Osnove JavaScript-a

JS je

- slabo tipizirani programski jezik
- iterpretirani jezik
- standardni jezik koga podržavaju svi web browser-i

Prilikom definisanja JS koda moguće je definisati:

1. unutar zasebnog .js fajla JavaScript koji se dodaje unutar .html fajla preko script taga i unutar src atributa kao vrednost dodeljujemo putanju (naziv) do .js fajla i
2. unutar html-a definišemo script tag gde dalje navodimo JS kod. Postoji atribut type kojem kao vrednost dodeljujemo "text/javascript".

Prekopirati sledeći kod u prvi.html fajl:

In [ ]:
```
<html>
    <head>
        <script src="script1.js"></script>
        <script type="text/javascript">
            console.log("Hi, from header!");
        </script>
    </head>


    <body>

    </body>

</html>
```

Prekopirati sledeći kod unutar script1.js fajla i pokrenuti prvi.html fajl

In [ ]:
```
console.log("Hi, from script1!");
```

Promenljive (var, let, const)

- var
  - promenljiva se vidi u okviru cele funkcije
- let
  - promenjljiva se vidi u okviru bloka
- const
  - konstanta koja se vidi u okviru bloka

Objekti

1. Ugrađeni
   - nizovi (var niz = new Array()), push, length, pop, slice, ...
   - String (var tekst = "tekst"), substring, split, indexOf, length ...
2. Korisnički definisani

- Object

In [ ]:
```
let student = new Object();
student.ime = "Pera";
student.prezime = "Perić";
alert(student.ime + " " + student.prezime);
```

- JSON

In [ ]:
```
let student2 = {
    ime: "Mika",
    prezime: "Mikić",
    messageBox: function() {
        alert("Poruka koja se prikazuje pozivom funkcije.");
    }
}
```

- pozivom konstruktora (iduće sedmice)

Funckije

1. Ugrađene
    - isNaN
    - eval
    - parseInt
    - alert
    - itd.
2. Korisnički definisane (radićemo detaljnije sledeće nedelje)
    - funkcije se definišu sa ključnom reć function
    - funkcija se može preneti po referenci, pa je tako možemo smestiti u neku promenljivu ili koristiti kao parametar neke druge funkcije

Poruke

Postoje tri vrste dijaloga

1. alert(tekst)
2. confirm(tekst)
3. prompt(tekst)

Događaji

Funkcije možemo da pozivamo prilikom događaja koje korisnik izaziva sa klijentske strane aplikacije (u veb čitaču). Ove funkcijse se zovu eng. event handlers.

Spisak događaja:

- onload
- onclick

- onsubmit
- onfocus
- itd.

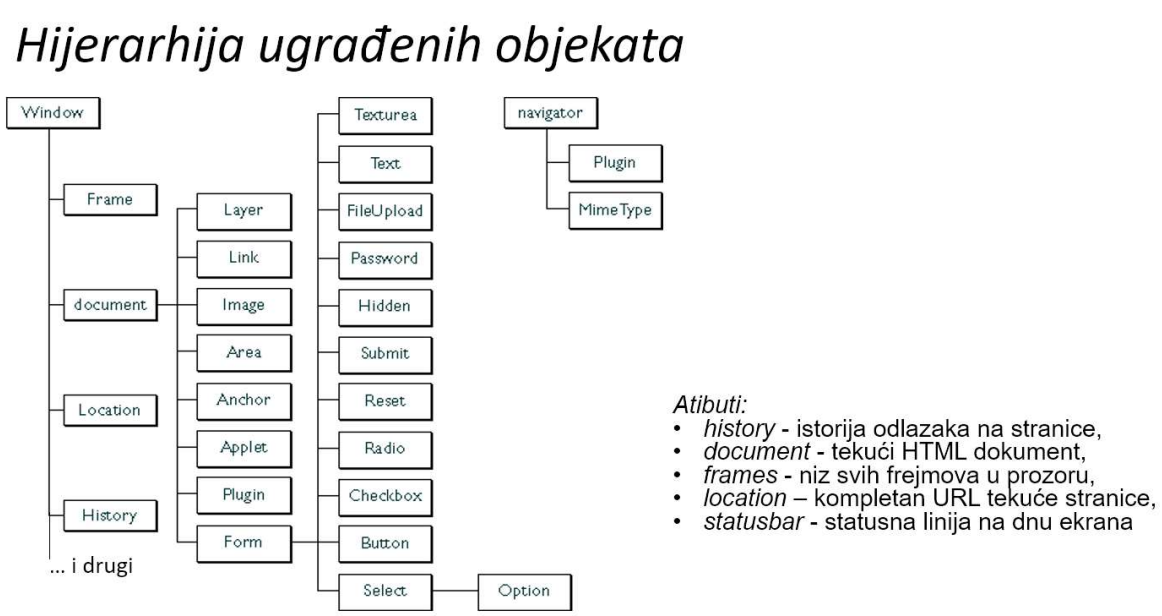
Prekopirati sledeći kod u drugi.html fajl

