1. **Što je CSS?**

CSS (Cascading Style Sheets) je jezik za oblikovanje web-stranica. Koristi se za definiranje izgleda i rasporeda elemenata HTML dokumenata, omogućujući dizajnerima i programerima da prilagode boje, fontove, margine, razmake i druge vizualne aspekte web-stranica.

1. **Navedite osnovne dijelove CSS pravila.**

Selektor, svojstvo, vrijednost

1. **Na koje sve načine možemo primijeniti CSS na HTML?**

Možemo primijeniti unutarnji ili vanjski CSS.

1. **Koji je aktualni CSS standard?**

CSS3 (kompatibilan s prijašnjim verzijama)

1. **Kako i zašto koristimo oznake proizvođača web preglednika?**

Razvoj CSS-a podijeljen je na module, neki se koriste službeno dok je drugi dio samo umjereno stabilan za uporabu. Dok se ne usvoje kao dio web standarda za postavljanje svojstva moramo koristiti oznake proizvođača web preglednika (npr. -webkit- za chrome, safari, opera)

1. **Na koje sve načine možemo definirati boju unutar CSS-a?**

Nazivom boje (red, blue…), vrijednošću boje (255,0,0 crvena…), heksadecimalnim kodom boje (#FF0000 crvena), hsl zapisom (0,100%,75% svijetlo crvena boja)

1. **Opišite osnovne vrste fontova – generički fontovi.**

Serifni - mali ukrasi na rubovima znaka, najviše se koriste za ispis povećanih dijelova teksta (Times New Roman)

Neserifni - nema rubne ukrase (povećana čitljivost), ispis većeg dijela sadržaja web stranica (Arial, Verdana…)

Ukošeni, dekorativni i fontovi fiksne širine

1. **Koja svojstva koristimo za uređivanje fonta znakova (stil i dizajn znakova)?**

font-family, @font-face, font-size, font-weight, font-style, line-height, text-align, text-decoration, text-transform, letter-spacing itd.

1. **Na koje sve načine možemo uređivati tekst (razmještaj i format znakova)?**

text-align, text-indent, text-transform, text-decoration, letter-spacing, word-spacing…

nsku boju koristimo komandu *background-color* (skraćeno *bg-color* itd. itd.). Da bude transparentna samo dodamo broj koliko da bude transparentno npr. *bg-color:* *rgba(255, 0, 0, 0.5)* 0.5 stoji za 50% transparentnosti.

1. **Koje svojstvo koristimo za primjenu pozadinske slike?**

*background-image (bg-image)*. Primjer: *bg-image: url(‘putanja do slike’)*

1. **Na koje parametre pozadinske slike možemo utjecati?**

Poziciju, veličinu, način postavljanja, dali je fixna ili floata sa sadržajem, način miješanja boja pojedinih slojeva sa pozadinom itd.

1. **Koja je inicijalna pozicija pozadinske slike?**

Inicijalna pozicija pozadinske slike u CSS-u je *background-position: 0% 0%;*. To znači da se slika pozicionira u gornjem lijevom uglu elementa.

1. **Možemo li određivati dimenzije pozadinske slike i na koje načine?**

Možemo komandom background-size (bg-size). Postoji cover (zauzme cijelu pozadinu), contain (skalira da se u potpunosti uklopi u element, čuva omjer slike). Specifične dimenzije tipa *bg-size: 300px 200px;; 50% 50%;; 10em 5em;; 50vw 50vh;;*

1. **Koje vrste gradijenta postoje u CSS-u?**

Linearni - pravocrtni smjer

Radijalni - koncentrični smjer u odnosu na postavljen centar

Konusni - rotacija tranzicije boja oko zadanog centra

1. **Opišite ukratko box model**

Box model je temeljni koncept u CSS-u koji opisuje kako se elementi prikazuju na web stranici.

1. **Što određuju svojstva width i height?**

Određuju širinu i visinu elementa/boxa

1. **Što određuje svojstvo padding?**

Svojstvo padding u CSS-u određuje prostor unutar elementa, između sadržaja elementa i njegovog okvira (border).

1. **Što je margina elementa?**

Određuje prostor izvan elementa, između njegovog okvira (border) i susjednih elemenata.

1. **Koji je zadani način kako web preglednici računaju dimenzije HTML elemenata?**

Putem standard box modela. Ukupna širina i visina elementa se zbroje sa visinom i širinom sadržaja, padding i border.

1. **Opišite dvije osnovne skupine HTML elemenata te objasnite kako se ovo svojstvo može promijeniti.**

Blok elementi i inline elementi. Mogu se promijeniti koristeći CSS svojstvo display. To svojstvo omogućuje kontrolu nad načinom na koji se elementi prikazuju i ponašaju u rasporedu.

1. **Što omogućuje CSS tehnika Flexbox?**

Omogućuje fleksibilan i efikasan način za stvaranje kompleksnih rasporeda i dizajna na web stranicama

1. **Što predstavlja fleksibilan spremnik (*Flex-container*), a što fleksibilan element (*Flex-item*)?**

**Flex-container** -  element koji postaje fleksibilni kontejner kada mu se dodijeli svojstvo display: flex; ili display: inline-flex;. Ovaj element postaje roditeljski kontejner za sve svoje direktne potomke, koji postaju fleksibilni elementi.  
**Flex-item** - Fleksibilni elementi su direktni potomci fleksibilnog kontejnera. Oni nasljeđuju svojstva fleksibilnog kontejnera i mogu imati dodatna svojstva koja utječu na njihov raspored i veličinu.

1. **Kako definiramo fleksibilan spremnik?**

Definira se dodjeljivanjem svojstva display: flex; ili display: inline-flex; HTML elementu. Ovo svojstvo pretvara element u fleksibilni kontejner, a sve njegove direktne potomke postaju fleksibilni elementi (flex items).

1. **Koji je zadani smjer raspoređivanja elemenata u fleksibilnom spremniku te kako se on može promijeniti?**

Dd lijevo na desno. Ovaj smjer se može promijeniti koristeći CSS svojstvo flex-direction.

1. **Koje svojstvo omogućuje raspoređivanje elemenata u više redova / stupaca?**

Koristi se svojstvo flex-wrap. Ovo svojstvo određuje da li će se fleksibilni elementi prelaziti u novi red kada nema dovoljno prostora u trenutnom redu ili stupcu.

1. **Koja svojstva određuju poravnanje elemenata po glavnoj i poprečnoj osi te raspodjelu slobodnog prostora?**

justify-content , align-items , align-content

1. **Kako možemo zadati dimenzije fleksibilnog elementa po glavnoj osi?**

Koristeći flex-basis. Ovo svojstvo određuje početnu veličinu fleksibilnog elementa prije nego što se primijene svojstva flex-grow i flex-shrink.

1. **Navedite i opišite nekoliko metoda transformacija.**

2D transformacije - traslate(x,y), rotate(deg), scale(x,y), skew(deg)...  
3D transformacije - rotatex(deg), rotatey(deg)

1. **Što omogućuju tranzicije?**

Omogućuje prijelaz elementa iz jednog stila u drugi, koji se

odvija postepeno u zadanom vremenskom trajanju

1. **Koje parametre je moguće postaviti tranziciji?**

transition-property , transition-duration , transition-delay , transition-timing-function

1. **Što su CSS animacije?**

CSS animacije omogućuju stvaranje kompleksnih animacija koje se mogu primijeniti na HTML elemente. Za razliku od tranzicija, koje su jednostavnije i koriste se za animiranje promjena svojstava između dvije stanja, CSS animacije omogućuju više kontrole i fleksibilnosti.

