

Autori: © Divna Krpan, Saša Mladenović, Goran Zaharija

OOP Vježbe 02

Uvod u JavaScript i okruženje za rad

1. Bilješke

1.1. Što ćemo naučiti?

- Osnovne naredbe JavaScript-a
- Korištenje konzole web preglednika za izvršavanje naredbi
- Postavljanje HTML stranice za izvršavanje JavaScript programa
- Korištenje uređivača kôda MS Visual Studio Code

1.2. Što je JavaScript?

JavaScript je jednostavan *interpreterski* programski jezik. U JavaScript-u su napisani programi koji se izvršavaju u web stranicama. Pomoću tih programa možemo kontrolirati izgled web stranice ili kako stranica reagira kad korisnik klikne na nešto ili pomakne miša.

Za pisanje i izvršavanje JavaScript programa dovoljno je imati:

1. uređivač programskog kôda (npr. MS Visual Studio Code),
2. web preglednik (eng. browser) (npr. Google Chrome, Mozilla Firefox).

Svaki web preglednik ima JavaScript *interpreter* (na engleskom ga nazivamo *engine*) koji može izvršiti program pisan u JavaScriptu. Google Chrome koristi *V8 engine*, Firefox koristi *SpiderMonkey*, a Safari koristi *Nitro*.

1.3. Sintaksa JavaScript-a

Svaki programski jezik ima pravila pisanja, a ta pravila nazivamo sintaksom programskog jezika. Važno je zapamtiti:

- U JavaScript-u razlikujemo velika i mala slova.
- Naredbe završavaju znakom **točka-zarez** (;).
- Blokovi kôda započinju i završavaju *vitičastim zagradama*: { }.

- Pravila za pisanje naziva varijabli i funkcija.

Premda upotreba znaka *točka-zarez* (;) nije obavezna, ipak se preporuča jer se kod izvršavanja nekog programa može dogoditi da interpreter ubaci točku-zarez gdje ne bi trebalo. Može se dogoditi da prelazak naredbe u novu liniju slučajnim ili namjernim pritiskanjem tipke Enter ne znači kraj naredbe. To može dovesti do neočekivanog izvršavanja programa. Ako pišemo točku-zarez, onda je jasno gdje naredba završava, čak i ako prelazi u novi red. Prema tome, točka-zarez funkcionira kao i točka u prirodnom jeziku kojom označavamo kraj rečenice.

Uvlačenje blokova kôda pomaže čitljivosti programa kao u C#-u, a ne utječe na izvršavanje naredbi kao što je to slučaj u Python-u.

1.3.1. Pravila pisanja naziva varijabli i funkcija

Osnovna pravila pisanja naziva varijabli i funkcija su slična onima u programskom jeziku C#:

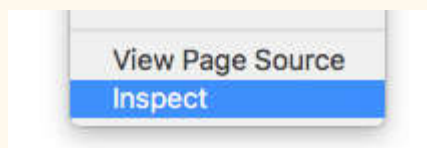
- Znak kojim počinje naziv mora biti slovo engleske abecede ili jedan od posebnih znakova: *podcrta* (_) ili znak dolara (\$).
- Naziv ne može početi znamenkom.
- Naziv ne može sadržavati razmake.
- Naziv ne smije biti ključna riječ programskog jezika.
- Treba izbjegavati nazive globalnih varijabli ili funkcija koje su predefinirane u samom jeziku (npr. Array, Date, Boolean, Object, String, ...).

Prema dogovoru, u programskom jeziku C# primjerice nazivi varijabli počinju malim početnim slovom, a nazivi metoda, klasa i svojstava započinju velikim početnim slovom. Poštivanje tih pravila pomaže čitljivosti programa jer prema nazivu možemo brže zaključiti o čemu se radi, no to nije sintaksna pogreška. Kod pisanja programa u JavaScript-u ćemo se nastaviti pridržavati tih dogovora.

1.4. Izvršavanje naredbi u konzoli

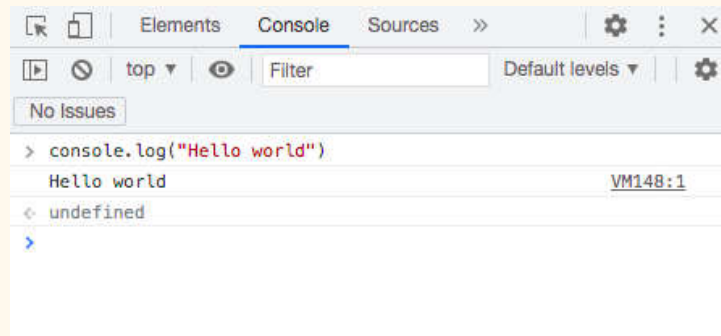
Konzoli web preglednika Chrome možemo pristupiti tako da:

- otvorimo Chrome praznu stranicu (ili *tab*),
- desnim klikom miša na praznom prostoru prozora otvorimo izbornik,
- odaberemo opciju **Inspect**.