In [ ]:
```
<html>
    <head>
        <script src="script2.js"></script>
        <script type="text/javascript">
            function f1(a1, a2) {
                console.log(a1);
                console.log(a2);
            }
        </script>
    </head>


    <body>
        <button onclick="f1('1', '2')">Click1</button>
        <button onclick="f2('3')">Click2</button>
    </body>

</html>
```

Prekopirati sledeći kod unutar script2.js fajla i pokrenuti drugi.html fajl

In [ ]:
```
function f2(a1) {
    console.log(a1);
}
```

Hijerarhija ugrađenih objekata



Slika 1. Hijerarhija ugrađenih objekata

BOM (eng. Browser Object Model)

1. window
    - informacije o tekućem prozoru

- metode:
  - alert
  - confirm
  - open
  - itd.
- atributi:
  - screen
  - document
  - location
  - screenX
  - screenY
  - itd.

2. document
   - informacije o tekućem dokumentu
   - metode:
     - write
   - atributi
     - title
     - forms
     - links
     - itd.
3. location
   - URL stranice
   - metode:
     - reload
     - replace
       - atributi
       - href
       - protocol
       - host
       - port
       - pathname
       - search
4. History
   - kontrolu pristupa već viđenim stranicama
   - metoe:
     - back
     - forward
     - go
   - atributi:
     - current
     - length
     - next
     - previous
5. Navigator
   - opisuje veb čitač koji koristi korisnik
   - atributi:

- appName
- appversion
- cookieEnabled
- language
- itd.

DOM (eng. Documet Object Model)

Opisuje HTML element u obliku stabla. Pored toga sadrži i liste predefinisanih objekata koji se mogu javiti u HTML stranici: forms[] (lista koja sadrži sve forme koje se nalaze u HTML dokumentu), achors[], images[], itd. Možemo da pristupamo elementima ili da dodajemo nove elemente. Svaki element u stablu se naziva čvor. Postoje tri vrste čvorova:

1. html tag
2. html atribut
3. tekst

Klase koje opisuju DOM:

Za opis HTML dokumeta DOM koristi iduće klase:

Node

Klasa Node opisuje jedan čvor u stablu, daje nam metode pomoću kojih možemo manipulisati potomcima čvora, kao i metode za rad sa atributima.

Document

Klasa Document opisuje HTML dokumnet. Klasa Document sadrži atribute ka značajnim delovima HTML dokumenta (head, body, title, forms, anchors, images,...), kao i metode za dobavljanje elemenata iz stabla i kreiranje novih elemenata, atributa ili tekstualnih čvorova.

Form

Klasa koja opisuje formu. Sadrži atribute: elements (niz elemeneata forme), length, action, method, name,... Pored atributa tu su i metode submit() za slanje forme i reset() koje reset-uje sva polja forme.

Input

Klasa koja opisuje input elemente forme. Atributi: value, defaultValue, type, name, form, itd.

