# Ispitna pitanja web2

1. Šta je nginx?
2. Kakve funkcionalnosti nginx omogućava?

NGINX is open source software for web serving, reverse proxying, caching, load balancing, media streaming, and more.

1. Šta je ECMAScript i koje verzije postoje?

ECMAScript is a JavaScript standard meant to ensure the interoperability of web pages across different web browsers. JavaScript is a language based on ECMAScript. Verzije od 1 do 12(2021).

1. Šta označava “use strict”?

The main reasoning for strict mode was not to prevent programming errors - it was to make JavaScript lexically scoped so it could be statically analysable.

1. Šta je forEach i kako funkcioniše?

Iterating array elements. Arr.forEach(function) syntax

1. Getters & Setters (person obj,language property)

var person = {  
  firstName: "John",  
  lastName : "Doe",  
  language : "NO",  
  get lang() {  
    return this.language;  
  },  
  set lang(value) {  
    this.language = value;  
  }  
};

1. JS prototype

Objects can have a prototype object, which acts as a template object that it inherits methods and properties from.

1. Object.create

The Object.create() method creates a new object, using an existing object as the prototype of the newly created object.

const me = Object.create(person);

1. var, let i const

A screenshot of a computer

Description automatically generated with low confidence

1. for, for in, for of

for (let i = 0; i < 9; i++) for( var|const|let variable of|in iterable) same syntax for for in

The JavaScript for/of statement loops through the values of an iterable objects. For in loops over enumerable property names of an object.

1. JS iterables

value (the next value)

done (true or false)

while (true) {

const result = iterator.next();

if (result.done) break;

}

1. JS Map & Set

A Map holds key-value pairs where the keys can be any datatype. Map remembers the original insertion order of the keys. Map has a property that represents the size of the map. A JavaScript Set is a collection of unique values. Each value can only occur once in a Set.Set can hold any value of any data type.

1. JS Class

Templates for objects.

1. JS Promise

The Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

const myPromise = new Promise(function(myResolve, myReject) {  
// "Producing Code" (May take some time)  
  
  myResolve(); // when successful  
  myReject();  // when error  
});  
  
// "Consuming Code" (Must wait for a fulfilled Promise).  
myPromise.then(  
  function(value) { /\* code if successful \*/ },  
  function(error) { /\* code if some error \*/ }  
);

1. Fetch API

The Fetch API interface allows web browser to make HTTP requests to web servers.

fetch(file)

.then(x => x.text())

.then(y => myDisplay(y));

1. Symbol type

Unique

let id = Symbol('id');

person[id] = 140353;

// Now person[id] = 140353

// but person.id is still undefined

1. Function default & rest

ES6 allows function parameters to have default values. Function(x, y = 10)

The rest parameter (...) allows a function to treat an indefinite number of arguments as an array function sum(...args)

1. NJS Async non blocking vs Sync blocking

Blocking methods execute synchronously and non-blocking methods execute asynchronously. …

1. NJS Async event queue loop

Svi pozivi idu u event queue. Neblokirajuće operacije se odmah izvršavaju i šalje se odgovor – response. Za pozive blokirajućih operacija koje se ne mogu odmah izvršiti i za koje se ne zna koliko će trajati

registruje se callback funkcija.

1. Implementacija blokirajućih operacija- Worker pool

Node.js uses the Worker Pool to handle "expensive" tasks. This includes I/O for which an operating system does not provide a non-blocking version, as well as particularly CPU-intensive tasks.

1. NJS Async callback funkcija

A callback is a function called at the completion of a given task; this prevents any blocking, and allows other code to be run in the meantime.

1. process.nextTick & setImmediate

process.nextTick(callback) zakazuje izvršavanje callback funkcije pre početka sledeće iteracije eventLoop petlje događaja. Any function passed as the setImmediate() argument is a callback that's executed in the next iteration of the event loop.

Use setImmediate if you want to queue the function behind whatever I/O event callbacks that are already in the event queue. Use process.nextTick to effectively queue the function at the head of the event queue so that it executes immediately after the current function completes.

1. EventEmitter

The EventEmitter is a module that facilitates communication/interaction between objects in Node. The concept is quite simple: emitter objects emit named events that cause previously registered listeners to be called.

1. JS closure (innerFunc is a closure in this example)

function outerFunc() {

let outerVar = 'I am outside!';

function innerFunc() {

console.log(outerVar); // => logs "I am outside!"