1. **Opišite dva osnovna koraka za definiciju CSS animacija.**

Prvi korak - pravilo @keyframes

- definira samu animaciju, njezine ključne kadrove i svojstva koja se animiraju   
Drugi korak - svojstvo animation

- dodjeljuje animaciju određenom elementu i definira kako je animiran

1. **Objasnite što predstavlja tok dokumenta. Koja metoda pozicioniranje je unaprijed zadana za HTML elemente?**

Tok dokumenta predstavlja zadani raspored HTML elemenata. Zadana metoda je da se elementi pozicioniraju prema toku sadržaja HTML dokumenta te pomaci nemaju efekta na statičan element.

1. **Koju metodu pozicioniranja koristimo ako želimo HTML element pozicionirati (pomaknuti) prema njegovoj statičnoj (normalnoj) poziciji?**

Koristimo metodu pozicioniranja relative. Ova metoda omogućuje elementu da se pozicionira relativno u odnosu na svoju normalnu poziciju u dokumentu.

1. **Koja metoda pozicioniranja omogućuje pozicioniranje HTML elementa prema granicama prvog elementa pretka, čija pozicija nije statična?**

Kada postavimo position: absolute; na element, element se pozicionira u odnosu na najbližeg predka čija pozicija nije static. Ako takav predački element ne postoji, element se pozicionira u odnosu na <html> element (tj. u odnosu na prozor preglednika).

1. **Koja je razlika između fiksnog i ljepljivog pozicioniranja?**

Fiksno se pozicionira prema prozoru web preglednika a ljepljivo se pozicionira ovisno o pomaku sadržaja stranice, element bude pozicioniran relativno ili fiksno.

1. **Na koji način se koristi CSS svojstvo z-index?**

Koristi se za kontrolu položaja elemenata duž z-osi, odnosno za određivanje koji element će se prikazati ispred ili iza drugih elemenata kada se preklapaju.

1. **Objasnite što su plutajući elementi te navedite primjere gdje se oni koriste.**

Omogućuju pozicioniranje elemenata na lijevi ili desni rub njihova elementa roditelja, uklanja se iz normalnog toka dokumenta te se ostali sadržaj omota oko float elementa.

Primjeri: omotavanje teksta oko slika, navigacijaski meni

1. **Čemu služi CSS svojstvo clear?**

Koristi se za sprečavanje plutajućih elemenata da utječu na element koji slijedi nakon njih. Ovo svojstvo je posebno korisno kada želite "očistiti" plutajuće elemente i vratiti normalan tok dokumenta, sprečavajući tako neželjena preklapanja ili rasporede.

1. **Objasnite kada je web rješenje responzivno.**

Kad se automatski prilagođava tipu i karakteristikama medija putem kojeg se koristi.

Kada omogućava dobro korisničko iskustvo, bez obzira na medij koji se koristi za pregledavanje

1. **Što predstavlja viewport? Što je potrebno postaviti preko istoimenog metapodatka svakoj HTML datoteci?**

Viewport je korisniku vidljivo područje web stranice.Moramo postaviti početni zoom (na 1.0) i širinu viewporta na širinu uređaja

1. **Objasnite na koji način se koristi @media pravilo za postizanje responzivnog dizajna.**

Koristi se za stvaranje responzivnog dizajna tako da omogućuje primjenu različitih stilova ovisno o karakteristikama uređaja, kao što su širina ekrana, orijentacija, rezolucija i drugo.

1. **Navedite smjernice za postizanje responzivnog web dizajna.**Koristiti fleksibilne jedinice, koristiti CSS flexbox i grid, koristiti media queries, uporaba responzivnih slika, postaviti viewport, responzivna navigacija, optimizacija preformansi, koristiti responzivan framework (npr. bootstrap)

**Što je HTML te koja je aktualna verzija HTML jezika?**

HTML (HyperText Markup Language) je standardni jezik za kreiranje web stranica i web aplikacija. Koristi se za strukturiranje i prikazivanje sadržaja na webu, uključujući tekst, slike, video zapise i druge multimedijalne elemente. HTML omogućava povezivanje stranica putem hiperveza, što čini web pretraživanje dinamičnim i interaktivnim.

Aktualna verzija HTML jezika je HTML5

1. **Što je HTML element te od čega se sastoji?**

HTML element je osnovna gradivna jedinica HTML dokumenta.   
 Svaki HTML element sastoji se od tri dijela:

1. Početna oznaka (start tag): Ova oznaka označava početak elementa i obično sadrži ime elementa unutar uglatih zagrada. Na primjer, <p> je početna oznaka za paragraf.
2. Sadržaj (content): Ovo je dio između početne i završne oznake. Sadržaj može biti tekst, drugi HTML elementi ili prazan prostor. Na primjer, tekst unutar paragrafa.
3. Završna oznaka (end tag): Ova oznaka označava kraj elementa i također sadrži ime elementa unutar uglatih zagrada, ali s kosom crtom ispred imena. Na primjer, </p> je završna oznaka za paragraf.

**3.Što su atributi?**

Atributi u HTML-u su dodatni informaciji koje se mogu dodati HTML elementima kako bi se definirala njihova ponašanja, izgled ili funkcionalnost. Atributi se nalaze unutar početne oznake elementa i obično imaju oblik ime\_atributa="vrijednost".

Primjer: <a href="https://www.primjer.com" target="\_blank">Posjeti web stranicu</a>

1. **Koji elementi čine osnovnu strukturu HTML dokumenta?**

<!DOCTYPE html>: Deklaracija tipa dokumenta koja definira verziju HTML-a. Za HTML5, ovo je obavezno i piše se na samom početku dokumenta.

<html>: Korijenski element koji obuhvaća cijeli HTML dokument. Sadrži sve ostale elemente.

<head>: Sadrži metapodatke o dokumentu, kao što su naslov stranice, poveznice na stilove (CSS), skripte (JavaScript), i druge konfiguracije. Ne prikazuje se direktno na stranici.

<title>: Definira naslov dokumenta koji se prikazuje u naslovnoj traci preglednika ili na kartici preglednika.

<body>: Sadrži sadržaj dokumenta koji se prikazuje u pregledniku. Ovo uključuje tekst, slike, video zapise, tablice i druge elemente.

1. **U koje dvije osnovne skupine dijelimo HTML elemente i koja je glavna razlika između njih?**

Dijelimo ih u redne i inline elemente.  
Glavna razlika između blokovnih i inline elemenata je u načinu na koji se prikazuju:

Blokovni elementi započinju na novom retku i zauzimaju cijelu širinu roditeljskog elementa.

Inline elementi ne započinju na novom retku i ne zauzimaju cijelu širinu roditeljskog elementa, već se prikazuju unutar teksta.

**6. Objasnite kako se vrši objavljivanje video i audio sadržaja na web stranici putem HTML5 standarda te opišite prednosti koje nosi takav način rada.**

Unutar elementa <source src=”” type=”” /> navodi se lokacija datoteke i oznaka njenog tipa (<video></video> i <audio></audio> elementi).  
Za lokaciju datoteke se može koristiti relativna putanja (preporučljivo) i apsolutna putanja.  
Primjer objave:  
<video width="640" height="360" controls>

    <source src="video.mp4" type="video/mp4">

    <source src="video.webm" type="video/webm">

    Vaš preglednik ne podržava video element.

</video>  
Preporuča se korištenje atributa width i height uvijek

1. **Koje formate video i audio datoteka koristimo? Ako morate odabrati jedan format za video i jedan format za audio sadržaj, koji bi to bio?**

video - mp4

audio -  mp3

**Zašto?**

• daju optimalan odnos kvalitete i veličine datoteka

• podržavaju ih svi moderni web preglednici

1. **Opišite namjenu ostalih elemenata za objavljivanje medijskog sadržaja na web stranicama (object, embed, iframe).**

Osim <video> i <audio> elemenata, HTML nudi i druge elemente za objavljivanje medijskog sadržaja na web stranicama: <object>, <embed> i <iframe>. Svaki od ovih elemenata ima svoju specifičnu namjenu i prednosti.  
 <object> Element

<object> element se koristi za ugrađivanje vanjskih resursa, kao što su dokumenti, slike, videozapisi, audio zapisi, ili drugi interaktivni sadržaji. On je vrlo fleksibilan i može se koristiti za ugrađivanje raznih vrsta sadržaja.

