INTERNET TEHNOLOGIJE I VEB DIZAJN

Rad sa HTML elementima u JavaScript-u

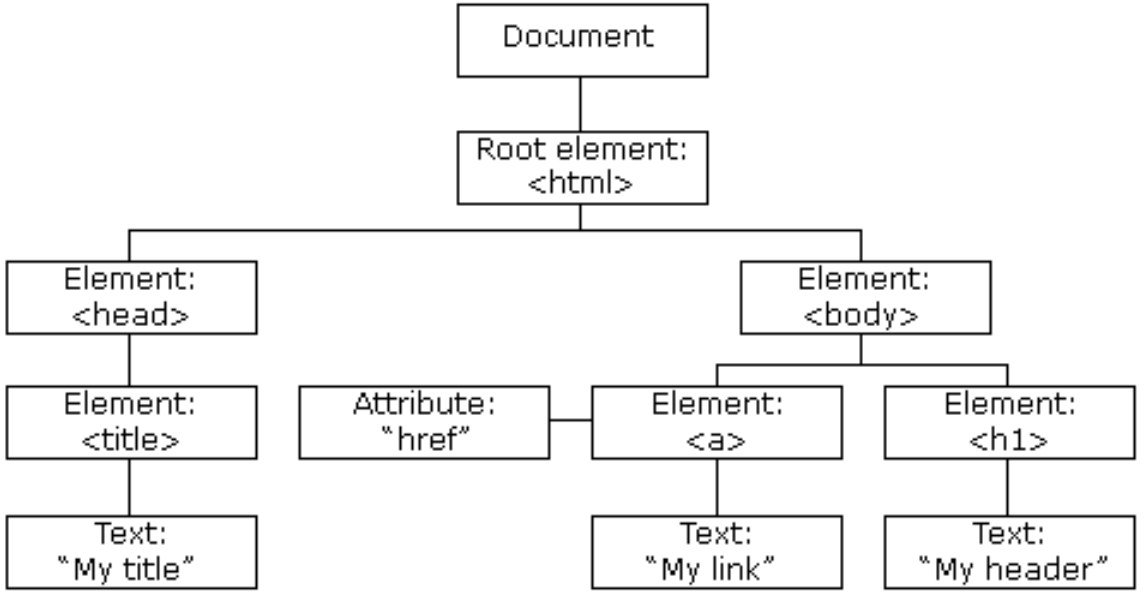
predavač: dr Miloš Antonijević

Objekat document

- Kada se HTML stranica učita u browser, njen sadržaj se prevede u objekat sa nazivom **document**.
- Elementima u HTML kodu se pristupa kroz **document** objekat korišćenjem DOM (Document Object Model) strukture.
- Korisnik vrši interakciju sa Veb stranicom kroz: 1. elemente HTML forme i 2. događaje koje browser detektuje i obrađuje (klik miša, prelazak preko nekog elementa...).
- **document** objekat zapravo predstavlja polje objekta **window** (https://developer.mozilla.org/en-US/docs/Web/API/Window)

DOM

- DOM se koristi za pristup i maniplaciju HTML dokumentima.
- DOM struktura predstavlja HTML dokument u obliku stabla



DOM (2)

- Entiteti (elementi, atributi, tekst...) u HTML dokumentu su u DOM strukturi predstavljeni uz pomoć **node**-ova (čvorova stabla).
- Čvorovi imaju svoja svojstva (polja) i metode.

