```
setTimeout( () => console.log(3) , 3000 ); // nakon 3s
let promise = new Promise( (resolve, reject) => {
            setTimeout( () => {
            console.log("nakon 3 sekunde...");
             if (false) {
                 resolve("dobro izvrsen");
              } else {
                reject("lose izvrsen");
            },
            3000);
});
promise.then(
  function(result) {console.log(result);},
  // ako resolve, result = "dobro izvrsen"
  function(error) {console.log(error);}
  // ako reject, result = "lose izvrsen"
);
promise.catch(
  function(error) {console.log(error);}// samo ako reject
)
promise.catch(
            function(error){console.log(error);}
).then(
            function(result){console.log("Resolve:"+result)},
            function(result){console.log("Reject:"+result)}
);     // catch ce uhvatit error, u then se ce pozvat prva funkcija s result = undefined
                                    //Fetch\\
let promise2 = fetch("https://web1lab2.azurewebsites.net/products?categoryId=1");
promise2.then(     // obraduje se promise od fetcha
            function(response) {
                  if (!response.ok) { throw new Error("Cannot load"); }
                  else { return response.json(); } // novo obecanje reponse.json()
            },
            function(error) { throw error; }
).then(          // obraduje se promise od response.json
            function(response) { console.log("Loaded JSON"); }
).catch(         // catch hvata gresku u bilo kojem promiseu
            function(error) { console.log(error); }
)
                                    //LoadJson\\
async function LoadJSON() {    // funkcija se izvodi asinkrono
    let promise = await fetch("https://web/categoryId=1");
      // unutar funkcije, await se izvodi sinkrono (ostatak funkcije ceka)
    if (!promise.ok) { throw new Error ("Cannot load"); }
    else { var jsonContents = await promise.json(); }
            console.log(jsonContents);
}
LoadJSON().catch(
            (error) => {console.log(error);}
)
            ******* prez  8 *********
GET{metoda} / predmet / rppzwpu HTTP{oznaka resursa}/1.1{oznaka protokola}
Host: www.fer.hr {ime posluzitelja}
HTTP - (hypermedia) prijenos u formatima -> html,meta-data,chunk,
Media Type -> text/html,image/jpeg,video/quickTime,application/javascript


                logo.png?
Browser ----------------------> www.fer.hr
                        Content-type image/jpeg
                        Content-length:1399
        <-------------------------


MME Type - (tip/podtip) -> application/javascript,  application/json, text/plain
Ciklus zahtjev-odgovor= jedna konverzacija
HTTPS port(443) --> HTTS --> TLS/SSL --> TCP --> IP
Uspostava komunikacije TLS - faza rukovanja(dogovor parametara),
            faza komunikacije(kljuc za sifriranje poruka)
Tijek komunikacije Server <-> Client
    <- salje zahtjev , -> odgovara certifikatom, <- provjerava certifikat, generira kljuc sjednice
    , salje kljuc sifriran javnim kljucem, -> desifrira kljuc sjednice, <-> koriste kljuc sjednice
```

```
Primjena HTTP - Cloud Computing, Rest, www, WOT
URI - uniformni (struktura zapisa)
            - identifikator (infromacija koja omogucuju razlikovanje resursa)
            - resurs (informacijski izvor)
URL,URN - podskup od URI
URN -jedinstvenost i trajnost  identifikacije
            pr. urn:ietf:rfc:2616
URL - sadrzi informaciju o lokaciji
            pr. http://www.ietf.org/rfc/rfc.txt
Primjeri URI-a (http://www.ietf.org/rfc/rfc2396.txt,mailto:John.Doe@example.com
                                        --> apsolutni (puno ime web adrese,www.fer.hr)
                                        --> relativni (skraceno, npr localhost)
            Analiza URI-a
http:{shema,nacin pristupa resursu(HTTP)}//www.fer.unizg.hr{host name(ip adresa ili ime)}/
|kako|                                  |gdje|
predmet/rppzwpu{put resursa}
    |sto se dohvaca|
shema:(http,ftp,urn,file)// <autoritet> <put {/predmet}> ? <upit {web=prag}> {put,upit isto
neobavezno}
pr. http://www.google.com:81/search?q=web{html#b3 -> skakanje po poglavljima}
