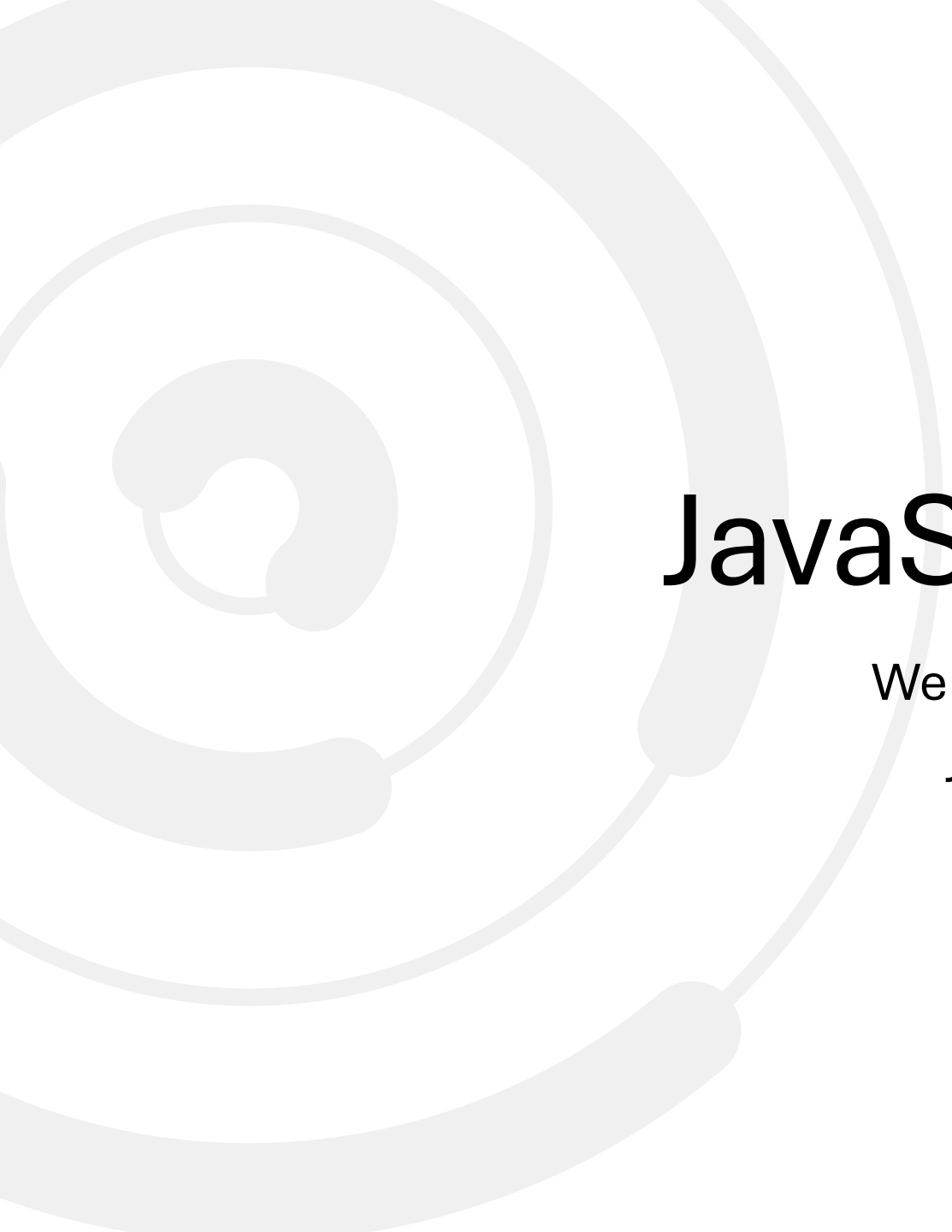


JavaScript, pt. 2

Web programiranje

Jurica Maltar

Document object model (DOM)

- Sučelje koje omogućuje upravljanje sadržajem nekog strukturiranog dokumenta unutar programskog jezika