Style

Klasa koja nam omogućava upravljanje CSS atributima čvora: display, color, width, backgroundColor, ...

Pretraga DOM stabla i pristup elementima forme na osnovu:

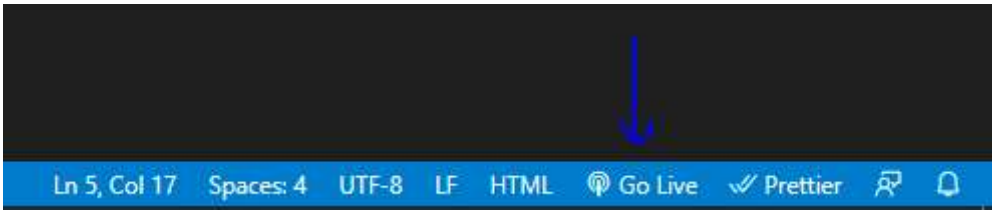
1. id elementa, funckija getElementById('id_value')

2. vrednosti atributa class, funckija getElementsByClass('class_name')[whole_number]

3. ime elementa, funkcija getElementsByName('name')[whole_number]
4. ime html elementa, funkcija getElementsByTagName('tag_name')[whole_number]
5. CSS selektora
    - funkcija querySelector('selector')
    - funkcija querySelectorAll('selector')[whole_number]

Upravljanje DOM-om

Za potrebe današnjih vežbi instalirajte live-server VS Code ekstenziju link. Ovaj server nam omogućava live reloading tj. osvježavanje servera svaki put kad sačuvamo nove promene, pa je iz tog razloga zgodan za razvoj. Da bi pokrenuli ovaj server potrebno je da pritisnemo GoLive dugme u statusnoj traci VS Code-a. Nakon toga se otvara vaš veb čitač i pokreće se server.



Sad ćemo proći kroz upravljanje DOM-om tako što ćemo vizualizovati stek. Prvi korak je da napravimo listu koju ćemo puniti stavkama i stavljati ih na vrh liste. Za početak prekopirajte idući kod u index.html:

In [ ]:
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
    <script src="index.js"></script>
  </head>
  <body>
    <h1>Stack</h1>
    <button onclick="pushToStack()">Push element to stack</button>
    <ul>
      <li>Some text</li>
      <li>Also some text</li>
    </ul>
  </body>
</html>
```

I idući kod u index.js:

In [ ]:
```
function pushToStack() {
  let ul = document.getElementsByTagName("ul")[0];
  let newLi = document.createElement("li");
  let text = document.createTextNode("New li with some text");
  let firstLi = document.getElementsByTagName("li")[0];
  newLi.appendChild(text);
  ul.insertBefore(newLi, firstLi);
}
```

Otvorite server i testirajte. Obratite pažnju da funkcija getElementsByTagName() vraća *listu* elemenata.

Idući korak je da omogućimo korisniku da uklanja elemente sa steka. Za to ćemo dodati dugme koje će reagovati na događaj pritiska miša i ukloniti element iz steka. Prekopirajte idući kod u index.html:

In [ ]:
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
    <script src="index.js"></script>
  </head>
  <body>
    <h1>Stack</h1>
    <button onclick="pushToStack()">Push element to stack</button>
    <button onclick="popFromStack()">Pop element from stack</button>
    <ul>
      <li>Some text</li>
      <li>Also some text</li>
    </ul>
  </body>
</html>
```

Dodajte funkciju popFromStack() u index.js:

In [ ]:
```
function pushToStack() {
  let ul = document.getElementsByTagName("ul")[0];
  let newLi = document.createElement("li");
  let text = document.createTextNode("New li with some text");
  let firstLi = document.getElementsByTagName("li")[0];
  newLi.appendChild(text);
  ul.insertBefore(newLi, firstLi);
}

function popFromStack() {
  let ul = document.getElementsByTagName("ul")[0];
  let firstLi = document.getElementsByTagName("li")[0];
  ul.removeChild(firstLi);
}
```