<a href="../djelatnost/nastava/intstv.html>Internet stvari </a> (popni se na folder vise, spusti na
djelatnost/nastava/ , otvori intstv.html

                    Poruke HTTP
request                                 reply
Get /pred/web HTTP 1.0          HTTP 1.1 200 OK Pocetni redak
...                                 Content-type  Zaglavlja
...                                 ...
prazan                              redak
...                             <!Doctype html> <html>  tijelo poruke

                        Metode zahtjeva
GET - dohvacanje sadrzaja, HEAD- dohvacanje podataka o resursu(nema sadrzaja u tijelu za
razliku od GET)
, POST(sign up, comment,burza grupa),PUT,DELETE
HTTP reply - HTTP/1.1 {inacica protokola} 404 {kod} Not FOund {opis}
Kod - Informacija ->(100 Continue), Uspjeh -> ( 200 Ok), Preusmjeravanje -> (300 Multiple
Choices, 301 Moved, 302 Found, 304 Not Modified), Pogreska na klijentu -> (400 Bad Request,
401 Unauthorized, 403 Forbidden, 404 Not Found), Pogreska na posluzitelju->(500 Internal server
error, 503 Service not available, 505 http version not supported)
GET koristi link, POST body

opera       dns server      www.fer.hr      | Validacija - moze se povuci js koji radi validaciju da se
fer.hr?                                     | ne salje na server
------------------>                         | Cilj cache-a -> smanjiti odziv,internetski
<------------------                         | promet, opterecenje
GET /pred/web                               | Uvjetni GET -> IF-Match,IF-None-Match,If-Range
-------------------------------->          |
 HTTP 200 OK +index.html                    |
<--------------------------------          |
GET css                                     |
-------------------------------->          |
GET js                                      |
-------------------------------->          |
<--------------------------------          |
<--------------------------------          |
```

```
            ************ prez 9,10 ***************
Procesni modeli i protokoli
  -> in-process (opasno,ISAPI,Apache Server Api,low usage)
  -> poseban proc(sporo,CGI,low usage)
  -> poseban proc s pool-om(Fast CGI,PhP)
  -> proc s 2 dretve
  -> proc s pool-om
  -> vanjski proc s pool-om dretvi
Arhitekture
  browser <---> server <---> vanjski intrepeter(python)
  browser <---> server <---> aplikacijski server(Node.js)
  Event Loop -> ako je fja async stavlja se u queue sve dok se sve ne obradi

Versioning  -> patch  ~version 1.2.3 -> [1.2.3, 1.3.0>
            -> minor ^version 1.2.3 -> [1.2.3, 2.0.0>
            -> major *version
```



```
                Promises
let makePromise=function (x) {
  return new Promise(function (res, rej) {
    try {
      setTimeout(function () {
        res(x);
      }, 1000)
    } catch (err) {
      //handle err
    }
  })
}
let afAll = async function(){
  let sum=0;
  let res = await Promise.all([
    makePromise(getRandomBetw([1,5])).then(function (r1){
      sum+=r1;
      })
    .catch(function (err){
      //handle err
    }),
    makePromise(getRandomBetw([1,5])).then(function (r2){
      sum+=r2;
    }).catch(function (err){
      //handle err
    })
          ])
}
let sum=0;
makePromise(getRandomBetw([1,5]))
.then(function (r1){
  sum+=parseInt(r1);
  return makePromise(getRandomBetw([1,5]));
}).then(function (r3){
  sum+=parseInt(r3);
  console.log(`sum is ${sum}`);
}).catch(function (err){
  //handle err
})
let asyF = async function () {
  let r1 = await makePromise(getRandomBetw([1,5]));
  let r2 = await makePromise(getRandomBetw([1,5]));
  let r3 = await makePromise(getRandomBetw([1,5]));
  console.log(`${r1}+${r2}+${r3}=${r1+r2+r2}`);
}

 ************ prez 11, 12 *****************
<%= x %> -> x
<%- @x %> -> @x
validacija - provjera ispravnosti podataka
      (moze se provesti na: serveru, bazi,  klijentu )
    -> forma(disabled, maxlength, max, min, step)
    -> js (regex, neka fja)

Stanja -> na razini citavog sustava(globalno)
        -> na razini korisnika sustava(kosarica)
        -> na razini sjednice izmedu korisnika i sustava(login)
Tranzijetna pohrana -> nema trajnog cuvanja stanja
