

Razvoj programske podpore za web i pokretne uređaje

- predavanja -
2021./2022.

09. Dinamički web

1/4

Creative Commons



- slobodno smijete:

- **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
- **prerađivati** djelo



- pod sljedećim uvjetima:

- **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
- **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
- **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.



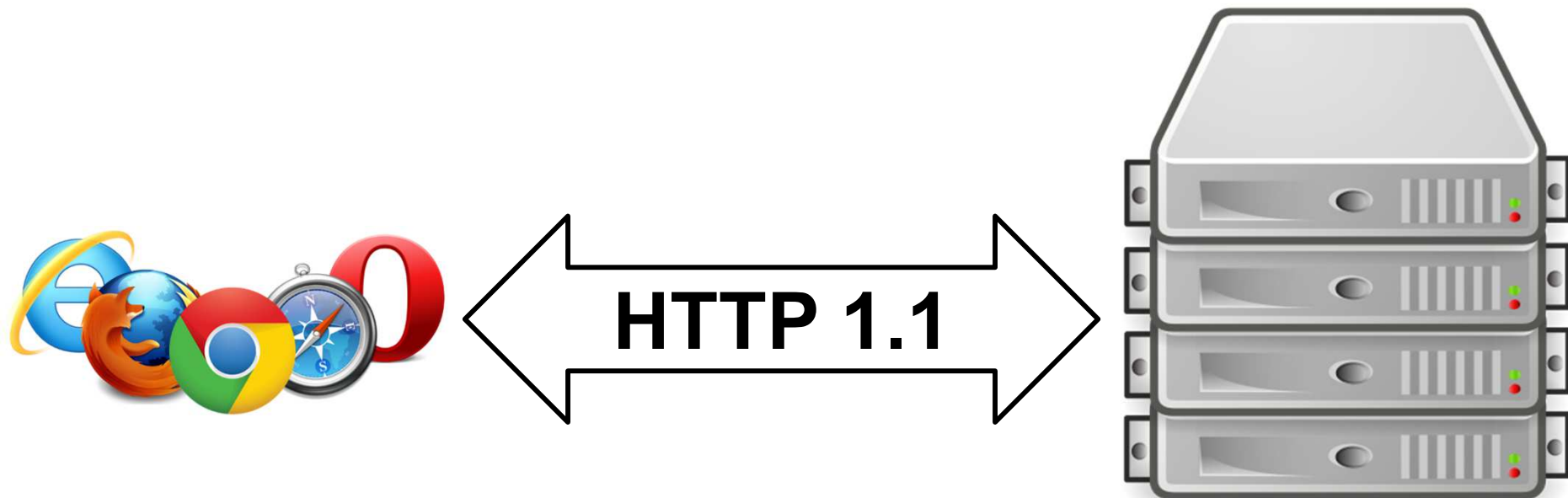
U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.

Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.

Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licence preuzet je s <http://creativecommons.org/>

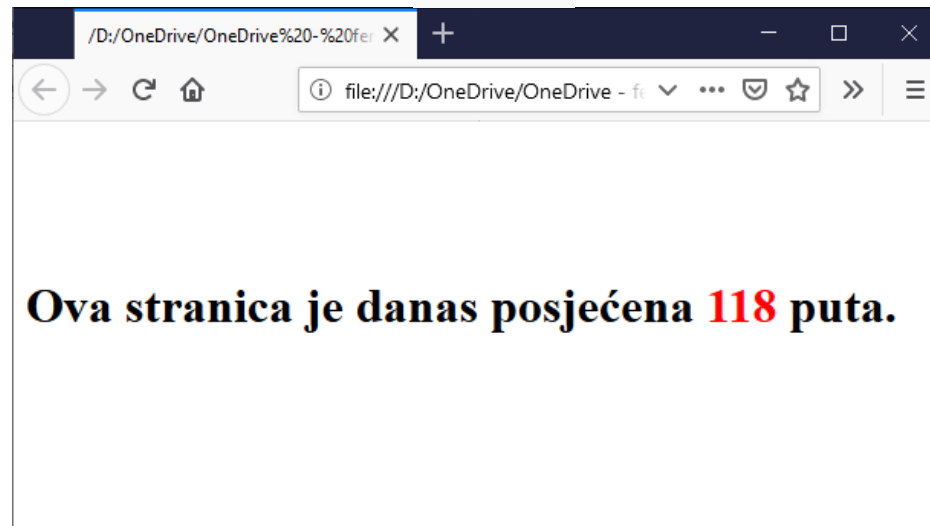
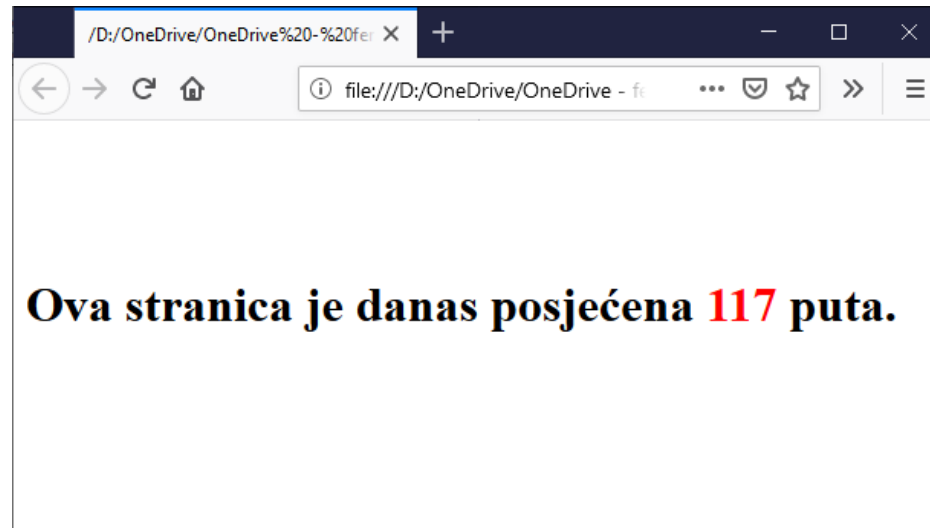
■ Znamo:



- Klijent
 - HTML
 - CSS
 - Javascript

- Poslužitelj
 - ?

Motivacija: kako bi napravili?

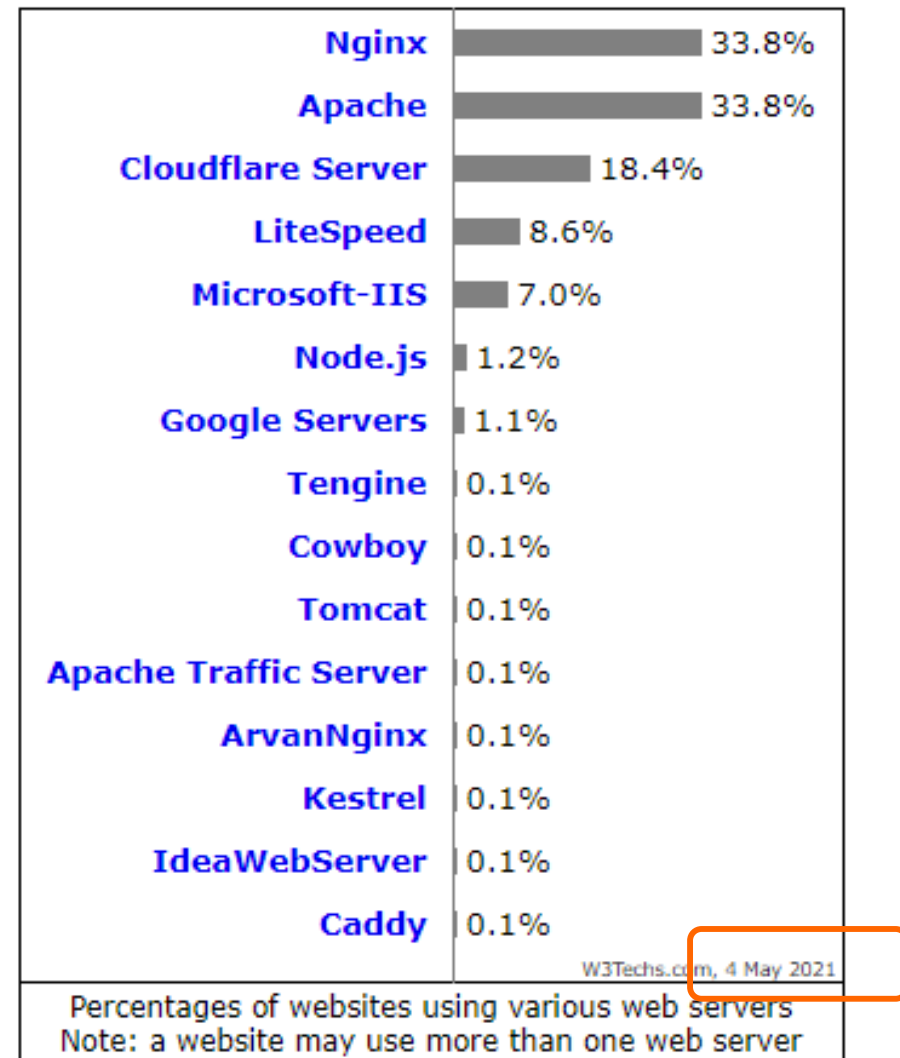
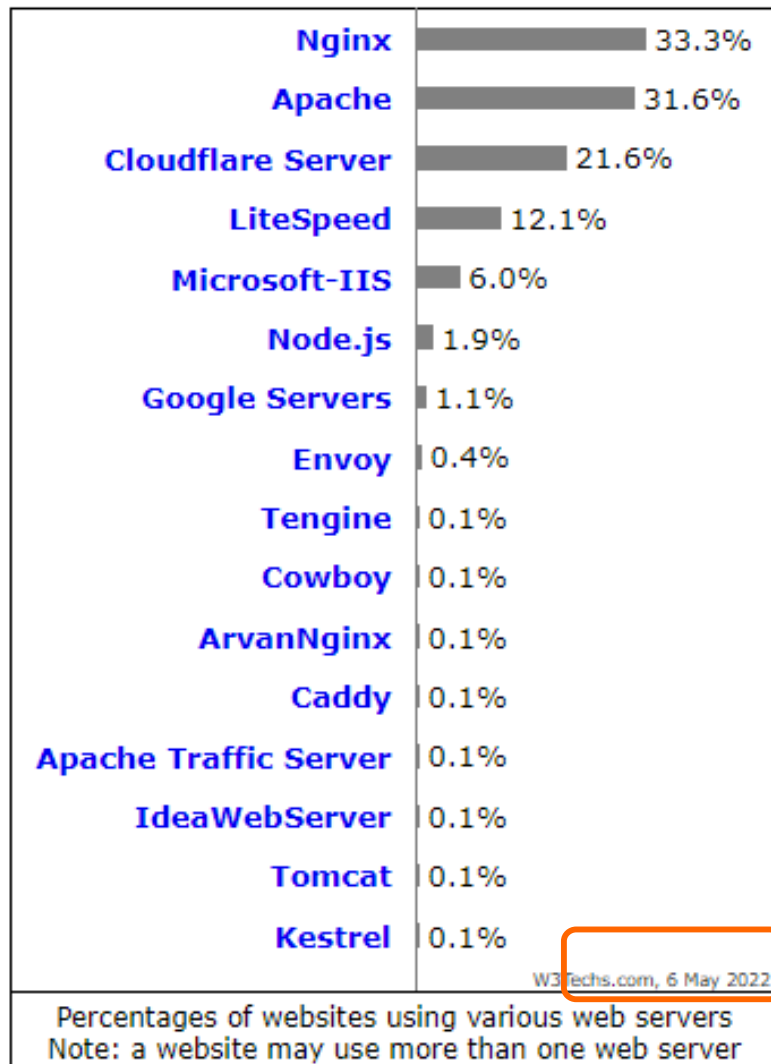


Dinamički web

- Motivacijski primjer će očigledno zahtijevati programiranje **na poslužiteljskoj strani** (*server-side*)
- Dinamički web - (dinamičko) generiranje prilagođenog HTML sadržaja na zahtjev
- Povezana terminologija na eng:
 - *Dynamic websites*
 - *Server-side programming*
 - *Backend*
- Prije dinamičkog sadržaja, pogledajmo detaljnije kako poslužiti **statički** sadržaj
 - (U 12. predavanju ćemo napraviti rudimentarni web poslužitelj u Javi!)
 - Zasad pogledajmo „gotove opcije”

