RAZVOJ PROGRAMSKE POTPORE ZA WEB

Internet – globalni sustav međusobno povezanih računalnih mreža koji za povezivanje koristi protokol IP. Obuhvaća razne usluge: elektronička pošta , udaljeni pristup računalima, razmjena datoteka, telefonija, **World Wide Web**. World Wide Web - internetska usluga korisnicima putem njihovih uređaja omogućava pristup „mnoštvu različitih digitalnih dokumenata danih na raspolaganje preko umreženih računala diljem svijeta".

Informacijski prostor weba čine informacijski izvori ili resursi međusobno povezani poveznicama.

Osnovne komponente WWW-a:

TKO traži? – klijent- Bilo koji uređaj ili program koji može slati zahtjeve poslužitelju weba, može biti: preglednik weba na bilo kojem računalu ili pokretnom uređaju, robot/crawler – program koji sistematično prolazi sjedištima weba, program koji dohvaća podatke s nekog sjedišta weba koristeći protokol HTTP. Klijent zahtijeva uslugu slanje zahtjeva poslužitelju protokolom HTTP, više klijenata šalje zahtjevež na jedan (ili više) poslužitelj(a).

KOGA će pitati? – poslužitelj- Programski sustav koji „poslužuje" zahtjeve koje upućuje klijent i vraća odgovore. Može, ali i ne mora biti na računalu različitom od klijenta. Svako računalo može biti poslužitelj. Klijent mora moći pristupiti poslužitelju, uporaba protokola TCP/IP, pristup poslužitelju pomoću IP-adrese, pristup točno određenoj usluzi, na vratima (port) koja su pridijeljena toj usluzi, komunikacija između klijenta i poslužitelja protokolom HTTP. Ovisno o resursu kojeg klijent zahtijeva: poslužitelj poslužuje statički dokument, ili obrađuje zahtjev slanjem drugom programskom okruženju, nakon obrađenog zahtjeva, poslužitelj klijentu vraća odgovor.

GDJE se nalazi? - adresiranje - identifikacija resursa: **URI. URI ≠ URL.** URI = Uniform Resource Identifier, URL = Uniform Resource Locator, način identificiranja (URI) / lociranja (URL) resursa na webu.

KAKO tamo doći? - način povezivanja i komunikacije: protokol **HTTP** - standardni internetski aplikacijski protokol. **H**yper **T**ext **T**ransfer **P**rotocol. HTTP – normirani način komunikacije na webu, format poruka koje se razmjenjuju: zahtjevi – HTTP-request i odgovori – HTTP-response.

ŠTO tražimo? - zapis resursa: **HTML:** jednostavan, prenosiv zapis teksta, čitljiv i ljudima i računalima**,** mogućnost umetanja poveznica, korištenje drugih medija (slike, audio, video) u izvornom obliku.

Klijent uvijek definira zahtjev, a poslužitelj odgovor - request/response. Klijent i poslužitelj postaju zavisni samo komunikacijom. Komuniciraju prosljeđivanjem pouka.

Za razvoj su nam potrebni: Klijentske tehnologije i jezici: HTML, CSS i JavaScript. Poslužitelji: Apache, Nginx, IIS. Poslužiteljske tehnologije i jezici: (Node.js - JavaScript) , PHP, ASP.Net, Python, Java, Ruby, razvojni okviri. Izvor podataka: relacijske baze podataka – za pohranu i tekstualni oblici podataka – za razmjenu: XML, JSON. Web API-ji, REST.

HTML-HyperText Markup Language, oznakama (tags) definira strukturu i sadržaj dokumenta, može definirati i dio semantike stranice. Ne koristi se za: definiranje izgleda/dizajna (iako je to moguće), programiranje dinamičkih svojstava stranice programiranje interakcija s korisnikom.

CSS-Cascading Style Sheets. Jezik kojim se opisuje prezentacija dokumenta pisanog u jeziku HTML. Odvaja prezentaciju/dizajn od sadržaja/strukture. Definira dizajn stranice, raspored elemenata, izgled elemenata. Prilagođava se svojstvima klijenta.

JavaScript- Programski jezik (skriptni) namijenjen uporabi na webu. Izvorna namjena: omogućavanje interaktivnosti na klijentskoj strani, npr. izmjena/dodavanje/brisanje elemenata stranice, provjera ispravnosti unesenih podataka u obrazac. Danas: moderna sjedišta weba nezamisliva su bez jezika JavaScript, omogućuje izradu cjelovitih aplikacija u pregledniku. JavaScript se danas izvodi i na strani poslužitelja. Okruženje Node.js.

Načelo modernih aplikacija je raspodijeljenost (distribuiranost) usluga i njihova međusobna komunikacija. Postoji li uobičajen način međusobne komunikacije iznad protokola HTTP: **Web API** – Application Programming Interface, **REST** – Representational State Transfer: stil arhitekture aplikacija weba sa skupom pravila koji omogućuju jednostavnu suradnju servisa, vrlo popularan.

HTML

Označni (markup) jezici: originalan sadržaj dokumenta (najčešće tekst) + oznake u istom obliku kao i sadržaj dokumenta (tekst). Oznake daju upute programima za obradu. Oznake mogu određivati: strukturu sadržaja dokumenta, značenje (semantiku) pojedinih dijelova sadržaja.

Vrste označnih jezika: 1) **Proceduralni**: oznake uključene u sadržaj dokumenta, određuju način obrade sadržaja programima koji dokument obrađuju, mogu uključivati programske konstrukte (funkcije, makronaredbe itd.), autor izravno upravlja i sadržajem i oznakama, korištenjem istih mehanizama (uređivanje teksta i sl.), primjeri proceduralnih označnih jezika: nroff/troff, Tex, PostScript, RTF. 2) **Deskriptivni**- oznake uključene u sadržaj dokumenta, označavaju dijelove dokumenta, ali ne određuju način njihove obrade, potiču definiciju strukture i semantike dijelova dokumenta, ne izgled i način tumačenja, autor izravno upravlja i sadržajem i oznakama, korištenjem istih mehanizama (uređivanje teksta i sl.). Primjeri deskriptivnih označnih jezika: HTML, XML, LaTeX. 3) **Prezentacijski**-oznake uključene u sadržaj dokumenta, ali nisu „pristupačne" korisniku kao u slučaju proceduralnih i deskriptivnih jezika (najčešće zapisane u binarnom ili složenom tekstnom obliku). Autor upravlja oznakama neizravno, korištenjem alata za obradu dokumenta (pristup WYSIWYG). Koristi se proceduralni i/ili deskriptivni princip tumačenja oznaka. Primjeri prezentacijskih označnih jezika: Office Open XML, Open Document Format –ODF.

Gradivene komponente HTML-a: Elementi-dijelovi stranice, Atributi- opisuju elemente. Sadržaj elemenata-(bez sadržaja), tekst (uključujući prazne znakove), elementi, miješani sadržaj; elementi i tekst. HTML definira dozvoljeni skup elemenata, uključujući: Oznake elementa(<h1>, <p>, <i>), Semantiku (značenje) elementa (<h1> - označava najznačajniji naslov, <p> - označava odlomak teksta, <i> - označava segment ukošenog teksta..), Dozvoljeni (ili očekivani) sadržaj elementa (sadržaj elementa <h1> čini tekst, sadržaj elementa <p> čini

tekst, sadržaj elementa <body> čine drugi elementi i/ili tekst, osim elemenata kojima se definira struktura HTML dokumenta (<html>, <body>,<head>,..).

W3C (standardizacijsko tijelo weba) definira različite kategorije elemenata.

Elementi moraju biti ispravno gniježđeni: Sintaksno – unutarnji element mora biti zatvoren završnom oznakom prije zatvaranja vanjskog element. Semantički – unutarnji element mora pripadati kategoriji elemenata (ili kategorijama) dozvoljenog sadržaja.

Element može imati definirano 0..n atributa unutar početne oznake. **Atribut** je definiran parom ime="vrijednost". Atributom se navode dodatne inf. o elementu. **Globalni atributi**: definirani za sve elemente HTML-a. Važniji globalni atributi: **id** - definiranje jedinstvenog identifikatora elementa unutar HTML dokumenta, **class** - definiranje pripadnosti elementa jednoj ili više grupa (klasa) elemenata unutar HTML dokumenta, **lang** - definiranje jezika sadržaja elementa **title** - definiranje dodatne informacije o elementu prikazivane uglavnom u obliku tooltipa, **style** - definiranje stila prikaza elementa (tzv. inilne stil CSS-a), **hidden** - ako je atribut definiran, sadržaj pripadajućeg elementa nije vidljiv, **data**-* - korisnički podaci u obliku ime-vrijednost, gdje je ime navedeno u nastavku imena atributa (*), a vrijednost je vrijednost atributa.

Zaglavlje sadrži podatke o dokumentu – meta-podatke. Ne definira sadržaj dokumenta, ali utječe na: njegov prikaz i interaktivnost unutar preglednika Weba, pretraživanje i indeksiranje od strane pretraživača Weba, ponašanje alata za strojno tumačenje sadržaja, ponašanje asistivnih alata.

Element zaglavlja **<title>** je obavezan. Definira naslov stranice kod prikaza u pregledniku, rezultatima pretrage, u knjižnim oznakama.

Element **<meta>**-Definira meta-podatke o dokumentu. Meta-podaci se ne prikazuju, no utječu na interpretaciju i prikaz dokumenta unutar preglednika Web-a. Također ih koriste drugi alati koji obrađuju HTML dokumente. Unutar zaglavlja može postojati 0..n elemenata <meta>.

Svaki meta-podatak (u pravilu) je u obliku para ime – vrijednost. Standardni oblik definiranja meta-podatka: Atribut name – ime meta-podatka, Atribut content – vrijednost meta-podatka. Uobičajeni meta-podaci o stranici: application-name – ime aplikacije Web-a kojoj stranica pripada, author – autor stranice, description - skraćeni opis stranice, generator – aplikacija koja je stvorila stranicu, keywords – ključne riječi stranice, viewport – upravljanje vidljivim dijelom stranice.

```
<head>
 <meta name="application-name" content="QuiltCMS-PWMU">
 <meta name="author" content= "Marko Ferković - Enter">
 <meta name="description" content="Programiranje za Web i mobilne uređaje">
 <meta name="generator" content="QuiltCMS">
 <meta name="keywords" content="HTML, CSS, Web, HTTP">
</head>
```

Atribut http-equiv omogućava redefiniranje vrijednost elementa zaglavlja HTTP odgovora: Content-type – tip resursa i kodiranje stranice, Default-style – poželjno korištenje navedenog CSS-a za prikaz stranice, Refresh – vremenski interval obnavljanja (ponovnog dohvaćanja) HTML stranice (u sekundama).

Element zaglavlja **<style>**-Definiranje stila stranice. Stilovi definirani na razini pojedinačne stranice. Definiranje stilova korištenjem CSS-a. Unutar zaglavlja može postojati 0..n <style> elemenata.

Element zaglavlja **<link>**- Referenciranje resursa u zasebnoj datoteci. Prilikom obrade stranice, preglednik Weba dohvaća referencirane resurse. Najčešće korišteno za CSS. Atribut rel – definiranje odnosa dokumenta i resursa (stylesheet –vanjski CSS), Atribut href – URI resursa.

```
<link rel="stylesheet" href="site.css">
```

Element zaglavlja **<script>** - Definira skripta na strani korisnika (jezik JavaScript). Unutar zaglavlja može biti 0..n elemenata <script>. Programski kôd skripta može biti: unutar elemenata <script>, u zasebnoj datoteci; element <script>, korištenjem atributa src, referencira datoteku navođenjem njena URI-ja.

```
<script>
 document.getElementById("AldoRaine").innerHTML = "Buongiorno"
</script>
<script src="renderTree.js"></script>
```

Element zaglavlja **<base>** - Definira podrazumijevani URI za sve relativne URI-je unutar dokumenta. Najviše jedan navod unutar zaglavlja HTML dokumenta. Atribut href definira apsolutni URI (koji zamjenjuje kontekstni bazni URI), Atribut target definira cilj hiperveza.

```
<base href="http://www.unizg.fer.hr/default/scripts">
```

Tijelo HTML dokumenta - Definira sadržaj (i, djelomično, izgled) dokumenta. Elementi čine hijerarhijsku strukturu – stablo, <html> kao korijenski element dokumenta. Elementi opisuju strukturu i semantiku sadržaja. Elementi ne bi trebali opisivati izgled dokumenta.