Prikaz izbornika sa slike se može razlikovati ovisno o tome koji operacijski sustav koristimo. Također, sam izgled prozora u kojem se nalazi *konzola* (eng. Console) razlikuje se ovisno o

web pregledniku i postavkama preglednika. Iza znaka `>` upisujemo naredbe koje želimo izvršiti.



Nakon svake naredbe koju želimo izvršiti potrebno je pritisnuti **Enter**.

Možemo upisivati jednostavne naredbe kao što su aritmetički izrazi:

```
5 + 3
5 / 2
(2 + 3) * 4.5
```

Nakon što pritisnemo tipku *Enter*, konzola će nam automatski ispisati rezultat naredbe.

Naredba koju ćemo često koristiti za ispisivanje sadržaja na konzolu (slično kao `Console.WriteLine()` u C#-u) je:

```
console.log("OOP");
```

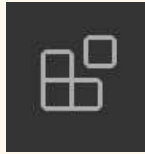
Naredba za ispis poruke (slična kao `MessageBox.Show()` u C#) je:

```
alert("Ovo je poruka");
```

Osim aritmetičkih izraza, možemo upisivati sve naredbe programskog jezika. Obzirom da se nakon pritiska tipke *Enter* naredbe koje smo upisali automatski izvršavaju, onda će nam u konzoli biti prilično nezgodno pisati programe koji se sastoje od više linija kôda.

1.5. Dodaci za VS Code

Visual Studio Code (skraćeno *VS Code*) ima mnogo ugrađenih mogućnosti koje nam omogućuju pisanje i uređivanje kôda u različitim programskim jezicima, ali i raznih alata kao što je npr. ugrađena “konzola” ili “terminal” (ovisno o operacijskom sustavu kojeg koristimo). Za sve što nam treba, a već nije ugrađeno, pretražujemo *dodatke* (eng. *extensions*) koji se nalaze u **Extension Marketplace**. Dodatke ili proširenja možemo tražiti na stranici (<https://marketplace.visualstudio.com/>) ili izravno u VS Code klikom na ikonu:



Klikom na gornju opciju možemo pretraživati dodatke ili pregledati već instalirane. Instalacija i uklanjanje dodataka je jednostavna, nakon što pronademo ono što želimo dodati, potrebno je odabrati opciju **Install** ili **Uninstall**, ako to želimo ukloniti. Jedan od dodataka kojeg ćemo koristiti za izvršavanje naših programa u JavaScriptu je **LiveServer**.

1.5.1. LiveServer

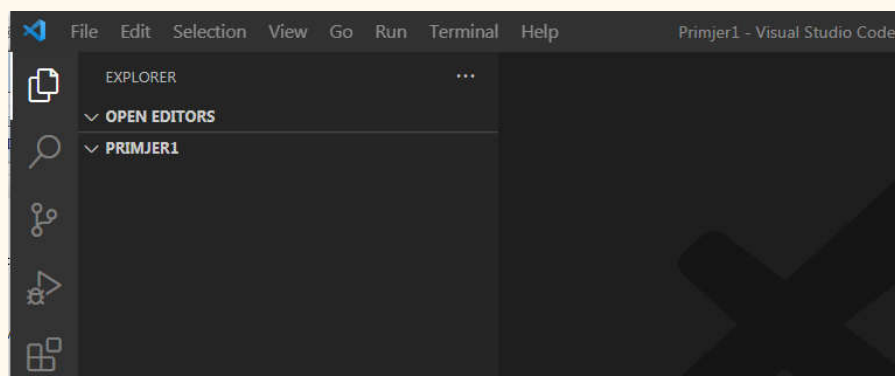
LiveServer simulira izvršavanje stranice na poslužitelju (eng. server). Prepoznat ćemo ga prema slici:



Na stranici <https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer> možete vidjeti i upute kako se koristi. Ako je stranica pokrenuta pomoću LiveServer-a, automatski će se osvježavati čim spremimo nove izmjene u programu.

