Programiranje u skriptnim jezicima (PJS)

Nositelj: doc. dr. sc. Nikola Tanković
Asistenti:

- Luka Blašković, mag. inf.
- Alesandro Žužić, mag. inf.

Ustanova: Sveučilište Jurja Dobrile u Puli, Fakultet informatike u Puli

[1] JavaScript osnove

JavaScript je visoko svestran programski jezik široko korišten u web razvoju. Prvobitno osmišljen za unapređenje interaktivnosti web stranica, danas se koristi i za razvoj serverskih aplikacija, desktop softvera, mobilnih aplikacija i drugih naprednih softverskih rješenja.

Posljednje ažurirano: 2.8.2024.

Sadržaj

1.1 Uvod

1. **Web stranica**: Zamislimo da je web stranica ljudsko tijelo.
    - **HTML** (Hypertext Markup Language) je kostur koji daje strukturu i podršku tijelu.
    - **CSS** (Cascading Style Sheets) je koža koja daje izgled tijelu.
    - **JavaScript** je skupina mišića i tetiva koja omogućuje kretanje tijela.
2. **Interaktivnost**: S JavaScriptom možemo izrađivati interaktivne komponente web stranice, poput:
    - formi koje reagiraju kada ih ispunjavamo,
    - izbornika koji se "spušta" kada kliknemo na njega ili
    - animacije koja se pokreće kad joj se približimo mišem.
3. **Running everywhere!**: Danas se JavaScript izvodi u raznim okruženjima, ne samo u web pregledniku! Može se izvoditi na:
    - serveru tj. poslužitelju
    - desktop aplikacijama
    - mobilnim uređajima
4. **Easy to learn, Hard to Master**: JavaScript je jedan od jednostavnijih jezika za naučiti. Ima jednostavnu sintaksu i rezultate izvođenja kôda možemo vidjeti gotovo odmah u web pregledniku.
5. **Bogat community**: JavaScript je jedan od najpopularnijih programskih jezika na svijetu. Ima veliku zajednicu developera, odlično je dokumentiran, ima puno biblioteka i razvojnih okruženja koji nam olakšavaju izradu web stranica/aplikacija.

1.2 Gdje pisati JavaScript kôd?

Pisanje JavaScripta na u web pregledniku (strana klijenta - eng. *client side*) možemo podijeliti na 3 načina:

1. **Inline JavaScript** - kôd se piše direktno unutar HTML elementa, npr. u atributu `onclick`:

```
<button onclick="console.log('Hello World!')">Hello World</button>
```

2. **Internal JavaScript** - kôd se piše unutar HTML dokumenta, ali u odvojenom `<script>` elementu:

```
<script>
  console.log("Hello World!");
</script>
```

3. **External JavaScript** - kôd se piše u odvojenom JavaScript dokumentu, npr. `script.js`:

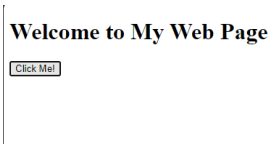
```
<!--index.html-->
<!DOCTYPE html>
<html>
  <head>
    <title>Moja web stranica</title>
    <script src="script.js"></script>
  </head>
  <body>
    <h1>Dobrodošli na Moju web stranicu</h1>

    <button onclick="showMessage()">Klikni me!</button>
  </body>
</html>
```

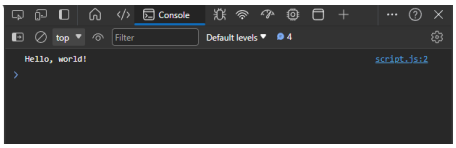
- Prednost ovog načina je što možemo koristiti isti kôd na više stranica, a i sam HTML dokument je čišći i pregledniji.
- Na isti način kao u kôdu iznad, `script.js` datoteku možemo uključiti i u druge HTML datoteke.

```
// script.js
function showMessage() {
  console.log("Hello World!");
}
```

1.3 Gdje je taj "Hello World"?

Kada otvorimo HTML dokument u web pregledniku, možemo otvoriti konzolu (F12) i vidjeti poruku "Hello World!", tako jednostavno!