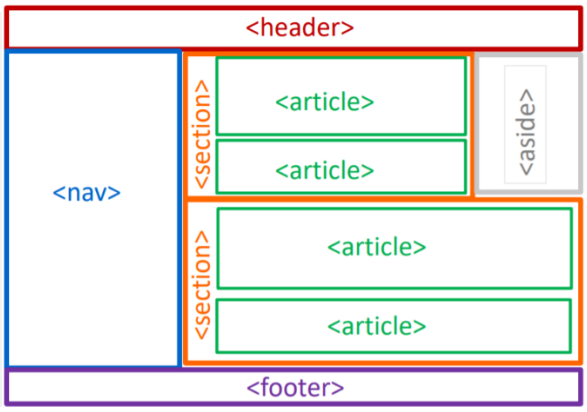
Preglednik ignorira više od jednog uzastopnog praznog znaka. Prazni znakovi su razmak, novi redak,...

Svaki HTML element je tipa prikaza **blok** ili **inline**. Element tipa blok uvijek započinje u novom retku i proteže se čitavom širinom stranice. Element tipa inline započinje od prvog slobodnog mjesta na stranici i širok je koliko je potrebno za njegov prikaz.

Grupiranje elemenata: Element ** ** za grupiranje elemenata. je element tipa prikaza block. Koristi se kao sadržnik (container) drugih elementata u dokumentu, grupirajućih ih u jednu logičku cjelinu. Najčešće se koristi uz navođenje globalnih atributa class, id, style. Pomoću vrijednosti atributa class i id referencira se određena grupa elemenata kojom se najčešće: definira stil prikaza (CSS), programski manipulira sadržajem, stilom prikaza i strukturom (HTML DOM + JavaScript).

Element **** za grupiranje tekstnog sadržaja. je element tipa prikaza inline. Koristi se, u pravilu, kao sadržnik (container) niza znakova ili slika (također inline sadržaja), grupirajućih ih u jednu logičku cjelinu. Najčešće se koristi uz navođenje globalnih atributa class, id, style. Pomoću vrijednosti atributa class i id referencira se određena grupa elemenata kojom se najčešće: definira stil prikaza (CSS), programski manipulira sadržajem, stilom prikaza i strukturom (HTML DOM + JavaScript). **
-skok u novi red, tip prikaza inline. **<hr/>- horizontalna crta, tip prikaza je block. **<!-komentar>**

```
<div class="stih">
  <p>Ar nigdar ni tak bilo da ni nam tak bilo,<br>
  pak nigdar ni nebu da <b>kmet</b> gladen
  <span class="rima">nebu</span>,</p>
  <p>kajti nigdar nebu na zemlji ni na
  <span class="rima"> nebu</span>,<br>
  pri koncu pak Turčin potukel nas
  <span class="rima">se bu</span>.</p>
</div>
```



Semantički elementi imaju značenje. Semantika elemenata koristi se za: pravilan prikaz sadržaja elemenata u pregledniku weba, tumačenje sadržaja od strane pretraživača, ispravno glasovno čitanje stranice. Iznad je **semantička struktura elementa**.

Odjeljak - **<section>** - Tematski grupiran sadržaj, vjerojatno s naslovom. Odgovara poglavlju knjige ili logičkom dijelu stranice weba. Korištenje elementa <section> je prikladno ako je njegov sadržaj vrijedan prikaza u kratkom pregledu dokumenta. Može sadržavati druge elemente za opis semantičke strukture dokumenta: <article>, <aside>, <nav>, <section>.

Članak - **<article>** - Članak – neovisna, cjelovita kompozicija sadržaja. Odgovara novinskom članku, postu na forumu, postu na blogu, komentaru čitatelja itd. Članci mogu biti ugniježđeni jedan unutar drugoga, s time da članak dijete zadržava kontekst članka roditelja. Može

sadržavati druge elemente za opis semantičke strukture dokumenta: <article>, <aside>, <nav>, <section>.

Zaglavlje i podnožje - **<header>** i **<footer>.** Element zaglavlja uobičajeno sadrži: Element naslova/podnaslova (<h1> / <h2> / …<h6>), sadržaj i/ili meta podatke stranice, odjeljka ili članka. Element podnožja uobičajeno sadrži: (Meta-) podatke o stranici, odjeljku ili članku. Oboje tip prikaza block.

Navigacija - **<nav>** - Ukoliko na HTML stranici postoji jedan ili više dijelova isključivo namijenjenih navigaciji (skup poveznica na druge stranice ili dijelove iste stranice), omeđuje se elementom <nav>. Može sadržavati druge HTML elemente, uključujući one za opis semantičke strukture.

Sporedni sadržaj - **<aside>-** Sporedni sadržaj, marginalno vezan uz lokalni kontekst, izdvaja se u element <aside>. Može sadržavati druge elemente za opis semantičke strukture dokumenta: <article>, <aside>, <nav>, <section>.

Odlomak - **<p>** - Grupira tekst u logičku cjelinu, vizualno odvaja odlomak od okolnog sadržaja stranice Weba. Tip prikaza: block.

Predformatirani tekst **<pre>**-Čuva originalni format teksta zajedno s praznim znakovima i prijelomom redaka, uglavnom proporcionalan font. Tip prikaza je block.

Stilizacija teksta – Eksplicitnom stilizacijom izravno određujemo izgled dijela teksta, npr. podvučenost, ukošenost... Implicitnom stilizacijom deklariramo dio teksta kao istaknut, aktivni stilovi (definirani unutar preglednika i/ili korisnički) definiraju izgled teksta. Implicitna stilizacija sadrži i semantičku informaciju o označenom tekstu. Oznake: Eksplicitne: , , <u></u>, , … Implicitne: , , <mark></mark>, <small></small>, <del></del>, <ins></ins>, … Tip prikaza: inline. Implicitna stilizacija teksta vezanog uz računala. Font je proporcionalan, ali nisu očuvani prijelomi redaka i uzastopni prazni znakovi. **<code>** - programski kod, **<samp>** - izlaz programa, **<kbd>** - unos u program, **<var>** - varijable.

Elementom **<time>** može se označiti dijelove teksta HTML dokumenta koji definiraju vrijeme i/ili datum. Označeni tekst je razumljiv čovjeku, ali zbog oblika zapisa ne nužno razumljiv i računalu. Računalu razumljiv zapis označenog vremena i/ili datuma moguće je navesti kao vrijednost atributa datetime.

```
Što se navodno desilo <time datetime="1969-07-20
20:17"> dvadesetog srpnja 1969. godine u 20 sati i 17 minuta?
```

Entiteti: predefinirani znakovi ili nizovi znakova. Referenciranje entiteta: &ime_entiteta; ili &#kod_entiteta. Tijekom prikaza HTML dokumenta reference na entitete se zamjenjuju njihovim vrijednostima. Npr. > se zamjenjuje znakom >. Češće korišteni entiteti: nbsp - non-breakable space – preglednik forsira prikazivanje praznog znaka, ne ignorira uzastopne prazne znakove, gt (>), lt (<), amp (&), quot ("), aps (‘), euro (€), copy (©), ...

Slika nije dio HTML kôda stranice. Referencira se vanjski izvor slike korištenjem URI i rezervira prostor na stranici za njen prikaz - nakon uspješnog dohvata resursa koji sadrži sliku zapisanu u nekom od podržanih oblika (gif, jpg, png …). Element `<img src="URI slike" alt="opis"/>`. Atributi src i alt su obavezni. Tip prikaza: inline. Definiranje dimenzija prikazane slike: Atributi height i width elementa img ili parametri height i width globalnog atributa style (preporučeno). Dimenzije slike na stranici: 1) Dimenzije nisu navedene: slika se prikazuje u izvornoj veličini, potreban prostor se određuje nakon dohvata, 2) Eksplicitno navedene i visina i širina: rezervira se prostor na stranici i prije dohvaćanja slike, vrši se prilagodba veličine dohvaćene slike na zadane dimenzije, 3) Eksplicitno navedena samo jedna od dimenzija: druga dimenzija se određuje tako da se čuvaju omjeri izvornih dimenzija, prostor se određuje nakon dohvata slike. Element **<figure>** označava sliku kao samostalan dio sadržaja strance (blok), povezan s lokalnim kontekstom. Npr. slika korištena za grafičku reprezentaciju poveznice ne smatra se samostalnim dijelom sadržaja stranice. **<figcaption>** - za opis slike koja je u <figure>.

```
<img src="Joshua_Tree.jpg" alt="Joshua2" height="400" width="200"/>
```

Tri vrste **lista**: Neuređene liste, Uređene liste, Opisne liste. Lista je definirana elementom liste (ovisno o vrsti) i članovima liste. Tip prikaza: Čitava lista je blok, pojedini član liste je blok.

Neuređena lista- Poredak članova liste nije bitan, oznaka elementa liste je grafičkog oblika (kvadrat, krug itd.) Lista je sadržana unutar elementa **<ul>.** Pojedini član liste je omeđen elementom **<li>.** Upravljanje oznakom: svojstvo globalnog atributa style list-style-type**.** Vrste oznake: disc, circle, square, none.

Uređena lista -Poredak članova liste je bitan, oznaka elementa liste (alfa-)numeričkog oblika (broj, slovo …). Lista je sadržana unutar elementa **<ol>.** Pojedini član liste je omeđen elementom **<li>.** Upravljanje oznakom elementa liste - atribut style: 1 - cijeli brojevi, A - velika slova, a - mala slova, I – veliki rimski brojevi, i – mali rimski brojevi. Početna vrijednost oznake elementa liste – atribut start. Preokrenut poredak brojanja elementa liste – booleovski atribut reversed.

Opisna lista-Element liste se sastoji od pojma i njegove definicije. Lista je sadržana unutar elementa **<dl>.** Pojam koji se opisuje naveden je unutar elementa **<dt>.** Definicija pojam navedena je unutar elementa **<dd>.**

Tablice- Tablični prikaz sadržaja unutar ćelija tablice, Tip prikaza: blok. Tablica definirana elementom **<table>**, sastoji se on 1..n redaka definiranih elementima **<tr>**. Svaki redak se sastoji od 1..m ćelija definiranih elementima **<td>.** U <style> na pocetku se mogu definirati stvari za sve tablice. **<th>**- za prvi redak tablice tj. redak zaglavlja. Atribut colspan određuje broj stupaca preko kojih se ćelija proteže (u desno). Atribut text-align za poravnanje teksta. S border se uređuje okvir, postoji border- collapse za spajanje okvira. Atribut rowspan određuje broj redaka preko kojih se ćelija proteže (prema dolje). Definicija grupiranja stupaca unutar elementa colgroup. Pojedine grupe stupaca (1..n stupaca) se određuju elementom col i atributom span (broj stupaca u grupi). **<thead>, <tbody> i <tfoot>** su za uređivanje zagravlja, tijela i podnožja tablice.

```
<table sty      width: 50%">
 <caption>    imjer HTML tablice</caption>
 <tr> <th colspan="4">Varijable</th></tr>
 <tr> <th>1</th> <th>2</th> <th>3</th> <th>4</th> </tr>
 <tr> <td>a11</td> <td>a12</td> <td>a13</td> <td>a14</td> </tr>
 <tr> <td>a21</td> <td>a22</td> <td>a23</td> <td>a24</td> </tr>
 <tr> <td>a31</td> <td>a32</td> <td>a33</td> <td>a34</td> </tr>
 <tr> <td>a41</td> <td>a42</td> <td>a43</td> <td>a44</td> </tr>
</table>
```

Poveznice vode korisnika na druge resurse. Poveznica može biti bilo koji vizualni element HTML stranice. Pokazivač nad poveznicom mijenja oblik, najčešće u simbol ruke. Poveznice su vizualno naglašene: neposjećena, posjećena, aktivna. Element sidro **<a>** (anchor). Atribut href: URI resursa – cilja poveznice. Poveznice mogu biti apsolutne i relativne.

```
<p><a href="http://www.fer.hr/index.html">FERWeb</a></p>

<p><a href="Hiperlinkovi2.html"><img src="link_icon.png"
     width="30px"></a></p>
```

Relativan

Bookmarks- I pojedini elementi stranice, ne samo čitava stranica, mogu predstavljati cilj poveznice. Praćenjem poveznice na bookmark, prikazuje se ciljna stranica, tako da element s bookmarkom bude odmah vidljiv Bookmark – potencijalno svaki element s definiranim globalnim atributom id. Poveznica na bookmark unutar istog HTML dokumenta. U definiciji poveznice navodi se #[id bookmarka]. U terminologiji URI-ja znak # označava fragment resursa.

```
<h2 id="b3">Poglavlje treće</h2>  <a href="#b3">LIBRO PARVO</a><br/>
 <a href="Bookmarks.html#b3">Relativna poveznica i bookmark</a>
</p>
<p>
 <a href="https://www.fer.unizg.hr/predmet/rppzwpu/primjeri/html/
  Bookmarks.html#b3">Apsolutna poveznica i bookmark</a>
```

Odabrani atributi poveznica: title – globalni atribut, definira tekst tooltipa poveznice, target – definira prozor/tab unutar kojeg će se prikazati cilj poveznice, dozvoljene vrijednosti atributa su:_self – umjesto trenutno prikazanog dokumenta, _blank – unutar novog prozora ili taba, ime_okvira – unutar imenovanog okvira (novi ili već postojeći), _parent, _top – unutar nadređenog ili korijenskog okvira (kod ugniježđenih okvira i elementa iframe, nije obrađeno).