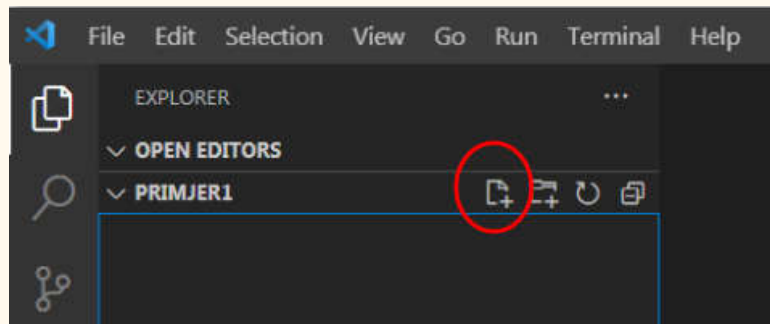
1.6. Postavljanje okruženja za rad

Nakon instalacije programske podrške: Chrome + VS Code + LiveServer, moramo napraviti još jedan korak kao bi mogli pisati programe van konzole. Obzirom da Chrome može izvršiti JavaScript program koji je dio neke web stranice, onda ćemo morati napraviti web stranicu. Za izradu web stranica potrebno je poznavanje HTML jezika, ali tu će nam pomoći VS Code koji ima mogućnost generiranja dijelova kôda u različitim programskim jezicima. Iskoristit ćemo mogućnost automatskog generiranja strukture HTML stranice. Najprije ćemo na disku napraviti novi folder (mapu) kojeg otvaramo pomoću VS Code.



Nakon toga, nastavljamo raditi u VS Code. Novu datoteku stvaramo klikom na ikonicu u **explorer** prozoru ili desnim klikom miša i odabirom opcije **New file**. Potrebno je upisati puni

naziv datoteke: **index.html**. Ekstenzija **html** označava vrstu dokumenta, a *index* je čest naziv za početnu stranicu. Ekstenzija je važna, ali sam naziv može biti bilo što.



Nakon stvaranja, nova datoteka će biti vidljiva u *explorer* prozoru, a otvaramo je klikom miša. Datoteka je u početku prazna. HTML dokument ima definiranu strukturu koju treba napisati kako bi ga mogli koristiti. Neki dijelovi kôda koji se često koriste definirani su u samom okruženju te ih možemo dobiti upisivanjem odgovarajuće kratice i tipke **TAB** na tipkovnici.

Kratice pomoću kojih možemo dobiti započeti HTML dokument u VS Code:

- !
- doc
- html:5

Dobije se sljedeći sadržaj:

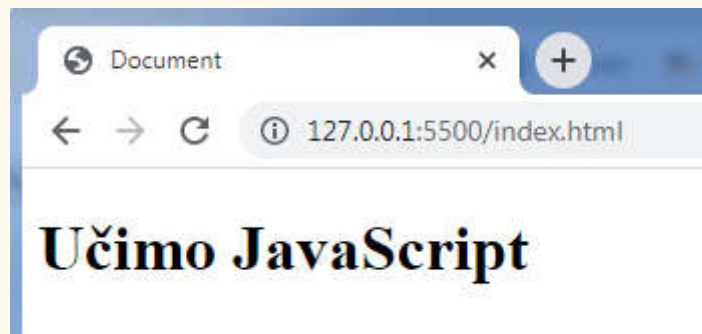
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

</body>
</html>
```

Svaki HTML dokument počinje *oznakom* (eng. tag) **<html>**, a završava oznakom **</html>**. Sastoji se od zaglavlja (**<head></head>**) i tijela (**<body></body>**). Unutar dijela **body** nalazi se sadržaj stranice koji će se prikazati u pregledniku. Za početak je dobro napisati neki tekst kako bi mogli vidjeti da stranica postoji. Upišimo naslov (eng. heading) koji se označava **<h1></h1>**.

```
<body>
  <h1>Učimo JavaScript</h1>
</body>
```

Spremimo stranicu i desnim klikom na *index.html* u prozoru *explorer* biramo opciju **Open with Live Server**. Stranica će se automatski otvoriti u zadanom pregledniku:



JavaScript programe možemo pisati unutar HTML dokumenta, ali to ćemo izbjegavati jer je dobro odvojiti izgled od sučelja stranice, a manja je vjerojatnost da ćemo napraviti pogreške. Nadalje, VS Code pruža bolju podršku za JavaScript ako JavaScript kôd spremamo u poseban **js** dokument.

Napravimo novu datoteku **kod.js**. Ekstenzija **js** označava da datoteka sadrži JavaScript kôd. Kako će naša početna stranica “znati” da treba uključiti *kod.js* prilikom pokretanja? Dodat ćemo blok **<script></script>**. Unutar tog bloka možemo pisati JavaScript programe, ali umjesto toga ćemo samo napisati *putanju* (eng. path) do *kod.js* datoteke. To ćemo napraviti na sljedeći način:

```
<body>
  <h1>Učimo JavaScript</h1>
  <script src="kod.js"></script>
</body>
```

Obzirom da se *kod.js* i *index.html* nalaze u istom folderu, dovoljno je napisati naziv. Ako niste sigurni kako se piše, onda će vam VS pomoći tako da unutar dvostrukih navodnika: **src=""** pritisnete tipke: **CTRL + Space** pa će VS Code pomoći padajućim izbornikom gdje jednostavno možete odabrati lokaciju vaše **js** datoteke.

Na prvi pogled **script** oznaka s imenom datoteke izgleda složeno, ako niste upoznati s HTML sintaksom, ali tu opet pomaže VS Code nadopunjavanjem. Možete generirati blok kôda upisivanjem kombinacije **script:src**, te pritiskanjem tipke TAB, a čim počnemo tipkati script, već sam VS Code predlaže što nam treba tako da ni to ne treba pamtit.

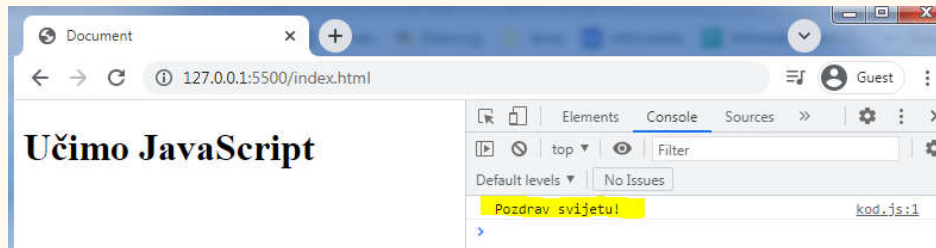
1.7. Prvi JavaScript program

Za razliku od programskog jezika C#, JavaScript program nema strogo definiranu strukturu. Iako struktura nije strogo zadana, pisat ćemo uredno i paziti jer je često redoslijed pisanja bitan.

Prva naredba kojom možemo započeti pisanje programa je standardni ispis poruke “Pozdrav svijetu!”

```
console.log("Pozdrav svijetu!");
```

Ako smo dobro postavili okruženje, ispisat će se poruka na konzoli:



1.8. Tipovi podataka i varijable

Podaci se odnose na informacije koje spremamo u programima (npr. ime, godina rođenja, adresa i sl.). U JavaScript-u postoje dvije kategorije tipova podataka:

- Vrijednosni
 - Tekst (string)
 - Brojevi (number)
 - Logički (boolean)
 - undefined
- Referentni
 - klase, nizovi, ...

Prema tipovima podataka znamo npr. koliko prostora u memoriji zauzima neki podatak ili koje su operacije moguće.

1.8.1. Deklaracija i inicijalizacija varijabli

U C#-u je svaku varijablu potrebno deklarirati pisanjem tipa i naziva varijable. Često je varijable potrebno i inicijalizirati prije nego što ih počnemo koristiti. U Python-u nije potrebno deklarirati varijable.

U JavaScript-u je potrebno deklarirati varijable, ali ne navodimo tip već samo ključnu riječ **let**.

```
let grad;  
let godina;  
let ime, prezime;
```

Varijabla koja je samo deklarirana ima vrijednost (i tip!) **undefined**.

Naredbom **typeof** možemo doznati koji je tip neke varijable.

```
grad = "Split";  
godina = 2000;  
console.log(typeof grad);  
console.log(typeof godina);  
console.log(typeof ime);
```

Ispis u konzoli:

```
string  
number  
undefined
```

Vrijednosti varijabli:

```
console.log(grad);  
console.log(godina);  
console.log(ime);
```

Ispis u konzoli:

```
Split  
2000  
undefined
```

Možemo primijetiti da je tip i vrijednost varijable *ime* **undefined**.

U C#-u varijabla ne može mijenjati tip nakon deklaracije. U JavaScript-u varijabla može promijeniti tip čim joj se pridruži podatak nekog drugog tipa. Ne možemo je deklarirati ponovo pomoću ključne riječi **let** unutar istog bloka kôda.

1.9. Objekt

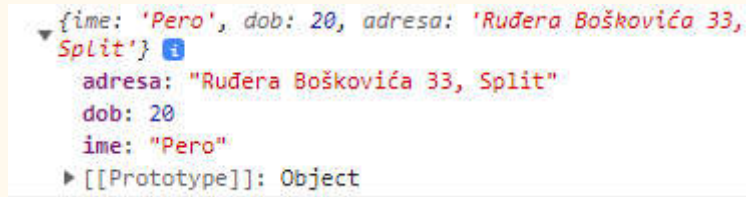
Objekt je složeni podatak koji sadrži više međusobno povezanih vrijednosti. Uzmimo na primjer *osobu* koja ima: *ime*, *dob*, *adresa*. Ako želimo spremiti podatke za jednu osobu možemo to za početak učiniti pomoću tri različite varijable:

```
let ime = "Pero";  
let dob = 20;  
let adresa = "Ruđera Boškovića 33, Split";
```

Međutim, osoba iz primjera je jedna povezana cjelina. Možemo je zapisati kao objekt u JavaScriptu:


```
let osoba = {
  ime: "Pero",
  dob: 20,
  adresa: "Ruđera Boškovića 33, Split"
};
console.log(osoba);
```

Ispis u konzoli:



```
{ime: 'Pero', dob: 20, adresa: 'Ruđera Boškovića 33, Split'}
  adresa: "Ruđera Boškovića 33, Split"
  dob: 20
  ime: "Pero"
  [[Prototype]]: Object
```

Dijelovima objekta pristupamo pomoću:

- Operatora točke - *dot notation* (primjer: **osoba.ime**)
- Zagrada - *bracket notation* (primjer: **osoba["ime"]**)

Objekt je u ovom slučaju zapisan slično kao *rječnik* (eng. dictionary) u C#, odnosno u obliku parova **ključ-vrijednost**.

1.10. Niz

Niz (eng. array) je složena struktura podataka koju ćemo često koristiti tako da će se ovdje dati samo kratki pregled osnovnih operacija koje ćemo koristiti. Može sadržavati više podataka kojima pristupamo indeksom. Prazan niz incijaliziramo:

```
let niz = [];
```

Ako znamo koje su vrijednosti već kod incijalizacije, onda ih možemo navesti unutar zagrada.

```
let drugiNiz = [2, 3, "Ana", 5.6];
```

Iz gornjeg primjera možemo vidjeti da je u JavaScript-u moguće miješati različite tipove podataka u istom nizu.

Rad s nizom:

- Veličina niza: **niz.length**
- Prvi element niza: **niz[0]**
- N-ti element niza: **niz[n]**
- Dodavanje elementa *novi* u niz na kraj: **niz.push(novi)**
- Uklanjanje zadnjeg elementa: **niz.pop()**, a prvog elementa **niz.shift()**
- Spajanje nizova: **niz3 = niz1.concat(niz2);**
- Traženje indeksa: **i = niz.indexOf(vrijednost)**

- Ako tražena vrijednost postoji u nizu, vraća poziciju
- Ako tražena vrijednost ne postoji u nizu vraća -1
- Rastavljanje stringa na niz: **niz = tekst.split(znak);**
- Spajanje niza u string: **tekst = niz.join(znak);**

Primjeri:

```
let brojevi = [2, 5, 3, 4, 7];
console.log(brojevi);
```

Ispis u konzoli:

```
► (5) [2, 5, 3, 4, 7]
```

Dodajmo par novih elemenata:

```
let novi = 10;
brojevi.push(novi);
brojevi.push(11);
console.log("Ispis nakon dodavanja: ");
console.log(brojevi);
```

Ispis u konzoli:

```
Ispis nakon dodavanja:
► (7) [2, 5, 3, 4, 7, 10, 11]
```

Probajmo kako radi **pop()**:

```
let izbacen = brojevi.pop(); //pop vraća element kojeg je izbacila
console.log("Izbacen: " + izbacen);
console.log(brojevi);
```

Ispis u konzoli:

```
Izbacen: 11
► (6) [2, 5, 3, 4, 7, 10]
```

Primijetimo da je izbačen zadnji element iz niza.

Za detaljnije informacije o nizovima možete pogledati dokumentaciju na stranicama:

- [W3schools JavaScript Arrays](#)
- [mdn web docs: Array](#)

1.11. Operatori

- Aritmetički: +, -, *, /, %
 - ++, --
 - +=, -=, *=, /=, %=
- Logički: &&, ||
- Relacijski: <, >, <=, >=, ==, !=, ===

1.12. Odluke i petlje

Odluke:

- if-else,
- switch-case

```
if (uvjet) {  
    // radi nešto  
}  
else {  
    // radi nešto drugo  
}
```

Petlje:

- for
- while

```
for (let index = 0; index < n; index++) {  
    // tijelo petlje  
}
```

```
while (uvjet) {  
    // radi nešto  
}
```

-