}

return innerFunc();

}

1. callBack chain

rekurzivno se izvrsava

1. NJS Buffer

Buffer objects are used to represent a fixed-length sequence of bytes. primeri korišćenja buffer – a za upis, čitanje, kopiranje, nadovezivanje i pristup delu buffer – a

1. NJS Stream

Tokovi – streams su memorijske strukture koje se koriste za i/o operacije kao što su rad sa datotekama, čitanje i slanje http zahteva

1. NJS kompresija (zlib modul)

Svaki metod za kompresiju (deflate, deflateRaw, gzip) ima odgovarajući metod za dekompresiju (inflate, inflateRaw, gunzip). Svi metodi rade asinhrono, tj. koriste callback, što se može videti u ispisu koji ne prati redosled poziva u listingu. Podaci komprimovani sa metodima deflate i gzip su dodatno

dekomprimovani i sa unzip.

1. NJS File read / write ( fs modul)

dva osnovna načina sinhroni i asinhroni (sinhroni se koristi kada preostali kod cita iz fajla,dok kod asinhronog treba biti pazljiv o citanju pre zavrsenog upisa)

1. File system promise

The fs.promises API provides an alternative set of asynchronous file system methods that return Promise objects rather than using callbacks. The API is accessible via require('fs').promises.

A key difference between the two is when using the callback approach, we’d normally just pass a callback into a function that would then get called upon completion in order to get the result of something. In promises, however, you attach callbacks on the returned promise object.

1. NJS http server
2. NJS http client

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client. It can also make HTTP requests to other servers.

1. NJS http predaja podataka

var http = require('http');

http.createServer(function (req, res) {

res.writeHead(200, {'Content-Type': 'text/html'});

res.write(req.url);

res.end();

}).listen(8080);

var options = {

host: 'www.nodejitsu.com',

path: '/',

port: '1338',

//This is the only line that is new. `headers` is an object with the headers to request

headers: {'custom': 'Custom Header Demo works'}

};

http.request();

1. https public / private key

https je protokol za šifrovanu komunikaciju između klijenta i servera. https koristi ssl (secure sockets layer) / tls (transport layer security) protocol. Komunikacija se šifruje korišćenjem tajnog ključa koji se razmenjuje preko PKI (Public Key Infrastrucure). PKI zamenjuje “sigurnog kurira” za razmenu simetričnog kratkoročnog tajnog ključa kojim se šifruje https komunikacija. Javni – public key predstavlja sertifikat koji dokazuje identitet pošiljaoca. Tajni i javni ključ izdaje CA (Certifiate Authority) koji potpisuje izdate ključeve

1. NJS sockets (net modul)

Nalaze se ispod http sloja i omogućavaju komunikaciju od tačke do tačke na mreži između servera i klijenta. Sve internet komunikacije koriste sockete za prenos podataka. Klijenti i serveri su jedinstveno označeni IP adresom i portom. NJS soketi implementiraju tok duplex između klijenta i servera za čitanje i pisanje.

1. NJS Express

Nodejs modul koji omogućava razne dodatne funkcionalnosti (u odnosu na http) kao što su rutiranje i serviranje statičkih fajlova.

1. Express static files (css, slike)

Serviranje statičkih file – ova se postiže korišćenjem metoda app.use koji navodi middleware funkciju express.static. express.static je ugrađena middleware funkcija koja definiše folder sa statičkim file -ovima

1. Express rutiranje method -> get, post, put, delete, itd

app.method(“/route”, (req, res, next) =>

{

… next();

});

1. Express predaja podataka (vrv se misli na parameter)

Parametri rute su imenovani delovi putanje – URL – a koji se koriste da private vrednosti navedene na poziciji gde se nalaze. Vrednosti parametara se prihvataju iz koda sa req.params i nazivom parametra kao ključem: req.params.userId, req.params.bookId

1. Express callback next

Calling next() with no arguments tells express to continue to the next matching middleware or route handler.

Graphical user interface, text, application

Description automatically generated

1. Express modularni ruteri

Graphical user interface, text, application

Description automatically generated

1. Express middleware

Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application’s request-response cycle. The next middleware function is commonly denoted by a variable named next.