Obrasci- Elementi grafičkog korisničkog sučelja unutar HTML stranice. Služe za unos podataka od strane korisnika. Svaki element obrasca ima dva bitna atributa: name – ime elementa, mora biti jedinstveno, value – vrijednost elementa, preuzima se iz unosa korisnika. Podaci iz obrasca šalju se poslužitelju weba. Šalju se odabranom resursu na strani poslužitelja i obrađuju se od strane resursa. Rezultati se vraćaju kao nova HTML stranica. Kod slanja podataka iz obrasca na poslužitelj, preglednik formira niz parova name=value, za sve elemente obrasca. Svaki element obrasca treba imati jedinstveno ime unutar obrasca. Ako ime nije navedeno, podaci iz elementa se ne šalju poslužitelju, ako imena nisu jedinstvena, poslužitelj ne može razlučiti kojim elementima obrasca su vrijednosti pripadale.

input Element
button
text control
number
radio button
checkbox
password
date
color

select Element
pull-down menu
list box

textarea Element
textarea control

Element **<form>** - Sadrži ostale elemente koji definiraju obrazac.Tip prikaza: block. Važni atributi: action – URI resursa kojem se prosljeđuje sadržaj obrasca, target – cilj rezultata obrade proslijeđenog sadržaja obrasca (isti prozor, novi prozor, imenovani prozor …), method – HTTP metoda za prosljeđivanja sadržaja iz obrasca (GET ili POST).

```
<form target="_self" action="/processForm.php”
   method="GET">
 … elementi obrasca …
</form>
```

Element **<input>**- Reprezentira različite tipove elemenata grafičkog korisničkog sučelja. Tip prikaza: inline. Bitni atributi elementa input: type – definira tip elementa**,** name – ime elementa obrasca, value – inicijalna vrijednost.

Opis elementa obrasca s elementom **<label>.** Tip prikaza: inline. Element obrasca navodi globalni atribut id s jedinstvenom vrijednošću unutar dokumenta. Labela referencira pripadajući element obrasca navođenjem njegove jedinstvene vrijednosti unutar atributa for. Grupiranje elemenata obrasca **<fieldset>.** Opis grupe **<legend>**. Oboje tip prikaza block.

```
<form target="_self" action="/processForm.php" method="GET">
<p><label>Korisničko ime: <input type="text" name="username" value="enter your username"
size="30"></label></p>
<fieldset>
<legend>Uloga</legend>
<p><input type="radio" id="admin" name="role" value="admin" checked>
<label for="admin">Administrator</label></p>
<p><input type="radio" id="user" name="role" value="user">
<label for="user">Korisnik</label></p>
<p><input type="radio" id="guest" name="role" value="guest">
<label for="guest">Gost</label></p>
</fieldset>
</form>
```

Tekstno polje - Tip: "text", tip: ”password” – umjesto unošenih znakova prikazuju se *. Često korišteni atributi: hidden, readonly, pattern, maxlength, size, name, required.

Radio Button- izbor jedne od navedenih opcija. Type=“radio“. Ima atribut name- grupa opcija mora imati istu vrijednost, ima atribut value-vrijednost prosljeđena poslužitelju. Može se inicijalno odabrati jedna opcija sa checked.

Checkbox-odabir nijedne ili više između navedenih opcija. Tip: ”checkbox", Svaka opcija bi trebala imati različitu vrijednost atributa name. Sa checked može se staviti više inicijalno odabranih. Sa disabled- onemogućena promjena, vrijednost neće biti slana na poslužitelj.

Button- definira gumb i akciju na pritisak gumba**.** Tip: ”button". Vrijednost će biti poslana na poslužitelj samo ako je aktiviranjem baš tog gumba i pokrenut sam proces slanja (odnosi se na gumbe tipa type= ”submit”). Postoji i type: “reset“.

```
<input type="button" name=”gumb" value="Klikni me!"
   onclick="alert('Zašto!?')">
```

Datum: Tip: ”date".

```
Dolazak: <input type="date" name="trip_start"><br>
Odlazak: <input type="date" name="trip_end">
```

Datoteka- Prijenos datoteke na poslužitelj. Moguće samo korištenjem HTTP metode POST (atribut method elementa form). Tip: ”file".

```
<label>Odaberite datoteku za predaju: <input type="file"
name="datoteka"></label>
```

Padajuća lista – element **<select>**. Odabir samo jedne ponuđene opcije. Elementima **<option>** definirane sve ponuđene opcije. Sa selected se postavlja inicijalno odabrana opcija. Atribut size odllučuje broj istovremeno vidljivih opcija. Atribut multiple za višestruk odabir opcija.

```
<select name="jezik" size="5" multiple>
  <option value="njemački">Njemački jezik</option>
  <option value="francuski">Francuski jezik</option>
  …
  <option value="kineski">Kineski jezik</option>
</select>
```

Područje teksta **<textarea>.** „Spin poruka“ je inicijalni sadržaj.

```
<form action="/processForm.php">
  <textarea name="tijelo" rows="10" cols="50">
  Spin poruka
  </textarea>
</form>
```

Dimenzije područja teksta (stupaca i redaka)

Klikom na gumb submit podaci iz obrasca se kodiraju i šalju na URI naveden u atributu action obrasca korištenjem HTTP metode navedene u atributu method.

Formiranje liste podataka za slanje. - Formira se niz parova name=value odvojenih znakom &. U listu se uključuju svi elementi obrasca po redosljedu navođenja u definiciji obrasca, uz sljedeće iznimke*: element obrasca ima naveden atribut disabled, element obrasca je tipa checkbox i nije odabran, element obrasca je tipa radio i nije odabran, element obrasca je tipa button i nije aktiviran, element obrasca nema navedenu vrijednost atributa name. Rezultirajući niz znakova se kodira metodom navedenom u atributu **enctype** elementa form, ako atribut nije naveden, koristi se podrazumijevana metoda. Tri metode kodiranja u upotrebi (norma HTML 5): application/x-www-form-urlencoded (podrazumijevana metoda), multipart/form-data, text/plain.

CSS

CSS je jezik kojim se opisuje prezentacija dokumenta pisanog u HTML-u. Pravila CSS-a se sastoje od selektora i proizvoljnog broja deklaracija (parovi svojstvo-vrijednost): **selector { property: value; property: value; …}.**

Opcije pisanja CSS-a: 1) Inline- Najlošija opcija - CSS se može pisati unutar elementa koji se želi stilizirati. Narušava se načelo "separation of concerns", stil se isprepliće sa sadržajem što nije dobro. 2) Unutar zaglavlja- CSS se piše u zaglavlju unutar oznake <style>. Odabiremo elemente unutar dokumenta i postavljamo im stil. 3) Posebna datoteka- CSS se piše u posebnoj datoteci koju referenciramo (uključujemo) iz zaglavlja pomoću oznake <link>. Smanjuje se HTML datoteka, CSS se može keširati. Preporučeni način.

site.css

```
h1 {
  color: blue;
  text-align: center;
}
```

site.html

```
<!DOCTYPE html> <html>
<head>
  <link rel="stylesheet" href="site.css">
</head>
```

Selektori se koriste za odabir elemenata kojima se želi pridijeliti stil. Moguće je odabrati nula, jedan ili više elemenata. Dijelimo ih u pet kategorija: 1) Jednostavni selektori 2)Atributni selektori 3) Kombinirani selektori 4) Selektori pseudo klasa 5) Selektori pseudo elemenata.

Jednostavni selektori- Univerzalni selektor(*)- odabire sve elemente na stranici, rijetko se koristi, Element selektor-odabire elemente na temelju imena elementa, ID selektor- odabire elemente na temelji indentifikatora, koristi se znak "#" i indentifikator, Class selektor- Odabire elemente na temelju klase. Koristi se znak "." i ime klase. Moguće je kombinirati element i selektor class -> h1.naslov odabiru se samo elementi h1 koji imaju klasu "naslov". Moguće je grupirati više elemenata, odvajaju se zarezom pri navođenju.

Atributni selektori - Sintaksa: element[atribut="vrijednost"], mogu se izostaviti i element i vrijednost.

Kombinirani selektori- jednostavne selektore je moguće kombinirati s četiri moguća kombinatora.

Operator (kombinator)	Primjer	Objašnjenje
_ (razmak)	div span	Odabire sve span element unutar div elementa, bez obzira koliko duboko
>	div>span	Odabire span elemente koji su djeca div elementa, tj. neposredno ispod
+	div+span	Odabire neposredne susjede iste razine (*siblings*), za ovaj primjer odabire span koji neposredno nakon diva i imaju istog roditelja
~	div~span	Odabire susjeda iste razine koji slijedi prvog i ne mora nužno biti neposredno iza prvog. Za ovaj primjer odabire sve span elemente koji slijede nakon diva i imaju istog roditelja.

Selektori pseudo-klasa- Pseudo-klase se koriste kako bi se opisala posebna stanja elementa, npr. posjećena poveznica, fokusirani element, itd. Kombiniraju se sa selektorima: **selektor:pseudo-klasa { … }.**

Selektor	Primjer	Objašnjenje
:hover	div:hover	Odabire div elemente nad kojima je pokazivač miša
:first-child	li:first-child	Odabire li element koji je prvo dijete svog roditelja
:required	input:required	Odabire sve input elemente koji imaju atribut required

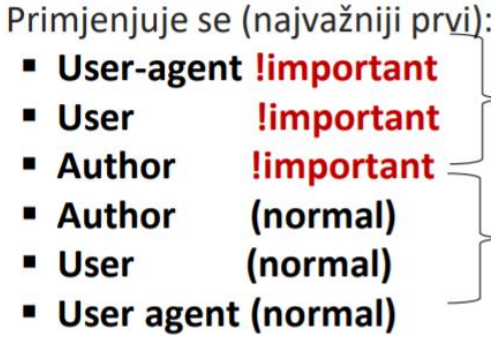
Selektori pseudo-elemenata - Pseudo-element se koriste kako bi se stilizirali određeni dijelovi elementa, npr. prvo slovo, prvi redak. Kombiniraju se sa selektorima: **selektor::pseudo-element { … }.**

Selektor	Objašnjenje
::after	Ubacivanje sadržaja nakon odabranog elementa
::before	Ubacivanje sadržaja prije odabranog elementa
::first-letter	Odabire prvo slovo
::first-line	Odabire prvi redak
::selection	Odabire dio elementa koji je odabrao korisnik

Chrome **DevTools** je skup alata koji su ugrađeni u preglednik Google Chrome. Uvelike olakšavaju razvoj, moguće je mijenjati stranicu uživo, pratiti mrežni promet, debagirati javascript, itd.

Kaskadnost - određivanje koje pravilo/svojstvo će se primijeniti, uzimaju se obzir tri koncepta: 1) Izvor odnosno važnost 2) Specifičnost 3) Poredak koda.

Izvor i važnost - pravila CSS-a mogu doći iz različitih izvora: User-agent: svaki preglednik ima neke postavke, Author: stilovi definirani unutar web-aplikacije, User: korisnici mogu lokalno premostiti određene postavke, rijetko korišteno, Važnost pravila je moguće modificirati naredbom tj. iznimkom "!important" (što se ne preporuča).



Specifičnost i poredak - Neformalno: "precizniji, specifičniji selektori imaju veći prioritet". Primjenjuje se (najvažniji prvi): Inline stilovi (1000), #ID selektori (100), .class, :pseudo-class i [attribute] selektori (10), <TAG> i ::pseudo-element selektori (1), <body> i * (0), Specifičnost se izračunava tako da se zbrajaju težine(brojevi u zagradama). Ako je specifičnost ista, gleda se poredak i pobjeđuje zadnji.

Kaskadnost - ukupan algoritam: Prema specifikaciji, kaskadna pravila se evaluiraju u četiri koraka: 1) Pronaći sve deklaracije koje se odnose na neki element odnosno svojstvo 2) Sortiraj po (1) važnosti i izvoru: (1. Ako su različite važnosti/izvora primijeni pravilo (i stani) 2. Inače, nastavi sa sljedećim korakom), 3) Sortiraj po (2) specifičnosti (1. Ako su različite specifičnosti, primjeni pravilo najveće specifičnosti (i stani) 2. Inače, nastavi), 4) Primijeni (3) poredak tj. zadnje navedeno pravilo.