- Hijerarhija unutar HTML-a može se prikazati kao n-arno stablo
  - Svaki je element čvor unutar stabla
  - `html` element je korijen stabla čija su djeca `head` i `body` elementi

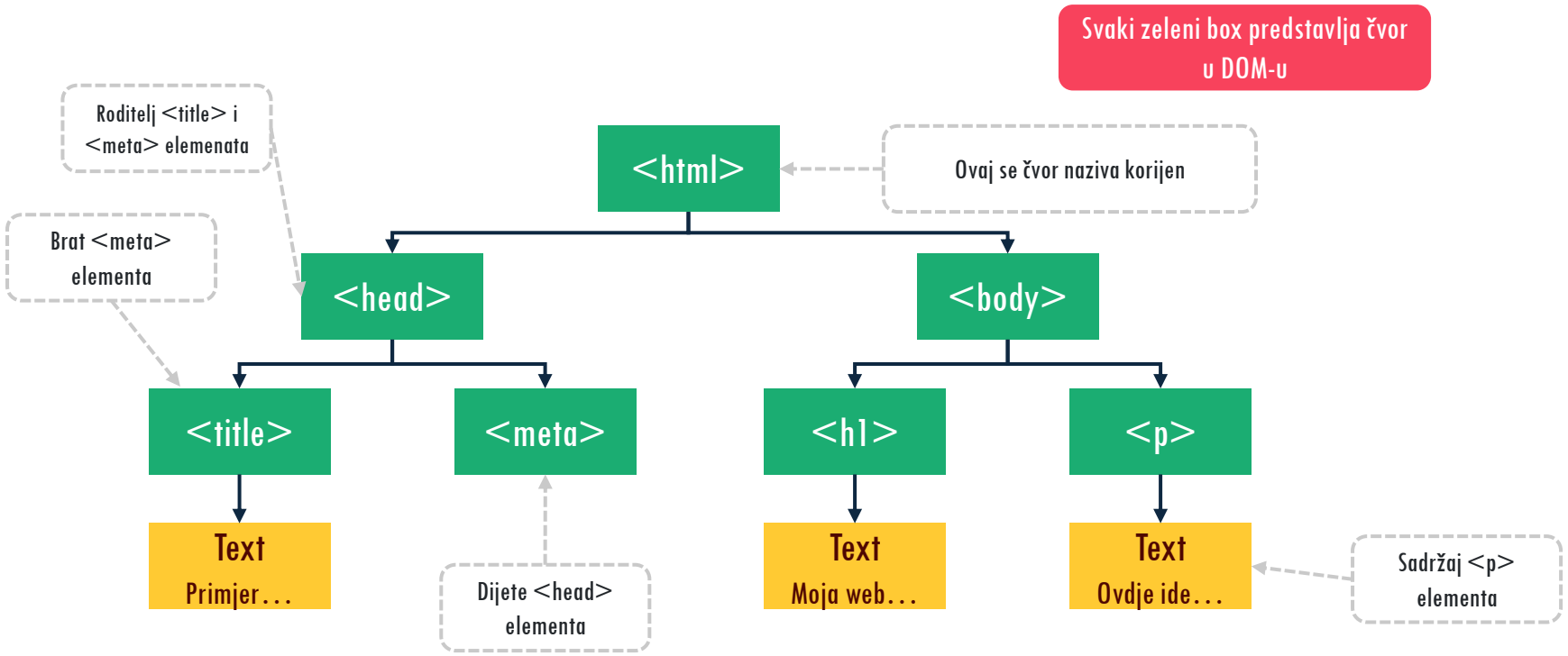
Document object model (DOM)

- Kada preglednik učita neki HTML dokument, on se pretvori u DOM strukturu
- Primjer:

```
<!DOCTYPE html>
<html>
<head>
    <title>Primjer jednostavne stranice</title>
    <meta name="author" content="Jurica">
</head>
<body>
    <h1>Moja web stranica</h1>
    <p>Ovje ide tekst moje osobne stranice</p>
</body>
</html>
```