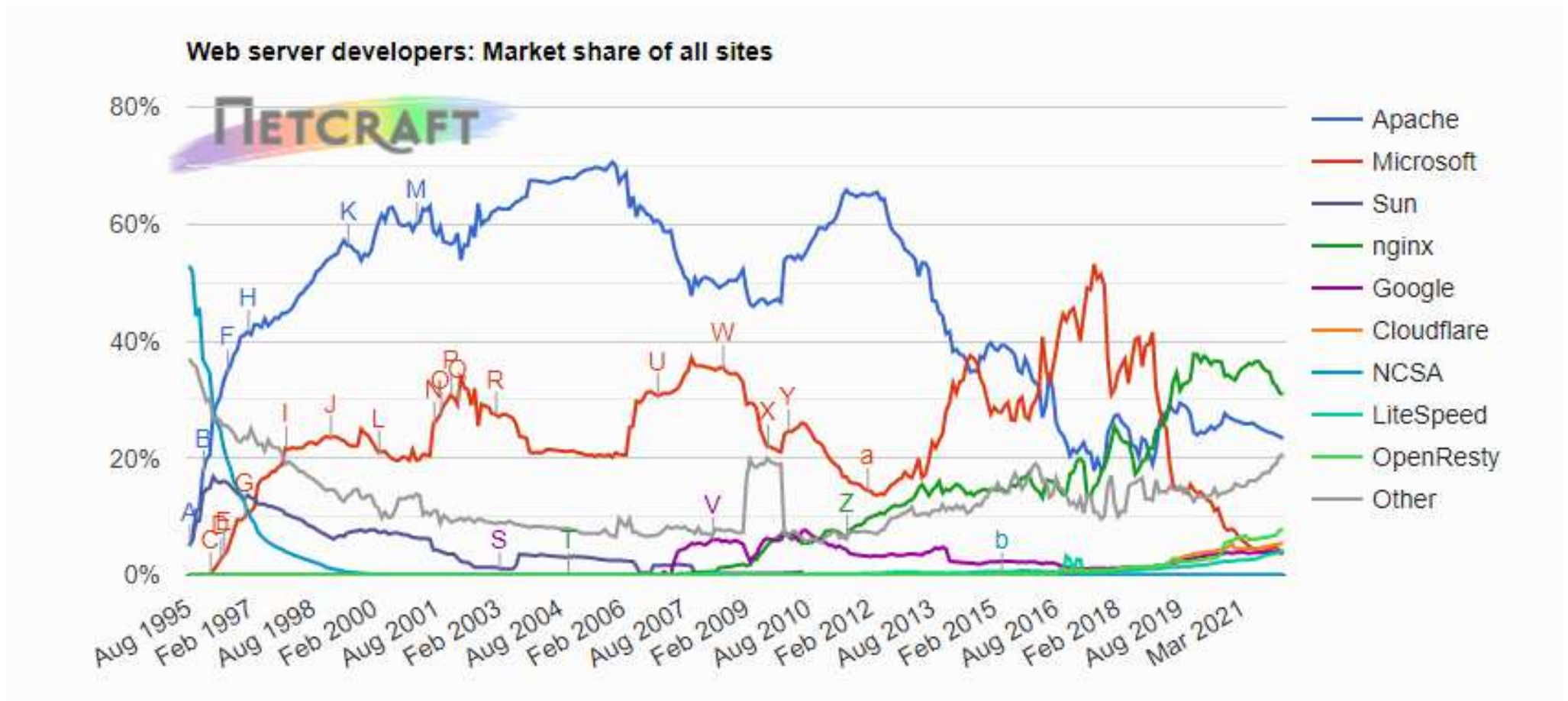
Web poslužitelji

- Udio na tržištu "danas" (https://w3techs.com/technologies/overview/web_server):



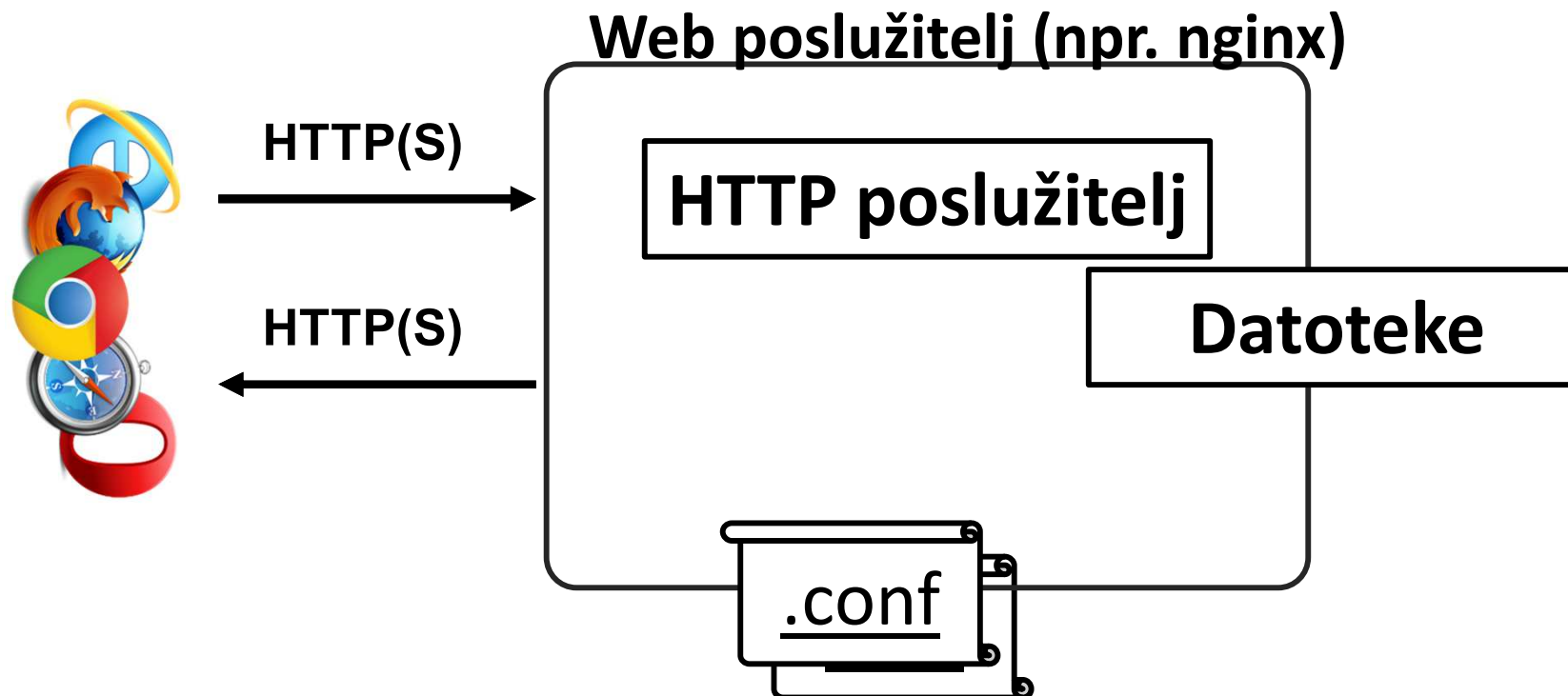
Web poslužitelji

- Udio na tržištu (<https://news.netcraft.com/archives/category/web-server-survey/>) u vremenu:



■ Onda statički sadržaj možemo poslužiti npr.:

- Instaliramo web poslužitelj, npr. nginx
- Konfiguriramo ga
 - Konfiguracije poslužitelja se tipično obavljaju izmjenama konfiguracijskih datoteka i ponovnim pokretanjem servisa (ili ponovnim učitavanjem konfiguracije)



■ Npr. dio konfiguracijske datoteke nginxa...

- Preuzeto s: http://nginx.org/en/docs/beginners_guide.html

Serving Static Content

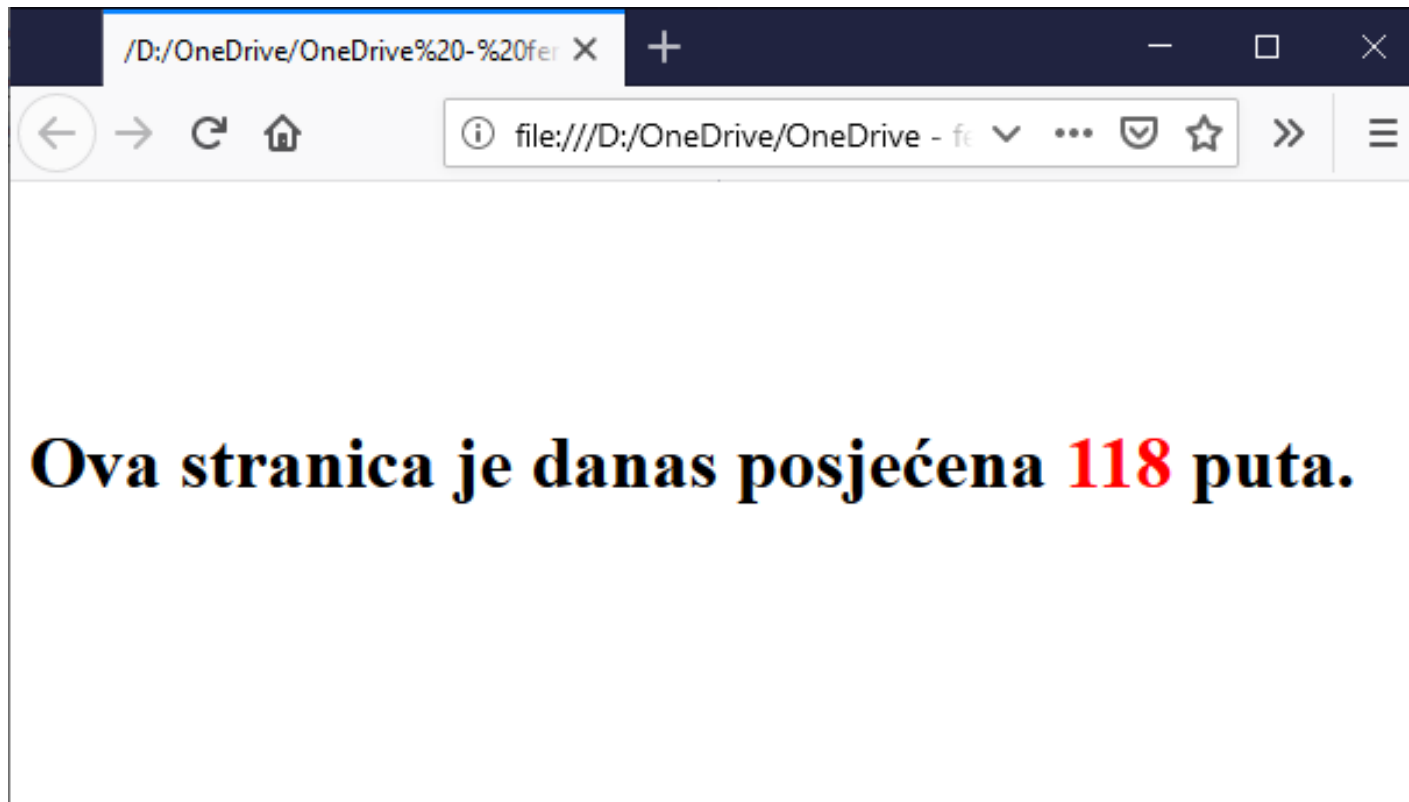
An important web server task is serving out files (such as images or static HTML pages). You will implement an example where, depending on the request, files will be served from different local directories: `/data/www` (which may contain HTML files) and `/data/images` (containing images). This will require editing of the configuration file and setting up of a [server](#) block inside the [http](#) block with two [location](#) blocks.

First, create the `/data/www` directory and put an `index.html` file with any text content into it and create the `/data/images` directory and place some images in it.

```
server {  
    location / {  
        root /data/www;  
    }  
  
    location /images/ {  
        root /data;  
    }  
    ...  
}
```

■ Vratimo se na motivacijski primjer...

- Znamo poslužiti statički sadržaj, ali kako:



Napravimo mali program index.py koji broji pozive:

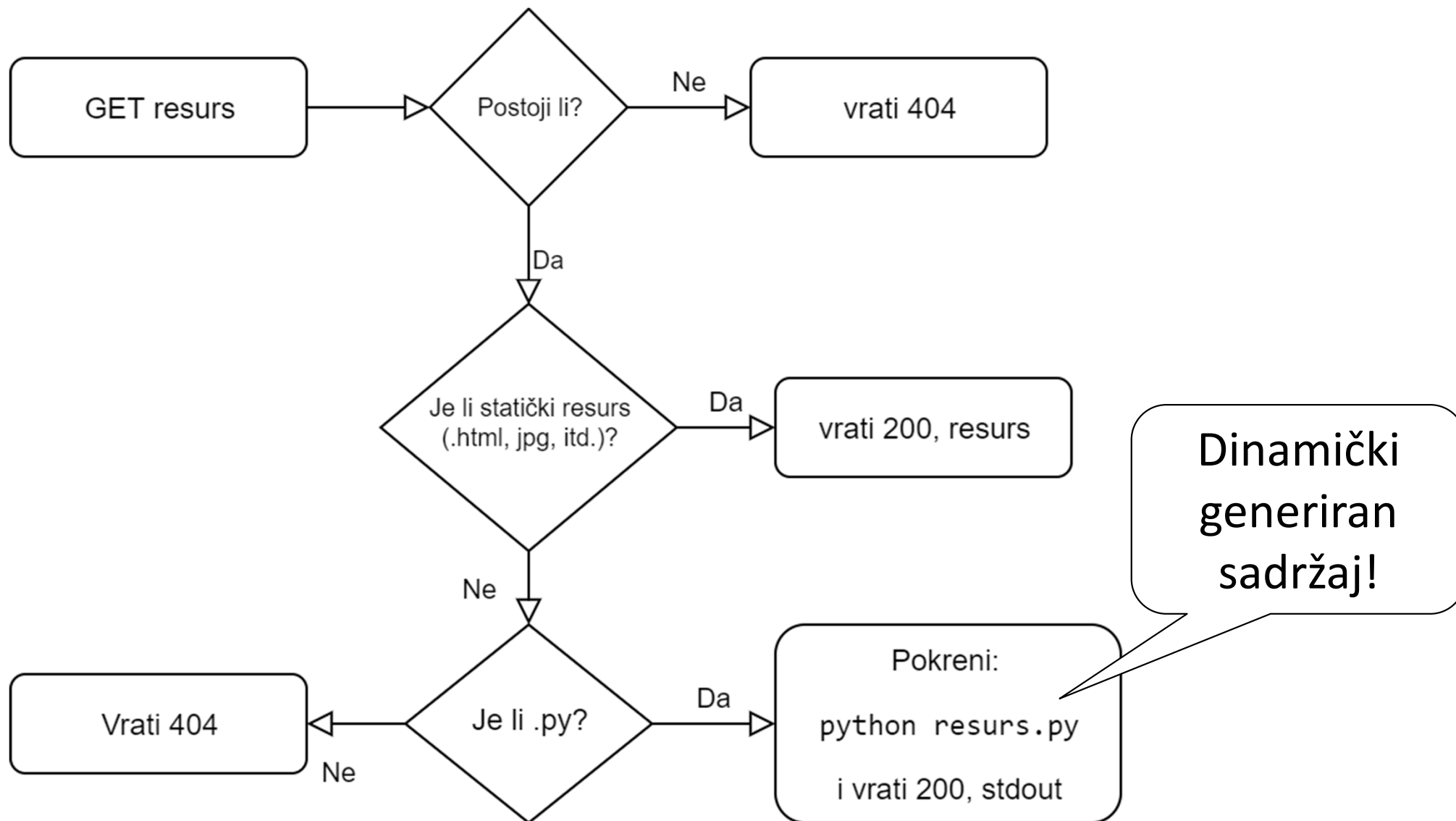
```
from datetime import date
today = date.today()
fc = today.strftime('%Y%m%d') + ".counter"
num = 0
try:
    f = open(fc, "rt")
    num = int(f.readline())
    f.close()
except IOError:
    pass
except:
    print("Other error")
num = num + 1
f = open(fc, "wt")
f.write(str(num) + "\n")
f.close()
print ("Ova stranica je posjecena " + str(num) + " puta.")
```

```
...>python index.py
Ova stranica je posjecena 1
puta.
```