Nasljeđivanje - Color i font-family se nasljeđuju, width, height, border, itd. se ne nasljeđuju. Naslijeđena svojstva imaju najmanji prioritet, manji čak od user-agent stilova. Moguće je kontrolirati nasljeđivanje s inherit-nasljeđuje od roditelja, initial-vraća na postavke preglednika i unset- vraća na default opcijama.

Font-4 su obitelji fontova: serif, sans-serif, cursive i monospace. Serif su oni sa crticama i produženjima na slovima. Postoje 3 opcije za korištenje fontova: 1) Fontovi s lokalnog (korisničkog) računala. Postoji mali skup fontova koji su zastupljeni na gotovo svim operacijskim sustavima, npr. TNR i Arial 2) Web fonts-Fontovi koji preglednik dohvaća preko mreže i koji su pohranjeni na poslužiteljima neke treće strane 3) Poslužiteljski fontovi poput prethodnog, samo što fontove poslužujemo s vlastitog poslužitelja.

```
<body>
  <h1 style="font-family: Consolas, monospace;">1. Consolas</h1>
  <h1 style="font-family: 'PT Mono', monospace;">2. PT Mono@Google</h1>
  <h1 style="font-family: mojPTMono, serif;">3. Moj PT Mono@localhost</h1>
</body>
 <link href="https://fonts.googleapis.com/css?family=PT+Mono&display=swap"
rel="stylesheet">
```

```
<style>
  @font-face {
    font-family: mojPTMono;
    src: url(PTMono-Regular.ttf);
  }
</style>
```
2
3

@font-face direktiva- možemo definirati novi font. Moguće je definirati još detaljnija svojstva fontova osim same familije, npr: font-weight: {normal, bold, 100, 200, …, 900}. Za svaku kombinaciju mora postojati datoteke (definicija) tog font-facea. Ima i font-size i font-style.

Svojstvo	Moguće vrijednosti
font-weight	normal|bold|bolder|lighter|number|initial|inherit
font-style	normal|italic|oblique|initial|inherit
font-size	medium|xx-small|x-small|small|large|x-large|xx-large|smaller|larger|*length*|initial|inherit

Formati fontova – TTF, OTF, EOT, WOFF, WOFF2-budućnost zbog 30% bolje kompresije. Koristiti WOFF2, pa WOFF pa TTF. Još neka svojstva teksta: text-align, text-indent, letter-spacing, line-height.

Boja- Model boja je apstraktni matematički model koji opisuje kako boje predstaviti uređenim n-torkama brojeva, tipično trima ili četirima vrijednostima odnosno komponentama boje. Neki modeli su RGB, HSL, HSV, CMYK, Munsell, Natural. CSS podržava RGB i HSL, a koristit ćemo RGB. RGB je aditivni model u kojem se boja postiže "zbrajanjem" crvene, zelene i plave komponente. Zadajemo intenzitete u rasponu od 0 do 255 (jedan bajt), 24-bitna boja, tzv. "truecolor", 2563 = 16.777.216 boja. Opcije: 1. rgb(r, g, b), 2. rgba(r, g, b, alpha) , alpha definira (ne)prozirnost, (opacity) u intervalu [0, 1], 3. Heksadecimalna notacija: #rrggbb, 4. Skraćena hex notacija (ako su obje hex znamenke iste): #rgb, 5. Putem imena, 140 imena.

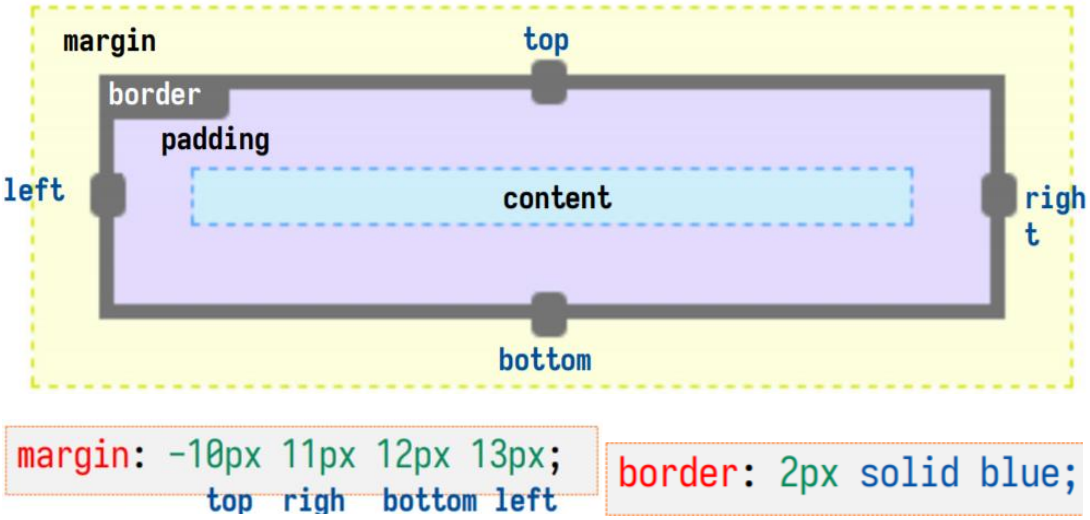
Slika kao pozadina - Moguće je postaviti N slika, podrazumijevano ponašanje: Gornji lijevi kut, ponavlja se horizontalno i vertikalno. Dakle, background je kratica za ova svojstva: background-color, background-image, background-position, background-size, background-repeat, background-origin, background-clip. background-attachment.

```
<head><style>
    body {
      background-color: #ccc;
      background-image: url('./fer-logo-xs.jpg'), url('./stars-xs.png');
      background-repeat: no-repeat, repeat;
    }
</style></head>
<body></body>
```
Relativni put od mjesta gdje je taj CSS napisan

Što bi se prikazalo da nema ove linije?

```
background: #ccc url('./fer-logo-xs.jpg') no-repeat;
```

Box model: margin, border, padding




```
margin: -10px 11px 12px 13px;
        top  righ  bottom left
```

```
border: 2px solid blue;
```

CSS omogućuje da se elementu promijeni narav prikaza pomoću display: block | inline | inline-block | … **Box-sizing-** omogućuje da se visina i širina primjenjuj na sve, a ne samo na sadržaj. Svojstvo **display** služi za stiliziranje linkova. In addition, links can be styled differently depending on what state they are in. The four links states are: a:link - a normal, unvisited link, a:visited - a link the user has visited, a:hover - a link when the user mouses over it, a:active - a link the moment it is clicked.

Uz standardne stvati **tablice** se mogu urediti i ovako: For zebra-striped tables, use the nth-child() selector and add a background-color to all even (or odd) table rows. A responsive table will display a horizontal scroll bar if the screen is too small to display the full content. Add a container element (like) with overflow-x:auto around the <table> element to make it responsive.

Liste- The list-style-type property specifies the type of list item marker. The list-style-image property specifies an image as the list item marker. The list-style-position property specifies the position of the list-item markers (bullet points- točke za natuknice ili kj si već odabrao), ima outside i inside. The list-style-type:none property can also be used to remove the markers/bullets. Note that the list also has default margin and padding. To remove this, add margin:0 and padding:0 to <ul> or <ol>. The list-style property is a shorthand property. It is used to set all the list properties in one declaration: To da je natuknica neka slika. When using the shorthand property, the order of the property values are: list-style-type (if a list-style-image is specified, the value of this property will be displayed if the image for some reason cannot be displayed), list-style-position (specifies whether the list-item markers should appear inside or outside the content flow), list-style-image (specifies an image as the list item marker). If one of the property values above are missing, the default value for the missing property will be inserted, if any. Na standardan način se dodaju boje.

Gumbovi- Sve standardne stvari. Zaokruženi su ako staviš border-radius. Možes - button:hover koristit. Sa box-shadow daješ sjenu. Use the opacity property to add transparency to a button (creates a "disabled" look). Tip: You can also add the cursor property with a value of "not-allowed", which will display a "no parking sign" when you mouse over the button. Remove margins and add float:left to each button to create a button group. Use display:block

instead of float:left to group the buttons below each other, instead of side by side. Još se razni efekti poput pressed mogu dodati gumbu.

Obrasci- možeš dodat standardne box model stvari. By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding outline: none; to the input. Use the :focus selector to do something with the input field when it gets focus. If you want an icon inside the input, use the background-image property and position it with the background-position property. Also notice that we add a large left padding to reserve the space of the icon. In this example we use the CSS transition property to animate the width of the search input when it gets focus. Use the resize property to prevent textareas from being resized (disable the "grabber" in the bottom right corner).

S **float** se mogu postići responzivni elementi. **Position** - static (default), relative, fixed, absolute, sticky- ostaje uvijek na ekranu, z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

Responzivni dizajn(RD) - je pristup dizajnu web stranica koji omogućava dobar prikaz na različitim uređajima i veličinama prozora ili zaslona. Može ga se ostvariti sa: Viewport/scale, Fluidne, proporcionalne širine, a ne fiksne - % umjesto px, Media queries, Odgovarajuće jedinice, responzivne slike i tekst(rem umjesto px), Tehnologija za responzivni razmještaj: CSS Flexgrid, CSS Grid, Radni okviri: Bootstrap, Semantic UI, Foundation, W3.CSS,…, Metodologije:mobile-first pristup, progressive enhancement.

Pikseli: Problem: pametni telefon ima puno piksela, ali su jako mali – ne možemo istovjetno koristiti piksele i na monitoru i na pametnom telefonu. 1in = 2.54cm = 96px.

Viewport je područje prozora na kojem se vidi sadržaj web stranice. Ako je sadržaj veći od viewporta, onda preglednik prikazuje scrollbare. Dvije klase problema, uski uređaji: Niske rezolucije: iscrtavaju sadržaj na širem, virtualnom viewportu, pa onda prilagođavaju na uži - dobro samo za "stare" stranice koje nisu uzimale u obzir pokretne uređaje, Visoke rezolucije: pikseli su jako maleni, jedan piksel na monitoru fizički npr. 3x veći od piksela na mobitelu.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Width: širina viewporta, može biti konstanta ili najčešće device-width što je širina ekrana u CSS pikselima kod uvećanja od 100%. initial-scale: inicijalno uvećanje (zum) kod učitavanja stranice, može se limitirati s maximum-scale, minimum-scale i user-scalable.

CSS jedinice- Koriste se u raznim CSS svojstvima, npr. font-size, margin, height. Nekad mogu biti negative vrijednost (npr. margin). Dvije kategorije: 1) Apsolutne (ne preporučuju se za ekrane): mm, cm, in 1in = 2.54cm = 96px, pt 1pt = 1/72th of 1in, pc 1pc = 1/6th of 1in, px 1px = 1/96th of 1in, ali kao što smo vidjeli iz prethodnih slajdova, px nije baš "apsolutan" - ovisi o uređaju: moguće je da više fizičkih piksela čini jedan softverski. 2) Relativne- Izračunavaju se u odnosu na neku drugu temeljnu vrijednost. Tako definirani stilovi se bolje ponašaju kod promjene veličine ekrana. Pored % (koji se odnosi na roditelja), to su: em-font size of the element, rem-font size of the root element, vw 1% of viewport's width, vh 1% of viewport's height, vmin 1% of viewport's smaller dimension, vmax 1% of viewport's larger dimension. Rem ili em: rem je jednostavniji, izračuni i otklanjanje pogrešaka je olakšano, em je pogodniji za modularne komponente. Tip: Koristiti em za svojstva koja trebaju skalirati s fontom, inače rem.

