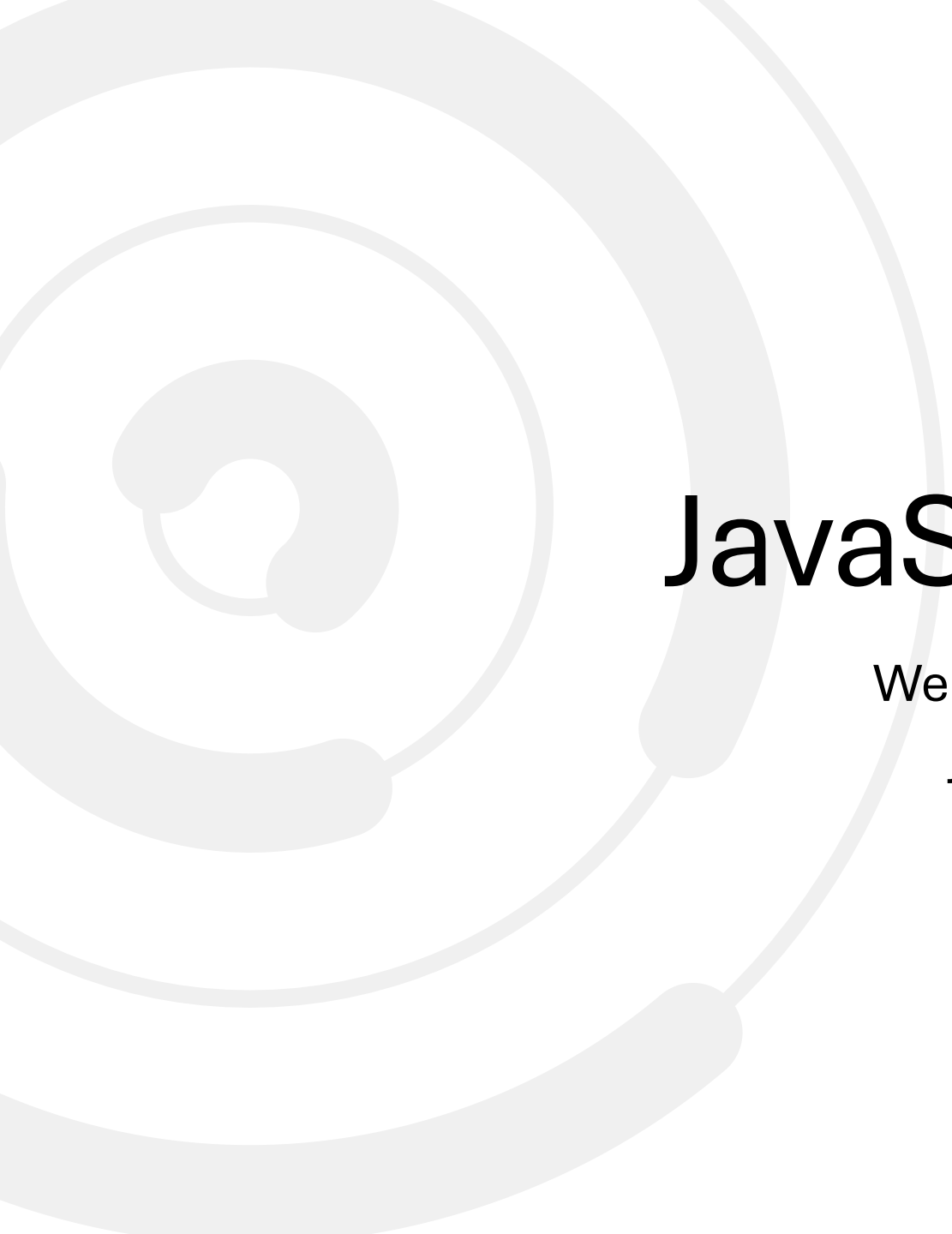


JavaScript, pt. 3

Web programiranje

Jurica Maltar

jQuery

- Open-source JavaScript biblioteka za pojednostavljeno upravljanje sadržajem
    - Upravljanje DOM-om
    - Upravljanje CSS-om
    - AJAX (asynchronous JavaScript and XML)

jQuery

- Tijek korištenja jQuery-a može opisati riječima “dohvati element i upravljaj te izvršavaj akcije nad njime”:

```
$(selector).action...
```

- jQuery dohvaćamo pomoću
  - cdn-a
  - direktno s web stranice i spremamo lokalno uz projekt
  - Putem npm-a (uskoro)

Hello World

```
$(document).ready(() => {
        // Here we are using jQuery
        console.log("I'm using jQuery");
});

// shorthand
$(() => {
        console.log("I'm using jQuery");
});
```

jQuery

- **Pitanje:** Kada ćemo korisiti minify-ane verzije jQuery-a (i generalno bilo koje druge biblioteke) i zašto?