Middleware functions can perform the following tasks:

* Execute any code.
* Make changes to the request and the response objects.
* End the request-response cycle.
* Call the next middleware function in the stack.

If the current middleware function does not end the request-response cycle, it must call next() to pass control to the next middleware function. Otherwise, the request will be left hanging.

1. Šta je type script (TS)?

Type script (TS) jezik i kompajler su nastali 2012 godine. Osnovna ideja sa TS je da se uvedu brojna ograničenja kao kod strogo tipiziranih jezika (C#, JAVA) koja će omogućiti razvoj velikih i složenih projekata u JS bez izmene samog JS jezika

1. TS deklaracije tipova

a : number; b : string; c: Boolean;

1. TS izvedeno tipiziranje
2. TS deklaracije - \*.d.ts file

File – ovi tipa \*.d.ts se koriste za povezivanje postojećeg JS koda sa novim TS kodom. Bez dodatne specifikacije postojećeg JS koda za novi TS, to nije moguće – TS prevodilac će prijaviti greške da promenljive / objekti iz postojećeg JS koda nisu definisani. Deklaracijom svih objekata / promenljivih u postojećem JS kodu sa “declare”, kao u prikazanom kodu iznad, TS prevodilac “zna” kog tipa su te promenljive / objekti, i može da prevede takav kod – kombinaciju starog / postojećeg JS koda koji se koristi sa novim TS kodom.

1. TS Any & enum

Enum abc { a, b, c }, any je bilo sta

1. TS default & rest parametri

fun(a : number, b :string = “test”), fun2(… argArray: number [])

1. Deklaracija funkcija

var f = function( a: number, b: number) : returnType – anonimna

function fja( a?:number) : boolean{} – opcioni argument

1. union & type

var unionType : string | number

typeof a === “string”

1. object res & spread

const { name, website, twitterHandle } = marius;

name; // Type string

website; // Type string

twitterHandle; //

const { twitterHandle, ...rest } = marius;

twitterHandle; // Type string

rest; // Type { name: string; website: string; }

1. Class & Interface

Interface IComplexType { id: number, name: string}

let cType : IComplexType; cType = { id: 1, name: ‘ime’};

Interface IPrint{Print(): void}

Function abc implements IPrint(){print(){ }

1. TS decorators

A Decorator is a special kind of declaration that can be attached to a class declaration, method, accessor, property, or parameter. Decorators use the form @expression, where expression must evaluate to a function that will be called at runtime with information about the decorated declaration.

1. TS generic types

function getArray<T>(items : T[] ) : T[] {

return new Array<T>().concat(items);

}

let myNumArr = getArray<number>([100, 200, 300]);

let myStrArr = getArray<string>(["Hello", "World"]);

1. TS promise

Graphical user interface, text, application

Description automatically generated

1. TS async / await

Graphical user interface, text, application

Description automatically generated

1. Šta je angular?

Angular je frontend razvojni okvir nastao 2012. definiše pogodnu strukturu klijentske web aplikacije, koja se sastoji od komponenata (components), koje su osnovni gradivni elementi Angulara(basic building blocks) i modula

1. Kako se startuje angular aplikacija?

ng new nazivProjekta

ng serve (pokretanje)

1. Struktura angular aplikacije

node\_modules i package json, src. U src:

* app (svi fajlovi aplikacije)
* assets (resursi)
* environments (okruzenje – razvojno, produkciono)

1. angular main.ts, index.html, index.css

glavni fajl aplikacije, enableProdMode – fja za produkcioni mod

1. angular assets

assets folder za smeštaj resursa kao što su slike, json fajlovi i slično

1. angular app.module.ts

grupiše elemente / komponente aplikacije

1. angular components

Component je dekorator koji označava klasu AppComponent kao Angular komponentu I sadrži metapodatke: selector je naziv elementa za referenciranje šablona komponente na html strani, templateUrl je naziv spoljnjeg šablona komponente, template je šablon koji se definiše lokalno u samoj komponenti, styleUrl je naziv eksternog stila komponente. Kao I template, i stil se može zadati lokalno sa style i definicijom stila u navodnicima, kao i za template. title je podatak član klase AppComponent koji se može prikazati na web strani.

1. angular interpolacija

Text interpolation lets you incorporate dynamic string values into your HTML templates. <h3>Current customer: {{ currentCustomer }}</h3>

1. angular pipes

Use pipes to transform strings, currency amounts, dates, and other data for display. Pipes are simple functions to use in template expressions to accept an input value and return a transformed value.