Prezistentna -> trajno cuvanje(pr. sustav i korisnik)

Sjednica -> slijed vremenski omeđenih i logički povezanih
            transakcija između pojedinog klijenta i poslužitelja
  1. pocetak sjednice(zahtijev klijenta prema serveru
     nakon duljeg vremena neaktivnosti)
  2. trajanje sjednice(logicki povezane transakcije
     izmedu klijenta i servera)
  3. zavrsetak sjednice(prestanak rada klijenta)

Identifikator sjednice(session token) -> odreduje sjednicu,
            dodan svakoj transakciji koja pripada sjednici

Prijenos sesssion tokena-a -> URI, header, body
```

```
Prijenost stanja -> hidden field, URL rewriting, cookies
Hidden field ->    <input name="naziv" type="hidden" value="SID=abc123">
   --> pros - podrzan na svakom browseru, ne moze se onemoguciti, performanse
   --> cons - vidljivi kod izvornog koda, prijenos kod svake transakcije, koristenje obrazaca
            //index.js\\
//implementacija session-a
router.use(session.sessionManager);
if(req.session.access_counter === undefined) // postavi access_counter koji smo izmislili
            req.session.access_counter = 0;
            //sessionFER\\
//session record store
let sessionStore = new Map();
            ***
//extract sessionID from GET or POST request
let sessionID = (req.query[sIDName] || req.body[sIDName]);
            ***
//fetch the session record
let sidRecord = sessionStore.get(sessionID);
            ***
if(!sidRecord) {
sidRecord = {id: uuid.v4(), created: Date.now()};
sessionStore.set(sidRecord.id, sidRecord)
//add the session record to the request object
req.session = sidRecord;
            ***
//pass the control to the next middleware layer
next();
Url rewriting -> mehanizam oznacavanja sjednica kada cookie nije dostupan
                (https://www.fer.unizg.hr/predmet/or?sid=234a3f0cc7)
        --> pros - neovisan o browseru, ne moze se onemoguciti na klijentu,jednostavan
        --> const - prijenos kroz URI, ogranicena kolicina, manja citljivost, dodatna funkcionalnost

//add sessionID parameter to URL query segment
return function(url) {
            let newURL = new URL(url)
            newURL.searchParams.append(sIDName, sessionID)
            return newURL.toString()
}
Cookies -> mala kolicina slobodno definiranih vrijednost, do 4kB
          -> stvara server, sprema klijent
            --> domena+put=doseg
            --> sadrzaj - ime=vrijednost,obvezno
            --> domena - ako nije definirano uzima se od servera, npr www.fer.unizg.hr/predmet/or
            --> put - ako nije definiran uzima se dio URI-a,fer.unizg.hr/nastava/or/labosi.html --> /nastava/or
            -->rok valjanosti, ogranicenje pristupa,sigurnost prosljedivanja(isto za druge domene) <-- opcionalno


                    GET /predmet/or
                    HOst www.fer.unizg.hr
Client---------------------------------------------------->
                    Set-cookie: sid=mileVOliDisko(sadrzaj);
                    Path=/nastava/or(put);Domain=www.fer.unizg.hr(domena);
                    Secure(sigurna veza);HttpOnly(nema lokalnog pristupa);
                    Expires: Wed, ...(istjece, moze i Max-age=3600)
<----------------------------------------------------Server
Uvjeti prosljedivanja cookie-a
  1. server pripada domeni (pr. www.fer.unizg.hr(*host-only),fer.unizg.hr,unizg.hr,hr ->da, carnet.hr->ne)
  2. sadrzan unutar puta (/nastava/or/labosi,nastava/or->da, nastava/oop-> ne)
  3. nije isteko rok trajanja, 4.ako je defirniran secure salje se kroz https(ne http)
  5. ako zabranimo, cookie nece bit proslijeden iz druge domene
                    GET /nastava/or
                    Host www.fer.unizg.hr
                    Cookie: sid=abc123
Client---------------------------------------------------->
                    HTTP/1.1 OK
                    Content-type: text/html
                    ...
<----------------------------------------------------Server
                    GET /intranet/or
                    Host www.fer.unizg.hr
                    Cookie: sid=abc123
Client---------------------------------------------------->
Trajni -> definiran rok valjanosti
```