- **Zadatak**: Integrirajte jQuery u projekt i ispišite “Hello World”

Odabir elemenata

```
let allElements = $("*");
let doc = $(document);
let allDivs = $("div");
let myElement = $("#myElement");
let allClassInstances = $(".myClass");
let specialParagraph = $("p#myElement");
let linksFromMenu = $("ul li a.nav");
let titlesAndParagraphs = $("h1, p");
```

Dodatno sužavanje odabira

```
// first of chosen
$("div.foo").first( );
// last of chosen
$("div.foo").last();
// div.foo elements that contain <p> tags
$("div.foo").has("p");
// h1 elements that don't have a class of bar
$("h1").not(".bar");
// unordered list items with a class current
$("ul li").filter(".current");
// the sixth
$("ul li").eq(5);
```

Događaji

```
$("#myButton").click((event) => {
        // when clicked do...
});

// alternative
$("#myButton").on("click", (event) => {
        // when clicked do...
});
```

Ulančavanje događaja

```
// it is possible to chain event listener
$("#myButton").click((event) => {
        // when clicked do...
}).mouseover((event)=>{
        // otherwise, do something else...
});

// alternative
$("#myButton").on("click mouseover", (event) => {
        // do the same for both click and mouseover
});
```

- **Zadatak**: Ustanovite o kojem se događaju radi koristeći `on` s `"click mouseover"` – ukoliko je pritisnut gumb, u HTML-u dodajte poruku “Pressed”, a ukoliko je prijeđeno mišem preko njega “Hovered”

Sadržaj elementa

- Općenita metoda za postavljanje/dohvaćanje vrijednosti atributa
  - `attr`
- Mnoštvo metoda za postavljanje/dohvaćanje (getter-a/setter-a) vrijednosti atributa
  - `val // get/set value of an input`
  - `html // get/set html content of an element`
  - `text // get/set text of an element`

- **Zadatak**: sakrijte paragraf nakon pritiska gumba koristeći `attr`
- **Zadatak**: ako je paragraf skriven, prikažite ga, a u suprotnom sakrijte
- **Zadatak**: prikažite u `alert` ono što ste upisali u `input` nakon što odznačite taj `input`
  - (proučite jQuery naredbu `blur`)
  - Koji je komplement za `blur`? Primjer korištenja tog komplementa?

JavaScript offtopic

- Sintaksa za stringove je `""` ili `''`
- U JavaScript ES6 standardu pojavljuje se tzv. template string ``` ``
  - Shortcut: Alt Gr + 7
  - Mogućnost definiranja u više redova

```
`
<ul>
    <li>Hello World</li>
    <li>Good Morning</li>
</ul>
`
```

  - Interpolacija

```
`I'm ${2025 - 1993} years old`
```

Dodavanje elemenata

- Stvaranje elemenata: (npr.) `let ul = $("<ul id='myList'></ul>");`
- Dodavanje na početak/kraj
  - `prepend`
  - `append`
- Dodavanje prije/nakon
  - `before`
  - `after`
- Uklanjanje elementa
  - `remove`
  - `empty`

- **Zadatak**: Dinamički popunite elemente liste `["tomato", "gold", "deepskyblue"]`, obojajte ih te dodajte naknadno jedan element prije zadnjeg

Obilazak

- Prethodnici
  - `parent`
  - `parents`
  - `parentsUntil`
- Sljedbenici
  - `children`
  - `find`
- Braća
  - `siblings`
  - `next`
  - `nextAll`
  - `nextUntil`
  - `prev`
  - `prevAll`
  - `prevUntil`

CSS klase i CSS stil

- Dodavanje klase
  - `addClass`
- Uklanjanje klase
  - `removeClass`
- Provjera postojanja klase
  - `hasClass`
- Toggle-anje klase
  - `toggleClass`
- CSS stil getter/setter
  - `css`

