

1) WWW – osnove (HTTP, HTML, URL)

WWW (World Wide Web) je sistem međusobno povezanih hipertekst dokumenata i multimedijalnog sadržaja kome se pristupa korišćenjem web pretraživača. Web stranice, multimedijalni sadržaj i drugi tipovi dokumenata se prenose putem HTTP-a (Hypertext Transfer Protocol). HTTP je stateless protokol, što znači da je svaki request između klijenta i servera nezavisan i sadrži sve informacije potrebne za obradu requesta.

HTML (Hypertext Markup Language) je standardni markup jezik za struktuiranje web stranice. On obezbeđuje način za opisivanje strukture stranice korišćenjem tagova. Tagovi se navode između znakova '<' i '>'.

URL (Uniform Resource Locator) je adresa koja se koristi za pristupanje resursima na internetu. URL se obično sastoji od naziva protokola kojim se podaci prenose (HTTP, HTTPS, FTP...), domena (human-readable adresa koja identifikuje resurs na internetu), puta do resursa (npr. "/proizvodi/laptopovi") i query parametara (omogućavaju prosleđivanje parametara resursu npr. "?q=search&category='elektronika'").

2) Web aplikacije

Web aplikacija je softverska aplikacija kojoj se pristupa preko interneta korišćenjem web pretraživača. Web aplikacija se sastoji od serverske i klijentske strane i mehanizma za perzistenciju (baze). Web aplikacije se razlikuju od standardnih desktop aplikacija jer ne zahtevaju instalaciju, već se hostuju na web serveru i pristupa im se korišćenjem url-a. Najznačajnije karakteristike web aplikacije su pristupačnost i nezavisnost od platforme, skalabilnost, održivost... Primeri web aplikacija su YouTube, Facebook, Lichess...

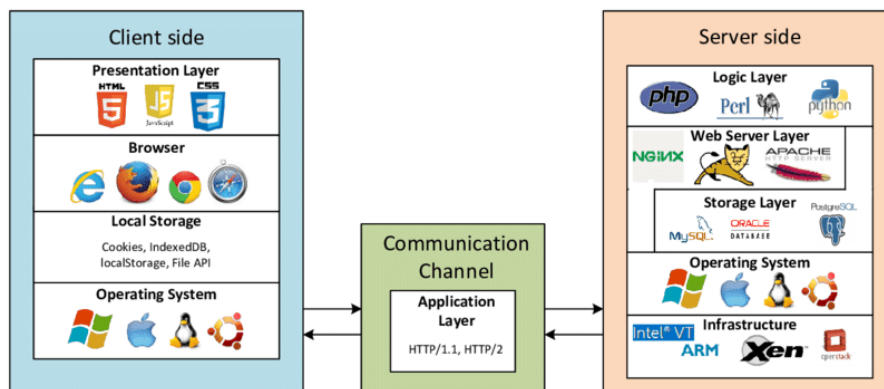
3) Desktop vs Mobilne vs Web aplikacije

Desktop aplikacije su dizajnirane da se izvršavaju na određenoj platformi (operativnom sistemu) i direktno se instaliraju na korisnički računar. Ažuriranje aplikacije zahteva dodatnu instalaciju, međutim mogu se koristiti i offline. Omogućavaju visoke multimedijalne performanse (3d igre), ali je dodavanje multimedijalnog sadržaja komplikovano.

Mobilne aplikacije su dizajnirane da se izvršavaju na određenoj mobilnoj platformi (najčešće iOS ili Android operativnom sistemu). Korisnici aplikacije najčešće preuzimaju iz odgovarajućeg app store-a, koji proizvođač operativnog sistema omogućuje. U zavisnosti od namene aplikacije moguće je koristiti ih i offline.

Web aplikacije ne zavise od platforme koju korisnik koristi. Izvršavaju se na web serveru i pristupa im se putem url adrese. Ažuriranje aplikacije ne zahteva dodatne instalacije od strane korisnika. Imaju limitiran GUI, ali se multimedijalni sadržaj jednostavno dodaje. Nije im moguće pristupiti bez pristupa internetu.