Svojstvo	"Preporučene" vrijednosti
font-size	rem
padding, margin,	rem, em
border	px
width, height	%, vw, vh
top, bottom, left, right	%

@media CSS pravilo- @media pravilo omogućuje uporabu različitih stilova na različitim uređajima. Pravila je moguće je definirati s obzirom na: Visinu i širinu vidljivog prozora (viewport), Visinu i širinu vidljivog uređaja, Orijentaciju (portret ili pejzaž), Rezoluciju. Navodi se medijski upit i CSS blok koji će se primijeniti ako je upit zadovoljen. Tipično se postavi više granica zbog mnoštva uređaja s različitim ekranima i rezolucijama.

```
@media screen and (max-width: 768px) {
  div.resp {
    width: 100%;
    background-color: orange;
    float: none;
  }
}
```

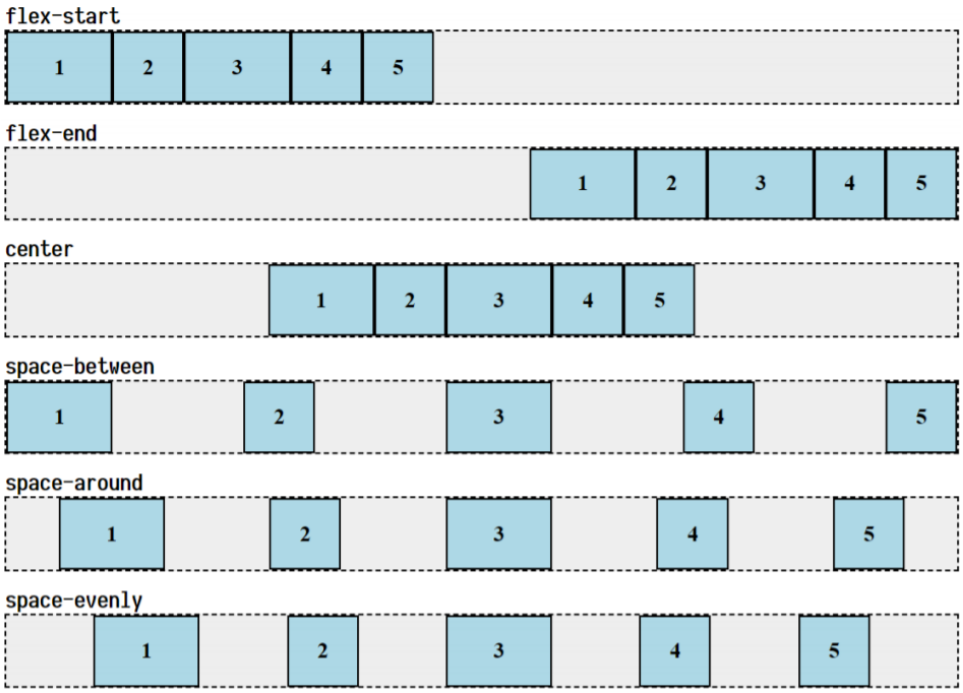
Media je moguće koristiti i kod <style>,<link>, <source> i drugim HTML elementima. Navodi se "media=" atribut.

Flexbox- Moderan i fleksibilan način responzivnog razmještaja elemenata. Danas je podržan u svim modernim preglednicima. Ustroj roditelj-djeca, osnovna ideja je da roditelj (container) može mijenjati veličinu (visina/širina) svoje djece kako bi što bolje popunio raspoloživu površinu (prilagođavajući se tako različitim uređajima i ekranima). Flexbox je: Jednodimenzionalni model razmještaja. U istom smislu je agnostičan po pitanju usmjerenja (direction agnostic) odnosno definira se u odnosu na os.

Flexbox - jednodimenzionalni model - Glavna (main axis) i sporedna os (cross axis) su okomite. Glavna os se definira s flex-direction i može biti: row, row-reverse, column, column-reverse.

Flexbox: flex-flow - Kombinacija flex-direction i flex-wrap. Flex-wrap definira kako će se elementi ponašati s obzirom na prelazak u nove linije (wrap), flex-wrap: nowrap | wrap | wrap-reverse, flex-flow default je: row nowrap.

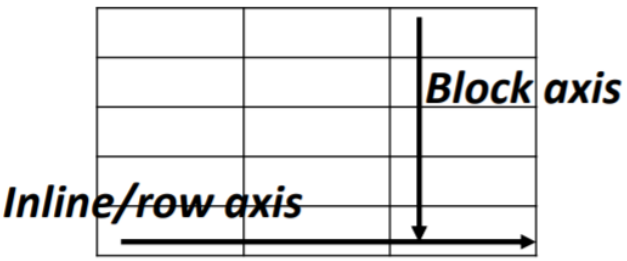
justify- definira poravnanje na gl. osi. Ima flex-start, flex-end, center, space-between, space-around, space-evenly. **align-content-** kao justify ali po sporednoj osi. Isti se modifikatori koriste. **align-items** - Definira kako poravnati po sporednoj osi u trenutnoj liniji. Ima .flex-start, .flex-end, .center, .baseline, .stretch.



Flexbox svojstva djece: **order:** <broj>- Elementi se inače prikazuju redom kako su navedeni u kodu, ovim je moguće promijeniti taj poredak. **flex**: <'flex-grow'> <'flex-shrink'>? ||<'flex-basis'>- Kratica za tri svojstva navedena gore. Flex-grow definira proporciju rasta elementa (cijeli broj bez jedinice), align-self: auto | flex-start | flex-end | center | baseline | stretch. Ovim svojstvom možemo nadjačati align-items svojstvo od roditelja te pojedini element drugačije poravanati.

CSS Grid - 2D sustav ("rešetka") sustav namijenjen oblikovanju web stranica: Block axis, Inline/row axis. Podržan u svim modernim preglednicima. Može se kombinirati s Flexboxom Sastoji se od: roditelja (display: grid | inline-grid), 1-N djece (neposredno ispod roditelja).

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
}
```



Praznina među redovima i stupcima je gap. Može se mjenjati sa grid-row-gap i grid-column-gap. Sa grid-column/row-start/end se može na veličinu utjecat. S grid-area se mogu i dodijeliti imena.

```
@media all and (min-width: 800px) {
  .wrapper {  grid-template-areas:
'header header header header header'
'aside1 main main main aside2'
'footer footer footer footer footer';
 } }
```

Još jedna CSS jedinica, ali samo za grid- **fr.** A flexible length or <flex> is a dimension with the fr unit, which represents a fraction of the leftover space in the grid container.

justify-items: poravnanje po inline/row osi. align-items: poravnanje po block osi.

Ove sve je eksplicitni grid. Postoji i implicitni. Ako postoji više elemenata nego ćelija u gridu, ili ako stavimo element izvan grida onda automatski nastaju nove staze (tracks) koje u kombinaciji s eksplicitno definiranima (ako postoje) čine implicitni grid.

Bootstrap- Najpopularniji CSS radni okvir za razvoj responzivnih, mobile-first web-aplikacija; sadrži: CSS i Javascript (opcionalno). Uključuje "sve što je potrebno" za brzi razvoj: Razne pomoćne CSS klase (poravnanja, boje, pozicije, …), Responzivni grid sustav, Gotove komponente (carousel, jumbotron, itd.), Ikone. Moguće je imati različite grafičke teme. Slobodan, otvorenog koda. Načelo: dodajemo klasu nekom elementu da ga oblikujemo.

Bootstrap grid- 2D sustav: koriste se container, row i column elementi tj. klase. U v4 temeljen na flexboxu. Container sadrži retke (rows), retci kolone (columns). Moguća su gniježđenja redaka i kolona. Mobile-first, responzivan Ugrađene četiri granice: xs < 576px, sm ≥576px, md ≥768px, lg ≥992px, xl ≥1200px.

Bootstrap: container klasa - Obavezna ako se koristi grid. Sadrže druge elemente, te postavljaju padding i opcionalno poravnavaju elemente. Tri vrste: container: ima postavljen max-width, container-fluid: uvijek 100%, container-{granica}: fluidan do granice, inače ima max-width.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
.container	100%	540px	720px	960px	1140px
.container-sm	100%	540px	720px	960px	1140px
.container-md	100%	100%	720px	960px	1140px
.container-lg	100%	100%	100%	960px	1140px
.container-xl	100%	100%	100%	100%	1140px
.container-fluid	100%	100%	100%	100%	100%

Bootstrap grid: row i col* klase - U container se stavljaju retci, a u retke kolone. Samo kolone mogu biti izravna djeca od redaka. Sam (HTML) sadržaj se smješta u kolone. Tri varijante: col - kolone jednake širine, col-{granica} vrijedi tek od granice na više, col-{granica}-{širina} kolona zadane širine, pri čemu je ukupna širina retka 12. Granica se može i izostaviti.

JAVASCRIPT (JS)

JavaScript je programski jezik koji se tipično koristi u izradi web-aplikacija. On je skriptni programski jezik. Skripni jezici se ne prevode uz pomoć prevoditelja, već se njihov kôd parsira i odmah izvodi. JavaScript se izvodi na tzv. stroju za JavaScript, koji se ponekad nalazi i u okviru internetskog preglednika i tada zove tzv. virtualnim strojem JavaScripta. JavaScript je standardiziran tzv. specifikacijom ECMAScripta. Chrome V8 je stroj koji izvodi JS na Chrome-u.

JavaScript se vrlo često izvodi u internetskom pregledniku što značajno ograničava skup dozvoljenih radnji. Rad s datotekama je vrlo ograničen, komunikacija dvije stranice je vrlo ograničena, JavaScript kôd može komunicirati sa vlastitim poslužiteljem, ali mu je komunikacija s drugim poslužiteljima ograničena.

JS se navodi unutar bloka **<script>** ili vanjska JS datoteka poveže sa HTML datotekom. Funkcija **alert()** korisniku vraća upozorenje tj. poruku, atribut onClick kod gumba izvodi zadanu funkciju pritiskom istog.

Console.log() ispisuje poruku unutar konzole (kartica Console) – može imati N argumenta bilo kojeg tipa (vrijednosti, varijable, čak i funkcije). U kartici Console također se prikazuju i greške.

Tri načina upravljanja stranicom: DOM – Document Object Model (model HTML stranice + manipulacija), CSSOM – Cascading Style Sheet Object Model (model CSS-a + manipulacija), BOM – Browser Object Model (ugrađene funkcije internetskog preglednika).

JavaScript je skriptni programski jezik slabog tipa-Varijable su promjenjivog tipa, tj. tip varijable se može mijenjati prilikom izvođenja programa, u jednu varijablu je moguće prvo pohraniti vrijednost jednog tipa, pa zatim novu vrijednost drugog tipa.

Varijable se deklariraju ključnom riječi let (ili var)- var ima doseg tijela funkcije, a let bloka u kojoj se nalazi. Ime varijable smije sadržavati samo znakove, znamenke i simbole $ i _.Prvi znak imena varijable ne smije biti znamenka, imenovanje je camelCase. Konstante se deklariraju ključnom riječi const-konvencija: Sve riječi naziva konstante pišu se velikim slovima, riječi se međusobno odvajaju donjom crtom (_).

Numerički tipovi podataka su number i bigInt. Number-do 15 znamenki, za decimalne i cijele brojeve, može pohraniti i Infinity, -Infinity i NaN. BigInt- pohrana cijelih brojeva proizvoljene duljine. Konstanta bigInt-a mora završiti s n.

Znakovni tipovi- string- Vrijednosti tipa string moraju biti omeđene navodnicima. Postoje tri vrste navodnika: dvostruki, jednostruki i ukošeni – svi se koriste ravnopravno. Ukošeni navodnici omogućavaju da se unutar vrijednosti tipa string ugradi proizvoljni izraz koji će se izvršiti. Ne postoji tip podataka za pohranu znaka – jedan znak se pohranjuje kao string duljine jednog znaka.

Boolean- prima true ili false.

Specijalni tipovi podataka: null- Kada varijabla ima vrijednost null ona je prazna i vrijednost joj nije poznata. Undefined-Kada je varijabla deklarirana, ali joj nije pridružena vrijednost. object- služi za grupiranje varijabli različitih tipova podataka. symbol- služi za definiranje jedinstvenih identifikatora za objekte.

U JavaScriptu se pretvorbe tipova podataka obavljaju automatski, sukladno potrebama. Moguće je napraviti i eksplicitne pretvorbe tipova podataka, ako su potrebne.

Pretvorbe u numerički tip podataka (number) se obavljaju po ovim pravilima: undefined postaje NaN (Not a Number), null postaje 0, true postaje 1, false postaje 0. Varijabla tipa string se nastoji pretvoriti u broj (tipa number), string se pri tome nastoji pročitati kao broj. Čisti prazni niz znakova se pretvara u broj 0. Ako se pojavi greška (npr. niz znakova "1a2" sadrži a u sredini) rezultat je NaN.

Pretvorbe u logički tip podataka (boolean) se obavljaju po ovim pravilima: undefined postaje false, null postaje false, NaN postaje false, 0 postaje false. prazan znakovni niz ("") postaje false, Sve ostale vrijednosti postaju true.

Većina operatora ima iste funkcije kao i oni u Javi. Operator potencije (**). Operator >>> je posmak uz punjenje 0. Tu su i === i !==, oni su boolean operatori koji u obzir uzimaju tipove varijabli. Za operator + vrijedi: ako je jedan operand tipa string, i ostali se pretvaraju u string. && je većeg prioriteta od | |. Usporedba null i undefined operatorom === uvijek rezultira s false. Usporedba null i undefined međusobno operatorom == daje true. Usporedba null ili undefined s ostalim vrijednostima operatorom == uvijek daje false. Prilikom usporedbe null i vrijednosti koje nisu null ili undefined operatorima <, >, <= i >= null se pretvara u 0, a undefined u NaN.

If-else if-else naredba slična je istoimenoj naredbi u jeziku C/C++. **Switch** naredba slična je istoimenoj naredbi u jeziku C/C++. **while i do-while** petlje slične su istoimenim petljama u jeziku C/C++. **for** petlja slična je for petlji u jeziku C/C++. **for...of** iterira po elementima (bilo kojeg iterabilnog podatka – npr. string, array), **for...in** iterira po ključevima (indeksima).