{{ datum | date }} {{ amount | currency:'EUR' }}

1. angular property binding

Property binding in Angular helps you set values for properties of HTML elements or directives. Use property binding to do things such as toggle button functionality, set paths programmatically, and share values between components.

<button [disabled]="isUnchanged">Disabled Button</button>

1. angular attribute, style & class binding

attribute:

<button (click)=”title=’zdravo’”>srb</button>

style: <nav [style.background-color]="expression"></nav>

class: [class.sale]="onSale"

1. angular event binding

<button (click)=”onClick()’”>save</button>

1. ngClass & ngStyle

Adds and removes CSS classes on an HTML element. <some-element [ngClass]="'first second'">...</some-element>

An attribute directive that updates styles for the containing HTML element. Sets one or more style properties, specified as colon-separated key-value pairs. <some-element [ngStyle]="{'font-style': styleExp}">...</some-element>

1. ngModel (direktiva)

Creates a FormControl instance from a domain model and binds it to a form control element. <input [(ngModel)]="name" #ctrl="ngModel" required>

1. ngIf, ngFor & ngSwitch

**ngIf** is a structural directive that conditionally includes a template based on the value of an expression coerced to Boolean. <div \*ngIf="condition; else elseBlock">Content to render when condition is true.</div>

<ng-template #elseBlock>Content to render when condition is false.</ng-template>

The core directive **ngFor** allows us to build data presentation lists and tables in our HTML templates.

<tr \*ngFor="let hero of heroes">

<td>{{hero.name}}</td>

</tr>

The **[ngSwitch]** directive on a container specifies an expression to match against. The expressions to match are provided by ngSwitchCase directives on views within the container.

<container-element [ngSwitch]="switch\_expression">

<some-element \*ngSwitchCase="match\_expression\_1">...</some-element>

...

<some-element \*ngSwitchDefault>...</some-element>

</container-element>

1. angular Input / Output

A common pattern in Angular is sharing data between a parent component and one or more child components. Implement this pattern with the @Input() and @Output() decorators. @Input() lets a parent component update data in the child component. Conversely, @Output() lets the child send data to a parent component.

Timeline

Description automatically generated

1. angular dependency injection / service

Dependencies are services or objects that a class needs to perform its function. Dependency injection, or DI, is a design pattern in which a class requests dependencies from external sources rather than creating them.

@Injectable({

providedIn: 'root',

})

constructor(private logger: Logger)

1. angular rutiranje

Text

Description automatically generated

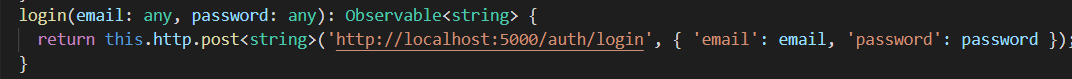
1. ng build

Sa ng build <projekat> [options] dobija se prevedena verzija projekta, tako što se generišu statički fajlovi \*.html, \*css i \*.js u folderu dist(ribution). Ovi fajlovi se mogu servirati bilo kojim web serverom.

1. angular HttpClient / Observable / subscribe – slanje zahteva serveru / prijem podataka

This service is available as an injectable class, with methods to perform HTTP requests.

this.httpClient.request('GET', this.heroesUrl + '?' + 'name=term', {responseType:'json'});



Text

Description automatically generated

1. Mongodb struktura baze

Sastoji se od jedne ili više kolekcij, kolekcije se sastoje od dokumenata, dokument je osnovna jedinica podataka MDB baze. Struktura MDB dokumenta je kao JSON koji se čuva u binarnom JSON formatu radi brzine i efikasnosti rada sa podacima

1. Kreiranje mongodb baze – nodeJS

var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/mydb";

MongoClient.connect(url, function(err, db) {

if (err) throw err;

console.log("Database created!");

db.close();

});

1. mongodb upiti – nodeJS

var query = { address: "Park Lane 38" };

dbo.collection("customers").find(query).toArray(function(err, result)

1. kako se kombinuju mongodb i express u nodeJS?

var express = require('express');

var app = express();

var mongoose = require('mongoose');

mongoose.connect('mongodb://localhost//:20717');

app.listen(port);

console.log(“Listening on port ' + port);