4) Arhitektura Web aplikacije (dijagram, cloud)



Cloud se odnosi na postupak korišćenja udaljenih servera i mreža, koji su hostovani na internetu, kako bi se pamtili, upravljani i obrađivali podaci, umesto da se oslanja na lokalne resurse. Cloud se koristi u raznim domenima, uključujući i treniranje modela veštačke inteligencije, prikupljanje i obrada podataka, IoT... Glavne prednosti korišćenja clouda su skalabilnost, fleksibilnost, accessibility, cost-efficiency...

5) Klijent strana (tehnologije, izazovi)

Klijentska strana web aplikacije se odnosi na deo aplikacije koji se izvršava u korisničkom pretraživaču. Osnovne komponente klijentske strane su HTML (Hypertext Markup Language) koji se koristi za struktuiranje web strane, CSS (Cascading Style Sheet) koji se koristi za stilizaciju web stranice i JavaScript – programski jezik kojim se dodaje interaktivnost i dinamičnost web stranice. Postoji i veliki broj JavaScript frameworkova koji služe za razvoj klijentske strane (React, Angular, Vue, jQuery).

PWA (Progressive Web Application) su web aplikacije koje kombinuju najbolje osobine web i mobilnih aplikacija. Omogućavaju offline rad (keširaju određene podatke) kao i push notifikacije.

WebAssembly je tehnologija koja omogućava izvršavanje aplikacija pisanih u različitim programskim jezicima da se izvršavaju u web browseru.

Izazovi koji postoje prilikom projektovanja klijentske strane su UX (obezbeđivanje user experience-a pr. response time vs responsiveness), UI (obezbeđivanje odgovarajućeg korisničkog interfejsa), obezbeđivanje adekvatnih performansi, omogućavanje cross-browser i cross-device izvršavanje (različite rezolucije ekrana – responsive design), kao i odabir odgovarajuće tehnologije koja je pogodna za konkretan slučaj korišćenja.

6) Server strana (tehnologije, izazovi)

Server strana aplikacije se odnosi na deo aplikacije koji se izvršava na serveru. Zadužena je za obrađivanje korisničkih requestova, izvršavanje osnovne poslovne logike aplikacije i pristup bazi podataka. Tehnologije koje se koriste za izradu logičkog sloja serverske strane su PHP, ASP.NET, Node.js, Python's Django framework, Ruby... Za sam web server se koriste IIS, Apache, Nginx... U zavisnosti od zahteva, mogu se koristiti relacione baze podataka, NO-SQL baze, graf ili in-memory baze...

Izazovi koje se prevazilaze prilikom izrade serverske strane su skalabilnost (kluster i load balanceri), sigurnost (autentifikacija, autorizacija, sql-injection, DDOS), perzistencija podataka....

7) Protokoli (HTTP 1.0, 1.1, 2.0, Websocket)

HTTP i Websocket su protokoli koji se koriste za uspostavljanje veze između klijenata i servera i razmenu podataka.

HTTP 1.0 je prva verzija ovog protokola. Omogućava jednostavnu request-response komunikaciju između klijenata i web servera. HTTP 1.0 je stateless protokol (protokol bez stanja) što znači da je svaki par request-response nezavisan, tj. svaki request sadrži dovoljno informacija kako bi bio obrađen.

HTTP 1.1 je uveden kao poboljšanje nad HTTP 1.0. Uvedena je keep-alive konekcija (perzistencija konekcije) kako bi se izbegao problem uspostavljanja konekcije pri svakom zahtevu. Uveden je pipelining, što omogućava da se šalje više requestova kroz jednu konekciju bez čekanja odgovora.

HTTP 2.0 je uveden kako bi se prevazišli problemi sa performansama koji postoje kod HTTP 1.1. Uveden je multiplexing (slanje više requesta i responsea kroz istu konekciju) i server push, što omogućava serveru da inicira slanje podataka.

Websocket je protokol koji omogućava full-duplex komunikacione kanale preko jedne tcp konekcije. Razlikuje se od HTTP-a po tome što omogućava bidirekcionu komunikaciju u realnom vremenu.

HTTPS (HTTP Secure) je ekstenzija standardnog HTTP protokola koja se koristi radi sigurnije komunikacije na internetu. HTTPS vrši enkripciju podataka koji se šalju kroz mrežu, čineći ih beskorisnim bilo kom osim korisniku. Kriptografija se vrši kriptografskim protokolima kao što su TLS kod HTTP-a i SSL kod HTTPS-a. Ovim je prevaziđena bojazan od Man-in-the-Middle napada.