Primjer korištenja <object> elementa:

<object data="dokument.pdf" type="application/pdf" width="600" height="400">

    Vaš preglednik ne podržava prikaz PDF dokumenata.

</object>

* data: Definira URL ili putanju do resursa koji se ugrađuje.
* type: Definira MIME tip resursa.
* width i height: Definiraju dimenzije ugrađenog sadržaja.
* Tekst unutar <object> elementa se prikazuje ako preglednik ne podržava ugrađivanje resursa.

**<embed> Element**

<embed> element se koristi za ugrađivanje vanjskih interaktivnih sadržaja ili plug-inova, kao što su Flash animacije, PDF dokumenti, ili drugi multimedijalni sadržaji. Ovaj element je manje fleksibilan od <object> elementa, ali je jednostavan za korištenje.

Primjer korištenja <embed> elementa:

<embed src="video.mp4" type="video/mp4" width="600" height="400">

* src: Definira URL ili putanju do resursa koji se ugrađuje.
* type: Definira MIME tip resursa.
* width i height: Definiraju dimenzije ugrađenog sadržaja.

**<iframe> Element**

<iframe> element se koristi za ugrađivanje drugih HTML dokumenata unutar trenutnog dokumenta. To omogućava prikazivanje vanjskih web stranica, interaktivnih karata, videozapisa s platformi kao što je YouTube, i drugih sadržaja.

Primjer korištenja <iframe> elementa:

<iframe src="https://www.youtube.com/embed/dQw4w9WgXcQ" width="560" height="315" allowfullscreen></iframe>

src: Definira URL stranice koja se ugrađuje.

width i height: Definiraju dimenzije ugrađenog sadržaja.

allowfullscreen: Omogućava prikazivanje ugrađenog sadržaja u punom ekranu.

1. **Koji element koristimo za izradu poveznica?**

**<a></a>**

1. **Koji su njegovi atributi i koji je od njih obavezan? Zašto?**

href atribut je obvezan - on definira URL na koji vodi poveznica, bez njega poveznica ne bi imala destinaciju  
Ima još atribute poput target, title, rel, download…

1. **Kako omogućiti otvaranje poveznice u novoj kartici internetskog preglednika?**

Da bi se omogućilo otvaranje poveznice u novoj kartici internetskog preglednika, koristi se target atribut unutar <a> elementa s vrijednošću \_blank. Ovo će reći pregledniku da otvori poveznicu u novoj kartici ili prozoru.

1. **Ukratko opišite četiri tipa poveznica.**   
   Vanjska poveznica:

Opis: Vodi na stranicu koja se nalazi na drugom web mjestu ili domeni.

           Primjer:

<a href="https://www.primjer.com">Posjeti primjer.com</a>

Namjena: Koristi se za povezivanje na vanjske resurse, druge web stranice, ili dokumente koji nisu dio trenutne web stranice.

Unutarnja poveznica:

Opis: Vodi na drugu stranicu unutar iste web aplikacije ili domene.

Primjer:

<a href="/kontakt">Kontakt</a>

Namjena: Koristi se za navigaciju unutar iste web stranice ili aplikacije, omogućujući korisnicima da lako pristupe različitim dijelovima sadržaja.

Poveznica na dio stranice (anchor link):

Opis: Vodi na određeni dio iste stranice, obično označen s id atributom.

Primjer:<a href="#sekcija1">Idi na sekciju 1</a>

<h2 id="sekcija1">Sekcija 1</h2>

Namjena: Koristi se za brzu navigaciju unutar dužih stranica, omogućujući korisnicima da brzo pristupe određenim sekcijama sadržaja.

Poveznica za slanje e-pošte (mailto link):

Opis: Otvara program za e-poštu korisnika s predodređenom e-mail adresom i mogućnošću predodređenog naslova i tijela poruke.

Primjer:

<a href="mailto:korisnik@primjer.com">Pošalji e-mail</a>

Namjena: Koristi se za olakšavanje komunikacije s korisnicima, omogućujući im da lako pošalju e-mail direktno s web stranice.

1. **Navedite pet standardnih formata grafičkih datoteka za web.**

JPEG, GIF, PNG, SVG, WebP

1. **Koje elemente koristimo za ugradnju grafika u HTML dokument?**

<img> ; <figure> ; <canvas> ; <svg> ; <picture>

1. **Koji atribut elementa img je obavezan za ugradnju grafike?**

src=”putanja do slike” alt=”opis slike”

1. **Na koji način možemo definirati adresu slike?**

*Apsolutna* - sadrži URL adresu grafike smještene u vanjskoj web domeni

*Relativna* - putanja do HTML dokumenta u kojem želimo prikazati sliku, do same slike, dohvaća se iz naše web mape

1. **Što omogućuju elementi svg i canvas te koja je razlika između njih?**

svg - vektorska grafika

canvas - rasterska grafika

1. **Objasnite što su metapodaci te nabrojite najčešće korištene.**

Metapodaci u HTML-u su informacije o dokumentu koje nisu vidljive korisnicima, ali pružaju važne detalje za preglednike i pretraživače. Nalaze se unutar <head> elementa i koriste se za opisivanje sadržaja, postavljanje konfiguracija i poboljšanje SEO-a. tipa <author>, <keywords>, <description>, <charset>, <viewport>

1. **Koliko razina naslova postoji u HTML-u?**

h1 - h6

1. **Koja je razlika između elemenata <p> i <pre>?**

<p> - prikazuje se kao običan paragraf, s automatskim prelaskom u novi red na kraju paragrafa

<pre> - unaprijed predformatirani tekst, prikazuje se točno onako kako je napisan, čuvajući sve razmake, nove redove i tabulatore

1. **Kako kreiramo novi red teksta?**

<br>

1. **Na koje sve načine možemo prikazati deblje otisnuti tekst?**

<b></b> koji samo podeblja tekst i <strong></strong> koji podeblja tekst i označava kao važan tj. ističe ga

1. **Koje vrste lista razlikujemo?**

Simboljička (<ul> <li> </ul>) ; brojčana (<ol> <li> </ol>) ; opisna (<dl> <dt> <dd> </dl>)

1. **Koje elemente koristimo za strukturiranje tablica?**

<table></table> - definiranje tablice

<tr></tr> - definiranje reda tablice

<th></th> - ćelija zaglavlja tablice

<td></td> ćelija s podacima tablice

<caption></caption> - definiranje naslova tablice

1. **Koja je razlika između elemenata <th></th> i <td></td>?**

<th> se koristi za ćeliju zaglavlja tablice   
<td> se koristi za ćelije s podacima tablice

1. **Kako možemo kreirati naslov tablice?**

Za kreiranje naslova tablice u HTML-u koristimo <caption> element. Ovaj element se stavlja unutar <table> elementa i pruža naslov ili opis tablice.

1. **Koje atribute koristimo za spajanje ćelija?**

colspan - spaja na razini reda (vodoravno)

rowspan - spaja na razini stupca (okomito)

1. **Čemu služe obrasci (forme)?**

Omogućuju unos ili ažuriranje podataka od strane korisnika web

rješenja, koji se šalju na poslužitelj i tamo obrađuju.