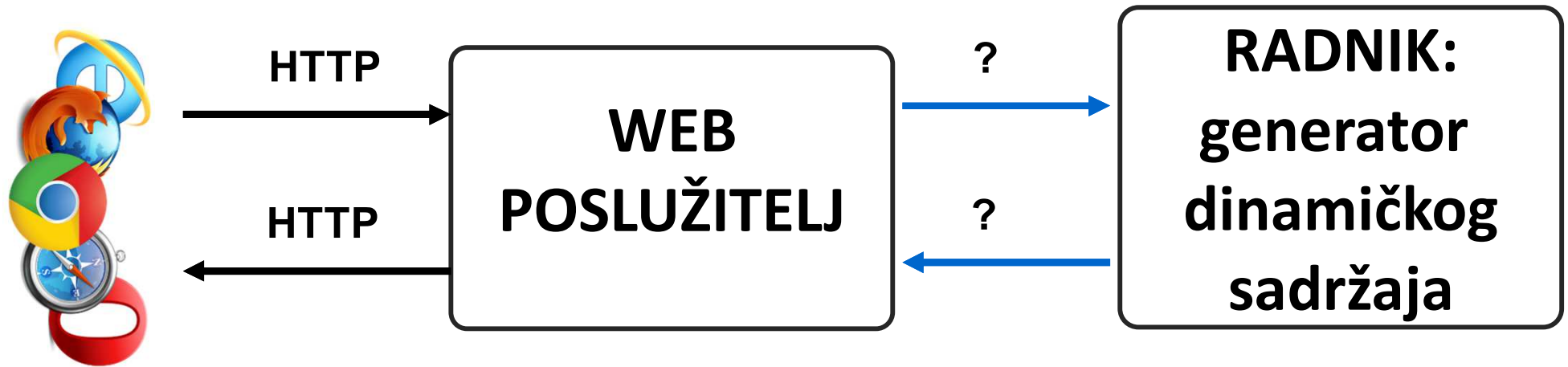
```
...>python index.py
Ova stranica je posjecena 2
puta.
```

```
...>python index.py
Ova stranica je posjecena 3
puta.
```

Konačno, mogli bi modificirati naš poslužitelj:

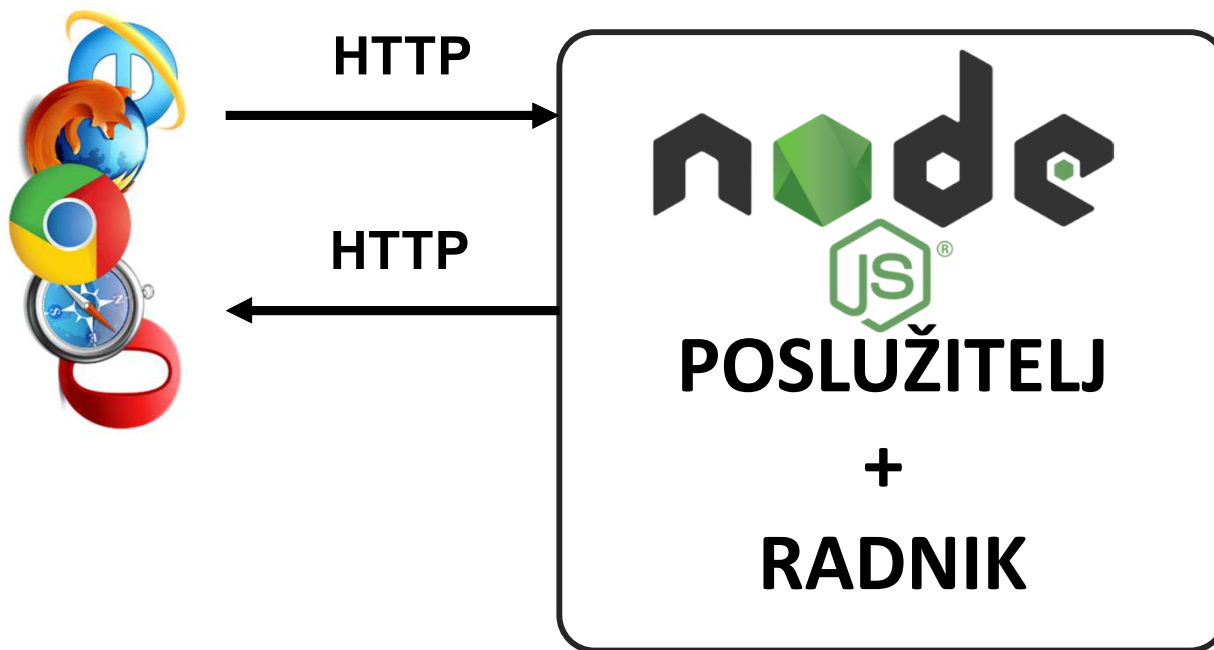


■ Načelno, tako se generira dinamički sadržaj:



- O ovim pitanjima više u 12. predavanju:
 - Kako WP komunicira s radnikom (*worker*)?
 - Procesni modeli: što je, kako nastaje i kakav je životni ciklus radnika?
 - Kako nastaje dinamički sadržaj?
 - Interpretiranjem skripti (PHP, Perl, Ruby, Python)
 - Unaprijed prevedenim programima (JSP, Asp.Net, itd.)

■ Mi ćemo sada nastaviti s:

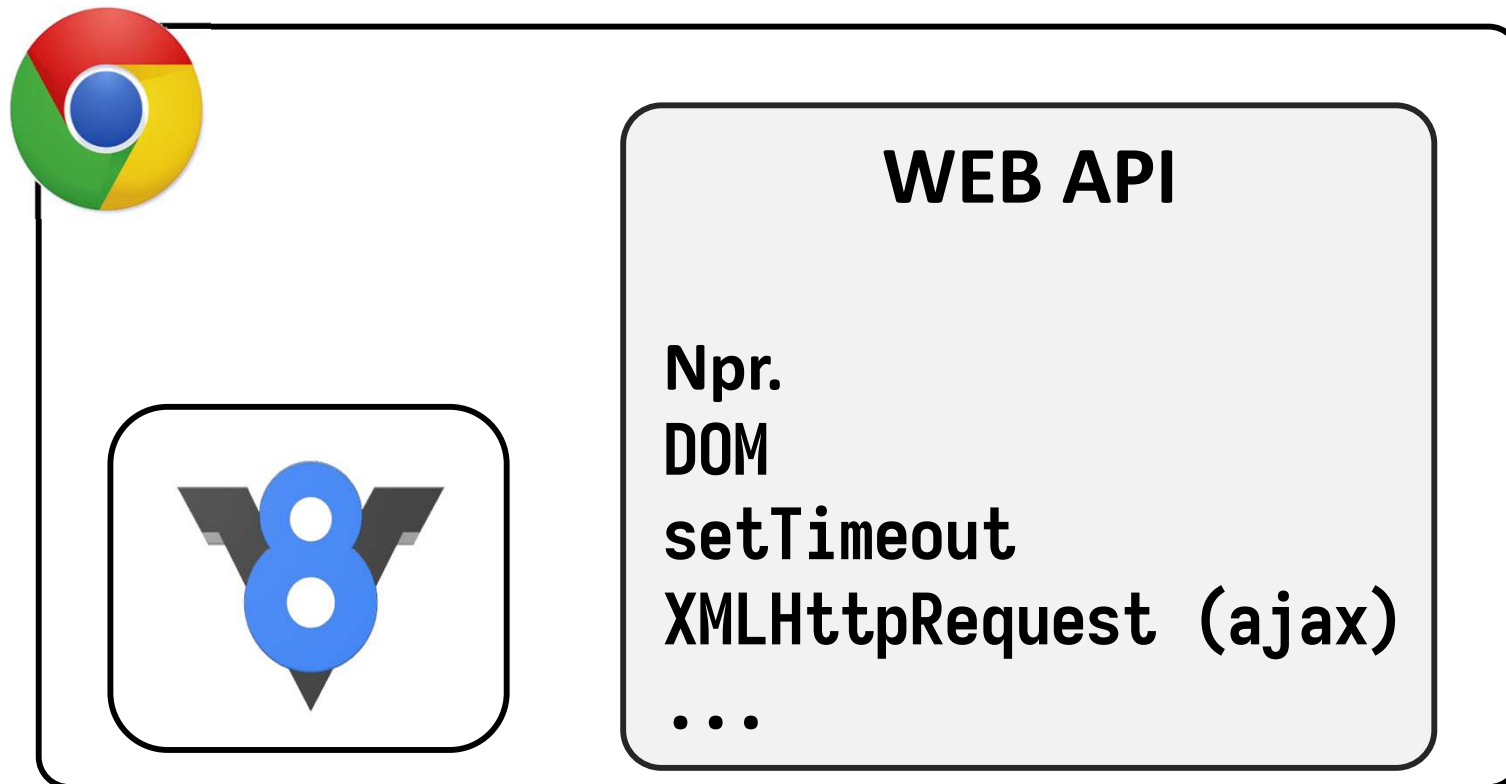


- S node.js-om se mogu ostvariti i drugačije konfiguracije, što se u produkciji tipično i radi...



Google V8

- *"V8 is Google's open source high-performance JavaScript and WebAssembly engine, written in C++."*
- Prva javna verzija 2008. godine



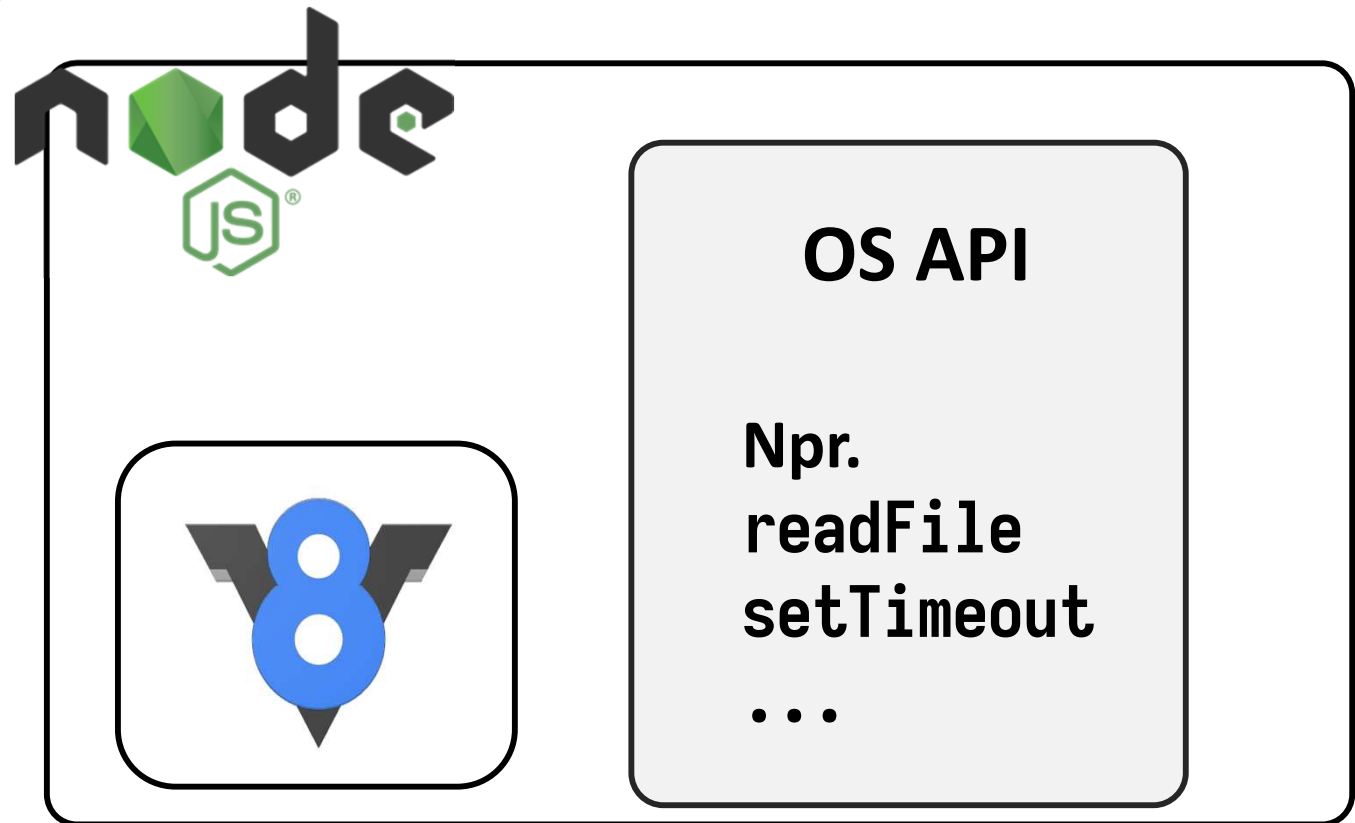
- **Prije 2011.: JS je ograničen na obavljanje u pregledniku**