8) HTTP – opis, metode zahteva, poruke odgovora

HTTP (Hypertext Transfer Protocol) je protokol na aplikativnom nivou koji se koristi za prenos hipertekst dokumenata (HTML) i drugih dokumenata. HTTP je osnova razmene podataka na webu. HTTP je stateless protokol (protokol bez stanja) što znači da je svaki par request-response nezavisan, tj. svaki request sadrži dovoljno informacija kako bi bio obrađen.

Osnovne HTTP metode su

- GET – koristi se za zahtevanje podataka od servera
- POST – koristi se za slanje podataka server (popunjena forma, upload dokumenata)
- PUT – koristi se za ažuriranje resursa na serveru
- DELETE – koristi se za uklanjanje resursa na serveru

Kod HTTP 1.0 svaki request zahteva uspostavljanje konekcije. Nakon slanja requesta vraća se poruka i to:

- 1xx – informativna
- 2xx – uspeh

- 3xx – redirekcija
- 4xx – greška kod klijenta
- 5xx – greška na serveru

9) HTML 5

HTML (Hypertext Markup Language) je podskup XML-a. Osnovna primena HTML-a je struktuiranje sadržaja na web stranicama. HTML dokument se sastoji od elemenata (`<div></div> ...`), atributa (`<div style='...'>...`), komentara (`<!-- ... -->`) i znakovnih referenci (`" = '`). HTML dokument se sastoji iz Doctype (informacije o dokumentu), zaglavlja (informacije o dokumentu) i tela dokumenta koje sadrži sadržaj tj skup HTML elemenata.

HTML elementi se sastoje iz početnog dela (taga `<div>`) i završnog dela (`</div>`). Neki elementi nemaju početni i krajnji deo (npr. `
`).

HTML elementi mogu biti blokovi ili inline. Blokovski elementi zauzimaju ceo red (širinu ekrana) u kome se nalaze. Inline elementi ne zahtevaju ceo blok. Svaki HTML element ima pridružen CSS objekat i uokviren je kutijom (box model) čiji je izgled definisam tim CSS objektom. Box model podrazumeva marginu, border, padding i content.

HTML5 je peta revizija HTML-a. Novi elementi koji se javljaju kod HTML 5 su semantički elementi (header, nav, section, article, footer, aside). Oni predstavljaju samostalni deo strane ili aplikacije koji može da se distribuira i koristi.

-

Da li bi sadržaj mogao samostalno da se prikaže kao RSS unos? Da => `<article>`

Da li je se sastoji od međusobno povezanog sadržaja? Da => `<section>`

Ukoliko ne postoji nikakva semantička veza, koristiti `<div>`

Aside se koristi za sadržaj koji nije glavni fokus artikla ili strane, ali **se odnosi** na artikal
Može biti napisan u okviru artikla

Ne znači da mora da bude lociran bočno od artikla

Nav predstavlja deo strane koji sadrži linkove ka ostalim stranama ili delovima na strani (navigacione linkove)

Ne treba svaku grupu linkova ubaciti u `<nav>`, već samo glavne navigacione blokove.

Npr, footer sadrži linkove ka copyright-u, uslovima korišćenja sajta i glavnoj strani. Njima ne treba `<nav>`, već je dovoljan sam `<footer>` element

-

Pravila za pisanje HTML 5:

Koristiti samozatvarajuće početne delove elemenata (oni koji imaju / na kraju i kojima ne treba završni deo, npr. `
`) samo za prazne elemente

Sva imena elemenata i atributa se pišu malim slovima

Svi atributi se pišu između apostrofa (') ili navodnika (")

Dodati blanko pre / kod elemenata koji imaju samozatvarajući početni deo, npr: `
` umesto `
`

Uključiti završne delove za elemente koji mogu imati sadržaj ali se ostavljaju prazni (npr. "", ne "")

10) CSS, opšte, progresivno poboljšanje

Progresivno poboljšanje je pristup razvoju web stranica koji podrazumeva da osnovna funkcionalna verzija web sajta dostavlja svim korisnicima, bez obzira na korišćeni uređaj, bio on PC, laptop ili telefon i bez obzira na korišćeni pretraživač. Počinje se sa osnovnim funkcionalnostima (HTML-om). Ceo sadržaj se semantički označava i dodaje mu se stil, korišćenjem spoljašnjeg CSS-a. Ponašanje se isključivo definiše preko eksternog javaskripta.