1. **Koji je osnovni element za kreiranje obrazaca? Koje atribute standardno dodjeljujemo tom elementu?**

<form></form>

Atributi: action, method, enctype, name, target

1. **Navedite nekoliko mogućih tipova <input /> elementa.**

text, password, checkbox, radio, file, button, image, reset, submit, hidden

1. **Koji element koristimo za kreiranje polja za unos više linija teksta? Kako kreiramo padajuće liste s ponuđenim vrijednostima?**

Za kreiranje polja za unos više linija teksta koristimo <textarea> element. Za kreiranje padajućih lista s ponuđenim vrijednostima koristimo <select> element u kombinaciji s <option> elementima.

1. **Kojim elementom kreiramo gumb za potvrđivanje obrasca?**

Za kreiranje gumba za potvrđivanje obrasca koristimo <button> element s atributom type postavljenim na "submit". Ovaj gumb šalje podatke obrasca kada se klikne. **<form action="/submit" method="post">**

**<button type="submit">Pošalji</button>**

**</form>**

1. **Čemu služi atribut required?**

Taj atribut označava da je polje za koje je naveden atribut obavezan unos/odabir podataka.

1. **Opišite karakteristike web servisa. Koja je razlika između web stranice i web sjedišta?**

Glavne karakteristike uključuju interoperabilnost, standardizirane protokole (npr. SOAP, REST), platformsku nezavisnost, skalabilnost i sigurnost.

1. **Što predstavlja URL adresa te od čega se ona sastoji?**

Uniform Resource Identifier je jedinstve

1. **Opišite funkciju HTTP protokola te objasnite što predstavlja klijent, a što  poslužitelj (server).**

Predstavlja lokaciju resursa na internetu i omogućava pristup tom resursu. URL se sastoji od nekoliko komponenti, uključujući shemu, domenu, putanju, upitne parametre i fragment

1. **Ukratko opišite funkciju pojedinih računalnih jezika strane klijenta.**

**HTML (HyperText Markup Language):**

Funkcija: Definira strukturu i sadržaj web stranice. Koristi se za označavanje elemenata kao što su naslovi, paragrafi, linkovi, slike i tablice.

**CSS (Cascading Style Sheets):**

Funkcija: Definira stil i izgled web stranice. Koristi se za kontrolu boja, fontova, rasporeda, margina, paddinga i drugih vizualnih aspekata.

**JavaScript:**

Funkcija: Dodaje interaktivnost i dinamičnost web stranici. Koristi se za manipulaciju DOM-a, obradu događaja, slanje i primanje podataka s poslužitelja, te izvođenje raznih skripti na strani klijenta.

1. **Što omogućuju računalni jezici strane poslužitelja u web razvoju?**

Računalni jezici strane poslužitelja u web razvoju omogućuju obradu podataka, upravljanje bazama podataka, stvaranje dinamičkog sadržaja i komunikaciju s klijentskim stranama.

Računalni jezici strane poslužitelja (**server-side languages**) koriste se za **dinamičko generiranje sadržaja**, obradu podataka i interakciju s bazama podataka prije nego što se stranica prikaže korisniku. Obrađuuju osjetljive podatke

1. **Navedite i ukratko opišite karakteristike kvalitetnog web rješenja.**

**-User friendly:**

            Intuitivno i jednostavno za korištenje, s jasnom navigacijom i razumljivim sadržajem.

**-Loading speed:**

Brzo učitavanje stranica i optimiziran performans za bolje korisničko iskustvo.

**-Sigurnost:**

Zaštita podataka korisnika i prevencija sigurnosnih ranjivosti.

**-Pristupačnost:**

Dostupno svim korisnicima, uključujući one s invaliditetom, koristeći pristupačne tehnologije.

**-SEO optimizacija:**

Dobro optimizirano za pretraživače kako bi se poboljšao rang na rezultatima pretraživanja.

**-Laka održivost:**

Čist i organiziran kod koji olakšava ažuriranja i održavanje.

**-Responsivni dizajn:**

Prilagođen prikaz na različitim uređajima i veličinama ekrana.

**-Skalabilnost:**

Sposobnost rasta i prilagodbe povećanom broju korisnika i zahtjevima.

1. **Navedite i ukratko opišite pet kategorija elemenata web rješenja.**

**-** **Strukturni elementi:**

Definiraju osnovnu strukturu web stranice

**-Sadržajni elementi:**

Sadrže tekstualni i medijski sadržaj

**-Interaktivni elementi:**

Omogućavaju korisničku interakciju

**-Stilski elementi:**

Definiraju izgled i stil web stranice

**-Meta elementi:**

Pružaju metapodatke o web stranici

1. **Što uključuje moderni pristup izradi web rješenja?**

Razdvojiti strukturu i sadržaj dokumenta od njegova stila

• prilagoditi web rješenje za različite platforme i profile korisnika

• smisleno i planski postaviti strukturu dokumenta, kako bi se

               osigurao:

• pregledniji i lakši za održavanje kod

• lakše upravljanje web stranicom

• dodjeljivanje stila sadržaju, postavljanje responzivne strukture dokumenta itd.

• brže učitavanje i prikazivanje u web pregledniku

**42. Zašto strukturiramo sadržaj web stranica?**Strukturiranje sadržaja web stranica poboljšava korisničko iskustvo, pristupačnost i SEO, te olakšava održavanje koda.

**43. Koji su nesemantički strukturni elementi?**

<div></div> ; <span></span>

1. **Navedite primjere semantičkih strukturnih elementa.**

<header>, <nav>, <main>, <article>, <section>, <aside>, <footer>, <figure> i <figcaption>

1. **Koji strukturni element koristimo ako sadržaj grupiramo samo zbog stiliziranja CSS-om, bez da mu damo semantičko značenje?**

Ako želimo grupirati sadržaj samo zbog stiliziranja CSS-om, bez dodavanja semantičkog značenja, koristimo <div> element. <div> je generički kontejner koji se koristi za grupiranje elemenata i primjenjivanje stilova bez ikakvog semantičkog značenja.

1. **Opišite JavaScript jezik te navedite što on omogućuje.**

JS je programski jezik koji se koristi za stvaranje interaktivnih i dinamičkih web stranica. Omogućuje upravljanje aplikacijom u koju je ubačen, ubacivanje programske logike u HTML stranicu, čitanje promjena HTML elemenata, reagiranje na događaj, provjeru podataka prije nego što se pošalju na server…

1. **Opišite moguće načine ugradnje JavaScript koda u HTML dokument te navedite primjere dobre prakse njegova pozicioniranja.**

Inline, između <script> i </script>  
Iz vanjske datoteke definirane src atributom <script> elementaŽ  
U atributu HTML elementa kojemu se dodjeljuje događaj (onclick, onmouseover

itd.)

U URL-u koji koristi poseban javascript: protokol (blokirano od strane modernih

web preglednika)

1. **Koja je razlika između JavaScript biblioteke (library) i razvojnog okvira (framework)?**

Razlikuje se od JavaScript biblioteke u načinu na koji ga koristimo: biblioteka nudi

funkcije koje bi mogle biti pozvane kad nam zatrebaju i pomažu nam u kodiranju, dok

okvir definira cjelokupni dizajn i arhitekturu aplikacije.

1. **Što su varijable? Kako možemo onemogućiti promjenu primitivne vrijednosti varijable?**

Varijable su kontenjeri za pohranu. Kada dodijelimo novu vrijednost varijabli (koja već ima vrijednost), stvara se nova vrijednost umjesto mijenjanja postojeće. Međutim, ako želite onemogućiti promjenu vrijednosti varijable, možemo koristiti const ključnu riječ.