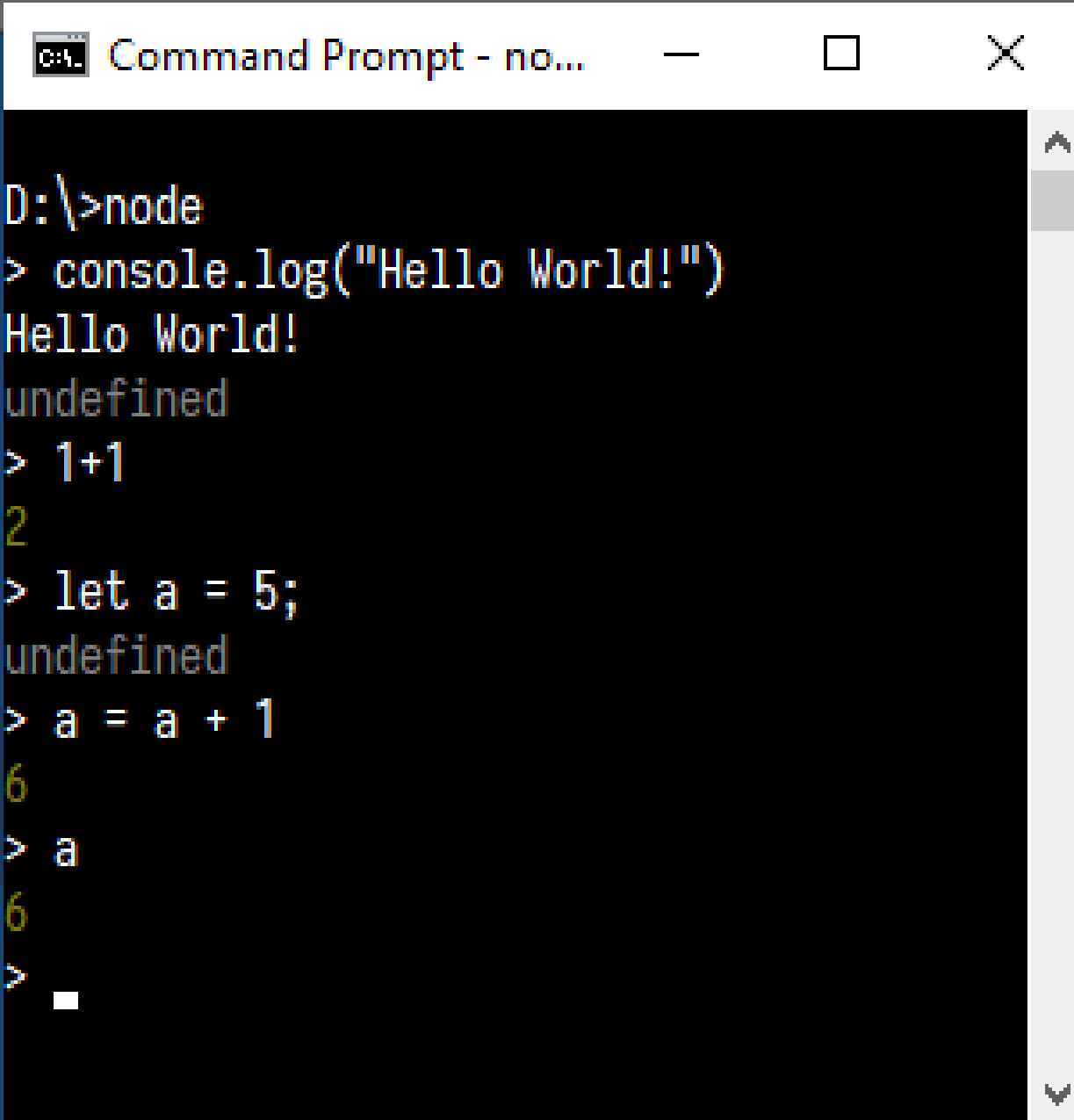
- **Ryan Dahl**

- 2009. demo, 2013. prva verzija v0.x
- *"Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient."*

- Sada se JS može obavljati **IZVAN preglednika** čime postaje općenamjenski programski jezik!



■ Primjer: REPL



```
D:\>node
> console.log("Hello World!")
Hello World!
undefined
> 1+1
2
> let a = 5;
undefined
> a = a + 1
6
> a
6
>
```

Primjer: kôd u datoteci

pr01.s19.js

```
for (let i = 0; i < 10; ++i) {  
    console.log(i + ". Hello World!");  
}
```

C:\Windows\System32\cmd.exe

0. DYNWEB-2>node pr01.s07.js

ello Worl d!

ello Worl d!

ello Worl d!

ello Worl d!

ello Worl d!

ello Worl d!

ello Worl d!

ello Worl d!

ello Worl d!

ello Worl d!

0. DYNWEB-2>_

Primjer: asinkrone operacije

```
console.log("1");  
setTimeout(() => {  
    console.log("2");  
}, 1000);  
console.log("3");
```

```
console.log("A");  
setTimeout(() => {  
    console.log("B");  
}, 0);  
console.log("C");
```

C:\Windows\System32\cmd.exe

```
X:\10. DYNWEB-2>node pr02.s08.js  
1  
3  
A  
C  
B  
2  
X:\10. DYNWEB-2>
```

Zašto je B iza C?

Petlja događaja: *Event Loop* ... ili "kako radi Javascript"?

```
console.log("A");  
setTimeout(() => {  
    console.log("B");  
}, 0);  
console.log("C");
```

A

stog(*stack*)

console.log("A");

webapi 
C++api 



Petlja događaja
(*event loop*)

red događaja (task queue)

Petlja događaja: *Event Loop* ... ili "kako radi Javascript"?



```
console.log("A");  
setTimeout(() => {  
  console.log("B");  
}, 0);  
console.log("C");
```

A

stog(*stack*)

```
setTimeout(() => {  
  console.log("B");  
}, 0);
```

async

webapi 
C++api 

 cllbck



Petlja događaja
(*event loop*)

red događaja (task queue)

cllbck


Petlja događaja: *Event Loop* ... ili "kako radi Javascript"?

```
console.log("A");  
setTimeout(() => {  
    console.log("B");  
}, 0);  
console.log("C");
```

A
C

stog(*stack*)

console.log("C");

webapi 
C++api 



Petlja događaja
(*event loop*)

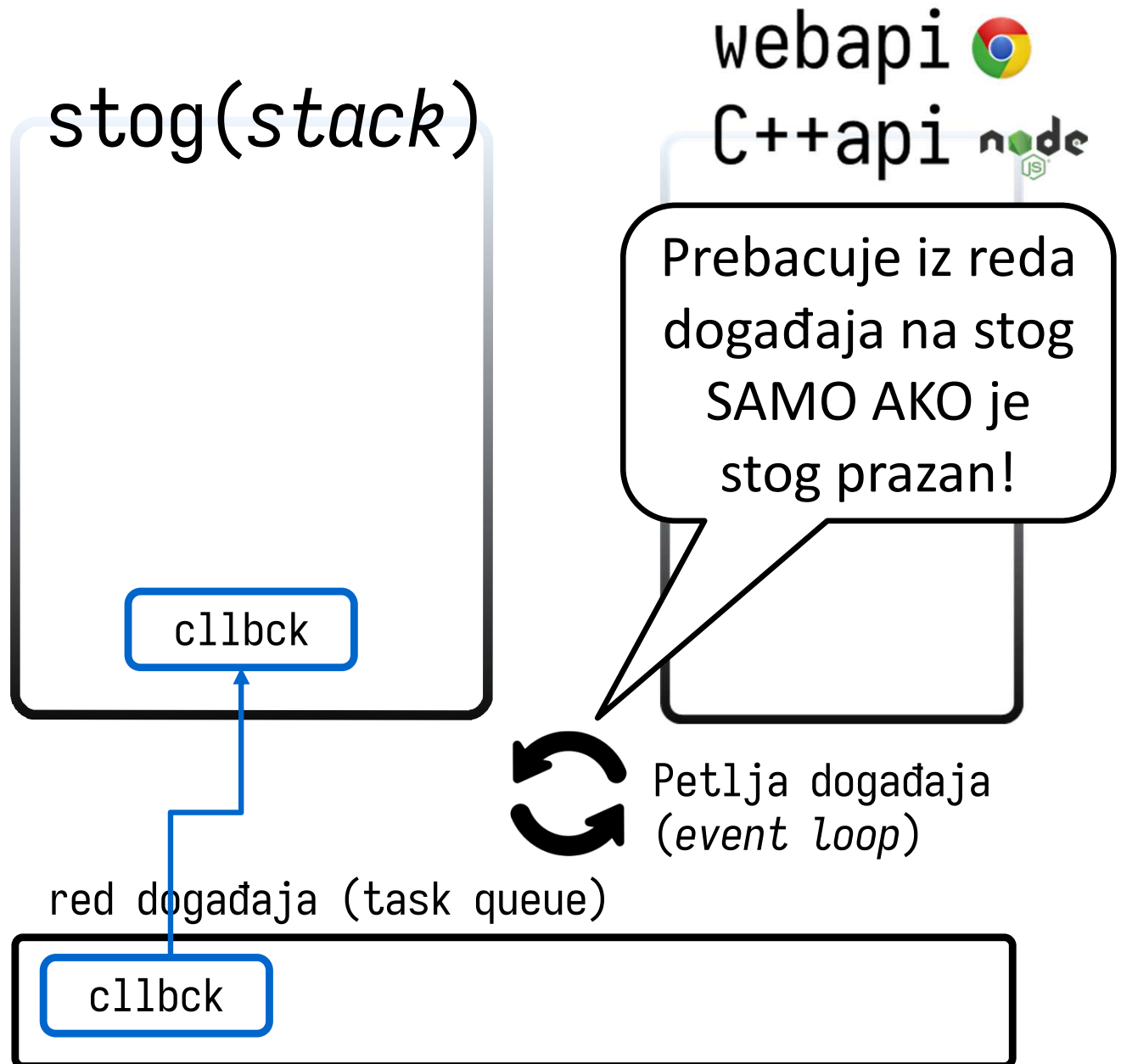
red događaja (task queue)

callback

Petlja događaja: *Event Loop* ... ili "kako radi Javascript"?

```
console.log("A");  
setTimeout(() => {  
    console.log("B");  
}, 0);  
console.log("C");
```

A
C
B



Što se obavlja asinkrono?

- "... *non-blocking I/O model*..."
- Prije svega **IO operacije** (<https://github.com/libuv/libuv>)
 - Dohvat podataka preko mreže
 - Čitanje i pisanje na disk
 - Itd.
- Ovisi i o operacijskom sustavu
- Ovisi i što programer koristi, node često nudi i dvije verzije funkcija, sinkrone i asinkrone, npr:
 - `fs.readFile(path[, options], callback)` <https://nodejs.org/api/fs.html>
 - `fs.readFileSync(path[, options])`
 - `crypto.pbkdf2(password, salt, iterations, keylen, digest, callback)`
 - `crypto.pbkdf2Sync(password, salt, iterations, keylen, digest)`
 - ...



npm - Node Package Manager

- Omogućuje upravljanje dodatnim programskim paketima (bibliotekama)
- `npm init` -> postavlja pitanja i generira `package.json`
 - Datoteka koja opisuje naš projekt (kako ga pokrenuti, koje dodatne pakete koristi, autor, verzija, itd.)
- `npm install ime_paketa`
 - Npr.:

A screenshot of a Windows command prompt window titled 'C:\Windows\System32\cmd.exe'. The prompt shows the command 'X:\10. DYNWEB-2\pr03.s14>npm install qrcode' being executed. The output includes a warning 'npm WARN qr_demo@1.0.0 No repository field.', followed by '+ qrcode@1.4.4', 'added 37 packages from 28 contributors and audited 37 packages in 3.149s', and 'found 0 vulnerabilities'. The prompt then returns to 'X:\10. DYNWEB-2\pr03.s14>'.