CSS (Cascading Style Sheets) je jezik koji se koristi za definisanje izgleda HTML dokumenta u pretraživaču. CSS objekti se sastoje od mnoštva atributa. Najkorišćeniji su border (izgled okvira), margin (udaljenost od drugih elemenata), padding (udaljenost okvira od sadržaja), font (izgled slova), background (izgled pozadine)... CSS stil se elementu može pridružiti na tri načina: 1) U style elementu u okviru strane

2) U style atributu u okviru samog elementa

3) U eksternom css fajlu.

Elementi se iz eksternog fajla mogu selektovati na više načina: selekcijom tipova, klasa, ID-a, HTML atributa, kombinacijom pravila.

11) CSS selektori

Elementi mogu biti selektovati na više načina:

- 1) Korišćenjem tipova – navođenjem tipa elementa, CSS stil će biti primenjen na sve elemente tog tipa. Tipove je moguće grupisati (h1, h2, h3 {...})
- 2) Selekcijom klasa – svaki element može pripadati jednoj ili više klasa. To se navodi u class atributu. Sintaksa koja se koristi za gađanje klasa je .<naziv klase>
- 3) Selekcijom id – Sintaksa koja se koristi je #<naziv klase>
- 4) Selekcijom HTML atributa - <element>[attr="attr_val"]
- 5) Selekcijom atributa *= sadrži pr. [class*="col-"]
^= počinje sa
\$= završava se sa
- 6) parent > child – Korišćenjem > selektuju se deca
- 7) parent desc – Korišćenjem ,, ,, selektuju se svi potomci

Svaki HTML element kaskadno nasleđuje karakteristike CSS objekta pridruženog roditeljskom HTML elementu.

JAVASCRIPT

1) JavaScript (JS), opšte

JavaScript je lightweight interpretirajući (ili just-in-time kompajlirani) programski jezik sa first-class funkcijama. Iako je JavaScript je najpoznatiji kao skripting jezik za razvoj web stranica, veliki broj non-browser okruženja ga takođe koristi (Node.js, Apache CouchDB, Adobe Acrobat). JavaScript je prototype-based, podržava više paradigmi, single-threaded, dinamički tipiziran.

JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation (via eval), object introspection (via for...in and Object utilities), and source-code recovery (JavaScript functions store their source text and can be retrieved through toString()).

Javascript – univerzalni jezik klijent strane

Skript se izvršava na klijentskoj mašini pošto se dokument učitava, ili u nekom drugom trenutku, npr. kad se aktivira link.

Skripte pružaju autoru sredstva za proširenje funkcionalnosti i interaktivnosti HTML dokumenata:

- dinamički menjati sadržaj dokumenta

- validacija korisničkih podataka u obrascu, kako bi se detektovale greške pre slanja podataka serveru.

- obrada događaja koji se odnose na dokument, kao što su učitavanje i zatvaranje dokumenta, fokus elementa, pokreti kursora miša, itd.

- Komunikacija sa serverom (AJAX, Web sockets)

Najpopularniji jezik Web-a, zastupljen na klijentu (Web pretraživači), na serveru (Node.js) ali i van Web-a: PDF, desktop widžeti

Standardizovan ECMAScript specifikacijom

Java i JavaScript su semantički potpuno različiti jezici

Sintaksa Javascripta je skoro identična Javi, C# ili PHP-u.

Na klijentu se tradicionalno implementira kao interpretatorski jezik

Prototipski skript jezik dinamičkih tipova

- Prototipski bazirano programiranje – stil objektno orijentisanog programiranja gde se nasleđivanje postiže kloniranjem postojećih objekata, koji služe kao prototipi

- Dinamički tipovi – ista promenjiva može da menja tip podatka

- `var k = 5;`

- `var name = "Hello";`

- `name = 4;`

- JS objekti su asocijativni nizovi

- `obj.x ⇔ obj['x']`

- Javascript je case-sensitive!