1. **Osim slova i znamenki, koje posebne znakove možemo koristiti u imenovanju varijabli?**

znak dolara $ i donja crta \_

1. **Opišite tri djelokruga varijabli.**Globalni djelokrug - definiraju se izvan funkcija, mogu se koristiti u čitavom prgramu, zatvaranjem stranice se brišu  
   Djelokrug funkcije - definiraju se unutar funkcije te se mogu koristiti samo unutar nje, izlaskom funkcije se brišu  
   Djelokrug bloka - definiraju se unutar bloka {} pomoću ključne riječi let, dostupne su samo unutar bloka u kojem su definirane, moraju se deklarirati prije upotrebe te se ne mogu ponovo deklarirati
2. **Navedite koji tipovi podataka postoje u JavaScript jeziku.**

Primitivni (nepromjenjivi) i objektni (promjenjivi).

1. **Navedite nekoliko metoda koja možemo primijeniti nad znakovnim nizovima (string).**

charAt(index): Vraća znak na određenom indeksu.

concat(...strings): Spaja dva ili više stringova.

includes(searchString): Provjerava da li string sadrži određeni podstring.

indexOf(searchValue): Vraća indeks prvog pojavljivanja podstringa.

lastIndexOf(searchValue): Vraća indeks posljednjeg pojavljivanja podstringa.

replace(searchValue, newValue): Zamjenjuje podstring s novim stringom.

slice(start, end): Izdvaja dio stringa.

1. **Koja je razlika između sljedećih operatora: =, ==, === ?**

= - dodjela vrijednosti

== - jednaka vrijednost

=== - nije jednaka vrijednost ili nije jednaki tip podataka

1. **Navedite tipove logičkih operatora.**

Logičko I (&&), logičko ILI (||), logičko NE (!)

1. **Što vraća operator modulo (%)?**

Vraća ostatak. To znači da ako imamo dva broja, a i b, izraz a % b će vratiti ostatak kada se a podijeli s b.

1. **Ukratko opišite razliku između kontrolnih struktura grananje i petlje.**

**Svrha:**

Grananje: Odabire jedan od više mogućih puteva izvršavanja na temelju uvjeta; omogućuje donošenje odluka na temelju određenih uvjeta; Program provjerava zadani uvjet i izvodi određeni blok koda samo ako je uvjet ispunjen. (if, else, if-else, switch)

Petlje: Omogućuju ponavljanje dijela koda više puta dok je određeni uvjet ispunjen; Koriste se kada treba iterirati kroz niz podataka ili ponavljati radnju dok se ne postigne određeno stanje. (for, while, do-while, forEach)

**Broj izvršavanja:**

Grananje: Blok koda se izvršava samo jednom, ovisno o uvjetu.

Petlje: Blok koda se može izvršiti više puta, ovisno o uvjetu.

**Primjena:**

Grananje: Koristi se za odlučivanje koji dio koda treba izvršiti na temelju uvjeta.

Petlje: Koristi se za iteraciju kroz nizove, ponavljanje zadataka, i slično.

1. **Što su objekti?**

Zbirka svojstava od kojih svako ima naziv (ključ) i vrijednost. Objekt se prepoznaje po vitičastim zagradama {}.  
Objekti su složeni podatkovni tipovi koji se koriste za pohranu podataka i funkcionalnosti u programskim jezicima poput JavaScript-a. Objekt se sastoji od svojstava (properties) i metoda (methods), pri čemu svojstva predstavljaju karakteristike objekta, a metode su funkcije koje obavljaju određene radnje.

1. **Za što služi ključna riječ *this*?**

Za dohvaćanje svojstva unutar definicije objekta koristimo ključnu riječ this;

Ključna riječ this u JavaScript-u odnosi se na trenutni objekt u kojem se nalazi. Omogućuje pristup svojstvima i metodama objekta unutar istog objekta.

1. **Za što koristimo JSON?**

Razmjena podataka između klijenta i servera, pohrana konfiguracijskih podataka, serilizacija i deserilizacija objekata, interoperabilnost;

JSON (JavaScript Object Notation) je format za razmjenu podataka koji omogućuje laganu pohranu i prijenos podataka između klijenta i poslužitelja. Koristi se u API komunikaciji te za pohranu struktura podataka u aplikacijama.

1. **Koji tipovi objekata postoje po podrijetlu?**

Nativni, host, korisnički definiran

Korisnički definirani objekti – kreirani od strane programera.

Ugrađeni (native) objekti – standardni objekti koje pruža JavaScript, poput Date, Math, Array.

DOM i BOM objekti – specifični za rad u web preglednicima, omogućuju manipulaciju dokumentom (document) i rad s prozorom preglednika (window).

1. **Što omogućuju sljedeći standardni (*native*) ugrađeni objekti: *Date*, *Math* i *RegExp*?**

Date - kreiranje objekata koji predstavljaju datume i vrijeme; omogućuje rad s datumima i vremenom.

Math -  pruža matematičke funkcije i konstante za izvođenje različitih matematičkih operacija; nudi matematičke funkcije i konstante.

RegExp - omogućuje pretragu traženog uzorka unutar znakovnog niza; omogućuje rad s regularnim izrazima.

1. **Što omogućuju objekti BOM-a (*Browser Object Model*)?**

BOM (Browser Object Model) omogućuje interakciju s prozorom preglednika, njegovim svojstvima i funkcijama.

Glavne funkcionalnosti BOM-a uključuju:

Manipulaciju prozorom preglednika (window.open(), window.close()).Rad s URL-ovima (window.location). Rad s kolačićima (document.cookie), Manipulaciju historijom preglednika (history.back(), history.forward()).

1. **Navedite primjere objekata koji pripadaju BOM-u.**

window, document, navigator, screen, history, location

window – glavni objekt koji predstavlja prozor preglednika.

navigator – omogućuje pristup informacijama o pregledniku i uređaju.

location – omogućuje pristup URL-u trenutne stranice.

history – omogućuje navigaciju kroz historiju preglednika.

screen – omogućuje dobivanje podataka o korisnikovom ekranu

1. **Što su nizovi?**

Niz je sortirana zbirka vrijednosti; Nizovi (Arrays) su posebna vrsta objekata koji omogućuju pohranu više vrijednosti unutar jedne varijable. Svaka vrijednost u nizu ima svoj indeks, koji počinje od 0. Nizovi omogućuju jednostavno čuvanje, manipulaciju i pristup podacima unutar programa.

1. **Kako se niz može kreirati? Kako pristupamo pojedinim elementima niza?**

let komandom u stavljanjem u uglate zagrade []. Npr. let brojevi = [1, 2, 3, 4, 5];. Elementi niza se pristupaju korištenjem indeksa unutar kvadratnih zagrada []. Indeksiranje počinje od 0, što znači da je prvi element niza na indeksu 0, drugi na indeksu 1, itd.

1. **Navedite nekoliko primjera metoda koje se koriste za upravljanje nizovima.**

lenght, push(), pop(), splice(), map, sort, filter, reverse

1. **Što su funkcije te koje su prednosti njihova korištenja?**

Funkcije su blokovi koda koji izvršavaju određenu radnju i mogu se ponovno koristiti unutar programa.

Prednosti korištenja funkcija:

Ponovna upotreba koda – izbjegava se ponavljanje istog koda.

Bolja organizacija – kod je modularan i lakše ga je održavati.

Lakše ispravljanje grešaka – promjene se rade na jednom mjestu, što smanjuje mogućnost grešaka.

Poboljšana čitljivost koda – funkcije omogućuju da program bude jasniji i strukturiraniji.

1. **Čemu služi ključna riječ *return*?**

Ključna riječ return koristi se u funkcijama za vraćanje vrijednosti kao rezultata izvođenja funkcije. Nakon return naredbe, izvođenje funkcije se prekida.

1. **Što predstavlja hoisting (izvlačenje)?**

Hoisting u JavaScript-u predstavlja mehanizam kojim preglednik "izvlači" deklaracije varijabli i funkcija na vrh svog dosega prije izvršavanja koda.