```
const array = ['a', 'b', 'c'];
for(const element of array){
  console.log(element)
}

// a
// b
// c
```

```
const array = ['a', 'b', 'c'];
for(const element in array){
  console.log(element)
}

// 0
// 1
// 2
```

Funkcija u JavaScriptu započinje ključnom riječi function. Parametre funkcije se odvaja zarezima. Funkcija završava s return. Parametri funkcije nemaju naveden tip, već se samo navode nazivi parametara. Primitivne varijable se iz pozivajućeg programa predaju po vrijednosti (dolazi do stvaranja kopije). Funkcije mijenjajući parametre mijenjaju samo lokalnu kopiju primitivnih varijabli (promjene nisu vidljive u pozivajućem programu). Ako se u pozivajućem programu ne navede vrijednost parametra, automatski se pri pozivu funkcije postavlja na undefined. Moguće je navesti i podrazumijevane vrijednosti parametara funkcija.

Stvaranje funkcija:

```
function double(value) {
  return value * 2;
}
let square = function(value) {
  return value * value;
}
```

Deklaracija funkcije

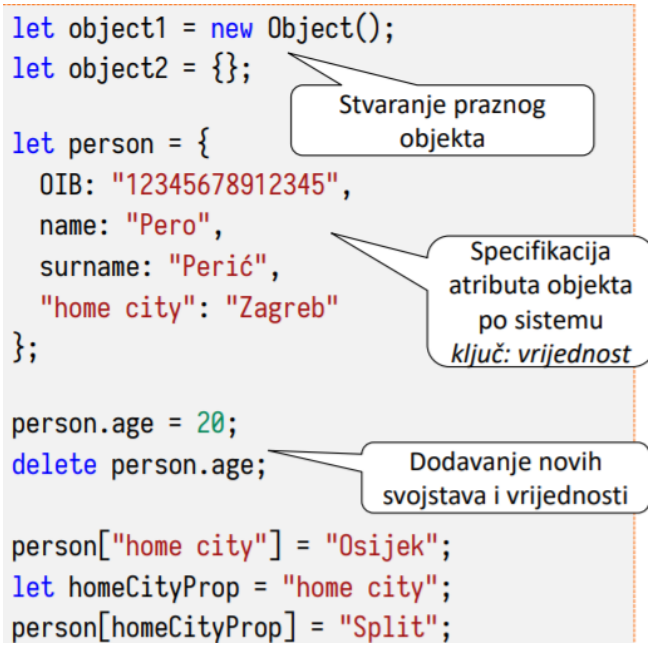
Funkcijski izraz

Moguće je stvoriti anonimne funkcije te koristiti lambda izraze.

```
let squareLambda = (value) =>
value * value;
```

```
let doubleArrayAnn = changeArray([1,2,3,4], function(value) {return value*2;});
```

Let i const imaju doseg bloka, a var ima globalni doseg.

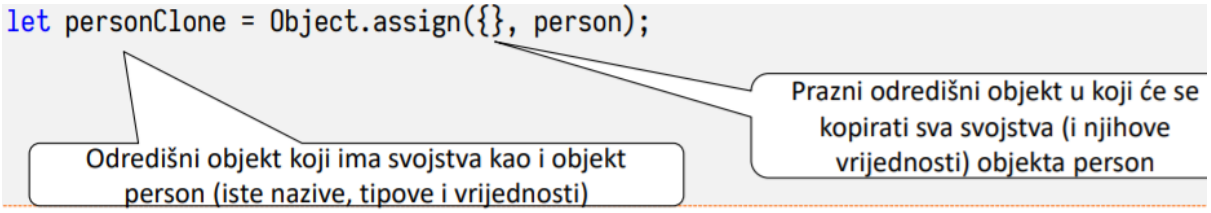


Sa for in se može iterirati po ključevima.

Provjera postojanja atributa radi se: usporedbom sa undefined ili ključnom riječi in (preporučuje se koristiti).

```
let existsNationality = personExtended.nationality !== undefined;
let existsNationality2 = "nationality" in personExtended;
```

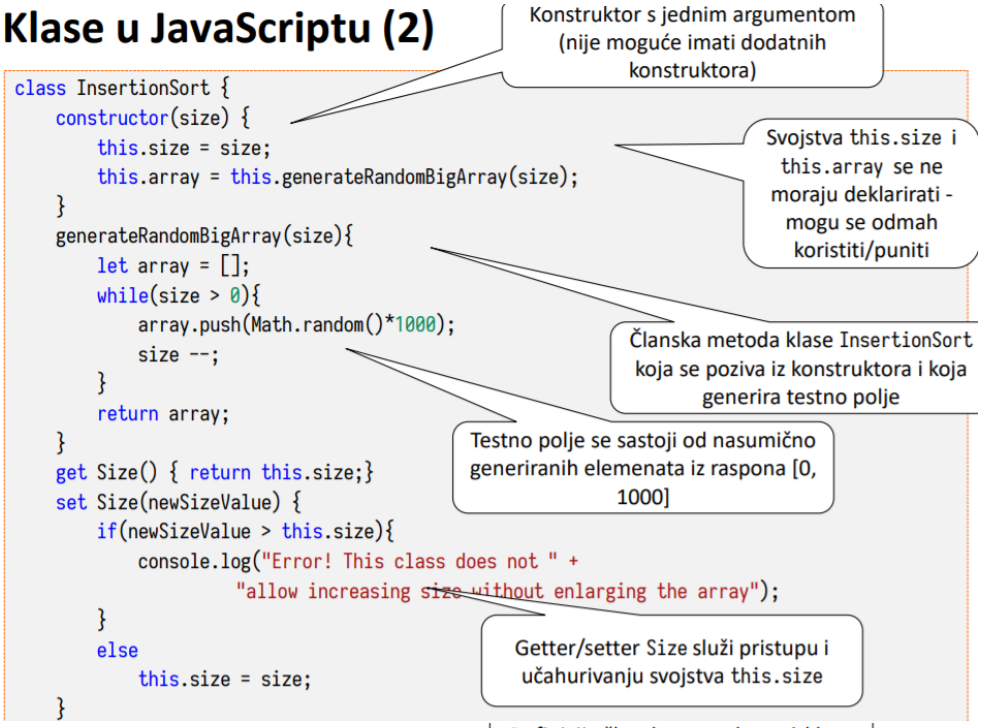
Operator pridruživanja (=) između objekata kopira samo referencu, a ne i svojstva. Za izradu kopije svojstava objekta koristi se metoda Object.assign.



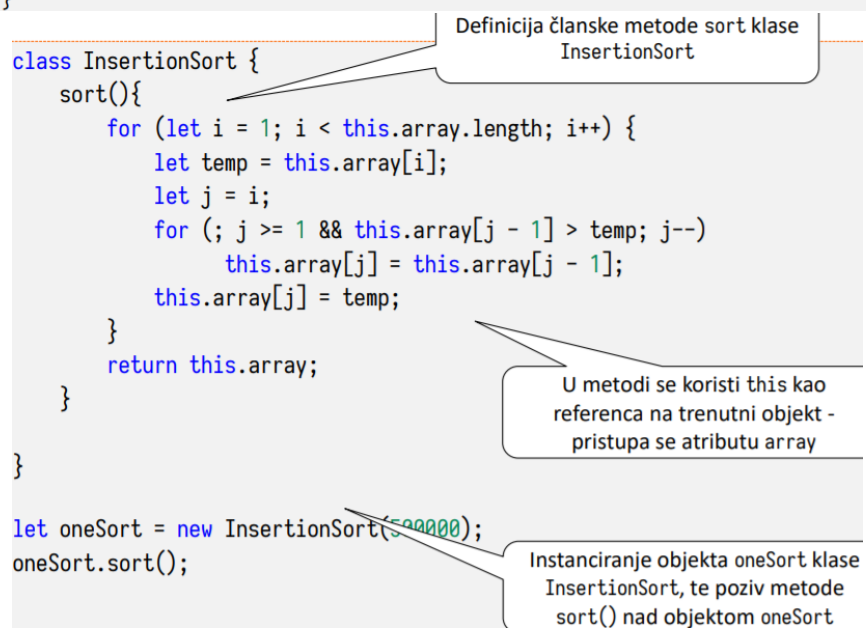
Postoje funkcije Object.keys(obj), Object.values(obj), Object.entries(obj). Varijabli označenoj s const se ne može promijeniti referenca!. U funkcijama standarno ko i u javi postoji this. On je nadostupan u lambda izrazima.

Klase u JavaScriptu slične su klasama u drugim programskim jezicima. Klase u JavaScriptu su interno funkcije. Klase u JavaScriptu su sintaksni šećer (eng. syntactic sugar) i mogu se izvesti i funkcijama. Nasljeđivanje se vrši ključnom riječi extends. Članske varijable i metode su neke klase dostupne su i u njoj izvedenoj klasi putem ključne riječi super. Nadjačavanje se izvodi tako što se u izvedenoj klasi navede metoda istog imena kao što je ima i metoda u osnovnoj klasi. Može se koristiti ključna riječ static za statičke metode. Ne postoje modifikatori vidljivosti poput private i protected. Za provjeru je li objekt dio klase koristi se `obj instanceof Class`.

Klase u JavaScriptu (2)




```
class InsertionSort {
    constructor(size) {
        this.size = size;
        this.array = this.generateRandomBigArray(size);
    }
    generateRandomBigArray(size){
        let array = [];
        while(size > 0){
            array.push(Math.random()*1000);
            size --;
        }
        return array;
    }
    get Size() { return this.size;}
    set Size(newSizeValue) {
        if(newSizeValue > this.size){
            console.log("Error! This class does not " +
                    "allow increasing size without enlarging the array");
        }
        else
            this.size = size;
    }
```




```
class InsertionSort {
    sort(){
        for (let i = 1; i < this.array.length; i++) {
            let temp = this.array[i];
            let j = i;
            for (; j >= 1 && this.array[j - 1] > temp; j--)
                  this.array[j] = this.array[j - 1];
            this.array[j] = temp;
        }
        return this.array;
    }