2) JS tipovi (Primitivni, objekti)

U primitivne tipove u JS spadaju string, number, boolean, undefined (let x), null (let y = null), symbol (uvedeni u ecma6, predstavljaju jedinstveni identifikator, const mySymbol = Symbol('description')).

U objekte spadaju:

- 1) Anonimni objekti const person = { name: 'John', age: 30 }
- 2) Nizovi const numbers = [1, 2, '3', 4, '5']
- 3) Funkcije function add(a, b) { return a + b; }
- 4) Date const curDate = new Date();
- 5) Regex
- 6) Map const myMap = new Map();
myMap.set('key1', 'val1');
- 7) Set

3) Promenljive

Promenljive se u JS mogu deklarirati na 3 načina.

- 0) var – istorijski, danas nije često u upotrebi
- 1) let – koristi se za promenljive koje je moguće redeklarirati
- 2) const – koristi se za promenljive koje ne bi trebale biti redeklarirane

Identifikator promenljive može da sadrži brojeve, slova, dolar i _. Ne sme da počinje brojem.

4) – 7) Funkcije, razlika između deklaracije i FE, hoisting, lambda fje

```
function max(x, y) {  
  if (x > y) return x;  
  return y;  
}  
const c = max(3, 4)
```

FE

```
var square2 = function(a) {  
  return a * a;  
}
```

Korišćenjem arrow notacije

```
var square2 = a => a * a
```

Hoisting je ponašanje u JS koje podrazumeva da su deklaracije promenljivih i funkcija pomerene na vrh u fazi kompajliranja. Ovo omogućava korišćenje varijabli i funkcija u kodu pre nego što su deklarirane. Hoistuju se samo deklaracije, ne i definicije.

Funkcije i FE se razlikuju po tome što su funkcije hoistovane, moguće im je pristupiti u kodu pre njihove deklaracije, dok sa FE to nije moguće. FE su korisni kada je funkciju potrebno proslediti kao parametar neke druge funkcije.

Lambda ili anonimne funkcije su funkcije bez imena. Moguće je pisati ih korišćenjem arrow notacije ili korišćenjem ključne reči function. Često se koriste kao callback funkcije ili IIFE (immediately invoked function expressions).

```
// Anonymous function as a callback
document.addEventListener("click", function() {
  console.log("Document clicked!");
});

// IIFE (Immediately Invoked Function Expression)
(function() {
  console.log("I am an IIFE!");
})();
```

8) Nizovi, ugrađene metode

Niz je struktura koja omogućava čuvanje i manipulaciju elementima koji se nalaze u njemu. Nizovi u JS mogu biti heterogeni, tj. sadržati različite tipove.

deklaracija niza: let niz = []

let niz = [1, 2, "str"]

Elementima se može pristupiti pomoću indeksa (niz[0])

Novi element se u niz dodaje korišćenjem metode .push()

Poslednji element se izbacuje korišćenjem metode .pop(),

Prvi element se izbacuje korišćenjem metode .shift()

Element se umeće na prvo mesto korišćenjem .unshift()

Slajsovanje .slice(od, do)

Splajsovanje .splice(od, do, vals)

Duzina niza .length

.foreach()

Filter metoda kreira novi niz koji sadrži elemente originalnog niza koji ispunjavaju uslov koji se nameće funkcijom koja se prosleđuje kao parametar. Ne modifikuje originalni niz. Dužina novog niza može biti različita od dužine originalnog.

Map metoda kreira novi niz koji sadrži elemente nad kojima je primenjena funkcija koja je prosleđena kao parametar. Ne modifikuje originalni niz. Dužina ovog niza jednaka je dužini originalnog.

Reduce metoda primenjuje prosleđenu funkciju nad akumulatorom, u redosledu sleva nadesno, dok ne ostane 1 element.

Metode je moguće nadovezivati.

const numbers = [1, 2, 3, 4, 5, 6];


```
// Chaining filter, map, and reduce
const result = numbers
  .filter(function(number) {
    return number % 2 === 0;
  })
  .map(function(number) {
    return number * 2;
  })
  .reduce(function(accumulator, currentNumber) {
    return accumulator + currentNumber;
  }, 0);
```

console.log(result);

Moguće je koristiti arrow notaciju.