1. **Koja je razlika između globalnih i lokalnih varijabli?**

Globalne varijable definirane su izvan funkcija i dostupne su u cijelom programu; Lokalne varijable definirane su unutar funkcije i dostupne su samo unutar te funkcije.

1. **Što znači da su funkcije u JavaScript jeziku objekti prve klase?**

Znači da se tretiraju kao bilo koja druga varijabla i da mogu imati vlastita svojstva i metode., mogu se vraćati iz druge funkcije

1. **Gdje sve koristimo anonimne funkcije?**

Funkcije bez imena. Može biti izravno dodijeljena varijabli, u event listenerima, u map metodi za manipulaciju nizovima, korištena kao argument druge funkcije ili obuhvaćena zagradama.

1. **Koji je doseg funkcija (closure)?**

Doseg (scope) varijabla funkcije je onaj unutar kojeg je funkcija definirana, a ne unutar kojeg je pozvana!

\*Closure funkcija ima pristup varijablama izvan svog tijela do njoj lokalne funkcije, odnosno lokalnog scopea.

1. **Zašto koristimo objekte u izradi programa?**

Zbog organizacije rada, ponovna iskoristivost, enkapsulacija, nasljeđivanje.

Objekti omogućuju **organizaciju i grupiranje** podataka i funkcionalnosti. Strukturirano pohranjivanje podataka

1. **Koja je razlika između izravnog kreiranja objekta i upotrebe funkcije konstruktora?**

Izravno kreiranje objekta – koristi {} za definiranje pojedinačnog objekta, kreira se putem literala.

Konstruktor omogućuje stvaranje više objekata s istim svojstvima i metodama.

1. **Što omogućuje nasljeđivanje?**

Ponovno korištenje koda, organizaciju, proširivost, polimorfizam, enkapsulacija

Nasljeđivanje omogućuje stvaranje novih objekata koji dijele svojstva i metode postojećih objekata. Time se kod optimizira jer nema potrebe za ponavljanjem istih metoda.

1. **Što predstavlja prototip te što on omogućuje?**

Prototip (Prototype) je mehanizam u JavaScript-u koji omogućuje nasljeđivanje svojstava i metoda između objekata. Svaki objekt u JavaScript-u ima prototipni objekt iz kojeg može naslijediti metode i svojstva. Kada tražimo svojstvo ili metodu na objektu, JavaScript prvo pretražuje taj objekt. Ako ga ne pronađe, traži ga u njegovom prototipu, nastavljajući prema gore kroz prototipni lanac.

1. **Objasnite što predstavlja Document Object Model (DOM).**

Document Object Model (DOM) je programski interface za HTML i XML dokumente. Predstavlja strukturu dokumenta kao stablo objekata, gdje svaki čvor u stablu predstavlja dio dokumenta (element, atribut, tekstualni čvor, itd.).

Kroz DOM možemo dinamički manipulirati sadržajem, stilovima i strukturom web stranice koristeći JavaScript.

1. **Navedite elemente od kojih se sastoji DOM stablo.**

Document čvor, Element čvorovi, Atribut čvorovi, Tekstualni čvorovi, Komentar čvorovi, DocumentType čvor, ProcessingInstruction čvorovi.

DOM stablo se sastoji od sljedećih vrsta čvorova (nodes):

Document – glavni korijenski čvor (predstavlja cijeli dokument).

Elementi (Element nodes) – HTML elementi (div, p, ul itd.)

Atributi (Attribute nodes) – svojstva elemenata (class, id)

Tekstualni čvorovi (Text nodes) – tekst unutar HTML elemenata.

Komentari (Comment nodes) – HTML komentari.

1. **Što predstavlja objekt document te što omogućuju njegova svojstva i metode?**

Objekt document predstavlja ulaznu točku za pristup Document Object Modelu (DOM) trenutnog HTML ili XML dokumenta. Kroz objekt document, programeri mogu pristupiti i manipulirati sadržajem i strukturom web stranice.

Objekt document je glavni ulaz u DOM i omogućuje manipulaciju HTML dokumentom.

Dohvaćanje elemenata (getElementById, querySelector itd.)

Promjena sadržaja (innerHTML, textContent)

Manipulacija atributima (setAttribute, getAttribute)

Stvaranje i brisanje elemenata (createElement, removeChild)

Rad s događajima (addEventListener

1. **Na koje sve načine možemo dohvatiti HTML element(e)?**

Po ID-u: getElementById(id):, po klasi: getElementsByClassName(className):, po nazivu taga: getElementsByTagName(tagName), po css selektorima: querySelector, querySelectorALl

1. **Koja je razlika između kolekcije HTML elemenata i kolekcije čvorova?**

HTMLCollection je specifična za HTML elemente i uvijek je živa, dok NodeList može sadržavati različite tipove čvorova i može biti statička ili živa.

HTML kolekcija (HTMLCollection) sadrži samo HTML elemente, ali se automatski ažurira kada se DOM promijeni. NodeList (NodeList) može sadržavati i elemente i tekstualne čvorove, ali se ne ažurira automatski (osim ako nije live).

1. **Koje sve operacije nad HTML elementima možemo izvršiti preko DOM-a?**

Dodavanje elemenata (createElement, appendChild, insertBefore).

Brisanje elemenata (removeChild, remove).

Izmjena atributa (setAttribute, removeAttribute).

Promjena sadržaja (innerHTML, textContent).

Manipulacija klasama (classList.add, classList.remove)

Selekcija elemenata, Manipulacija sadržajem, Manipulacija stilom, Manipulacija strukturom, Manipulacija događajima

1. **Kako sve možemo promijeniti stil HTML elementa?**

Inline stilovima, dodavanjem/uklanjanjem klasa, korištenjem setAttribute metodama, korištenje CSSom

Izravno kroz style svojstvo (element.style.color = "red";)

Dodavanjem klase: (element.classList.add("novaKlasa");)

Kroz CSS datoteku: (.novaKlasa { color: blue; })

1. **Što su događaji? Na koje sve načine ih možemo postaviti?**

\* Događaji (events) u JavaScriptu su akcije ili interakcije korisnika s web stranicom, poput klika mišem, pritiska tipke, pomicanja miša, učitavanja stranice, slanja forme i mnogih drugih.

Korisnički događaji → click, mouseover, keydown, keyup, submit

Događaji stranice → load, resize, scroll, unload

Događaji obrasca → change, focus, blur

Događaji tipkovnice → keydown, keypress, keyup

Događaji miša → click, dblclick, mouseover, mouseout

**Što predstavlja propagacija događaja te kako se njome upravlja?**

Propagacija događaja označava kako se događaji šire kroz DOM stablo. Postoje dva načina:

Bubbling (mjehurićanje, zadano ponašanje) – događaj se prvo aktivira na ciljanom elementu, a zatim se "penje" prema gore.

Capturing (hvatanje) – događaj se prvo aktivira na najvišem roditeljskom elementu i zatim se spušta prema dolje. Propagacijom se upravlja pomoću stopPropagation().

**Što predstavlja API?**

API (Application Programming Interface) je skup pravila i metoda koji omogućuje komunikaciju između aplikacija ili komponenti.

Primjeri API-ja u web razvoju: DOM API – omogućuje manipulaciju HTML dokumentom. Fetch API – omogućuje dohvaćanje podataka s poslužitelja. Geolocation API – omogućuje dobivanje lokacije korisnika.

1. **Što je AJAX te koje prednosti nosi njegova primjena u web razvoju?**

AJAX je kombinacija XMLHttpRequest objekta ugrađenog u preglednik (za traženje podataka s web poslužitelja) i HTML DOM-a (za prikaz ili upotrebu podataka). Prednosti su brže i dinamičnije korisničko iskustvo, smanjenje opterećenja poslužitelja, poboljšana interaktivnost, bolja korisnička iskustva.