Property	Description
attributes	A NamedNodeMap containing the attributes of this node (if it is an Element)
baseURI	Returns the absolute base URI of a node
childNodes	Returns a NodeList of child nodes for a node
firstChild	Returns the first child of a node
lastChild	Returns the last child of a node
nextSibling	Returns the node immediately following a node
nodeName	Returns the name of a node, depending on its type
nodeType	Returns the type of a node
nodeValue	Sets or returns the value of a node, depending on its type
ownerDocument	Returns the root element (document object) for a node
parentNode	Returns the parent node of a node
prefix	Sets or returns the namespace prefix of a node
previousSibling	Returns the node immediately before a node
textContent	Sets or returns the textual content of a node and its descendants

Pristup elementima forme

- Provera ispravnosti podataka koje korisnik unosi u elemente forme se može izvršiti na strani servera ili na strani klijenta.
- Prednost provere na strani klijenta je što se server rasterećuje suvišnih operacija.
- Sa druge strane, aplikacija koja se izvršava na serverskoj strane **ne sme** da veruje u ispravnost podataka koje dobije od klijenta.
- Najoptimalnije rešenje je vršiti proveru na klijentskoj strani, kako se neispravni podaci ne bi nepotrebno slali na server, ali istovremeno obavljati validaciju primljenih podataka na strani servera.
- Podaci koji su uneti kroz elemente HTML forme se na klijentskoj strani, putem JS-a, proveravaju tako što se njihova vrednost smesti u promenjivu koja se zatim obradi ili filtrira.

Pristup elementima forme (2)

- Svim HTML elementima se može pristupiti kroz **document** objekat na dva načina:
  1. Korišćenjem metoda getElementById(), getElementsByClassName(), getElementsByTagName():
     - getElementById() vraća element sa id-em koji je naveden kao argument metode
     - getElementByClassName() vraća niz elemenata koji imaju klasu koja je navedena kao argument
     - getElementsByTagName() vraća niz elemenata sa tagom navedenim u argumentu metode

  2. Ako se HTML element **forme** nalazi **u okviru forme**, navođenjem „putanje“ do svojstva **document** objekta, koje nosi naziv imena elementa (navedenog u **name** atributu) (npr. document.ime_forme.ime_elementa):
     - Napomena: ovo važi samo za elemente forme koji se nalaze unutar forme, u protivnom se koristi opcija 1.
     - U ovom sličaju atributi elementa predstavljaju polja kojima se može direktno prostupiti (npr. document.forma.ime.value='Ovo radi!')

Primer pristupa checkhox elementima

```
<form name="forma">
        <input type="text" name="proba" value="Tekst">
        <br>Musko: <input id="musko_chk" type="checkbox" name="musko">
        <br>Zensko: <input type="checkbox" name="zensko" checked>
</form>

<p onclick="document.forma.musko.checked = true">Klikni Ovde</p>
<p onclick="document.getElementById('musko_chk').checked = true">Ili Klikni
Ovde</p>
```

Primer pristupa radio elementima

```
<form name="forma">
        <br>Musko: <input id="musko_rb" type="radio" name="pol" value="musko">
        <br>Zensko: <input type="radio" name="pol" value="zensko" checked>
</form>

<p onclick="document.forma.pol[0].checked = true">Klikni Ovde</p>
<p onclick="document.getElementById('musko_rb').checked = false">Ili Ovde</p>
```

Primer pristupa select elementima

```
<select id="sel">
        <option>Cat</option>
        <option>Dog</option>
        <option>Fish</option>
</select>


document.getElementById('sel').getElementsByTagName('option')[2].selected =
true;
document.getElementById('sel').options[1].selected = true;
```

Dinamičko dodavanje elemenata

- Dodavanje elemenata u HTML DOM strukturu se može obaviti korišćenjem metode **createElement()** objekta **document**.
- Metodi createElement() se prosleđuje argument sa imenom čvora, odnosno elementa koji želimo da kreiramo.
- Nakon kreiranja, element dodajemo kao dete u željenom DOM elementu korišćenjem **appendChild()** metode roditeljskog elementa.

```
var paragraf = document.createElement("P");
paragraf.innerText = "Ovo je tekst paragrafa";
document.body.appendChild(paragraf);
```

Dinamičko dodavanje elemenata (2)

- Osim kao dete nekog čvora, element se može dodati kao njegov „srodnik“ (dele istog roditelja) korišćenjem **insertBefore()** metode roditeljskog elementa.
- insertBefore() metoda kao prvi argument prihvata element(čvor) koji želimo da dodamo, a kao drugi, čvor ispred koga želimo da dodamo novi element.

```
var ul_weeks = document.getElementById('weeks');
var day_to_add = document.createElement('li');
day_to_add.innerHTML = 'Tuesday';
weeks.insertBefore(day_to_add, ul_weeks.childNodes[2]);
```

- Ako se insertBefore() metoda koristi na način da je prvi parametar postojeći element u DOM strukturi, taj element će biti **premešten** na novu lokaciju, u odnosu na navedeni drugi argument metode.