SameSite
-> none (cookie se salje na drugu domenu)
-> strict(cookie preko druge domene se ne salje, link na nju ne radi)
-> lax (cookie preko druge domene se ne salje ali radi link na nju)

```
    //cookies.js\\
res.cookie(req.query.name,
        req.query.value, { path: req.query.path })
res.clearCookie(req.query.name,
        {path: req.query.path})
    //app.js\\
const cookieParser = require('cookie-parser')
//cookie parser middleware
app.use(cookieParser());
    //page.js\\
router.get('/*', function(req, res, next) {
res.render('page', {
path: req.path,
cookies: req.cookies
});
```

*******server.js*******

```
const express = require('express');
const app = express();
const path = require('path');
const pg = require('pg')
const db = require('./db')
const session = require('express-session')
const pgSession =
        require('connect-pg-simple')(session)
const router = require('./routes/router');
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
app.use(express.static
(path.join(__dirname, 'public')));
app.use(express.urlencoded({ extended: true }));
app.use(session({
   store: new pgSession({
      pool: db.pool,
   }),
   secret: 'fer-web-lab4',
   resave: false,
   saveUninitialized: true,
        cookie: {maxAge: 24 * 60 * 60 * 1000}
}));
app.use('/register', router);
app.listen(3000);
```

Opis modela : upisujemo mail i password i tako se registriramo, ispisujemo stare mailove i trenutnog, req.session.user = {email: req.body.email}; moglo se i req.session.user=req.body.email ali ovako mozemo dodat i neki dodatni parametar kojem mozemo pristupit preko ejs-a, npr username: req.body.username

************MI************************

```
<body>
   <p id="i1" class="c2"> Paragraf </p>
   <button onclick="decr()">-</button>
   <input id="cnt" type="text"
         value="5" readonly/>
</body>

function decr(){
   let cnt=document.getElbyid("cnt");
   cnt.value=Number(cnt.value)-1;
   let p = document.getElementById("i1");
   p.style.backgroundColor = "Red";
   p.innerHTML = "promjena teksta";
   let newP = document.createElement("p");
   body.appendChild(newP);
```