\*AJAX (Asynchronous JavaScript and XML) omogućuje učitavanje podataka s poslužitelja bez osvježavanja stranice.

1. **Koji tip JavaScript objekta upravlja asinkronom komunikacijom između klijenta i servera?**

JavaScript koristi XMLHttpRequest objekt i Fetch API za upravljanje asinkronom komunikacijom između klijenta i poslužitelja. XMLHttpRequest – starija metoda, omogućuje slanje HTTP zahtjeva bez osvježavanja stranice. Fetch API – modernija alternativa, koristi Promise objekte i omogućuje čišći i jednostavniji kod.

1. **U kojem sve formatu se mogu prenijeti podaci između klijenta i servera?**

JSON, XML, HTML, plain text, CSV

Podaci između klijenta i poslužitelja mogu se prenositi u različitim formatima: JSON (JavaScript Object Notation) – najčešće korišten format zbog jednostavnosti i fleksibilnosti. XML (eXtensible Markup Language) – stariji format, rjeđe se koristi. HTML – može se slati kao odgovor na zahtjev. Plain text – običan tekst. Form Data – koristi se za slanje podataka iz HTML formi (application/x-www-form-urlencoded ili multipart/form-data).

1. **Koje sigurnosno ograničenje je unaprijed postavljeno u primjeni AJAX tehnologije?**

Najvažnije sigurnosno ograničenje u AJAX tehnologiji je CORS (Cross-Origin Resource Sharing) politika. Same-Origin Policy (SOP) – preglednici dopuštaju JavaScript-u dohvaćanje podataka samo s iste domene (isteg podrijetla: protokol, domena i port). CORS omogućuje da poslužitelj eksplicitno dopusti zahtjeve iz drugih domena pomoću HTTP zaglavlja (Access-Control-Allow-Origin).

1. **Što predstavljaju obrasci u programiranju?**

Obrasci u programiranju (engl. design patterns) su unaprijed definirana rješenja za često susretane probleme u softverskom razvoju.

1. **Što je karakteristično za module obrazac?**

Koristi se za kreiranje JS paketa, tj. samostalnih dijelova koda, koji se mogu dodavati, zamijeniti, micati po volji bez opasnosti da se nešto poremeti u ostatku aplikacije/web stranice. Modul obrazac omogućuje grupiranje povezane funkcionalnosti i zaštitu privatnih podataka. Korištenjem modula možemo organizirati kod u manje, odvojene dijelove koji se lako ponovno koriste.

1. **Koja je svrha direktive "use strict"?**

Naznačiti da je kod koji slijedi strogi kod, Strogi način rada olakšava pisanje “sigurnog” JS-a, Strogi način rada prethodno naznačenu “lošu sintaksu” pretvara u stvarne pogreške u kodu.

1. **Koji tip varijable ćemo dobiti ako u njezinoj deklaraciji koristimo ključnu riječ let ili const?**

Korištenje ključnih riječi let i const u deklaraciji varijable daje varijable s blokovnim opsegom. To znači da su ove varijable dostupne samo unutar bloka koda u kojem su definirane (npr. unutar petlje, funkcije ili {}).

1. **Što će se ispisati u konzoli?**
   1. **var iznos = 1000;**
   2. **var poruka = `Iznos računa: ${iznos}$`;**
   3. **console.log(poruka);**

Iznos računa: 1000$

1. **Što omogućuju operatori "Spread" i "Rest"?**

Omogućuju rad s nizovima i objektima na fleksibilan način.

**Spread** - omogućuje širenje elemenata niza ili svojstava objekta. Koristi se za kopiranje, kombiniranje ili proširivanje nizova i objekata.

**Rest** - omogućuje grupiranje preostalih elemenata niza ili svojstava objekta u jedan niz ili objekt. Koristi se za funkcijske parametre i destrukturiranje.

1. **Što omogućuju Arrow funkcije?**

Omogućuju kraću sintaksu za definiranje funkcija, Arrow funkcije (=>) su kraći i moderniji način deklariranja funkcija u JavaScriptu, Arrow funkcije ne stvaraju vlastiti this, već nasljeđuju this iz okolnog opsega.

\*Kraći i čitljiviji kod Automatsko vezivanje this konteksta

1. **Kod upotrebe klasa u JavaScript jeziku što predstavlja metoda constructor? Pomoću koje ključne riječi implementiramo nasljeđivanje?**

Metoda constructor u JavaScript klasama automatski se pokreće kad stvaramo novu instancu klase. Koristi se za inicijalizaciju svojstava objekta i izvođenje početnih operacija. Nasljeđivanje se implementira pomoću ključne riječi **extends**.

**ECMAScript** (ES) je standardizirani skriptni programski jezik koji definira pravila i specifikacije za JavaScript. ECMAScript osigurava kompatibilnost među preglednicima.

1. **Opišite ReactJS.**

ReactJS je JavaScript biblioteka razvijena od strane Facebooka (sada Meta) za izradu korisničkih sučelja (UI). Omogućava izradu komponenti koje se mogu ponovno koristiti, a koristi virtualni DOM kako bi optimizirao ažuriranje korisničkog sučelja. React je posebno popularan za razvoj jednostraničnih aplikacija (SPA) jer omogućava brze i dinamične promjene u sučelju bez potrebe za ponovnim učitavanjem stranice.

1. **Koja je razlika u načinu upravljanja HTML DOM-om između klasičnog JavaScript koda i ReactJS biblioteke?**

U klasičnom JavaScript-u manipulacija DOM-om se obavlja direktno pomoću funkcija poput document.getElementById ili document.querySelector, što može biti sporo i neefikasno jer prepisuje cijeli DOM čvor. React koristi **Virtual DOM**, što znači da kreira virtualnu kopiju stvarnog DOM-a, uspoređuje ga s prethodnim stanjem i ažurira samo one dijelove stranice koji su se promijenili. Ovaj pristup značajno poboljšava performanse i optimizira rad aplikacije.

1. **Kako pomoću samo jedne naredbe kreirati ReactJS aplikaciju?**

npx create-react-app my-app

1. **Ukratko opišite za što služe sljedeće skripte definirane unutar ReactJS projekta:**
   1. Start - Pokreće razvojni server i omogućava prikaz aplikacije u pregledniku uz automatsko osvježavanje pri izmjeni koda.
   2. Test - Pokreće testove definirane unutar projekta koristeći testni okvir (npr. Jest).
   3. Build - Generira optimiziranu verziju aplikacije spremnu za produkciju u direktorij build.
   4. Eject - Trajno izvlači sve konfiguracijske datoteke i ovisnosti iz react-scripts, omogućavajući ručnu prilagodbu, ali bez mogućnosti povratka.
2. **Ukratko opišite strukturu ReactJS aplikacije:**
   1. datoteka package.json - Sadrži informacije o projektu, ovisnostima, verzijama paketa te skriptama koje se mogu izvršavati.
   2. mapa src - Glavni direktorij za razvoj aplikacije; ovdje se nalaze React komponente, CSS datoteke i druge programske datoteke.
   3. mapa public - Sadrži statične datoteke poput index.html, favicon ikonu i slike koje ne prolaze kroz Webpack obradu.
   4. mapa node\_modules - Sadrži sve instalirane pakete i njihove ovisnosti koje aplikacija koristi.
3. **Što predstavlja npm te što on omogućuje web programerima?**

**npm (Node Package Manager)** je upravitelj paketa za JavaScript koji omogućuje preuzimanje, instalaciju i upravljanje vanjskim bibliotekama i alatima. Omogućava jednostavno dodavanje novih paketa, upravljanje verzijama i dijeljenje vlastitih modula. **npm (Node Package Manager)** je alat koji dolazi uz Node.js i služi za upravljanje paketima (libraries, moduli, ili funkcionalnosti) u JavaScript okruženju. To je najveći svjetski ekosustav otvorenog koda, gdje korisnici mogu preuzimati, objavljivati, dijeliti i upravljati paketima softvera.