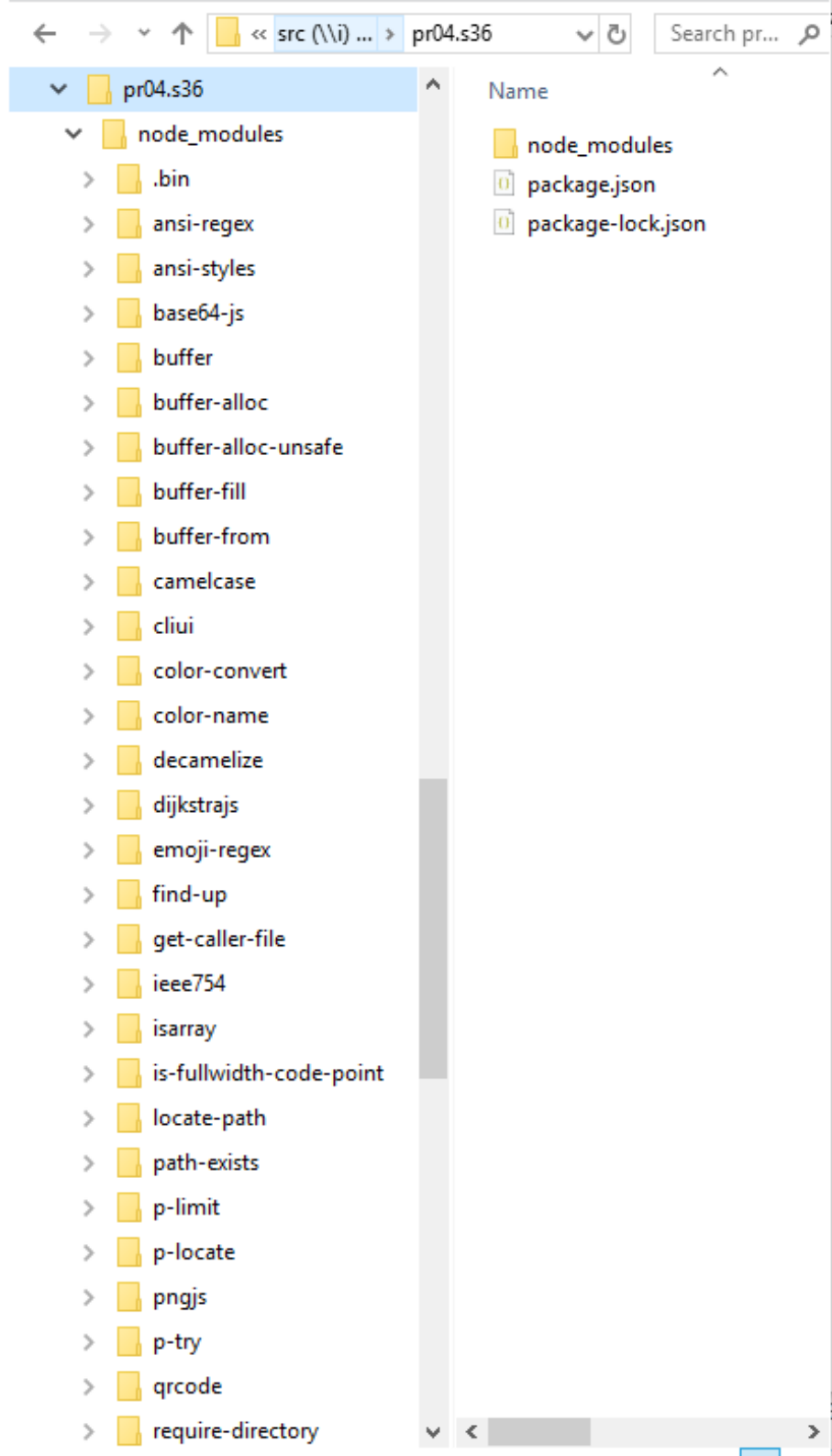
```
C:\Windows\System32\cmd.exe

X:\10. DYNWEB-2\pr03.s14>npm install qrcode
npm WARN qr_demo@1.0.0 No repository field.

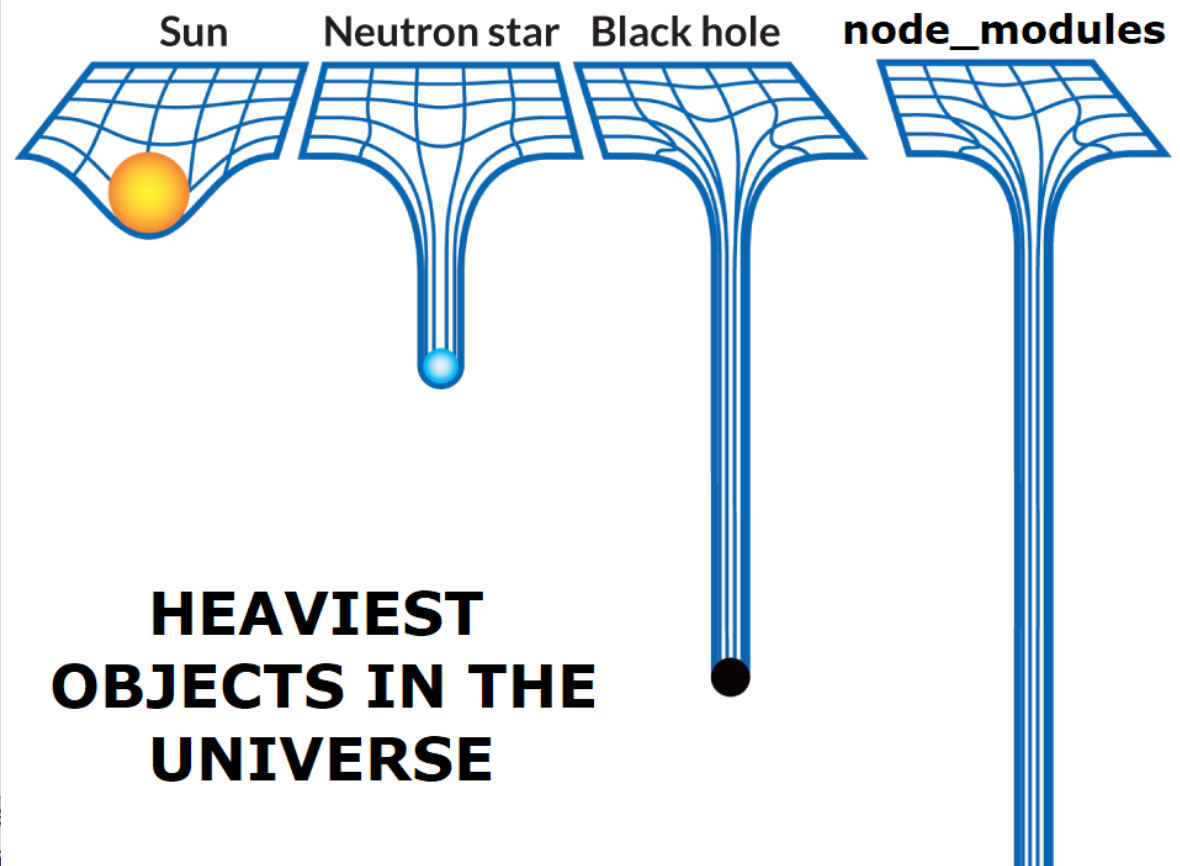
+ qrcode@1.4.4
added 37 packages from 28 contributors and audited 37 packages in 3.149s
found 0 vulnerabilities

X:\10. DYNWEB-2\pr03.s14>
```

- Paketi se pohranjuju u `node_modules` direktorij



node_modules



package.json

Dodali naredbu za pokretanje,
tako da možemo pokrenuti s:
`npm run dev`

```
{  
  "name": "qr_demo",  
  "version": "1.0.0",  
  "description": "QR code demo for Web1",  
  "main": "index.js",  
  "scripts": {  
    "dev": "node index.js",  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "igor",  
  "license": "MIT",  
  "dependencies": {  
    "qrcode": "^1.4"  
  }  
}
```

Netom instalirani paket, koristi se semantičko verzioniranje <https://github.com/npm/node-semver#versions>:

~version "Approximately equivalent to version", will update you to the next patch version.

~1.2.3 will use releases from 1.2.3 to <1.3.0.

^version Will update you to the next minor version.

^2.3.4 will use releases up to 3.0.0.

Primjer - generiranje QR koda

index.js

```
const QRCode = require('qrcode');

const genQR = async(data, fileName) => {
  try {
    await QRCode.toFile(fileName, data);
  } catch (err) {
    console.error(err)
  }
}

genQR('https://www.fer.unizg.hr/'
  , './fer-qr.png');
```



■ Node moduli (1/2)

- ES6 podržava module putem import/export
- Stariji "standard" su CommonJS moduli koje koristi node.js (ES6 je tek iz 2015. i nije još univerzalno podržan)
 - ES6 import/export je sada podržan i u novijim verzijama nodea (v14+)!
- Svaka datoteka svoj modul
- Instalirane moduli (pomoću npm install...) se smještaju u node_modules direktorij i uključuju s require bez putanje
 - Npr. `const QRCode = require('qrcode');`
- Vlastiti moduli se uključuju tako da se navodi putanja i ime
 - Ako se datoteka zove `index.js` onda je dovoljno sam navesti direktorij

Node moduli (2/2)

common/index.js

```
const os = require('os');
const moment = require('moment');
const chalk = require('chalk');
const getUsername = () => {
  return os.userInfo().username; }
const getHostName = () => {
  return os.hostname; }
const getCurrentTimestamp = () => {
  return moment().format("HH:mm"); }
const fancyLog = (msg) => {
  console.log(
    chalk.bgYellow.blue(getCurrentTimestamp()) +
    ' (' + chalk.blueBright(getUsername()) +
    '@' + chalk.magentaBright(getHostName()) +
    '): ' + chalk.greenBright(msg)
  ); }
module.exports = {
  log: fancyLog
};
```

index.js

```
//const fancy = require('./common/index.js');
const fancy = require('./common'); // kraće

console.log('Hello world!');
fancy.log('Hello world!');
```

common

JS index.js

> node_modules

JS index.js

{ } package-lock.json

{ } package.json

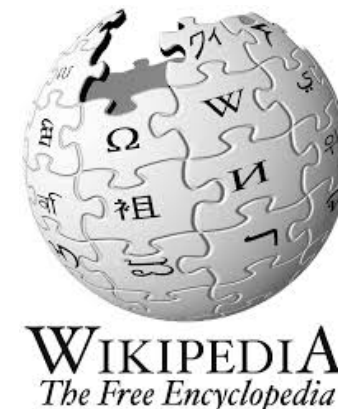
C:\Windows\System32\cmd.exe

```
X:\10. DYNWEB-2\pr04 node modules>node index.js
Hello world!
11:05 (Igor@i): Hello world!
X:\10. DYNWEB-2\pr04 node modules>
```

Ipak, Node.js se dominantno koristi za web

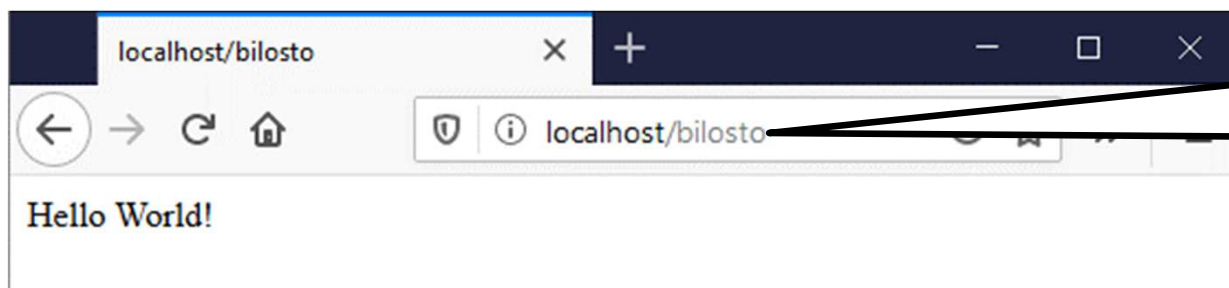
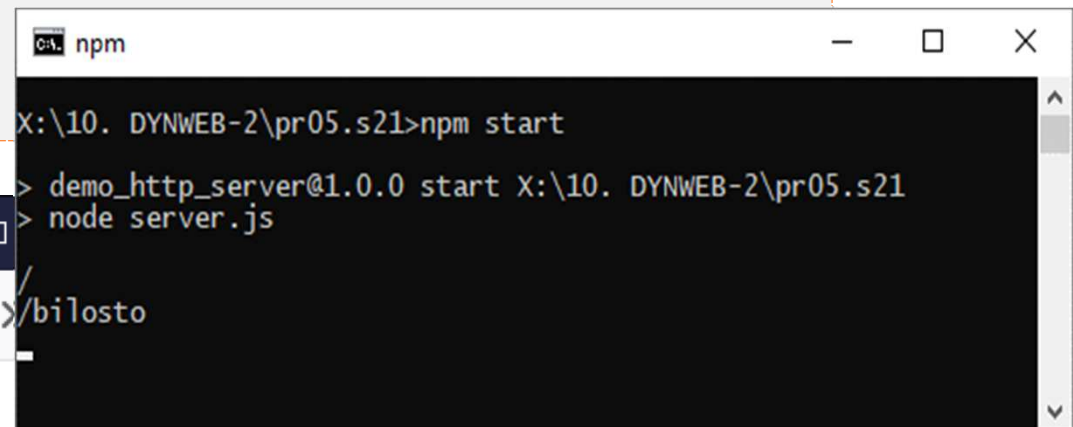
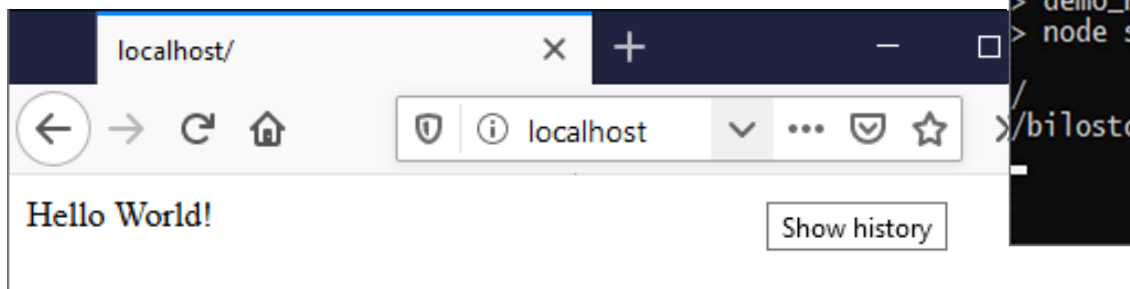


Node.js is an [open-source](#), [cross-platform](#), [JavaScript](#) runtime environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for [server-side scripting](#)—running scripts server-side to produce [dynamic web page](#) content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,^[6] unifying [web-application](#) development around a single programming language, rather than different languages for server- and client-side scripts.