Primer premeštanja elemenata

```
<ul id="myList1"><li>Kafa</li><li>Caj</li></ul>
<ul id="myList2"><li>Voda</li><li>Mleko</li></ul>

<p>Kliknite na dugme da premestite element iz jedne liste na drugu<p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
        var node = document.getElementById("myList2").lastChild;
        var list = document.getElementById("myList1");
        list.insertBefore(node, list.childNodes[0]);
}
</script>
```

Kreiranje atributa elementa

- Kreiranje atributa postojećem elementu se može obaviti korišćenjem createAttribute() metode objekta **document**.
- Argument metode predstavlja naziv atributa.
- Dodavanje atributa željenom elementu se vrši upotrebom setAttributeNode() metode roditeljskog elementa.

```
var link_elem = document.getElementById("singi_link");
var att = document.createAttribute("href");
att.value = "https://singidunum.ac.rs";
link_elem.setAttributeNode(att);
```

Promena slike i kreiranje komentara

- Slika na Veb stranici se može promeniti izmenom src atributa elementa img.

```
document.getElementById("myImageId").src="newSource.png";
```

- Kreiranje komentara željenom elementu se vrši upotrebom **createComment()** metode objekta **document**, nakon čega se komentar dodaje elementu korišćenjem appendChild() metode.

```
var c = document.createComment("Dodati komentar");
document.body.appendChild(c);
```

Otvaranje novog prozora

- Kreiranje novog prozora iz JS koda se obavlja korišćenjem metode **open()** objekta **window**.
- Prilikom kreiranja, kroz argumente metode open(), moguće je kontrolisati različite osobine prozora (https://www.w3schools.com/jsref/met_win_open.asp).
- Da bi se pristupalo svojstvima i metodama novootvorenog prozora neophodno ga je, prilkom kreiranja, smestiti u promenjivu.

```
var myWindow = window.open("", "MsgWindow", "width=200,height=100");
myWindow.document.write("<p>Ovo je novi prozor.</p>");
myWindow.document.title = "This is the new";
```

Osvežavanje stranice, preusmeravanje i štampanje

- Osvežavanje sadržaja (refresh) stranice iz JS se obavlja korišćenjem **reload()** metode polja **location** objekta **window**.

```
window.location.reload()
```

- Preusmeravanje (redirect) korisnika na drugu stranice se obavlja na dva načina:
  - Promenom vrednosti svojstva **href** polja **window.location** (simulira klik miša)
  - Upotrebom metode **replace()** polja **window.location** (simulira HTTP redirekciju)

```
window.location.href = "https://singidunum.ac.rs"
window.location.replace("https://singidunum.ac.rs")
```

- Štampanje trenutno aktivne Veb stranice iz Veb pregledača se obavlja korišćenjem **print()** metode objekta **window**.

Provera postojanja atributa

- Node objekat u JavaScript-u poseduje dve metode za vršenje provere da li određeni element poseduje tražene atribute.
- Prva metoda **hasAttribute()** proverava da li atribut čije je ime prosleđeno kao argument funkcije postoji u okviru elementa. Ukoliko postoji metoda vraća true, u suprotnom metoda vraća false.

```
<button id="dugme" onclick="func()">Klikni me</button>
<script>
function func() {
	console.log(document.getElementById("dugme").hasAttribute("onclick"));
}
</script>
```

- Druga metoda **hasAttributes()** proverava da li element ima bilo koji atribut. Ukoliko ima metoda vraća true, u suprotnom vraća false.

PITANJA?