1. **Putem kojeg sučelja se vrši upravljanje paketima?**

Upravljanje paketima se vrši putem naredbenog retka (CLI – Command Line Interface) pomoću npm ili alternativno yarn. Ove naredbe omogućuju instalaciju (npm install), uklanjanje (npm uninstall), ažuriranje (npm update) i druge operacije s paketima.

1. **Koja je funkcija datoteke package.json?**

Datoteka package.json služi kao konfiguracijska datoteka za ReactJS (ili bilo koji Node.js) projekt. Sadrži informacije o aplikaciji, instaliranim paketima, skriptama za pokretanje i ostalim konfiguracijama potrebnim za razvoj i izvođenje aplikacije.

1. **Što su JavaScript moduli? Koje su glavne prednosti njihove upotrebe?**

JavaScript moduli omogućuju podjelu koda na manje, samostalne dijelove koji se mogu lako uvoziti i koristiti u drugim datotekama. Glavne prednosti uključuju:

* Poboljšanu organizaciju koda
* Ponovnu upotrebu koda
* Izbjegavanje sukoba između varijabli
* Lakše testiranje i održavanje

1. **Što je potrebno napraviti kako bi se programski kod napisan u jednoj datoteci mogao koristiti u drugoj datoteci?**

Potrebno je koristiti export u datoteci koja sadrži kod i import u datoteci koja ga koristi.

1. **Na koji način datoteka index.js olakšava uvoz (import) modula?**

Datoteka index.js često služi kao centralna točka za izvoz više modula iz jednog direktorija

1. **Što predstavlja životni ciklus komponente?**

Životni ciklus komponente u Reactu odnosi se na različite faze kroz koje komponenta prolazi od trenutka kada je kreirana do trenutka kada je uklonjena iz DOM-a.

1. **Koje su tri osnovne faze životnog ciklusa komponente? Ukratko ih opišite.**

 **Mounting (Montaža)** – Ova faza se događa kada se komponenta prvi put kreira i dodaje u DOM. React tada poziva metode koje omogućuju inicijalizaciju stanja i dohvaćanje podataka prije nego što se komponenta prikaže korisniku.

 **Updating (Ažuriranje)** – Ova faza nastupa kada se stanje (state) ili svojstva (props) komponente promijene. React ponovo renderira komponentu kako bi prikazao najnovije podatke.

 **Unmounting (Demontaža)** – Ova faza se događa kada se komponenta uklanja iz DOM-a, najčešće kada korisnik napusti stranicu ili kada se komponenta više ne koristi. Ovdje se može obaviti čišćenje resursa kako bi se spriječili memorijski curenja.

1. **Navedite u kojem redoslijedu će se pozvati metode faze *mounting*.**

Prilikom montiranja komponente, React poziva sljedeće metode redoslijedom:

1. **constructor()** – Inicijalizira stanje (state) i povezuje metode.
2. **static getDerivedStateFromProps()** – Ažurira stanje komponente na temelju promjena props.
3. **render()** – Vraća JSX koji određuje izgled komponente.
4. **componentDidMount()** – Izvodi operacije nakon što je komponenta dodana u DOM, kao što su dohvaćanje podataka iz API-ja.
5. Navedite u kojem redoslijedu će se pozvati metode faze *updating*.

Prilikom ažuriranja komponente, React poziva sljedeće metode redoslijedom:

1. **static getDerivedStateFromProps()** – Ažurira stanje na temelju novih props.
2. **shouldComponentUpdate()** – Određuje hoće li se komponenta ponovo renderirati.
3. **render()** – Prikazuje JSX s ažuriranim podacima.
4. **getSnapshotBeforeUpdate()** – Hvata informacije prije nego što se stvarni DOM ažurira.
5. **componentDidUpdate()** – Izvršava operacije nakon što je ažuriranje dovršeno, poput novih API poziva ili interakcije s DOM-om.
6. **Koja metoda je jedina dostupna u fazi *unmounting*?**

Jedina dostupna metoda u fazi unmounting je:

**componentWillUnmount()** – Koristi se za čišćenje resursa prije nego što se komponenta ukloni iz DOM-a, kao što su prekidanje API poziva, uklanjanje event listenera i oslobađanje memorije.

1. **Što omogućuje *useEffect hook***

useEffect je React hook koji omogućuje izvršavanje sporednih efekata unutar funkcionalnih komponenti.

\*useEffect je React hook koji omogućuje izvršavanje koda nakon što se komponenta prikaže na ekranu

1. **Što omogućuju hook funkcije?**

Hook funkcije omogućuju korištenje React značajki, poput stanja (state) i životnog ciklusa, unutar **funkcionalnih komponenti**. Prije hookova, samo su **klasne komponente** mogle koristiti lokalno stanje i metode životnog ciklusa.

1. **Što omogućuje hook funkcija useState()? Koje dvije vrijednosti ona vraća?**

useState omogućuje upravljanje **lokalnim stanjem** unutar komponenti definiranih funkcijom.

Vrati **dvije vrijednosti**:

1. **Trenutnu vrijednost stanja** (state).
2. **Funkciju za ažuriranje stanja** (setState).
3. **Što predstavljaju događaji?**

Događaji su **interakcije korisnika** s web-stranicom, poput:

* Klikova (onClick),
* Unosa teksta (onChange),
* Pritiska tipki (onKeyDown),
* Pokreta miša (onMouseOver), itd.

1. **Kako se tumači ključna riječ *this* unutar metode komponente definirane klasom, s obzirom na način kako smo definirali samu metodu?**

U React klasnim komponentama, this se odnosi na **trenutnu instancu komponente**. Međutim, ako this nije eksplicitno vezan, može rezultirati undefined.

U standardnoj definiciji metoda koje rukuju događajem thisvraća undefined

•zbog implementacije React-a izgubi se implicitno zadani kontekst vezanja (bind)

•umetodama životnog ciklusa thispredstavlja objekt koji je definira –implicitno se provede bind

•ako koristimo arrowfunkciju, thispredstavlja objekt koji definira samu metodu (Reactkomponenta definirana klasom)

1. **Na koja dva načina možemo proslijediti argumente funkciji koja rukuje događajem?**

Arrow funkcija unutar JSX-a, Korištenjem bind metode:

1. **Opišite JSX.**

**JSX (JavaScript XML)** je sintaksni ekstenzija JavaScript-a koja omogućuje pisanje HTML-a unutar JavaScript koda. Omogućuje dinamičko iscrtavanje sadržaja pomoću varijabli i izraza.

JSX se prevodi u standardni JavaScript pomoću React.createElement().

1. **Što predstavlja uvjetno iscrtavanje (Conditional rendering)? Na koje načine ga je moguće realizirati?**

Uvjetno iscrtavanje omogućuje **prikazivanje različitog sadržaja** na temelju stanja (state) ili svojstava (props). Korištenjem if izraza, Ternarni operator (? :), Kratki spoj (&&)

1. **Što su React komponente? Koje tipove React komponenata razlikujemo?**

React komponente su glavni građevni dijelovi svake React aplikacije. One omogućuju da korisničko sučelje podijelimo na manje, ponovo upotrebljive dijelove. Klasne i funkcionalne komponente

1. **Kod izgradnje React aplikacije u koji međusobni odnos se postavljaju komponente (prema React standardu)?**

React aplikacija se gradi **hijerarhijski** pomoću **parent-child (roditelj-dijete) odnosa** između komponenti.

*props*(„propertyji”) sunačinnakojikomponentemeđusobnokomuniciraju

*propsi*se koristekakobi se informacijeprenijeleiz*parent*komponenteu *child*komponentu