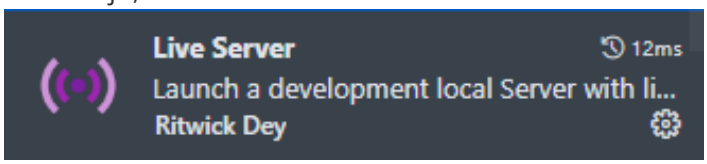


Idemo sada izmijeniti tekst koji nam ispisuje funkcija `showMessage()`. U `script.js` datoteci promijenimo tekst u `Hello JavaScript!`:

```
// script.js
function showMessage() {
  console.log("Hello JavaScript!");
}
```

Možemo primijetiti da se ponovnim klikom na gumb, tekst u konzoli nije promijenio. To je zato što je kôd iz `script.js` datoteke izvršen samo jednom, prilikom učitavanja stranice. Da bi promjena bila prikazana, moramo osvježiti stranicu (F5).

Naporno je svaki put osvježavati stranicu da bi vidjeli naše promjene. Iz tog razloga ćemo preuzeti Live Server ekstenziju za Visual Studio Code. Ona će nam omogućiti da otvorimo HTML dokument u web pregledniku i da se svaka promjena u kôdu automatski osvježi u web pregledniku. Nakon što instaliramo ekstenziju, kliknemo desnim klikom na HTML dokument i odaberemo `Open with Live Server`.



2. Izrazi, tvrdnje, varijable, tipovi podataka i operatori

Varijable su mjesta u memoriji u koje spremamo podatke. Svaka varijabla ima svoje ime i vrijednost. Vrijednost varijable može se mijenjati tijekom izvođenja programa.

Varijable možemo deklarirati na 3 načina: `var`, `let` i `const`. Varijable deklarirane s ključnim riječima `var` i `let` su varijable koje se mogu mijenjati, dok je `const` konstanta koja se ne može mijenjati.
U pravilu koristimo `const` za deklariranje varijabli, osim ako znamo da će se vrijednost varijable mijenjati, tada koristimo `let`. `var` izbjegavamo, budući da ga je `let` zamijenio u ES6 standardu JavaScripta. Koga zanima više zašto je uveden `let`, može pročitati ovdje.

```
let x = 5;
console.log(x); // 5

x = 10;
console.log(x); // 10

const y = 15;
console.log(y); // 15

y = 20; // TypeError: Assignment to constant variable.
console.log(y);
```

2.1 Tipovi podataka

JavaScript je slabo tipizirani jezik (eng. ***weakly typed***), što znači da razlikuje različite tipove varijable, no ne moramo ih strogo navoditi prilikom deklaracije varijable. Tip podatka varijable određuje se automatski prilikom dodjele vrijednosti varijabli.
Za provjeru tipa podatka varijable koristimo `typeof` operator.

```
let a = 5; // number
let b = "5"; // string
let c = true; // boolean

console.log(typeof a); // number
console.log(typeof b); // string
console.log(typeof c); // boolean
```

Varijable definirane s `const`:

- ne mogu se ponovno deklarirati (eng. ***redeclare***)
- ne mogu se ponovno dodijeliti (eng. ***reassign***)
- moraju se inicijalizirati prilikom deklaracije (eng. ***initialize***)
- imaju blokovski opseg (eng. ***block scope***)

Konstante se ne mogu ponovno deklarirati

```
const PI = 3.141592653589793;
PI = 3.14; // Baca grešku!
PI = PI + 10; // Baca grešku!
```

Konstante se moraju inicijalizirati prilikom deklaracije

```
const PI = 3.141592653589793; // Točno!

const PI; // Netočno!
```

2.2 Operatori

2.2.1 Izrazi (eng. *expressions*) vs tvrdnje (eng. *statements*)

U JavaScriptu, **izraz** (eng. ***expression***) je bilo koji valjani kôd koji se evaluira/razlaže (eng. ***resolve***) u vrijednost.
Primjer izraza može biti bilo koja matematička operacija, npr. za `x = 3`, `5 + 5`, ili `x = 7`, ili `x = x + 5`.
Navedeni izrazi se evaluiraju u vrijednosti: `3`, `10`, `10` i `12`.
Izrazi ne moraju biti samo brojevi! Evo još primjera izraza da bude jasnije:

- aritmetički izrazi: `5 + 3` ili `4 * 2`
- izrazi znakovnog niza: `"Hello " + "World"`
- logički izrazi: `true && false`
- funkcijski izrazi: `function() { console.log("Hello World!"); }`

Najjednostavnije rečeno, računalni program je popis "instrukcija" koje računalo treba "izvršiti". U programiranju, te "instrukcije" nazivaju se **tvrdnje** (eng. ***statements***). JavaScript program je popis tvrdnji koje se izvršavaju redom. Tvrdnje mogu biti: deklaracije varijabli, izrazi, kontrolne strukture, petlje, pozivi funkcija, ključne riječi, komentari itd.

2.2.2 Tablica osnovnih JavaScript operatora

Operatori su simboli koji se koriste za izvođenje operacija nad podacima, preciznije: sve kompleksnije izraze spajamo pomoću operatora, poput `=` i `+`. Postoji više vrsta operatora, mi ćemo se baviti samo nekima od njih:

- **Aritmetički operatori** - primarno se koriste za izvođenje aritmetičkih operacija nad brojevima
- **Operatori pridruživanja** - koriste se za pridruživanje vrijednosti varijablama
- **Operatori usporedbe** - koriste se za usporedbu vrijednosti
- **Logički operatori** - koriste se za izvođenje logičkih operacija
- **Operatori tipa (eng. *type*)** - koriste se za provjeru tipa podatka

Operator	Vrsta	Broj operanada	Opis	Primjer
Osnovni aritmetički `+`, `-`, `*`, `/`	Aritmetički	binarni (2)	Standardni aritmetički operatori.	`2 + 3` vraća `5`, `5 * 6` vraća `30`
Unarni `+`	Aritmetički	unarni (1)	Pokušava pretvoriti operand u broj, ako već nije.	`+"3"` vraća 3, `+true` vraća `1`
Unarni `-`	Aritmetički	unarni (1)	Vraća negaciju operanda.	ako je `x=3`, `-x` vraća `-3`
Inkrement `++`	Aritmetički	unarni (1)	Povećava svoj operand za 1, vraćajući novu vrijednost ako se koristi kao prefix (`++x`), ili izvornu vrijednost ako se koristi kao postfix (`x++`).	ako je `x = 3`, onda `++x` postavlja `x` na `4` i vraća `4`. Ali, `x++` vraća `3` i nakon toga postavlja `x` na `4`.
Dekrement `--`	Aritmetički	unarni (1)	Umanjuje svoj operand za 1, vraćajući novu vrijednost ako se koristi kao prefix (`--x`), ili izvornu vrijednost ako se koristi kao postfix (`x--`).	ako je `x = 3`, onda `--x` postavlja `x` na `2` i vraća `2`. Ali, `x--` vraća `3` i nakon toga postavlja `x` na `2`.
Ostatak `%`	Aritmetički	binarni (2)	Vraća cjelobrojni ostatak dijeljenja dva operanda.	ako je `x=3`, `-x` vraća `-3`
Eksponiranje `**`	Aritmetički	binarni (2)	Računa eksponent kao `baza^eksponent`.	`2 ** 3` vraća `8`, `10 ** -1` vraća `0.1`
Pridruživanje `=`	Pridruživanja	binarni (2)	Pridružuje vrijednost varijabli ili svojstvu.	`x = 2`, `y = f(x)`
Zbroji i pridruži `+=`	Pridruživanja	binarni (2)	Zbroji vrijednosti 2 operanda i rezultat pridruži lijevom operandu.	`a = 2`, `a=+3` vraća `5`
Oduzmi i pridruži `-=`	Pridruživanja	binarni (2)	Oduzmi vrijednosti 2 operanda i rezultat pridruži lijevom operandu.	`a = 2`, `a-=3` vraća `-1`
Pomnoži i pridruži `*=`	Pridruživanja	binarni (2)	Pomnoži vrijednosti 2 operanda i rezultat pridruži lijevom operandu.	`a = 2`, `a*=3` vraća `6`
Podijeli i pridruži `/=`	Pridruživanja	binarni (2)	Podijeli vrijednosti 2 operanda i rezultat pridruži lijevom operandu.	`a = 2`, `a/=2` vraća `1.5`
Ostatak i pridruži `%=`	Pridruživanja	binarni (2)	Izračunaj cjelobrojni ostatak vrijednosti 2 operanda i rezultat pridruži lijevom operandu.	`a = 3`, `a%=2` vraća `1`
Jednako `==`	Usporedni	binarni (2)	Vrati `true` ako su operandi jednaki.	`1 == 1` vraća `true`, `'hello' == 'hello'` vraća `true`, `5 == '5'` također vraća `true`
Nejednako `!=`	Usporedni	binarni (2)	Vrati `true` ako operandi nisu jednaki.	`1 != 1` vraća `false`, `'hello' != 'world'` vraća `true`
Identično `===`	Usporedni	binarni (2)	Vrati `true` ako operandi su operandi jednaki i istog tipa podatka.	`1 === 1` vraća `true`, `'hello' === 'hello'` vraća `true`, `'1' === 1` vraća