}

let oneSort = new InsertionSort(500000);
oneSort.sort();
```

Polja- Služe pohrani kolekcije vrijednosti ili objekata. Metode: push (… items) - dodaje element(e) na kraj, pop() - briše i vraća element s kraj, shift() - briše i vraća element s početka, unshift(…items) - dodaje element(e) items na početak polja, splice(index, num) - od pozicije index briše num elemenata, slice(index1, index2) - vraća novo polje s elementima od pozicija indeks1 do index2, concat(...items) – nadodaje element(e) items u polje i vraća kopiju novog polja, indexOf/lastIndexOf(item, pos) – traži element item, počevši s pozicijom pos, includes(value) – provjerava je li value u polju, find/filter(func) – vraća sve vrijednosti koje zadovoljavaju uvjet (funkciju) func, findIndex(func) - isto kao i find, ali vraća indeks, forEach(func) – poziva funkciju func za svaki element polja, map(func) – zove func za svaki element polja i stvara novo polje temeljem rezultata te ga vraća, sort(func) – sortira polje, te ga vraća, reverse() – obrće polje, te ga vraća, split()/join() – pretvara string u polje i obrnuto.

Mape- Metode: new Map() – stvara novi objekt – mapu, map.set(key, value) – pohranjuje vrijednost value pod ključ key, map.get(key) – dohvaća vrijednost za dani ključ (undefined ako ne postoji ključ), map.has(key) – provjerava postoji li ključ key u mapi, map.delete(key) – briše vrijednost za dani ključ key, map.clear() – briše sve iz mape, map.size – vraća broj parova ključ-vrijednost pohranjenih u mapi.

Iznimke u JavaScriptu slične su iznimkama u drugim programskim jezicima. Try-catch-finally funkcionira ko u Javi. Neke specifičnosti JavaScripta- Radi samo sinkrono - greške koje se dogode u kôdu asinkrono pokrenutom iz bloka try se neće uhvatiti u bloku catch koji je nadovezan na taj blok try, catch može koristiti informaciju o grešci preko objekta greške koji mu se automatski predaje. Objekt za greške tipa Error se sastoji od tri glavna svojstva: name - vrsta greške (npr. SyntaxError), message - poruka o detaljima greške, stack - lanac poziva metoda sa sistemskog stoga koji je doveo do greške. Objekti greške (Error) se po potrebi mogu stvoriti i aktivirati (baciti, eng. throw). Može se koristiti klasa Error, ali i SyntaxError, ReferenceError, TypeError, RangeError itd.

Internetski preglednici su tzv. okolina domaćina (eng. Host environment) za HTML stranice. Svaki internetski preglednik omogućava prikaz DOM-a.

Document Object Model (DOM) je programska reprezentacija internetske stranice. Glavni objekt je document i on predstavlja internetsku stranicu koja je učitana u internetski preglednik. Svaki element internetske stranice je predstavljen objektom koji se nalazi unutar glavnog objekta document. Struktura navedenih objekata je hijerarhijska i prati strukturu HTML-a. DOM-u se tipično pristupa iz JavaScripta (npr. manipulira se objektima DOM-a kako bi se napravile promjene internetske stranice). Glavni objekt DOM-a je document i ima niz podobjekata: document.documentElement-predstavlja oznaku <html>, document.body-predstavlja oznaku <body>, može biti null ako stranica nije u potpunosti učitana, document.head- predstavlja oznaku <head>. Nad objektima DOM-a moguće je pozivati brojne metode i koristiti brojna svojstva. Npr. .childNodes (svojstvo koje je kolekcija), .firstChild (objekt koji je prvo dijete nekog objekta), .lastChild (objekt koji je zadnje dijete nekog objekta), .nextSibling (sljedeći objekt na istoj razini), .previousSibling (prethodni objekt na istoj razini), .parentNode (roditeljski objekt). Svojstva i metode koje rade samo s elementima (oznakama), a ne i tekstom i ostalim objektima: .children (elementi djeca), .firstElementChild (prvo dijete element), .lastElementChild (zadnje dijete element), .previousElementSibling (prethodni element na istoj razini), .nextElementSibling (sljedeći element na istoj razini), .parentElement (roditeljski element). DOM je moguće pretraživati kako bi se pronašli specifični elementi. Takvi elementi moraju imati atribut id postavljen na neku vrijednost. Pretraživanje se obavlja koristeći: `document.getElementById(element id value)` - Koristi se id elementa `document.querySelectorAll(CSS query)` i `document.querySelector,` za složenije slučajeve pretraživanja. QuerySelector, QuerySelectorAll. Uzimaju referencu na element/elemente iz DOM-a i omogućavaju rad s njima u JavaScriptu.

```
// Select by element type - First element of type body
const body = document.querySelector('body');
```

Document.createElement-Stvara HTML element u JavaScriptu. Element se može dodati bilo gdje na stranicu.

```
const div = document.createElement("div");

const body = document.querySelector("body");
body.appendChild(div);
```

```
function changeWeather(){
 let tvrdnjaPara = document.getElementById("tvrdnja");
  if(tvrdnjaPara.style.color == "green")
    tvrdnjaPara.style.color = "red";
  else
    tvrdnjaPara.style.color = "green";
  if(tvrdnjaPara.innerHTML.indexOf("nije lijep") != -1)
    tvrdnjaPara.innerHTML = "Danas je <i>lijep</i> dan.";
  else
    tvrdnjaPara.innerHTML = "Danas <i>nije lijep</i> dan.";
  tvrdnjaParaSecondSibling = tvrdnjaPara.nextSibling.nextSibling.nextSibling;
  if(tvrdnjaParaSecondSibling.nodeValue == "")
    tvrdnjaParaSecondSibling.nodeValue = "Idemo svi u prirodu!";
  else
    tvrdnjaParaSecondSibling.nodeValue = "";
}
```

Dohvat elementa po id-u

Promjena stila (boje) elementa-tvrdnje

Svojstvo innerHTML služi pristupu HTML-u unutar elementa <p>

Atributom nodeValue pristupa se tekstu objekta koji je tekstualnog tipa

Objekt s tekstom "Idemo svi u prirodu!" je treći objekt nakon <p>. Prvi je '\n', drugi <hr/>.

U JavaScriptu možemo uređivati stilove, tekst, slušače događaja, itd.

Browser Object Model (BOM)- Sadržava globalne objekte preglednika koji su dostupni u cijeloj web-aplikaciji. Objekti omogućavaju korištenje ugrađenih funkcija preglednika. Dostupni objekti su: window(roditelj svim ostalima), location, screen, history, navigator, cookie. Objekt window sadržava metode za upravljanje internetskim preglednikom. Neke od funkcija su: Blur() – oduzima fokus trenutnom prozoru, Focus() – daje fokus trenutnom prozoru, Close() – zatvara trenutni prozor (“tab”) ResizeTo()/MoveTo() – mijenja poziciju/veličinu prozora na zadane koordinate/dimenzije. location- Sadrži informacije o lokaciji prozora. Lokacija = trenutni URL prozora. Lokacija != pozicija prozora.

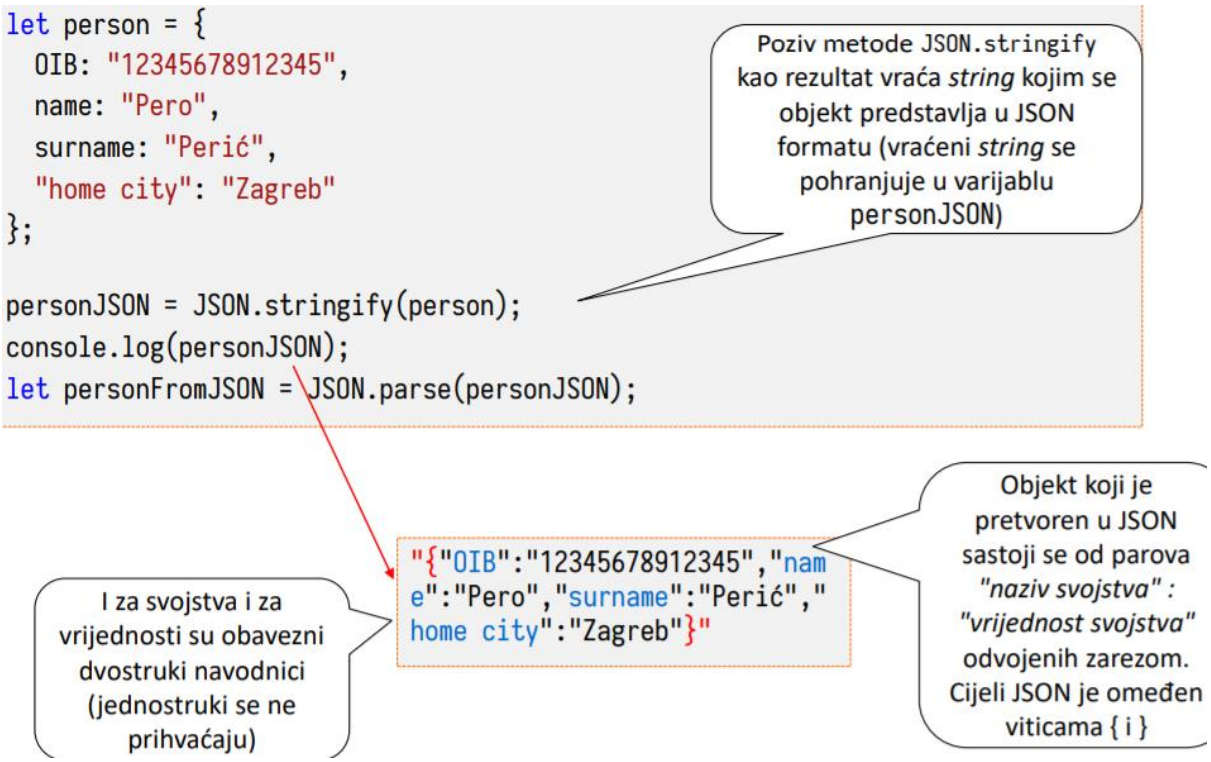
```
// Extracting username from URL
const params = location.search.substr(1).split('&');
const username = params[0].split('=')[1];
```

history- Sadrži informacije o povijesti lokacija prozora. Koristi se za navigaciju na druge lokacije. Prati stanje lokacije i omogućuje dodavanje novog stanja (pushState), zamjenu stanja (replaceState) te vraćanje na prethodno posjećena stanja (back, go).

Komunikacija s korisnikom - alert(message); Prikazuje poruku korisniku i očekuje potvrdu s "U redu". prompt(message, default answer); Postavlja pitanje korisniku i nudi podrazumijevani odgovor (default answer), podrazumijevani odgovor je opcionalan, vraća uneseni odgovor ili null ako je korisnik pritisnuo ESC. confirm(question); Postavlja pitanje korisniku i očekuje od njega odgovor "OK" ili "Cancel", vraća true za odabir "OK", a false za "Cancel" ili ESC.

```
alert("Hello from JavaScript");
let promptAnswer = prompt("JavaScript needs an answer", "default");
let confirmResponse = confirm("A question from JavaScript?");
```

JSON (JavaScript Object Notation) je standard koji se koristi za predstavljanje vrijednosti i objekata. često koristi za razmjenu podataka klijenta i poslužitelja. U JavaScriptu postoje dvije glavne metode: JSON.stringify - pretvara objekte u JSON format, JSON.parse - učitava objekte iz JSON formata. Pretvorba objekta u znakovni niz u JSON formatu naziva se serijalizacijom.



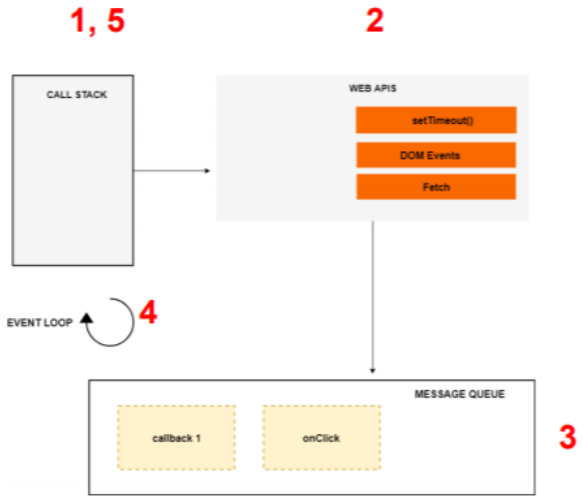
JSON.stringify prihvaća i može serijalizirati: objekte (omeđene s { i }), polja (omeđena s [i]), stringove, brojeve, logičke vrijednosti, null. JSON.stringify prilikom serijalizacije preskače: funkcije, simbole, sva svojstva u kojima je pohranjen undefined. JSON.stringify generira grešku ako dođe do pojave cirkularnih referenci (npr. objekt A referencira objekt B, koji referencira objekt A).

Lokalni spremnik omogućava pohranu podataka na klijentskoj strani u internetskom pregledniku. Koristi se kada je potrebno očuvati podatke uslijed osvježavanja ili ponovnog pokretanja internetskog preglednika. Većina preglednika dozvoljava pohranu barem 2 MB podataka u lokalnom spremniku. Poslužitelj ne može direktno mijenjati podatke pohranjene u lokalnom spremniku, već se promjene izvode samo na klijentu korištenjem JavaScripta. U DevTools može se vidjeti stanje lokalnog spremnika. Lokalni spremnik ime šest metoda: setItem(key, value) – pohranjuje par ključ-vrijednost, getItem(key) – dohvaća vrijednost po ključu, removeItem(key) – briše ključ i njegovu vrijednost, clear() – briše cijeli spremnik, key(index) – dohvaća naziv ključa na danoj poziciji index, length – dohvaća broj pohranjenih parova ključ-vrijednost u lokalnom spremniku. U lokalni spremnik se mogu pohranjivati samo stringovi.

Jednodretvenost i sinkronost u JavaScriptu- u principu JS je jednodretveni jezik u kojemu se naredbe izvršavaju sinkrono. Navedeno može izazvati problem ako dretva počne izvršavati naredbu koja dugo traje, čime dretva postaje blokirana, i s izvršavanjem programa se čeka.

Iako je sinkron, JavaScript omogućava nekoliko asinkronih načina izvedbe kôda: ▪ Brojač (eng. timer) – metoda setTimeout, Događaji DOM-a, Dohvat resursa - metoda fetch, Obećanja (eng. promises).

Prilikom izvedbe asinkronog kôda slijedi se ovaj tijek: 1. Naredba koju treba izvršiti se stavlja na sistemski stog i pokuša izvršiti sinkrono. Ako je naredba asinkrona preusmjerava se na web APIs kako bi započelo započinje njeno asinkrono izvršavanje. 2. Web APIs preuzima asinkronu naredbu i osigurava da se naredba izvrši (što može trajati). 3. Nakon što je naredba izvršena, povratna funkcija (eng. callback) se stavlja u red čekanja poruka. 4. Petlja događaja kontinuirano provjerava red čekanja poruka, i ako u njemu naiđe na povratnu funkciju, stavlja ju na stog. 5. Započinje sinkrono izvršavanje povratne funkcije koju je petlja događaja postavila na sistemski stog.



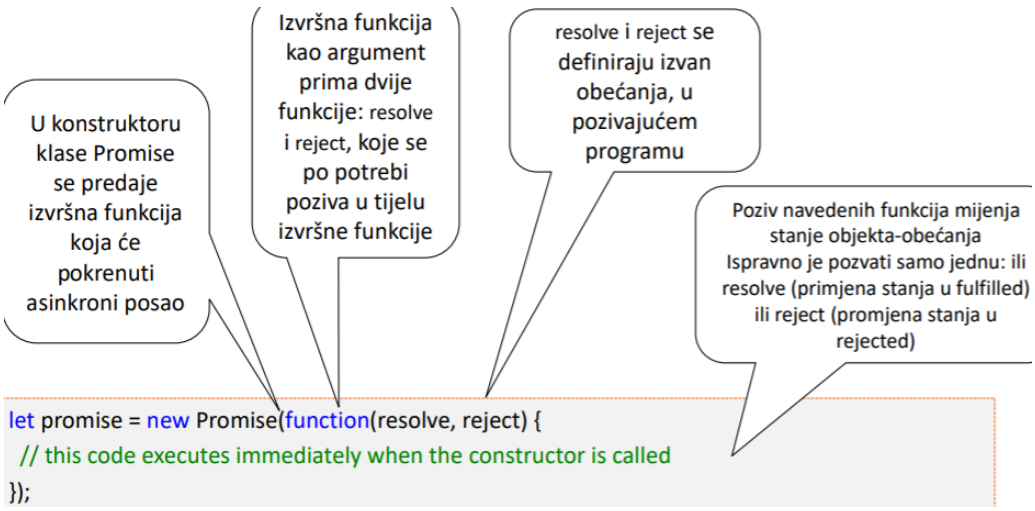
Brojač- tijek radnji tokom izvedbe koda: 1. timerFunction i setTimeout se stavljaju na sistemski stog. timerFunction i setTimeout odmah završavaju. 2. Čeka se 5 sekundi na asinkrono izvršavanje tijela setTimeout 3. Nakon 5 sekundi se kôd iz setTimeouta stavlja u red čekanja 4. Petlja događaja uzima kôd iz reda čekanja i stavlja ga na stog 5. Kôd iz setTimeout se skida sa stoga i izvršava.

