka stranica učitana browser kreira **D**ocument **O**bject **M**odel te stranice.

HTML DOM definiše:

- HTML elemente kao objekte
- Svojstva svih html elemenata
- Metode za pristupanje svim HTML elementima
- Događaje za sve HTML elemente

Drugim rečima HTML DOM predstavlja standard koji nam omogućava da dohvatimo , promenimo ,dodamo ili obrišemo html element.

Document

Ako se pitate zašto uvek kada koristimo JavaScript da manipulišemo sa nekim html elementom koristimo reč document rešenje na to pitanje je veoma jednostavno.

Document je objekat koji predstavlja našu web stranicu , nalazi se iznad svih drugih objekata na našoj stranici.

DOM metode

HTML DOM metode su akcije koje mogu biti izvršene nad HTML objektima.

Neke od ovih metoda smo već ranije pominjali ali ih nismo detaljno objašnjavali.

Pogledajmo sledeći primer metode za pronalaženje html elementa pa ga zatim analizirajmo:

```
document.getElementById("prviParagraf").innerHTML="Pozdrav svima!";
```

Sada kada znamo dosta stvari iz JavaScripta možemo videti da je ***document*** jedan objekat , zatim vidimo da je ***getElementById()*** metoda kojoj prosleđujemo kao parametar id html elementa koji dohvatamo. Ova metoda predstavlja jednu od najčešće korišćenih metoda za dohvatanje html elementa.
Zatim u ovom primeru vidimo i svojstvo **.innerHTML** kome dodeljujemo novu vrednost "Pozdrav svima!". Ovo svojstvo može se koristiti da se promeni vrednost bilo kog html elementa.

Takođe neke HTML elemente možemo pronaći i uz pomoć njihovog naziva taga.

Primer pronalaženja elemenata na ovaj način:

```
let a = document.getElementsByTagName("p");
```

U primeru iznad vidimo da opet koristimo *document* iz ranije navedenih razloga. Zatim sledi metoda **getElementsByTagName()** kojoj prosleđujemo naziv nekog html taga.

Ukoliko želite da pronađete sve HTML elemente koji pripadaju nekoj klasi možete koristiti sledeću metodu:

```
document.getElementsByClassName("naslovi");
```

Ova metoda getElementsByClassName dohvatiće sve elemente koji imaju atribut class="naslovi".

Vrednost nekog html elementa možemo menjati tako što dohvatimo neki objekat i promenimo njegovo svojstvo innerHTML.

Ovakav primer pokazan je već nekoliko puta pa ga nećemo dodatno objašnjavati.

```
document.getElementById("p1").innerHTML="Promenjena vrednost!";
```

Ukoliko želimo da promenimo vrednost atributa nekog html elementa to možemo uraditi na sledeći način:

```
<a id="link" href="https://www.facebook.com">Klikni</a>
document.getElementById("link").href="https://www.google.rs";
```

Potrebno je da pretpostavimo da u našem html fajlu imamo link tag koji ima id "link" i vodi na neku web adresu. Adresu na koju naš link vodi menjamo na način što dohvatamo taj html element i menjamo vrednost njegovog atributa href.

Menjanje CSS uz pomoć JavaScripta

DOM model omogućava JavaScriptu da menja i stilizaciju nekog html elementa.

Primer promene stilizacije:

```
document.getElementById("paragraf").style.color="green";
```

Potrebno je da dohvatimo neki html element uz pomoć neke od metoda za dohvatanje html elemenata zatim pišemo **style** pa atribut koji menjamo , u ovom slučaju **color**.

addEventListener() metod

Ovaj metod dodaje se nekom html elementu sa namerom da čeka na neki događaj i da reaguje kada se taj događaj desi.

Jednom elementu može se dodeliti više Event hendlera.

Sintaksa:

element.addEventListener(događajNaKojiSeReaguje, instrukcije)

Primer:

```
<p id="para">Ovo je paragraf!</p>
<script>
function reakcija(){
        a.style.color="red";
        }
let a=document.getElementById("para");
a.addEventListener("click", reakcija);
```

U primeru iznad imamo paragraf koji ispisuje "Ovo je paragraf!". U našem script tagu definisali smo funckiju koja menja stilizaciju odnosno boju paragrafa. Zatim smo dodali Event Listener kome smo rekli da pozove funkciju kada se dogodi klik.

Takođe moguće je direktno u Event Listeneru napisati funkciju kojom ćemo reagovati na neki događaj. Ako bi se nadovezali na prethodni primer to bi izgledalo ovako:

```
        a.addEventListener("mouseover", function(){
a.style.color="blue";
})
```