				`false`, `0 === false` vraća `false`
Identično nejednako `!==`	Usporedni	binarni (2)	Vrati `true` ako su operandi jednaki ali različitog tipa, ili ako su različiti i istog tipa podatka.	`1 !== 1` vraća `false`, `'hello' !== 'hello'` vraća `false`, `'1' !== 1` vraća `true`, `0 !== false` vraća `true`
Veće od `>`, manje od `<`	Usporedni	binarni (2)	(`>`) Vrati `true` ako je lijevi operand veći od desnog operanda. (`<`) Vrati `true` ako je lijevi operand manji od desnog operanda.	`5 > 2` vraća `true`, `'ab' > 'aa'` vraća `false`, `5 < 3` vraća `false`
Veće ili jednako od `>=`, manje ili jednako od `<=`	Usporedni	binarni (2)	(`>=`) Vrati `true` ako je lijevi operand veći ili jednak desnom operandu. (`<=`) Vrati `true` ako je lijevi operand manji ili jednak desnom operandu.	`5 >= 3` vraća `true`, `'ab' >= 'aa'` vraća `true`, `3 <= 3` vraća `true`
Logički AND `&&`	Logički	binarni (2)	Za skup boolean operanada rezultat će biti `true` samo i samo ako su oba operanda `true`. Ako generaliziramo, vraća vrijednost prvog `falsy` operanda kod evaluacije s lijeva na desno, ili vrijednost zadnjeg operanda ako su svi `true`.	za `a = 3` i `b = -2`, izraz `(a > 0 && b > 0)` vraća `false`, za izraz `5 && 6` vraća 6, ali `4 && false` vraća `false`
Logički OR `||`	Logički	binarni (2)	Za skup boolean operanada rezultat će biti `true` ako je jedan ili više operanada `true`. Ako generaliziramo, vraća vrijednost prvog `truthy` operanda kod evaluacije s lijeva na desno, ili vrijednost zadnjeg operanda ako su svi `false`.	za `a = 3` i `b = -2`, izraz `(a > 0 || b > 0)` vraća `true`, `true || 0` vraća `true`, ali `false || 0` vraća `0`
Logički NOT `!`	Logički	unarni (1)	Mijenja `true` izraz u `false` i obrnuto. Tipično se koristi sa boolean operandima, ali kada ne, vraća `false` kada se dodaje na tzv. `truthy` izraze, u suprotnom vraća `true`.	za `a = 3` i `b = -2`, izraz `(!(a > 0 || b > 0))` vraća `false`. `!""` vraća `true`, ali `!"Hello World"` vraća `false`
Operator tipa `typeof`	Type	unarni (1)	Vraća niz znakova koji označava vrstu operatora.	`typeof(2)` vraća `"number"`, `typeof("Banana")` vraća `"string"`, `typeof(someFunction)` vraća `"function"`