*******router.js*******

```
const express = require('express');
const router = express.Router();
const {body,validationResult} = require('express-validator');
const db = require('../db');
router.get('/', async function (req, res, next) {
   let rsp = await db.query('SELECT email FROM users');
            //router.get('/:id', function(req,res,next) {
            //id = parseInt(req.params.id)
            // rsp.rows[i] pristup elementima,
            // rsp.rows[i].atribut pristup atributu
// www.testovi.com/test/:1?a=b&c=d --> req.query.a -> b, req.query.c -> d
                                          //req.param.id -> 1
   res.render('register', {
      title: 'Register',
      err: undefined,
      users: rsp.rows,
      user: req.session.user
   });
});
router.post('/', [
body('email').trim().isEmail(),
body('pass').trim().isLength({ min:3, max:20 })
//body('employedsince').toInt().isInt({min:1970,max:2021}),
],
async function (req, res, next) {
const errors = validationResult(req);
if (!errors.isEmpty()) {
      res.render('register', {
            title: 'Register',
            err: "Invalid input!",
            users: [],
            user: req.session.user
      });
} else {
      try {
            await db.query('INSERT INTO
            users(email, password) VALUES ($1, $2)',
                  [req.body.email, req.body.pass]);
            req.session.user = {email: req.body.email};
            res.redirect('/register');
      } catch (err) {
            console.log(err);
            res.render('register', {
                  title: 'Register',
                  err: "Database error!",
                  users: [],
                  user: req.session.user
            });
      }
}
});
module.exports = router;
```

KOMUNIKACIJA S BAZOM PODATAKA
Driver održava bazen (pool) N unaprijed otvorenih konekcija s istim korisničkim računom-> CONNECTION POOLING
MVC – MODEL (UserMOdel s labosa)
- zaduženje: dohvat I manipulacija podacima
- tipično u suradnji s bazom podataka
- JS file
MVC – VIEW (register.ejs)
- EJS file – Embedded JavaScript Template
- zaduženje: prezentacija dostavljenih podataka
MVC – CONTROLLER (router.js)
- povezuje Model I View
- prima zahtjev, pribavlja model, predaje ga u view kako bi se generirao HTML koji se onda vraća

*******register.ejs*******

```
<html>
<head>
   <title> <%= title %> </title>
</head>
<body>
<%- include(`partials/header`); %>
<form action="/register" method="POST">
<fieldset>
      <legend>Registration data</legend>
      <div>
      <label for="email">Email:</label>
      <input type="text" name="email" id="email"
                        maxlength="20"
                        minlength="2" size="30">
      </div>
      <div>
      <label for="password">Password:</label>
      <input type="text" name="pass" id="password"
                        maxlength="20"
                        minlength="2" size="30">
      </div>
      <div>
      <input class="btn" type="submit" value="Submit">
      <input class="btn" type="reset" value="Reset">
      </div>
      <% if (err !== undefined) { %>
      <div>
                  <%= err %>
      </div>
      <% } %>
      </fieldset>
</form>
<div>
   Used emails:
   <% for (usr of users) { %>
      <%= usr.email %>
   <% } %>
</div>
<% if (user !== undefined) { %>
<div>
   This session:
   <%= user.email %>
</div>
<% } %>
</body>
</html>
```

-----------Dodatna Teorija-----------------------

*********Cookies**************

Sigurnost: informacije unutar kolačića su vidljive i čitljive kod prijenosa nesigurnim komunikacijskim kanalom, presretanje(krađa) kolačića, Cross-site scripting(Session cookie theft, Session hijacking), Cross-site request forgery, Cookie poisoning -> third-party cookies su problem privatnosti
Trajnost kolačića: Trajni (persistent) kolačići (imaju definirano vrijeme isteka, brišu se nakon roka isteka valjanosti) I Privremeni (transient) – session cookies (nemaju definirano vrijeme isteka valjanosti, najčešće imaju maxAge, brišu se nakon prestanka rada klijenta)
Brisanje kolačića – "ručno" (korisnik putem opcije u pregledniku), automatsko brisanje istekom roka valjanosti, promjenom sadržaja kolačića
Pristup kolačiću – udaljen (poslužitelj, mehanizmom prosljeđivanja kolačića putem HTTP odgovora) ili lokalni (klijent, programski)
- ako je postavljeno svojstvo HttpOnly nije moguć lokalni pristup kolačiću
- ograničenje prosljeđivanja samo na sigurne komunikacijske kanale (Secure)