HTTP server

```
var http = require('http'); // ugrađeni modul, ne treba install
http.createServer(function(req, res) {
  console.log(req.url);
  res.writeHead(200, { 'Content-Type': 'text/html' }); // header
  res.write('Hello World!'); // body
  res.end();
}).listen(80); // sluša na portu 80
```

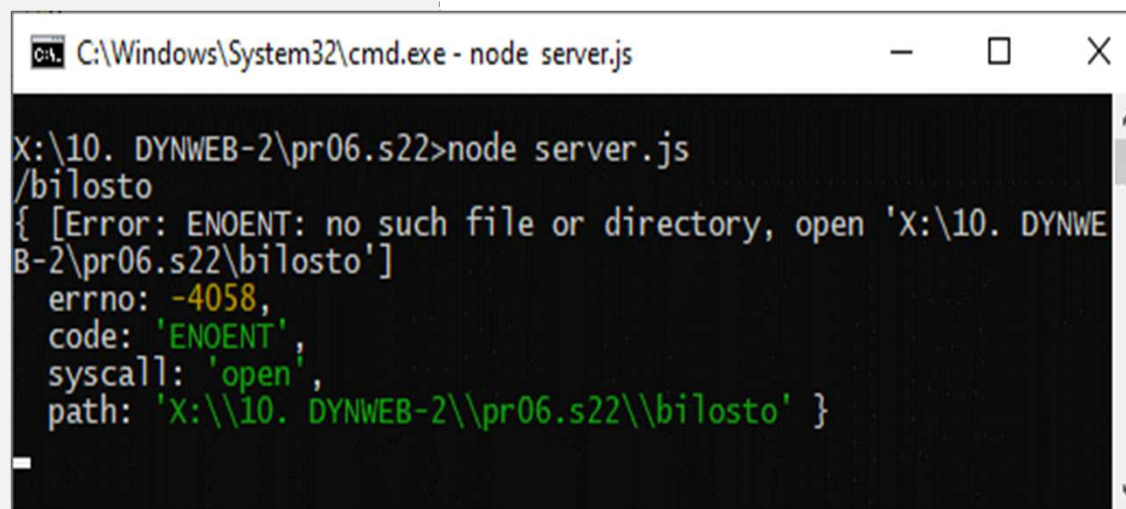


Uvijek vraćamo
isto 😊.
Popravimo →

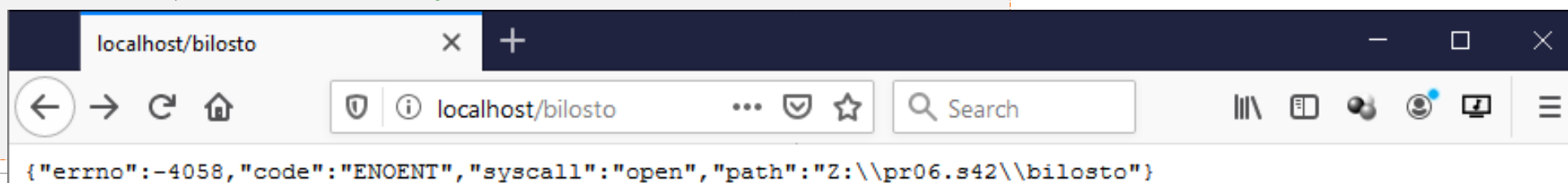
HTTP server - posluživanje statičkih datoteka

```
const http = require('http');
const fs = require('fs');

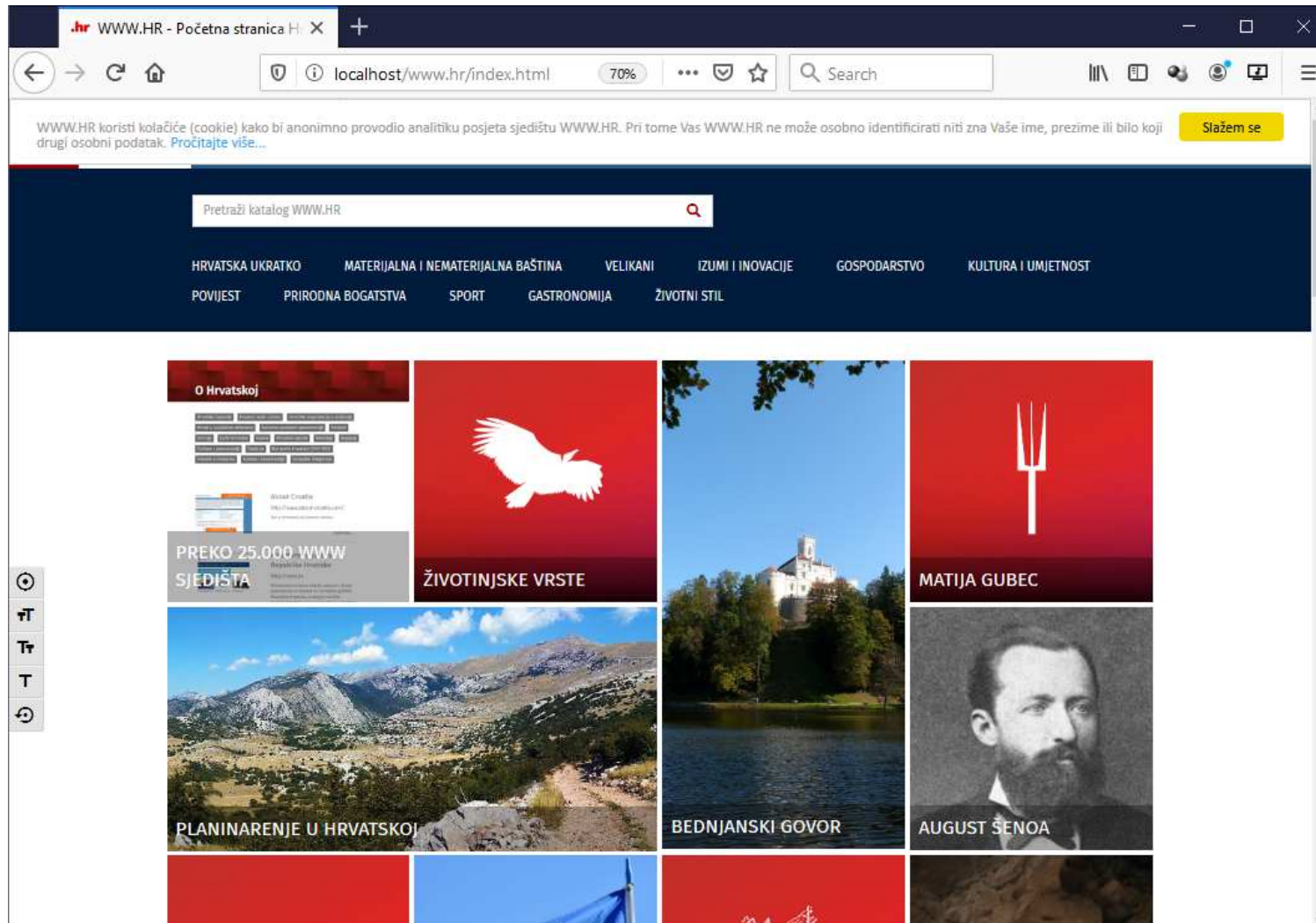
http.createServer(function(req, res) {
  console.log(req.url);
  fs.readFile(__dirname + req.url, function(err, data) {
    if (err) {
      console.error(err);
      res.writeHead(404);
      res.end(JSON.stringify(err));
    } else {
      res.writeHead(200 /* MIME? */ );
      res.end(data);
    }
  });
}).listen(80); // sluša na portu 80
```



```
C:\Windows\System32\cmd.exe - node server.js
X:\10. DYNWEB-2\pr06.s22>node server.js
/bilosto
{ [Error: ENOENT: no such file or directory, open 'X:\10. DYNWEB-2\pr06.s22\bilosto']
  errno: -4058,
  code: 'ENOENT',
  syscall: 'open',
  path: 'X:\\10. DYNWEB-2\\pr06.s22\\bilosto' }
```



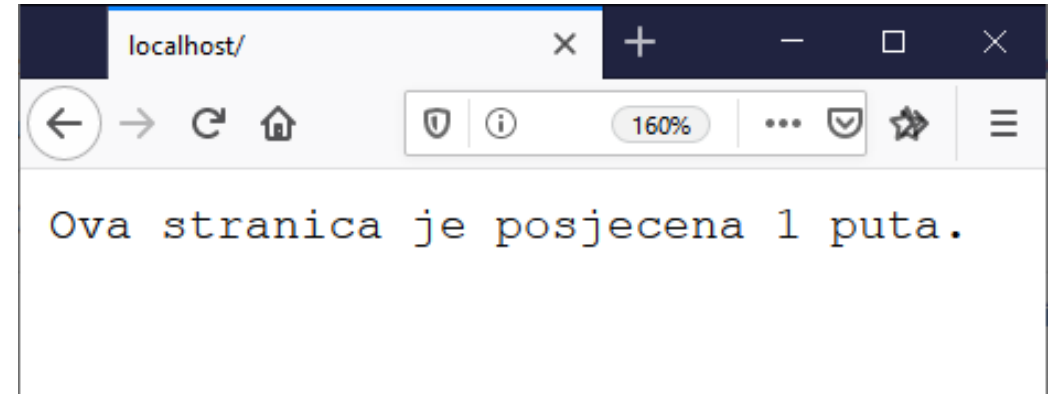
Ako kopiramo www.hr u poddirektorij i otvorimo <http://localhost/www.hr/index.html>



```
cat npm
/www.hr/index.html
/www.hr/w.hr_files/jquery.js
/www.hr/w.hr_files/bootstrap_002.js
/www.hr/w.hr_files/plugins.js
/www.hr/w.hr_files/cookieconsent.js
/www.hr/w.hr_files/bootstrap.css
/www.hr/w.hr_files/bootstrap.js
/www.hr/w.hr_files/plugins.css
/www.hr/w.hr_files/style.css
/www.hr/w.hr_files/infografike-stil.css
/www.hr/w.hr_files/icon.css
/www.hr/w.hr_files/css.css
/www.hr/w.hr_files/font-awesome.css
/www.hr/w.hr_files/light-top.css
/www.hr/w.hr_files/logo-2018.png
/www.hr/w.hr_files/katalog-screen.png
/www.hr/w.hr_files/info-zivotinje.png
/www.hr/w.hr_files/bednjanski.png
/www.hr/w.hr_files/info-gubec.png
/www.hr/w.hr_files/planinarenje.png
/www.hr/w.hr_files/senoa.png
/www.hr/w.hr_files/info-plitvice.png
/www.hr/w.hr_files/vindija.png
/www.hr/w.hr_files/ris.png
/www.hr/w.hr_files/infografika-strudla-butto
n.png
/www.hr/w.hr_files/Tesla.png
/www.hr/w.hr_files/kravata.png
/www.hr/w.hr_files/nogomet.png
/www.hr/w.hr_files/FaustVrancic.png
/www.hr/w.hr_files/dodaj-sjediste.png
/www.hr/w.hr_files/live-zg.png
/www.hr/w.hr_files/Plitvice.png
/www.hr/w.hr_files/DiokecijanP.png
/www.hr/w.hr_files/klima.png
/www.hr/w.hr_files/akvatorij.png
/www.hr/w.hr_files/otonivekovic.png
/www.hr/w.hr_files/vina.png
/www.hr/w.hr_files/bastina.png
/www.hr/w.hr_files/soc-drzava.png
/www.hr/w.hr_files/turizam.png
/www.hr/w.hr_files/Kulen_gastro.png
```

Rekreirajmo "posjecena N puta" primjer:

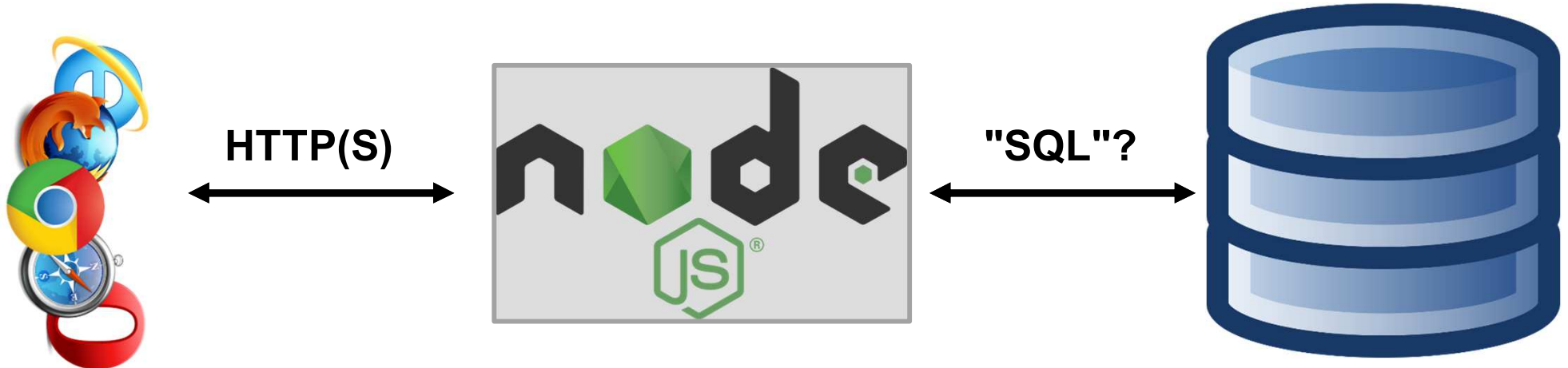
```
const http = require('http');
let counter = 0;
http.createServer(function (req, res) {
  res.writeHead(200);
  res.end(`Ova stranica je posjecena
${++counter} puta.`);
}).listen(80);
```



- Zapravo nije istovjetno - ovdje smo koristili brojač u memoriji
 - Ako ponovno pokrenemo server.js počinje opet brojati od 1
- Primijetimo:
 1. Radna memorija - **privremena**, *non-persistent*, brza
 2. Datoteke - **trajna**, *persistent*, sporija