2.2.3 Dodatni primjeri korištenja operatora

2.2.3.1 Aritmetički i Pridruživanja

```
const a = 5; // Operator pridruživanja
const b = 10;
console.log(a + b); // 15

// Vrijede ista pravila o prioritetu izvođenja operacija kao i u matematici

console.log(a + b * 2); // 25
console.log((a + b) * 2); // 30

let a = 20;
let b = 2;
console.log(a / b); // 10 - količnik
console.log(a % b); // 0 - ostatak pri dijeljenju

let a = 5;
let b = 10;
```

```
console.log(a / b); // 0.5 - količnik
console.log(a % b); // 5 - ostatak pri dijeljenju

let c = 5;
c += 10; // Isto kao da smo napisali c = c + 10
console.log(c); // 15

let d = 5;
d -= 10; // Isto kao da smo napisali d = d - 10
console.log(d); // -5

let e = 5;
e *= 10; // Isto kao da smo napisali e = e * 10
console.log(e); // 50

let f = 5;
f /= 10; // Isto kao da smo napisali f = f / 10
console.log(f); // 0.5

let g = 5;
g %= 10; // Isto kao da smo napisali g = g % 10
console.log(g); // 5

let h = 10;
let h_kvadrirano = h ** 2;
console.log(h_kvadrirano); // 100

let brojac = 0;
brojac++; // Isto kao da smo napisali brojac = brojac + 1
console.log(brojac); // 1

brojac = 10;
brojac--; // Isto kao da smo napisali brojac = brojac - 1
console.log(brojac); // 9

let i = 5;
let j = i++; // j = 5, i = 6 - prvo se dodjeljuje vrijednost i, a zatim se povećava za 1

let k = 5;
let l = ++k; // l = 6, k = 6 - prvo se povećava za 1, a zatim se dodjeljuje vrijednost k

console.log(j, i, l, k); // 5 6 6 6
```

2.2.3.2 Usporedni i Logički

```
// Usporedni operatori
let a = 5;
let b = 10;

console.log(a == b); // false
console.log(a != b); // true
```

```
let c = 5;
let d = "5";
console.log(c == d); // true
console.log(c === d); // false - različiti tipovi podataka (number i string)

let e = 5;
let f = 10;
console.log(e > f); // false
console.log(e < f); // true
console.log(e >= f); // false
console.log(e <= f); // true

// Logički operatori
let g = true;
let h = false;
console.log(g && h); // false
console.log(g || h); // true
```

Što ako se ne koriste uz boolean operande?
JavaScript će pokušati pretvoriti operande u boolean vrijednosti (npr. `0` u `false`, `1` u `true`, prazan string u `false`, string sa sadržajem u `true` itd.
Googlaj: javascript type coercion)

```
// Logički AND
console.log(5 && 6); // 6 (Pogledati u tablici - '&&' evaluira s lijeva na desno i vraća zadnji koji je 'true')
console.log(0 && 7); // 0 (Pogledati u tablici - '&&' evaluira s lijeva na desno i vraća prvi koji je 'false')
console.log(false && 0); // false

// Logički OR

console.log(5 || 6); // 5 (Pogledati u tablici - '||' evaluira s lijeva na desno i vraća prvi koji je 'true')
console.log(0 || 7); // 7 (Pogledati u tablici - '||' evaluira s lijeva na desno i vraća prvi koji je 'true')
console.log(false || 0); // 0 (Pogledati u tablici - '||' evaluira s lijeva na desno i vraća zadnji koji je 'false')

// Logički NOT

console.log(!true); // false
console.log(!false); // true
console.log(!"Hello World"); // false
console.log(!""); // true
console.log(!0); // true
console.log(!5); // false
```

Naglasili smo da je izraz (eng. ***expression***) u JavaScriptu bilo koji valjani kod koji se evaluira u vrijednost.