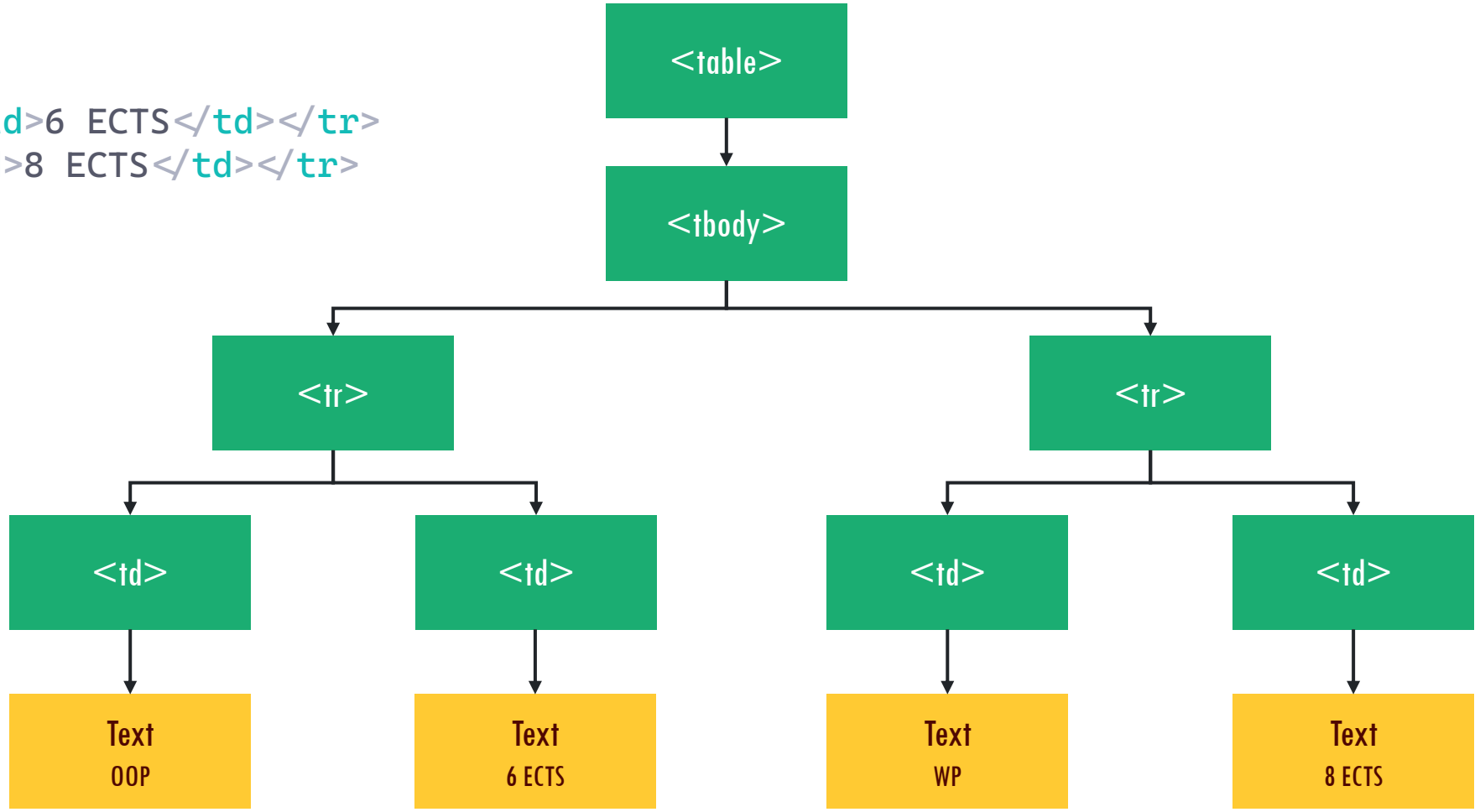
- Pogledajmo sada kako izgleda DOM ovog dokumenta.