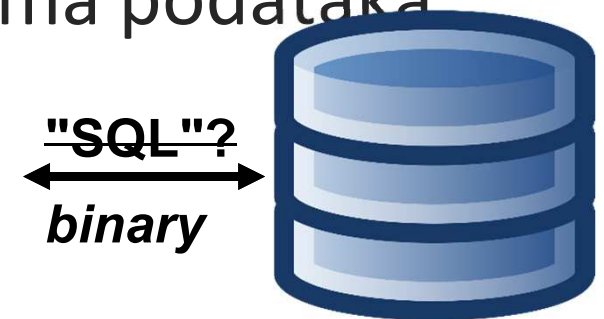
Baza podataka - perzistentna memorija

- Kada datoteke nisu dovoljne (tj. u 99.99% slučajeva)
- Prednosti (relacijskih) baza podataka nećemo navoditi ovdje -> <https://www.fer.unizg.hr/predmet/bazepod>
- Iole složenije web aplikacije imaju prateću bazu podataka u kojoj pohranjuju sadržaj
- Npr. portal pohranjuje vijesti u bazu podataka i onda se naslovna stranica generira od najnovijih N vijesti



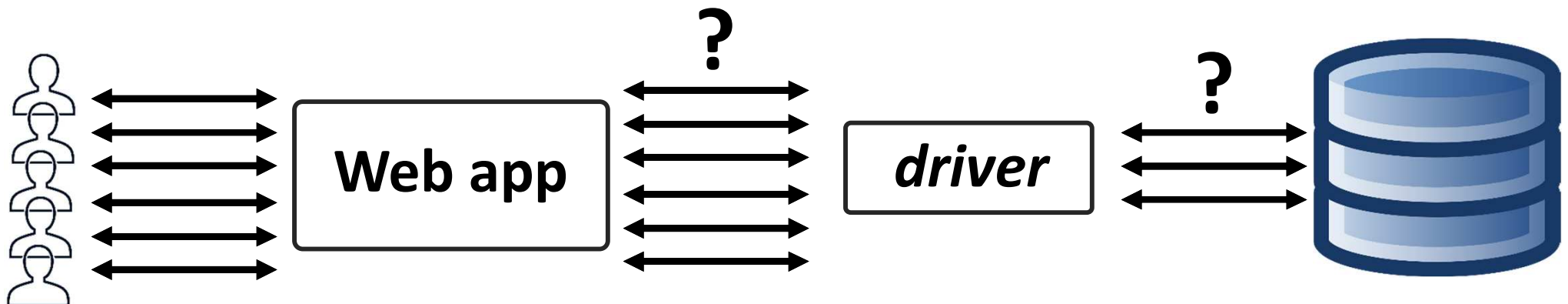
Općeniti problem: kako (**tehnički**) komunicirati s bazom? (1/3)

- Programski jezik X **<--?-->** Baza podataka Y
- TL;DR: treba nam **upravljački program** (*driver*)
- Driver apstrahira pristup bazi, tj. s driverom iz koda "pričamo" uvijek na isti način (neovisno o operacijskom sustavu i proizvođaču baze podataka)
- Driveri se mogu temeljiti na različitim protokolima i specifikacijama
- Najpoznatiji: **ODBC** - API za pristup bazama podataka
 - -> onda imamo ODBC drivere za X, Y, ...
- JDBC, OLEDB, ADO, ADO.NET, ...



Općeniti problem: kako (**tehnički**) komunicirati s bazom? (2/3)

- **Koliko i kada** otvaramo **konekcije** na bazu?
 - I s kojim korisničkim računom?
- Činjenice:
 - Uspostava konekcije je skupa (*handshake*, ~30ms)
 - Baze podataka ne mogu podnijeti proizvoljan broj konekcija
 - Upiti na jednoj konekciji se obavljaju serijski
 - Baze podataka su optimizirane za paralelni rad (s mjerom 😊)



Općeniti problem: kako (**tehnički**) komunicirati s bazom? (3/3)

- **Koliko i kada otvaramo konekcije na bazu?**
 - I s kojim korisničkim računom?
- **Opcije:**
 - Svaki zahtjev jedna konekcija - **LOŠE**
 - Svi zahtjevi jedna konekcija - **LOŠE**
 - Kompromis:
 - Driver održava **bazen (*pool*)** N unaprijed otvorenih konekcija s istim (zašto?) korisničkim računom
 - Tzv. ***connection pooling***



Connection pooling ;-)



■ Kako **logički** komunicirati s bazom?

- SQL ~ *lingua franca*
- 2+1 opcije:
 1. SQL naredbe koje pišemo u kodu
 2. Koristeći ORM (*Object-Relational Mapping*) softver
 - Zbog diskrepancije objektnog i relacijskog modela često se koristi softver koji prebacuje podatke iz jednog modela u drugi
 - Npr. *Hibernate, Entity Framework, ActiveRecord, Sequelize...*
 - Tipično imaju i prateći objektni upitni jezik koji se prevodi u SQL, npr. HQL, Linq, itd.
 3. Kombinacija 1 i 2 😊

Primjer: spojimo se na PostgreSQL bazu (1/4)

- Uzmimo driver: <https://node-postgres.com/>
 - Instalacija: `npm install pg`
- Po preporuci, stavit ćemo kod za obavljanje upita u direktorij `db` i sve naredbe obavljati putem njega
- Npr. tako možemo elegantno sve logirati:
 - Vrijeme obavljanja
 - Postavljeni upit
 - Rezultat

```
node_modules
- db/
  - index.js
- index.js
```

Primjer: spojimo se na PostgreSQL bazu (2/4)

db/index.js

```
const { Pool } = require('pg');
const pool = new Pool({
  user: 'musicbrainz',
  host: 'localhost',
  database: 'musicbrainz',
  password: 'musicbrainz',
  port: 5432, // prilagoditi...
})
module.exports = {
  query: async (SQL, params) => {
    try {
      const start = Date.now();
      let res = await pool.query(SQL, params);
      const duration = Date.now() - start;
      console.log('executed query', {
        SQL, params, duration, rows: res.rowCount });
      return res;
    } catch (error) { console.log(error) }
  }
}
```

helloworld.js

```
const db = require('./db');
(async() => {
  let res = await db.query(
    'SELECT id, name, description FROM place_type');
  console.log(res);
})(); // IIFE: https://developer.mozilla.org/en-US/docs/Glossary/IIFE
```

[ispis](#)

Primjer: spojimo se na PostgreSQL bazu (3/4)

```
C:\Windows\System32\cmd.exe

Z:\pr09 Node.js MusicBrainz>node helloworld.js
executed query { text: 'SELECT id, name, description FROM place_type',
  params: undefined,
  duration: 44,
  rows: 8 }
Result {
  command: 'SELECT',
  rowCount: 8,
  oid: null,
  rows:
    [ { id: 3, name: 'Other', description: null },
      { id: 1,
        name: 'Studio',
        description:
          'A place designed for non-live production of music, typically a recording studio.' },
      { id: 2,
        name: 'Venue',
        description:
          'A place that has live artistic performances as one of its primary functions, such as a concert hall.' },
      { id: 4,
        name: 'Stadium',
        description:
          'A place whose main purpose is to host outdoor sport events, typically consisting of a pitch surrounded by a structure for spectators with no roof,
or a roof which can be retracted.' },
      { id: 5,
        name: 'Indoor arena',
        description:
          'A place consisting of a large enclosed area with a central event space surrounded by tiered seating for spectators, which can be used for indoor s
ports, concerts and other entertainment events.' },
      { id: 6,
        name: 'Religious building',
        description:
          'A place that has worship or religious studies as its main function. Religious buildings often host concerts and serve as recording locations, espe
cially for classical music.' },
      { id: 7,
        name: 'Educational institution',
        description:
          'A school, university or other similar educational institution (especially, but not only, one where music is taught)' },
      { id: 8,
        name: 'Pressing plant',
        description:
          'A place (generally a factory) at which physical media are manufactured.' } ],
  fields:
    [ Field {
      name: 'id',
      tableID: 18910,
      columnID: 1,
```

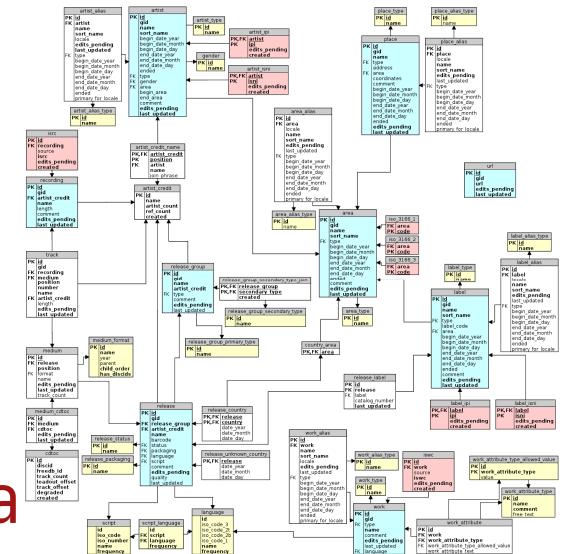
■ Primjer: spojimo se na PostgreSQL bazu (4/4)

- Kako koristiti argumente?
- Savjet: pripazite na SQL injection

```
(async() => {  
    let rows = await db.query(`SELECT id, name, description  
        FROM place_type  
        WHERE id BETWEEN $1 AND $2`, [1, 3]);  
    console.log(rows);  
})();  
// IIFE: https://developer.mozilla.org/en-US/docs/Glossary/IIFE
```


Primjer - MusicBrainz

- *"MusicBrainz is an open music encyclopedia that collects music metadata and makes it available to the public."*
<https://musicbrainz.org/>
- Relativno složen model s velikim brojem zapisa (milijuni)
 - Ali nema veze - nećemo ga proučavati
- Dovoljno je znati da postoje:
 - Izvođači (**artist_credit**)
 - Izdanja (**release_group**)
 - godina izdanja (**first_release_date_year**)
 - ocjena izdanja (**rating**)
- Može se skinuti gotova VM: https://musicbrainz.org/doc/MusicBrainz_Server/Setup
- 2022.: pg_dump baze s 5 tablica na Teams/General/Class materials



Plan!

GET ... ?year=2018&rating=88

GET ... ?year=2018

GET ...
?rating=88

2019 (6123)
2018 (4753)
2017 (1323)
2016 (1423)
2015 (4123)
2014 (6123)
2013 (1723)
2012 (1723)
2011 (1273)
godina(broj)

...

Filtrirano po
trenutnom
ratingu

autor: release
autor: release
autor: release
autor: release
autor: release
autor: release
autor: release
autor: release
autor: release
autor: release

rating
rating
rating
rating
rating
rating
rating
rating
rating
rating

Filtrirano po trenutnom
ratingu i godini

100 (6123)
97 (4753)
95 (1323)
90 (1423)
88 (4123)
85 (6123)
80 (1723)
70 (1723)
80 (1273)
rating(broj)

...

Filtrirano po
trenutnoj godini

Rješenje (1/4)

```
getHomePage = async(year, rating) => {  
  let [left, center, right] =  
    await Promise.all([  
      getSideBarYears(rating),  
      getCenterPage(year, rating),  
      getSideBarRatings(year)  
    ]);  
  return `></head>  
  <body><div class="container">  
    <div class="row">  
      <div class="col"></div>  
    </div><hr>  
    <div class="row">  
      <div class="col-3">${left}</div>  
      <div class="col-6">${center}</div>  
      <div class="col-3">${right}</div>  
    </div>  
  </div>  
</body></html>`;
```

```
http.createServer(async function(req, res) {  
  var q = url.parse(req.url, true).query;  
  res.writeHead(200);  
  res.end(await getHomePage(q.year, q.rating));  
}).listen(80);
```

Usput: hoćemo li i
inače zbilja ovako
stvarati HTML?
-> NE

Rješenje (2/4) - pomoćne funkcije

```
getPreparedParams = (year, rating, params) => {  
  let where = '';  
  if (year) {  
    where += ' AND first_release_date_year = $1';  
    params.push(year);  
  }  
  if (rating) {  
    where += ' AND rating = $' + (params.length + 1);  
    params.push(rating);  
  }  
  return where;  
}
```

Proizvede npr.:

- Za 2018, 95:
where =
`AND first_release_date_year
= \$1 AND rating = \$2`
params= [2018, 95]
- Za undefined, 95:
where = `rating = \$2`
params= [95]

Proizvede npr.:

- Za 2018, 95:
`/?year=2018&rating=95`
- Za undefined, 95:
`/?rating=95`

```
getHref = (year, rating) => {  
  let args = [];  
  if (year) {  
    args.push('year=' + year);  
  }  
  if (rating) {  
    args.push('rating=' + rating);  
  }  
  return '/' + args.length ? '?' + args.join('&') : '';  
}
```

Rješenje (3/4) - pomoćne funkcije

Jer želimo sve godine

```
getSideBarYears = async(rating) => {  
  let params = [];  
  let where = getPreparedParams(undefined, rating, params);  
  const { rows } = await db.query(`SELECT first_release_date_year as year,  
                                     count(*) as cnt  
                                     FROM release_group_meta rgm  
                                     WHERE first_release_date_year between 1990 and 2020  
                                     ${where}  
                                     GROUP BY first_release_date_year  
                                     ORDER BY first_release_date_year desc`, params);  
  return '<ul>' +  
    rows  
    .map(x => `<li><a href="${getHref(x.year, rating)}">${x.year}(${x.cnt}) </a></li>`)  
    .join('') +  
    '</ul>';  
}
```

getSideBarRatings je vrlo slična, nećemo pokazivati ovdje (pogledajte u službenom repozitoriju)

Svaki li je link, npr.:

```
<li><a href="?year=2018">2018(23117)</a></li>  
<li><a href="?year=2017">2017(54084)</a></li>  
<li><a href="?year=2016">2016(60137)</a></li>
```

...