Primjer 1:

- `5 + 5` je izraz koji se evaluira u `10`,
- kao i izraz `5 < 10` koji se evaluira u `true`,
- ili `9 < 9` koji se evaluira u `false`.

Logički operatori `&&`, `||` i `!` su također izrazi, koji se evaluiraju u `true` ili `false`, kako smo već prikazali u tablici operatora.

Primjer 2:

- Izraz `true && true` se evaluira u `true`,
- Izraz `true && false` se evaluira u `false`.
- Izraz `true || false` se evaluira u `true`,
- Izraz `false || false` se evaluira u `false`.

Jednako tako se izrazi iz primjera 1 mogu koristiti kao operandi u izrazima iz primjera 2. Vrlo je važno pritom pametno imenovati varijable, kako bi se izrazi mogli čitati kao rečenice.

Primjer 3: Želimo definirati logički izraz i nekoliko varijabli kako bi zaključili jesmo li pročitali broj stranica knjige koji smo si zadali kao cilj za ovaj tjedan.

```
let brojStranicaProcitano = 100;
let ciljaniBrojStranica = 200;

let ciljPostignut = brojStranicaProcitano >= ciljaniBrojStranica; // false
```

Primjer 4: Želimo definirati logički izraz i nekoliko varijabli kako bi zaključili jesmo li obavili sve zadatke prije nego što možemo krenuti na putovanje.

```
let kupljeneAvionskeKarte = true;
let rezerviraniSmjestaj = true;

let spremniZaPutovanje = kupljeneAvionskeKarte && rezerviraniSmjestaj; // true
```

Recimo da postoji opcija i da idemo s vlakom.

```
let kupljeneKarteZaVlak = true;
let kupljeneAvionskeKarte = false;
let rezerviraniSmjestaj = true;

let spremniZaPutovanje =
  (kupljeneAvionskeKarte || kupljeneKarteZaVlak) && rezerviraniSmjestaj; // true - jer je
bar jedan od uvjeta prijevoza ispunjen
```

Međutim i uvjet prijevoza možemo logično definirati kao varijablu!

```
let kupljeneKarteZaVlak = true;
let kupljeneAvionskeKarte = false;
let rezerviraniSmjestaj = true;
let uvjetPrijevoza = kupljeneAvionskeKarte || kupljeneKarteZaVlak; // true - jer je bar
jedan od uvjeta prijevoza ispunjen
let spremniZaPutovanje = uvjetPrijevoza && rezerviraniSmjestaj; // true - sada oba moraju
biti ispunjena!
```

Primjer 5. Želimo definirati nekoliko logičkih izraza i varijabli kako bi zaključili jesmo li zadovoljili sve uvjete za prolazak kolegija na fakultetu. Dani su sljedeći uvjeti:

- student mora imati više ili točno 50% bodova na završnom pismenom i više ili točno 50% bodova na završnom usmenom ispitu ili mora imati ukupno 50% bodova ostvarenih tijekom semestra
- student mora biti prisutan na više od 80% predavanja
- student mora predati projektni zadatak
- projektni zadatak mora biti ocijenjen s pozitivnom ocjenom

Kako možemo definirati prolaz preko ispita?

```
// Bodovi na pismenom i usmenom ispitu
let bodoviNaPismenom = 60;
let bodoviNaUsmenom = 40;

// Maksimalni broj bodova na pismenom i usmenom ispitu
let pismeniMaxBodova = 100;
let usmeniMaxBodova = 100;

// Prisustvo na predavanjima
let ukupniBrojPredavanja = 15;
let brojPrisustva = 14;

// Projektni zadatak
let predanProjektniZadatak = true;
let ocjenaProjektnogZadatka = 3;
```

Prvo ćemo definirati nekoliko logičkih i usporednih izraza, kako bi lakše ispisali konačan rezultat.

```
let prolazNaPismenom = bodoviNaPismenom / pismeniMaxBodova >= 0.5;
let prolazNaUsmenom = bodoviNaUsmenom / usmeniMaxBodova >= 0.5;

let prisutnostZadovoljavajuca = brojPrisustva / ukupniBrojPredavanja > 0.8;

let projektRijesen = predanProjektniZadatak && ocjenaProjektnogZadatka > 1;

let prolaz =
  prolazNaPismenom &&
  prolazNaUsmenom &&
  prisutnostZadovoljavajuca &&
  projektRijesen; // false
```