HTML

<pre>	preformated
<!-- -->	comment
<div>	block okvir
 	line break
<span>	inline okvir
<a>	link (href= target= ...)
	unordered list
	ordered list (reversed start= type=[1AaIi])
	list item
<dl>	data list
<dt>	dl title
<dd>	dl description
<table>	
<tr>	table row
<th>	table heading (colspan= rowspan=)
<td>	table data (-||-)
<caption>	
<colgroup>	(span=)
<thead>	table head
<tbody>	table body
<tfoot>	table foot
<form>	formular (method= action=)
<input/>	(disabled required name= pattern= value= type=[1]..)
<textarea>	(-||-*)
<button>	(-||-*)
<select>	(-||-*)
<option>	(value= -||-*)
<optgroup>	(label= disabled)
<fieldset>	(disabled ...)
<legend>	caption za fieldset
<label>	(for= ...)
<output>	(for= name= ...)
<meta/>	
<title>	
<style>	
<link>	(href= rel=(stylesheet))
<b>	bold
<i>	italic
<u>	underline
<p>	paragraph
<hX>	header teksta
	horizontal rule
<figure>	
<figcaption>	
<img/>	(src= alt= ...)
atrinut data-*	

```
<!DOCTYPE html>
<html>
<head>
    <title>Flower power!</title>
    <meta http-equiv="content-
type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-
width, initial-scale=1">
</head>
```

[1] button checkbox color email hidden date file image month number password radiorange reser submit text ...

CSS

selektori

*	sve
h1	<h1>
#x	id=x
.c	class=c
?, ?	primijeni na oba
h1[attr=?]	<h1 attr=?>
div span	unutar
div>span	dijete, neposredno
div+span	prvi sibling
div~span	svi sibling
:hover :focus :required :visited :nth-child() :checked	
::first-line ::first-letter ::before ::after ::selection	

specifičnosti:

inline	1000
#id	100
.class : [attr]	10
<tag> ::	1
<body> *	0

font

font-family	
font-weight	normal bold bolder lighter
font-style	normal italic oblique
font-size	medium small large smaller larger
text-align text-indent letter-spacing line-height ...	

position: static absolute fixed relative

flex

container	item
display: flex	order ?
flex-direction: (row|column)[-reverse]	flex-grow: n
flex-wrap: [no]wrap	flex-shrink: n
flex-flow=-direction+-wrap	flex-basis ?
justify-content: flex-(start|end) center space-(between|around|evenly)	flex=-grow+-shrink+-basis
align-items: flex-(start|end) center strech baseline	align-self: -||- od ← al it
align-content: -||- od jus	

boje: rgb() rgba() ##rrggbb $rgb
box model: content padding border margin
box-sizing: border-box

grid

container	item
display: grid	grid-(row|column): n [/ span m]
-template-rows: (fr)	grid-area=-row±column
-(row|column)-gap	
justify-items: start end center strech	justify-self: -||- ←
align-items: start end center strech	align-self: -||- ←
place-items: al. + just.	place-self: al. + just.
justify-content: (start|end) center strech space-(between|around|evenly)	
align-content: -||-	
place-content	

```
@media [not|only] mediatype and (mediafeature and|or|not
mediafeature) {
  CSS-Code;
}
```

mediatype: all print screen speech

initial inherit unset

JS

Array

.push() .pop()	
.slice(a, b)	[a:b]
.splice(a, k)	del [a:a+k]
.shift()	del [0]
.unshift(items)	.appendleft(items)
.concat(arr)	
.includes(e)	
.sort(?cmp)	
.reverse()	
.filter(x => bool)	

Map
.set(k, v) .get(k) .has(k) .delete(k) .clear() .size()
.keys() .values() .entries()

localStrorage
.setItem(k, v) .getItem(k) .removeItem(k) .clear() .length

```
Promise( (resolve, reject) => {
    ...
    if yes then resolve() else reject()
})
.then( (result => any), (result => any) )
.catch( error => any )
.state: pending fulfilled rejected
```