Rješenje (4/4) - getCenterPage

(potencijalno)
filtriramo po svemu

```
getCenterPage = async(year, rating) => {  
  let params = [];  
  let where = getPreparedParams(year, rating, params);  
  const { rows } = await db.query(`SELECT rg.name as release, ac.name as artist, rating  
    FROM release_group rg  
    JOIN release_group_meta rgm on rg.id = rgm.id  
    JOIN artist_credit ac ON rg.artist_credit = ac.id  
    WHERE rating is not null ${where}  
    ORDER BY first_release_date_year DESC,  
              first_release_date_month DESC,  
              first_release_date_day DESC  
    LIMIT 200`, params);  
  return '<ul class="list-group">' +  
    rows.map(x => `<li class="list-group-item d-flex justify-content-between align-items-center">  
      ${x.artist}: ${x.release}  
      <span class="badge badge-primary badge-pill">${x.rating}</span>  
    </li>`).join('') +  
    '</ul>';  
}
```

Proizvodi npr.:

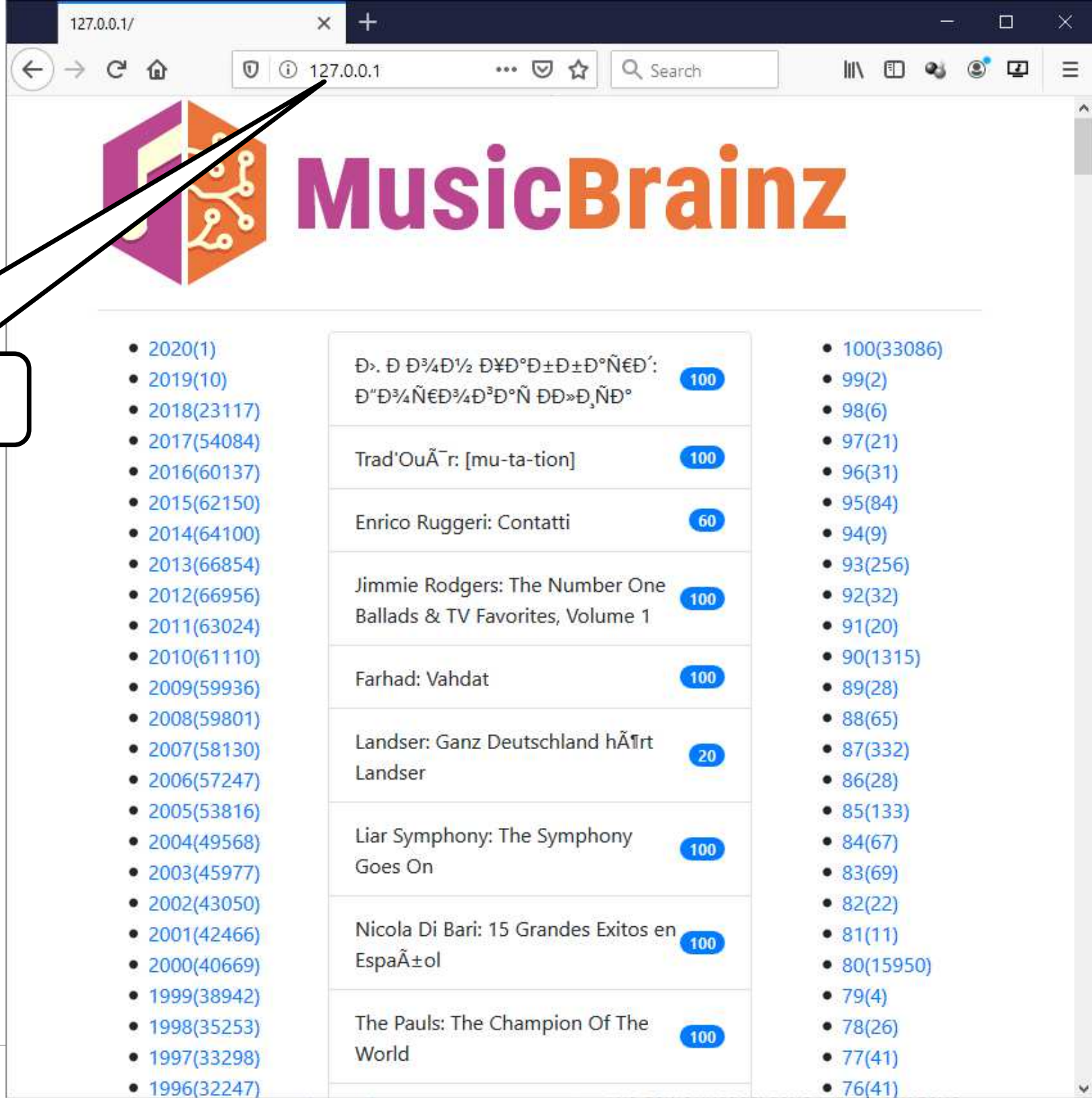
```
<ul><li class="list-group-item d-  
flex justify-content-between align-  
items-center">
```

Bee Gees: How Deep Is Your Love

```
<span class="badge badge-  
primary badge-pill">100</span>  
</li> <li>...</li> ...</ul>
```

 **Konačno:**

http://localhost



127.0.0.1/

127.0.0.1

MusicBrainz

- 2020(1)
- 2019(10)
- 2018(23117)
- 2017(54084)
- 2016(60137)
- 2015(62150)
- 2014(64100)
- 2013(66854)
- 2012(66956)
- 2011(63024)
- 2010(61110)
- 2009(59936)
- 2008(59801)
- 2007(58130)
- 2006(57247)
- 2005(53816)
- 2004(49568)
- 2003(45977)
- 2002(43050)
- 2001(42466)
- 2000(40669)
- 1999(38942)
- 1998(35253)
- 1997(33298)
- 1996(32247)

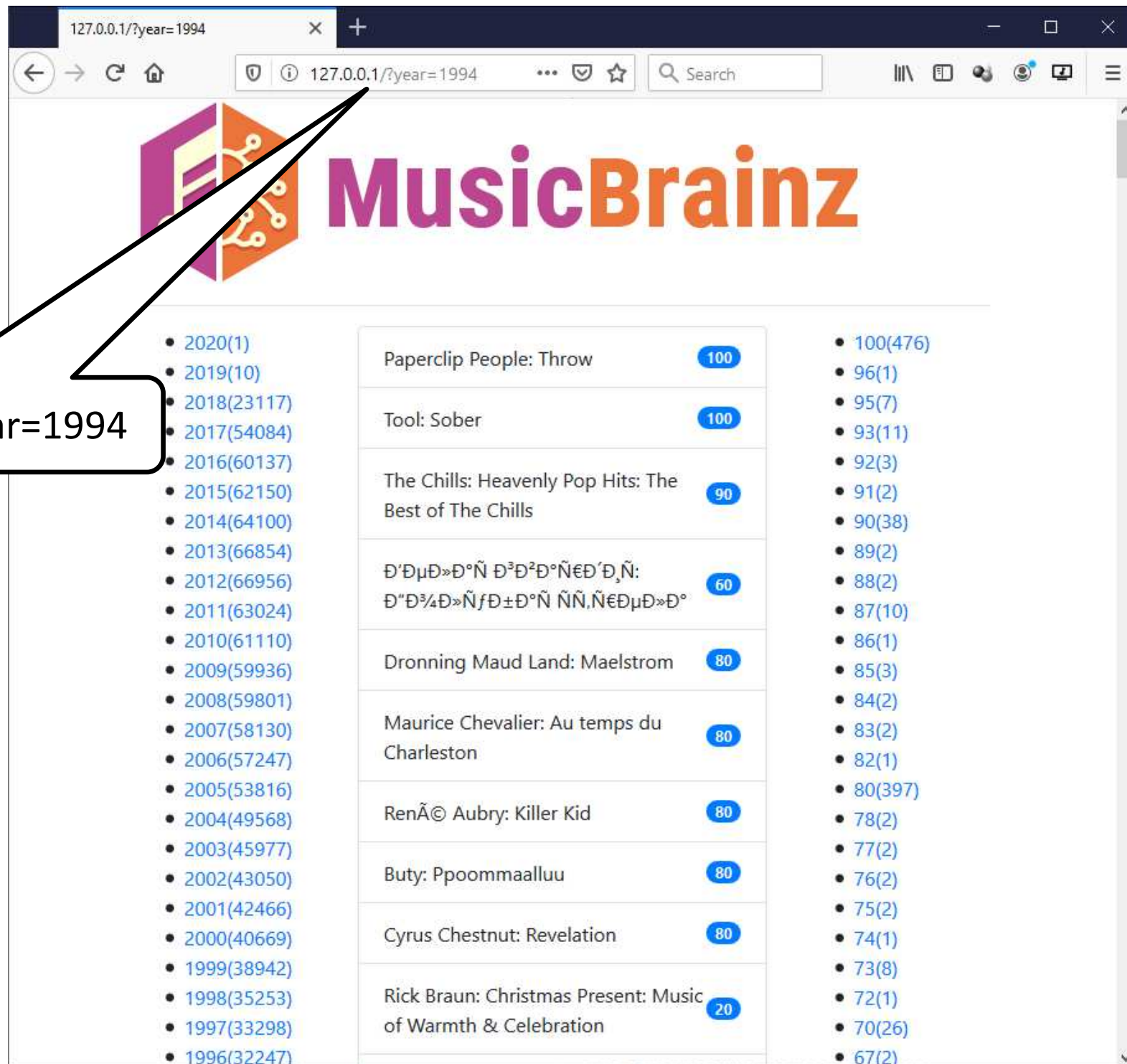
Đ>. Đ Đ¾¼Đ½ Đ±Đ°Đ±Đ±Đ°Ñ€Đ': Đ°Đ¾¼Ñ€Đ¾¼Đ³Đ°Ñ ĐĐ»Đ,ÑĐ°	100
Trad'OuÃ~r: [mu-ta-tion]	100
Enrico Ruggeri: Contatti	60
Jimmie Rodgers: The Number One Ballads & TV Favorites, Volume 1	100
Farhad: Vahdat	100
Landser: Ganz Deutschland hÃ¼rt Landser	20
Liar Symphony: The Symphony Goes On	100
Nicola Di Bari: 15 Grandes Exitos en EspaÃ±ol	100
The Pauls: The Champion Of The World	100

- 100(33086)
- 99(2)
- 98(6)
- 97(21)
- 96(31)
- 95(84)
- 94(9)
- 93(256)
- 92(32)
- 91(20)
- 90(1315)
- 89(28)
- 88(65)
- 87(332)
- 86(28)
- 85(133)
- 84(67)
- 83(69)
- 82(22)
- 81(11)
- 80(15950)
- 79(4)
- 78(26)
- 77(41)
- 76(41)

UNIZG-FER

 **Konačno:**

<http://localhost/?year=1994>



127.0.0.1/?year=1994

127.0.0.1/?year=1994

MusicBrainz

- 2020(1)
- 2019(10)
- 2018(23117)
- 2017(54084)
- 2016(60137)
- 2015(62150)
- 2014(64100)
- 2013(66854)
- 2012(66956)
- 2011(63024)
- 2010(61110)
- 2009(59936)
- 2008(59801)
- 2007(58130)
- 2006(57247)
- 2005(53816)
- 2004(49568)
- 2003(45977)
- 2002(43050)
- 2001(42466)
- 2000(40669)
- 1999(38942)
- 1998(35253)
- 1997(33298)
- 1996(32247)

Paperclip People: Throw	100
Tool: Sober	100
The Chills: Heavenly Pop Hits: The Best of The Chills	90
Đ'ĐμĐ»Đ°Ñ Đ³Đ²Đ°Ñ€Đ'Đ,Ñ: Đ"Đ³⁄₄Đ»ÑfĐ±Đ°Ñ ÑÑ,Ñ€ĐμĐ»Đ°	60
Dronning Maud Land: Maelstrom	80
Maurice Chevalier: Au temps du Charleston	80
RenÅ© Aubry: Killer Kid	80
Buty: Ppoommaalluu	80
Cyrus Chestnut: Revelation	80
Rick Braun: Christmas Present: Music of Warmth & Celebration	20

- 100(476)
- 96(1)
- 95(7)
- 93(11)
- 92(3)
- 91(2)
- 90(38)
- 89(2)
- 88(2)
- 87(10)
- 86(1)
- 85(3)
- 84(2)
- 83(2)
- 82(1)
- 80(397)
- 78(2)
- 77(2)
- 76(2)
- 75(2)
- 74(1)
- 73(8)
- 72(1)
- 70(26)
- 67(2)

Konačno:

The screenshot shows the MusicBrainz website interface. The browser's address bar displays the URL `localhost/?year=1994&rating=95`. A callout box with a black border and white background contains the same URL: `http://localhost/?year=1994&rating=95`. The website header features the MusicBrainz logo and the text "MusicBrainz". Below the header, a list of albums is displayed, each with a rating of 95. The albums listed are:

- Sade: The Best of Sade
- Bon Jovi: Cross Road: The Best of Bon Jovi
- Radiohead: My Iron Lung
- SinÃ©ad O'Connell: Universal Mother
- NOFX: Punk in Drublic
- Hole: Live Through This
- Therapy?: Troublelegum

Each album entry includes a blue circle with the number "95" indicating its rating. To the right of the album list, a vertical list of ratings is visible, including 100(476), 96(1), 95(7), 93(11), 92(3), 91(2), 90(38), 89(2), 88(2), 87(10), 86(1), 85(3), 84(2), 83(2), 82(1), 80(397), 78(2), 77(2), and 76(2).

Osvrnimo se...

- Upoznali smo se načelom rada HTTP poslužitelja
 - Kako generirati **dinamički** sadržaj
 - (Procesni modeli, web poslužitelji) -> P12, za nestrpljive - video
- Node.js
 - Okolina za obavljanje JS koda
 - Opće namjene, ali najviše se koristi za razvoj web-aplikacija
 - *Event loop, node modules*
 - Rudimentarni HTTP poslužitelj i web-aplikacija
 - Spajanje i dohvat podataka iz baze podataka
 - *Connection pooling*
 - ALI:
 - Radimo samo s korijenskom ('/') putanjom!?
 - Sav kôd je više-manje u jednoj datoteci?
 - Ispreplićemo kôd i HTML-a i dohvat podataka?
 - Kako poslužujemo statičke datoteke, npr. MB logo?