Dodat ćemo i alternativu polaganja putem kontinuiranog praćenja.

```
let kolokvij1 = 40;
let kolokvij2 = 60;
let kolokvijiMaxBodova = 200;
let prolazNaKolokvijima = (kolokvij1 + kolokvij2) / kolokvijiMaxBodova >= 0.5;

let prolaz =
  ((prolazNaPismenom && prolazNaUsmenom) || prolazNaKolokvijima) &&
  prisutnostZadovoljavajuca &&
  projektRijesen; // true
```

2.2.4 Typeof operator

Primitivni tipovi podataka u JavaScriptu predstavljaju vrijednosti koje se spremaju u memoriju bez dodatnih metoda i svojstava. Primitivni tipovi su:

- `string`
- `number`
- `boolean`
- `undefined`

`typeof` operator može vratiti jedan od tih primitivnih tipova.

```
// typeof
console.log(typeof 5); // number
console.log(typeof "5"); // string
console.log(typeof true); // boolean
console.log(typeof undefined); // undefined
console.log(typeof null); // object
```

Zašto je `typeof null` = objekt? U JavaScriptu, `null` doslovno predstavlja "ništa". Nažalost, `typeof` funkcija će vratiti da je tip podatka `null` objekt. Radi se o bugu koji je prisutan od samih početaka ovog jezika.

Kojeg će tipa biti sljedeća varijabla?

```
const secret_number;
```

▶ Spoiler Warning!

Odgovor je `undefined`. `undefined` je tip podatka koji se koristi kada varijabla nije inicijalizirana, dok je `null` je tip podatka koji se koristi kada varijabla nema vrijednost.

Vježba 1

Idemo napraviti kratku vježbu onoga što smo dosad prošli. U `script.js` datoteci deklarirajte varijable `a`, `b` i `c` i dodijelite im vrijednosti `5`, `"5"` i `true`. Ispišite vrijednosti varijabli u konzolu i provjerite njihove tipove. Kôd dodajte unutar funkcije `showMessage()`.
Nakon toga, `typeof` operatorom provjerite tipove varijabli i u konzolu ispišite tvrdnju za svaku varijablu, npr. "Varijabla a je tipa number". Izraze u `console.log()` možete spojiti pomoću `+` operatora.

Zašto `console.log(a == b)` vraća `true`? Objasnite.

Rezultat:

```
5                                          script.js:6
5                                          script.js:7
true                                       script.js:8
Varijabla a je tipa number                 script.js:10
Varijabla b je tipa string                 script.js:11
Varijabla c je tipa boolean                script.js:12
```

Vježba 2

Idemo sada napraviti jednostavan kalkulator. U `script.js` datoteci deklarirajte varijable `a` i `b` i dodijelite im vrijednosti `5` i `10`. Izračunajte zbroj, razliku, umnožak i količnik varijabli `a` i `b` i ispišite ih u konzolu. Dodatno, ispišite u konzolu ostatak pri dijeljenju varijabli `a` i `b` i rezultat eksponiranja varijable `a` na potenciju varijable `b`.

Rezultat:

```
Zbroj a i b je: 15                                          script.js:5
Razlika a i b je: -5                                        script.js:6
Umnožak a i b je: 50                                        script.js:7
Količnik a i b je: 0.5                                      script.js:8
Ostatak pri dijeljenju varijable a sa b je: 5               script.js:9
Rezultat eksponiranja varijable a sa b je: 9765625          script.js:10
```

2.3 Koncept varijable u JavaScriptu

Varijable u JavaScriptu mogu sadržavati bilo koju vrijednost, neovisno o tipu podatka. To znači da varijabla može sadržavati broj, string, boolean, objekt, funkciju, itd.
Ista varijabla može sadržavati i više različitih tipova podataka!
Važno je razumjeti što se dešava "ispod haube" kada deklariramo varijablu i dodijelimo joj vrijednost.
Bez tipova podataka, računalo neće znati interpretirati (na siguran način) sljedeće:

```
let x = 16 + "Volvo";
```