Odabir - razno

- Pozovi funkciju za svaki dohvaćeni element
    - `each`
- Pretvori odgovarajuće elemente u polje
    - `toArray`

Efekti i animacije

- Prikaži / sakrij element
  - `$(selector).show(speed, callback);`
  - `$(selector).hide(speed, callback);`
- Ako je skriven, prikaži, a u suprotnom sakrij element
  - `$(selector).toggle(speed, callback);`
- Slide
  - `$(selector).slideDown(speed, callback);`
  - `$(selector).slideUp(speed, callback);`
  - `$(selector).slideToggle(speed, callback);`
- Fade
  - `$(selector).fadeIn(speed, callback);`
  - `$(selector).fadeOut(speed, callback);`
  - `$(selector).fadeTo(speed, opacity, callback);`

Efekti i animacije

- Animacija
  - `$(selector).animate({ params }, [duration], [callback]);`

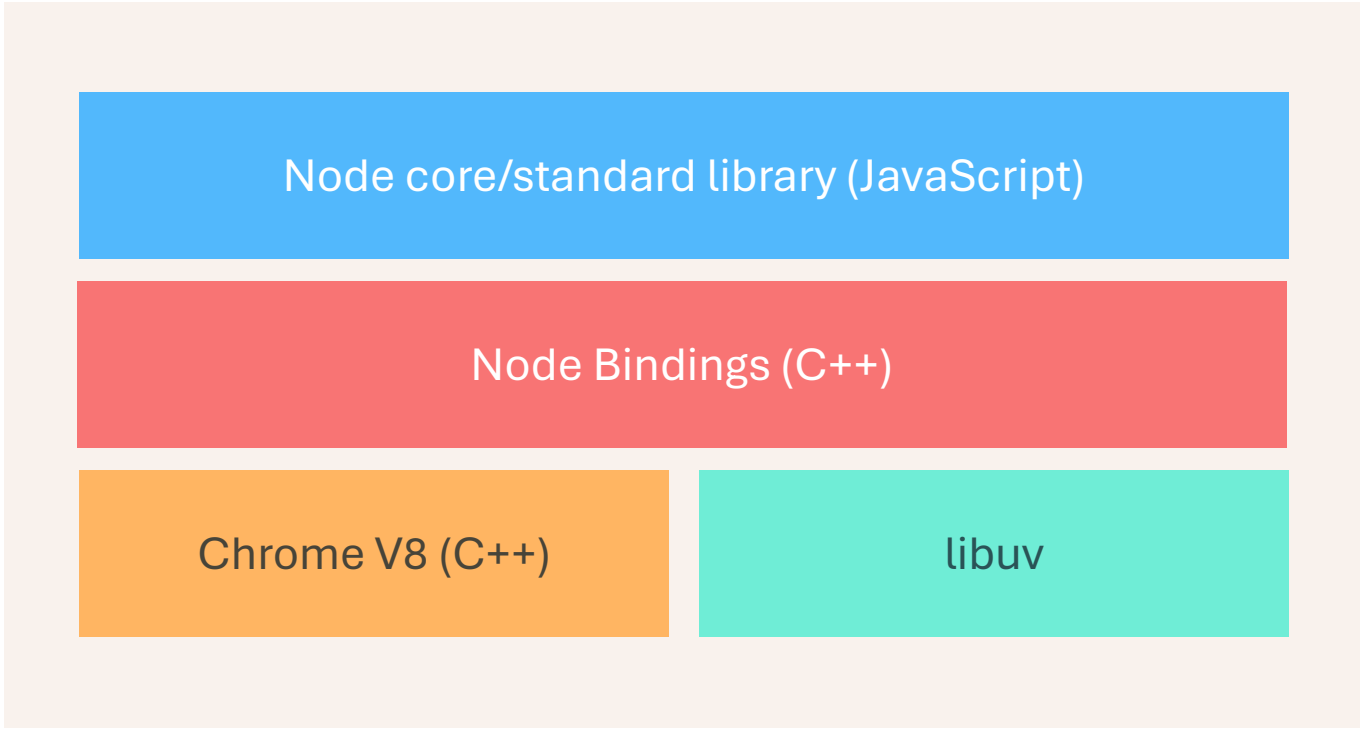
- **Zadatak**: Napravite animaciju kao u video-zapisu
  - Hint: Ako ne radi s `border-radius`, umjesto `border-radius` koristite `border-top-left-radius`, `border-top-right-radius`, `border-bottom-left-radius` i `border-bottom-right-radius`

Node.js

- Node.js je JavaScript runtime engine zasnovan na Chrome V8 engine-u koji omogućava izvršavanje JS-a izvan preglednika

- Node.js koristit ćemo za
  - izgradnju klijentske web aplikacije
  - implementaciju poslužitelja koristeći Express.js

Arhitektura Node.js-a



Za što se u pravilu koristi Node.js?