DOM - struktura jednostavne web stranice

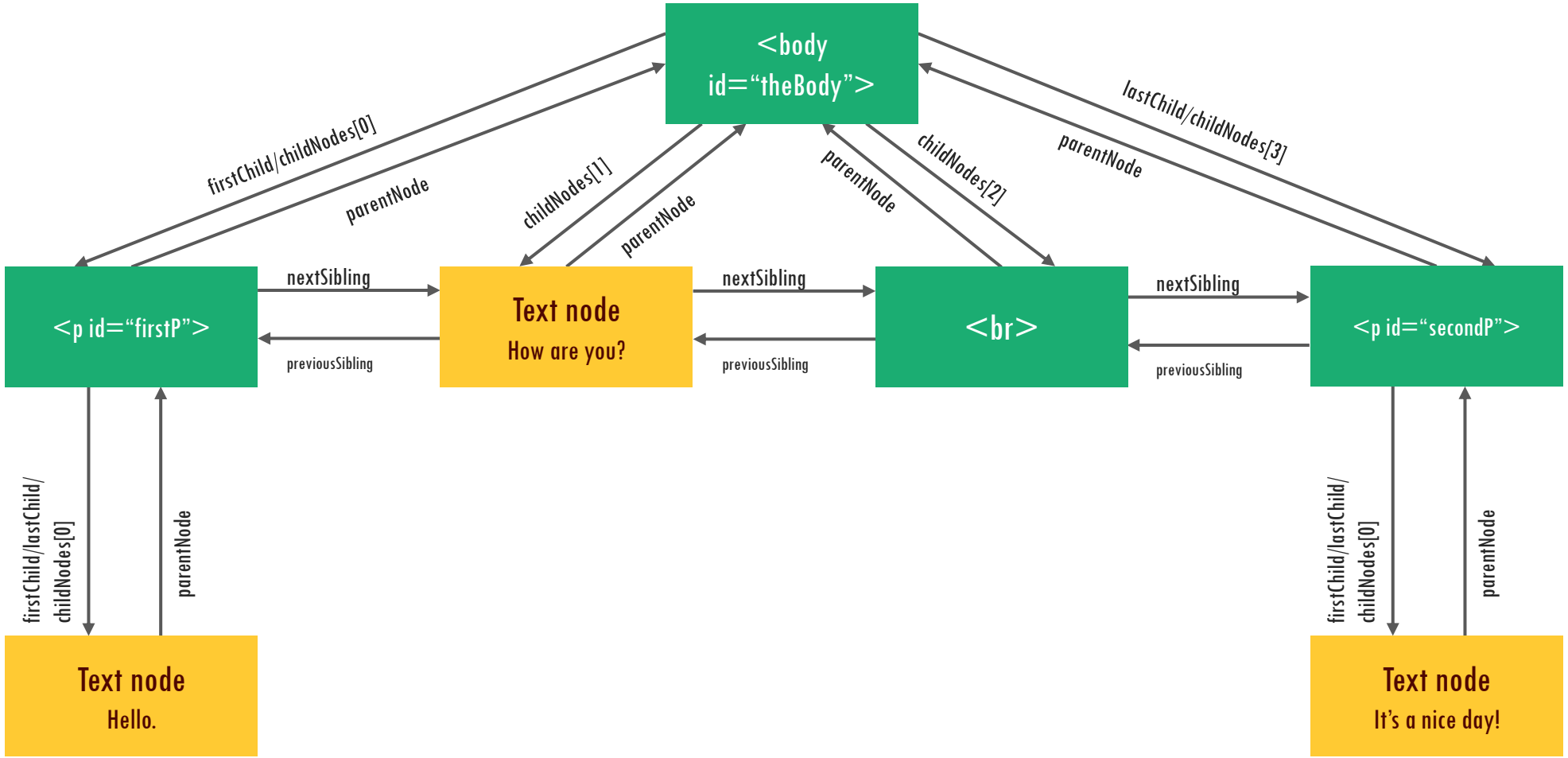


DOM - struktura dijela neke druge stranice

```
…
<table>
    <tbody>
    <tr><td>OOP</td><td>6 ECTS</td></tr>
    <tr><td>WP</td><td>8 ECTS</td></tr>
    </tbody>
</table>
…
```