```
let timerFunction = () => {
  setTimeout(() => {
    console.log('Asynchronous code execution start after 5 sec');
  }, 5000);
 };
console.log('Before timerFunction call');
timerFunction();
console.log('After timerFunction call');
```

Događaj DOM-a- tijek radnji tokom izvedbe koda: 1. -. 2. Slušač događaja čeka u okolini web APIs na aktivaciju događaja. 3. Nakon što se dogodi klik na gumb, kôd sadržan u onclick se stavlja u red čekanja. 4. Petlja događaja uzima kôd iz reda čekanja i stavlja ga na stog 5. Kôd se skida sa stoga i izvršava.

Asinkrone funkcije se mogu pokrenuti: Nakon nekog događaja. Nakon isteka određenog vremena korištenjem ugrađene funkcije setTimeout.

Obećanja- programski entiteti unutar kojih se postavlja posao koji može potrajati. Posao se obavlja unutar tzv. izvršne funkcije (eng. executor). Nakon što je posao završen obećanje prelazi u stanje uspjeha (fulfilled) ili neuspjeha (rejected). Stvara instanciranjem objekta Promise (pomoću new Promise).



Obećanja se koriste tako da se na njih pretplati (registrira) korištenjem metoda: **.then** - aktivira se ako je izvršna funkcija vratila uspjeh ili neuspjeh, **.catch** - aktivira se samo ako je izvršna funkcija vratila neuspjeh, **.finally** - aktivira se ako je izvršna funkcija vratila uspjeh ili neuspjeh (ali za razliku od then ne daje mogućnost obrade uspjeha ili neuspjeha).

```
promise.then(
  function(result) { /* handle a successful result */ },
  function(error) { /* handle an error */ }
);
```

Drugu

Umjesto slanja teksta poruke kao argumenta metode reject, može se poslati instancirani objekt tipa Error. **catch** (promise.catch()) kao argument prima anonimnu funkciju koja će sepozvati kada (i ako) izvršna funkcija obećanja pozove reject. Ako izvršna funkcija pozove resolve, ova anonimnafunkcija se ne poziva.

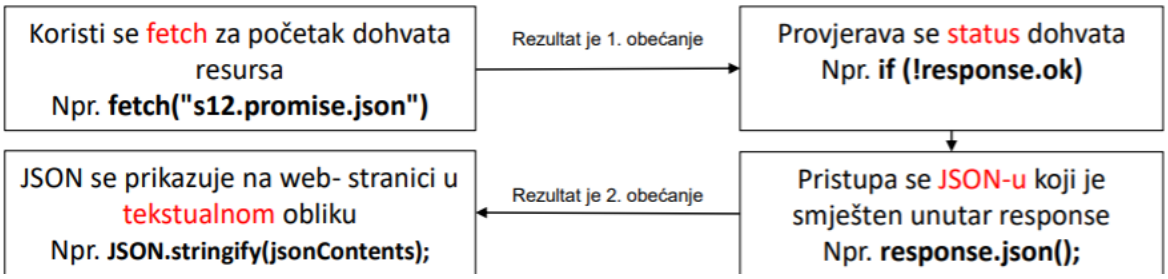
Ako želimo adekvatno koristiti ulančane pozive funkcija then/catch/finally potrebno je iz svake od njih vratiti novi objekt tipa Promise, na taj način se objekti tipa Promise ulančano prenose u iduću razinu i ponovno omogućuju rad metoda then/catch/finally.

```
promise.catch(
  function(error){
    return new Promise(function(resolve, reject){
      reject(error);
    });
})
.then(
  function(result){ console.log("Resolve:" + result) },
  function(result){ console.log("Reject:" + result) }
);
```

Unutar catch se stvara novi objekt tipa Promise, koji se linijom reject(error) postavlja u stanje greške i prosljeđuje u then.

then sada prima gore stvoreni objekt tipa Promise, prepoznaje da je u stanju greške te se ispisuje:
`Reject:Error: This is generated from the promise executor!`

Metoda **fetch** u JavaScriptu je dio internetskog preglednika i služi učitavanju različitih resursa.
1) fetch vraća objekt tipa Promise koji sadrži status dohvata (je li dohvat uspio) – 1. obećanje.
2) Dohvaćeni sadržaj je također dostupan kao novo 2. obećanje.



Koristi se fetch za dohvat datoteke u kojoj je pohranjen JSON

Prvo obećanje vraća odgovor za koji je potrebno provjeriti svojstvo ok (koje govori da li je učitavanje uspjelo)

U prvom then-u se obrađuje obećanje dobiveno od fetch-a

```
let promise = fetch("s12.promise.json");
promise
.then(
 function(response) {
  if (!response.ok) { throw new Error("Cannot load json file"); }
  else { return response.json(); }
 },
 function(error) { throw error; }
)
.then(jsonContents => {
  let spanElement = document.createElement('pre');
  spanElement.innerHTML = JSON.stringify(jsonContents);
  document.body.appendChild(spanElement);
 })
.catch(err => {
  console.log("An error occured while loading json");
});
```

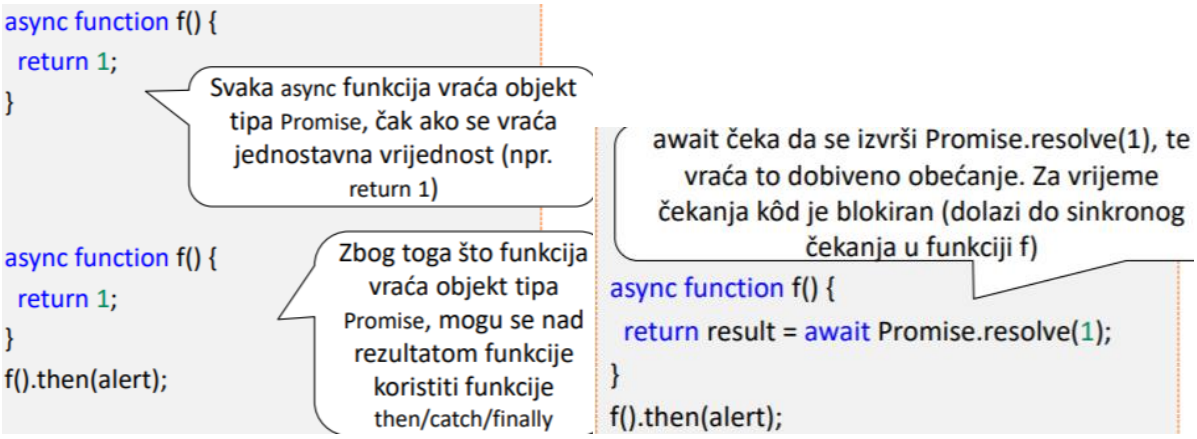
S response.json() se stvara novo obećanje koje se prosljeđuje u idući then blok na daljnju obradu

U drugom se then-u obrađuje obećanje dobiveno od response.json

Prikaz JSON-a na web-stranici

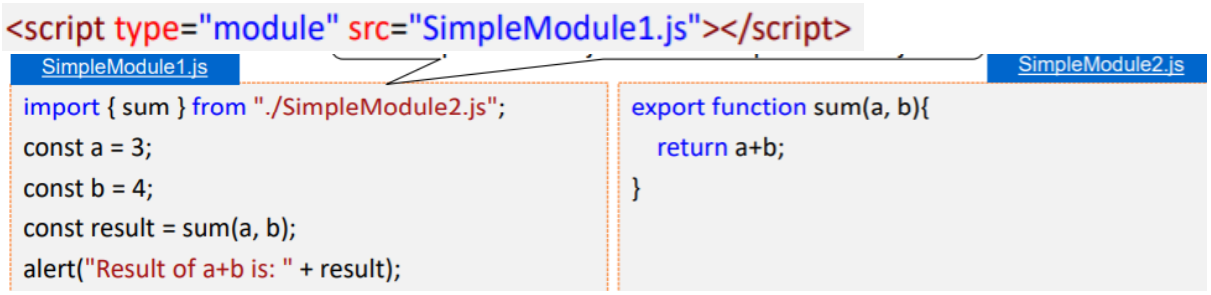
U slučaju greške bilo u bilo kojem od then blokova ispisuje se poruka na konzolu

Funkcije s async/await- Ključne riječi async i await se tipično koriste u okviru iste funkcije. Ključnom riječi await se čeka na izvršavanje neke operacije (tipično dugotrajne). await je jedino moguće koristiti unutar funkcije obilježene s async. Svaka funkcije obilježena s async mora vratiti objekt tipa Promise. Programska linija s await se ponaša sinkrono (blokira se tijek programa unutar funkcije i čeka se na izvršavanje).



Obećanja i async/await su ravnopravni programski konstrukti, i koriste se ovisno o tome što se programom želi postići. Obećanja su načelno asinkrona i često se ulančavaju korištenjem then/catch/finally. Async/await u odnosu na obećanja ima nešto jednostavniji kôd , ali uzrokuje potencijalna čekanja u kôdu. await čeka na sinkrono izvršavanje naredbi i blokira kôd unutar funkcije sve dok se naredba sa await ne izvrši. U slučaju da se await ekstenzivno koristi, moguće je sporije izvršavanje kôda. Obećanja i async/await moguće je kombinirati.

Moduli- Moduli služe organizaciji kôda u JavaScriptu i tipično se koriste kod programskih rješenja s puno programskih linija. Moduli su datoteke s kôdom u JavaScriptu i tipično imaju ekstenziju .js. Svaka datoteka je jedan modul. Iz jednog je modula moguće po potrebi učitati drugi modul. Pristup iz jednog modula drugom modulu se regulira direktivama: **expor**t - označavaju se programski entiteti (varijable, funkcije, klase….) koji su vidljivi izvan modula u kojem se nalaze. **import** - omogućava uvoz funkcionalnosti iz drugog modula. Moduli se tipično koriste kada je aplikacija postavljena na web-poslužitelj.



Ako se jedan te isti modul uveze više puta unutar jednog programa, njegov se kôd izvodi samo jednom, prilikom prvog uvoza. Prilikom učitavanja modula m se stvaraju programski entiteti obilježeni s export, te su od tada dostupni ostalim modulima koji uvoze modul m. Ključna riječ this nije dostupna unutar modula.