1. Programiranje poslužiteljskih strana web aplikacija
    - Web server, pristup bazi podataka
    - express.js, sails.js, nest.js...

Express

2. Alati za izgradnju web aplikacija (task runner-i)
    - bower, gulp, grunt, webpack...

Node.js

- Node.js možemo pokrenuti unutar naredbenog retka

- `node`
  - Naredba za pokretanje node.js-a (otvara se okruženje koje nalikuje okruženju u developerskog konzoli Chrome-a)
  - Node.js exe datoteka treba se nalaziti u putanji Windowsa (automatski se postavlja pri instalaciji)

- `node -v`
  - Verzija node-a

- `npm -v`
  - npm – “node package manager”
  - Verzija npm-a

Node.js

- Kod možemo pisati u node konzoli (nepraktično za “ne-oneline-re”)

- Kod možemo pisati u `.js` datoteku, a nakon što napišemo kod, pokrećemo ga s:
    - `node moja_datoteka.js`

Zadatak 1.

- Napišimo zajedno kod koji rješava kvadratnu jednadžbu $ax^2 + bx + c = 0$ za dane a, b, c
- Prisjetimo se:
  - $D = b^2 - 4ac$ (diskriminanta)
  - Za realne korijene: $a \neq 0, D \geq 0$

- $x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$

- Primjedba: za matematičke funkcije unutar JavaScripta koristimo `Math` biblioteku

Node.js moduli

- Kao i u slučaju drugih programskih jezika, kod JavaScript projekta možemo posložiti u više datoteka, odnosno modula.

- Export:

```
let myFunction = () => {
    console.log("Hello World");
}

module.exports = myFunction;
```

- Import:

```
let myFunction = require("./fun");
myFunction();
```

Node.js moduli

- Export:

```
let info = "Hello World";
let anotherFunction = (i) => {
    return ++i;
}
let list = [1, 2, 3, 4, 5];

module.exports = {info, anotherFunction};
```

- Import:

```
let anotherFunction = require('./something_else').anotherFunction;
let info = require('./something_else').info;
```

Node paketi

- https://www.npmjs.com/

- Pakete možemo pretražiti na gornjoj web adresi, guglati ili unutar cmd-a

- Kako bismo instalirali pakete za naš projekt, potrebno je inicijalizirati node direktorij:
  - `npm init`
  - Dalje slijedimo pravila...

Node paketi

- Unutar projekta stvara se odgovarajući `project.json` fajl
  - Naredbe za pokretanje koda
  - Dependency-i
  - Info o projektu
  - ...

- Pakete instaliramo naredbom `npm install`:
  - `npm install`
  - `npm install some-package some-other-package`
  - `npm install -g some-global-package`
  - `npm uninstall some-package`
  - …

- **Zadatak**: Koristeći npm instalirajte nodemon biblioteku

Node paketi

- Unutar git repozitorija ne commit-amo `node_modules` direktorij i ostale poddirektorije (`.gitignore`)
    - Kako clone-amo postojeći direktorij, naređujemo “`npm install`”

Zadatak 2.

- Kreirajte node projekt zvan “integrator”.
- Stvorite modul `integrate.js` te ondje smjestite funkciju koja računa integral, a prihvaća string `f` koji definira funciju koju integriramo, raspon od `x_from`, raspon do `x_to` te `dx`. Npr.

```
const x_from = -5;
const x_to = 5;
const dx = 0.001;
const f = "Math.sin(#x) + Math.pow(#x, 2)";
```

- Integrirajte neku matematičku funkciju u glavnom dijelu programa

- Primjedba: složene funkcije možete zadati preko stringa, a za takvo evaluiranje matematičkih izraza koristite funkciju `eval`