10) Closure i kontekst izvršenja funkcija

Kontekst izvršenja se sastoji od skupa promenljivih funkcije, lanca opsega i this pokazivača. Skup promenljivih funkcije čine unutrašnje promenljive, deklaracije funkcija i parametri funkcije. Lanac opsega predstavlja skup promenljivih funkcije i svih roditeljskih opsega.

Closure je osobina javascripta koja omogućava da funkcija zadrži pristup promenljivama iz spoljašnjeg opsega, čak i kada je funkcija iz spoljašnjeg opsega završila izvršavanje. Ovo se omogućava postojanjem scope chain-a u kontekstu izvršenja. Closure se može koristiti za enkapsulaciju podataka, i kreiranje privatnih promenljivih.

Execution context	
Variable object	{ vars, function declarations, arguments... }
Scope chain	[Variable object + all parent scopes]
thisValue	Context object

```
function counter() {
  let count = 0;

  return function() {
    count++;
    console.log(count);
  };
}

const increment = counter();
increment();
increment();
```

```
function createPerson(name) {
  let privateName = name;
  return {
    getName: function() {
      return privateName;
    },
    setName: function(newName) {
      privateName = newName;
    }
  };
}

const person =
createPerson('John');
console.log(person.getName());
person.setName('Jane');
console.log(person.getName());
```

11) IIFE (+klasa sa privatnim članovima)

IIFE (Immediately Invoked Function Expression) je funkcijski izraz koji se definiše i izvršava odmah nakon kreiranja. Koristi se za kreiranje privatnog scope-a za promenljive, kako ne bi došlo do polluting global scope-a, aliasing članova (davanje drugih imena) i za globalne objekte (window u browseru?).

```
const counter = (function (pocetnaVrednost) {
  let counter = pocetnaVrednost;
  function display() {
    console.log(`brojac = ${counter}`);
  }
  function reset() {
    counter = 0;
    console.log("resetovan");
  }
  function increaseCounter() {
    counter++;
    display();
  }
  return {
    increase: increaseCounter,
    reset: reset
  }
})(5);
```

Kreiranje privatnih članova pre uvođenja standarda ES-6 vršilo se korišćenjem funkcija i iskorišćavanjem closure-a. Primer u odgovoru 10.

12) Objekti literali

Objekti literali su comma-separated lista key-value parova, oivičeni sa { }. Svaki key-value par predstavlja polje objekta. Pr.

```
const person = {
  firstName: 'John',
  lastName: 'Doe',
  age: 30,
  isStudent: false,
  greet: function() {
    console.log(`Hello, my name is ${this.firstName} ${this.lastName}.`);
  }
};
```

Mogu se i referencirati promenljive. Pr firstName. Vrednosti ključeva mogu biti i dinamičke. U tom slučaju navode se između []. Pr [key]. ES6 omogućava da se unutar definicije objekta literala definiše funkcija. Moguć je i unzip (kao tuple u pythonu).

9) DOM, definicija i najčešće funkcije API-ja

DOM (Document Object Model) je standardizovani API (način manipulacije) za strukturu dokumenata (XML, HTML...). DOM nudi standardni skup objekata za predstavljanje HTML i XML dokumenata, kao i standardni interfejs za njihovu manipulaciju i pristup. Pre DOM-a svaki web pretraživač imao je drugačiji pristup objektima, što je sprečavalo cross browser aplikacije. DOM svaki dokument vidi kao stablo.

Najčešće korišćeni načini za pretragu HTML dokumenata u DOM stablu su

`document.getElementById("nekiId")`

Vraća HTML element sa navedenim ID-om.

`element.getElementsByTagName("button")`

Vraća niz HTML elemenata navedenog tipa koji su potomci element-a

`document.querySelectorAll("table td input[type='checkbox']")`

Izvršava CSS selekciju i vraća niz elemenata

`document.querySelector("table td input[type='checkbox']")`

Izvršava CSS selekciju i vraća prvi nađeni element

Najčešće korišćeni atributi su:

atribut	objašnjenje
style	pristup CSS objektu trenutnog HTML objekta
innerHTML	sadržaj elementa (između početnog i krajnjeg dela elementa)
value	(kod input elementa) vrednost elementa
parentNode	HTML objekat koji sadrži trenutni HTML objekat
childNodes	Niz potomaka, HTML objekata (elemenata, tekstualnih čvorova i za formatiranje)
children	Niz potomaka, isključivo HTML elemenata