DOM - veze među čvorovima



Pristupanje čvorovima DOM-a

- Svaki je html element spremljen u strukturi DOM-a
  - Možemo dodavati, brisati, kopirati ili mijenjati čvorove

- Kako bismo to učinili, trebamo znati pristupati čvorovima

- Pristup čvorovima:
  - `document.getElementById`
  - `document.getElementsByTagName`
  - `document.getElementsByClassName`

Pristupanje čvorovima DOM-a

- **Zadatak:** pristupimo čvorovima:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Click to change me</h2>
    <button id="abs-path-btn">Via abs. path</button>
    <button id="tag-name-btn">Via tag name</button>
    <button id="id-btn">Via id</button>
</body>
</html>
```

Dodavanje čvora DOM-u

1. Stvaranje HTML elementa
   - `document.createElement("hr")`
   - `document.createTextNode("Hello World")`

2. Za dodavanje čvora na odgovarajuće mjesto u DOM-u koristiti neku od metoda
   - `nodeRef.insertBefore`
   - `nodeRef.appendChild`

- **Zadatak**: Na kraju dokumenta stvorimo `<p>` s nekim tekstom (za dodavanje teksta koristiti atribut `innerText`) i nakon njega `<hr>`

Uklanjanje čvora iz DOM-a

- Ideja:
  1. Pristupiti odgovarajućem čvoru (nazovimo ga `c`)
  2. Pristupiti roditelju od `c`, i iz njega izbrisati sam `c`

- `parent.removeChild(c);`

- **Zadatak**: Uklonite treći gumb

Događaji (events)

- Izvršavanje događaja započinje korisnikovom interakcijom
- Događaj pokreće neku *callback* funkciju
- Događaji se registriraju unutar oznake elementa, npr:

```
<p onmouseover="changeColor()">I'm red on hover</p>
```

- Ili unutar JavaScripta:

```
elRef.addEventListener("click", () => { openAlert(); });
```

Mouse events

1. `onclick` - klik na lijevu tipku miša
2. `onmousedown` - pritisak na lijevu tipku miša, tipka stisnuta...
3. `onmouseup` - pri otpuštanju lijeve tipke
4. `onmouseover` - miš se pomiče iznad objekta
5. `onmouseenter` - miš je stigao na objekt
6. `onmouseleave` - miš je uklonjen s objekta

- **Zadatak:** pritiskom na prvi gumb promijenite boju naslova u crvenu, a pritiskom na drugi u plavu (`el.style.color` / `el.setAttribute("style", "color: " + color + ";");`)
- **Zadatak**: Napravite div kojemu se background mijenja u ovisnosti je li miš iznad njega ili ne

Keyboard events

1. `onkeydown` – čim korisnik pristisne tipku
2. `onkeypress` – kako korisnik stišće tipku
3. `onkeyup` – čim korisnik otpusti tipku

- **Zadatak**: Dinamički stvorite div čiji je unutarnji tekst ono što korisnik upisuje u dinamički stvoren input element + UPPERCASE-ano

`onload` event

- Evocira se pri inicijalizaciji dokumenta
- Primjer:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body onload="renderList()">
    <script src="./script.js"></script>
</body>
</html>
```

- **Zadatak** - dinamički popunite sadržaj neke liste

Timer events

- Pomoću `setInterval` izvršavamo neku funkciju svakih `n` milisekundi:

```
let timer = setInterval(callback, n);
```

- Pomoću `setTimeout` izvršavamo neku funkciju nakon `n` milisekundi:

```
let timer = setTimeout(callback, n);
```

- Timer se može resetirati:

```
clearInterval(timer);
clearTimeout(timer);
```

Timer events

- **Zadatak**: Napravite sat koji prikazuje sate, minute i sekunde u stvarnome vremenu i prikažite to unutar preglednika. Poslužite se s `new Date()`.
- **Zadatak**: Napravite odbrojavanje od 10 sekundi s razmakom od 10 milisekundi i prikažite to unutar preglednika. Poslužite se s `Date.now()`.