Express.js

- Jedan od najpopularnijih node paketa za izradu server aplikacija

- Koristeći Express.js kreiramo HTTP server koji će
  - Otvoriti HTTP konekciju i prisluškivati na odgovajućem portu
  - Posluživati statičke datoteke s diska
  - Odgovarati na HTTP zahtjeve (GET, POST, PUT, PATCH, DELETE)

Express.js – “Hello World”

```
const express = require("express");

let app = express();

app.get("/", (req, res) => {
    res.send("Hello World");
});

app.listen(3000, () => {
    console.log("Listen on port 3000");
});
```

Express.js

- Kao i na svakom poslužitelju, odgovaramo na zahtjeve i prosljeđujemo odgovarajuće datoteke
- Posluživanje datoteka:

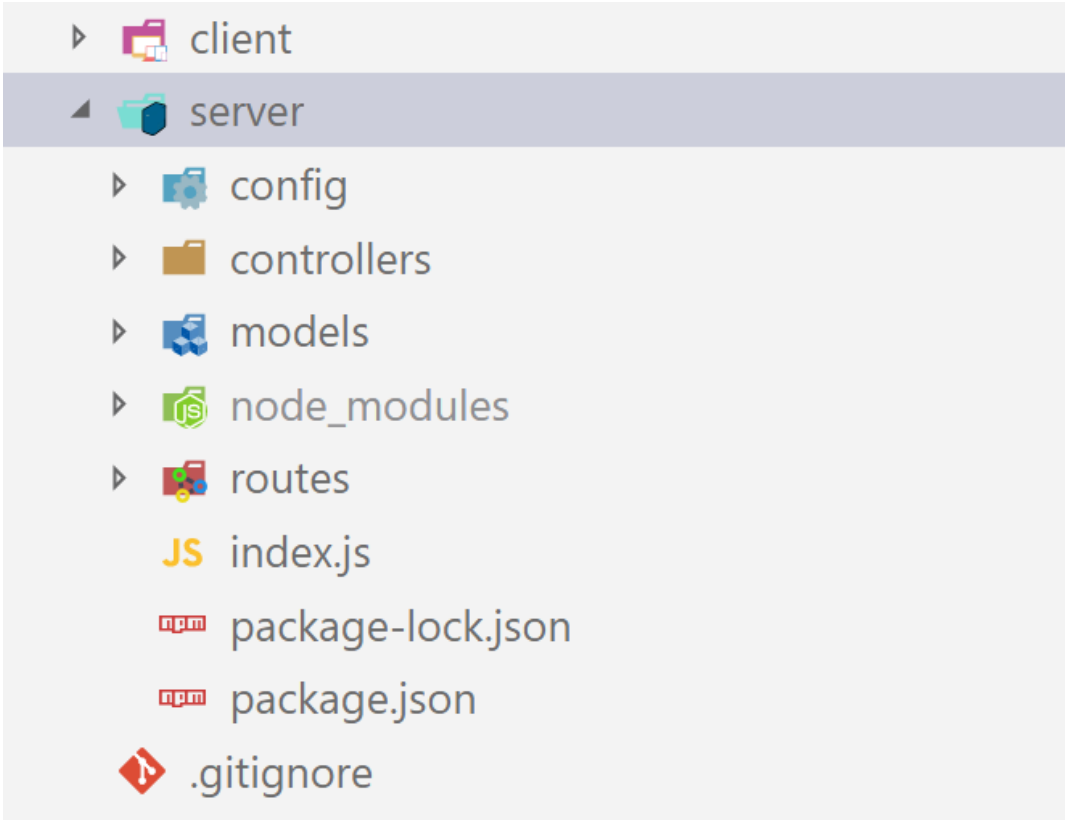
```
app.use(express.static(path.join(__dirname, "./www")));
```

- Definiranje ruta:

```
app.get(someRoute, someFunction);
```

Express.js

- Poslužiteljska aplikacija sadrži:
  - Povezivanje na bazu podataka (SQL/NOSQL)
  - Autentifikacija (JWT/passport)
  - API
  - Serviranje klijentske aplikacije
  - Slojevita arhitektura



AJAX

- Asynchronous JavaScript and XML

- Običnи JavaScript na klijentu: `fetch`
- u jQuery-u: `$.get`, `$.post` ili generalno `$.ajax`

- u Express.js-u: `app.get`, `app.post`, `app.delete`, `app.post`, `app.patch`