Ima li smisla? Hoće li ovo biti broj ili string? Ili ćemo dobiti grešku?
Kada JavaScript vidi da se koristi operator `+` na broju i stringu, on će automatski pretvoriti broj u string i spojiti ih. Ovo se zove **implicitna konverzija**.

```
let x = "16" + "Volvo";
```

Uzmimo za primjer sljedeći izraz?

```
let x = 16 + 4 + "Volvo";
```

Koji će biti rezultat? `"164Volvo"` ili `"20Volvo"`?

A ovdje?

```
let x = "Volvo" + 16 + 4;
```

▶ Spoiler Warning!

```
let x = 16 + 4 + "Volvo";
console.log(x); // 20Volvo

let x = "Volvo" + 16 + 4;
console.log(x); // Volvo164
```

Imajte na umu da prioritet i asocijativnost operatora utječu samo na redoslijed evaluacije **operatora**, ali ne i na redoslijed evaluacije **operanada**. **Operandi se uvijek evaluiraju s lijeva na desno!**, međutim njihovi rezultati se sastavljaju prema redoslijedu prioritera operatora!

JavaScript tipovi su dinamički, što znači da se tip podatka varijable može promijeniti tijekom izvođenja programa.

```
let x;
console.log(typeof x); // undefined
x = 5;
console.log(typeof x); // number
x = "Petar";
console.log(typeof x); // string
```

2.3.1 JavaScript Strings

String je tekstualni podatak, radi se o nizu znakova. String možemo definirati s jednostrukim ili dvostrukim navodnicima.

```
let x = "Petar";
let y = "Petar";
```

Možemo koristiti i navodne znakove unutar stringa, ali moramo paziti da se ne podudaraju s vanjskim navodnicima.

```
let x = "Petar je rekao: 'Dobar dan!'";
```

Možemo koristiti i varijable unutar stringa, ali onda moramo koristiti *backtickse* `` ` `` te `${}` za prikaz same varijable. Ovakva sintaksa se zove **template literals**.

```
let ime = "Petar";
let predstavljanje = `Moje ime je ${ime}`;
console.log(predstavljanje); // Moje ime je Petar
```

Istu stvar možemo dobiti i sa `+` operatorom, ali `template literals` sintaksa je jednostavnija i puno čitljivija!

```
let ime = "Petar";
let predstavljanje1 = "Moje ime je " + ime;
let predstavljanje2 = `Moje ime je ${ime}`;

console.log(predstavljanje1 == predstavljanje2); // true
```

Još jedan primjer s brojevima!

```
const a = 5;
const b = 10;
console.log(`Petnaest je ${a + b} a ne ${2 * a + b}.`);
// Petnaest je 15 a ne 20.
```

2.4 Eksponencijalna (znanstvena) notacija

Eksponencijalna notacija se koristi za prikazivanje jako velikih ili jako malih brojeva. Zapisujemo ju koristeći `e` ili `E`.

```
let y = 123e5; // 12300000
let z = 123e-5; // 0.00123
```

Primjerice, broj 100 možemo zapisati kao:

```
100 = 10e1 //čitaj 10 puta 10 na prvu
```

Broj 1 možemo zapisati kao:

```
1 = 10e-1 //čitaj 10 puta 10 na minus prvu
```

Decimalni broj 200.5 možemo zapisati kao:

```
200.5 = 2.005e2 //čitaj 2.005 puta 10 na drugu
```

2.5 BigInt [DODATNO]

Random Fact, ali nije loše za zapamtiti:
Većina programskih jezika ima različite tipove podataka za:

1. Cijele brojeve

- byte (8-bit)
- short (16-bit)
- int (32-bit)
- long (64-bit)

2. Brojeve s decimalnim zarezom

- float (32-bit)
- double (64-bit)

Svi Javascript brojevi su uvijek istog tipa! A to je `double` (64-bit floating point).

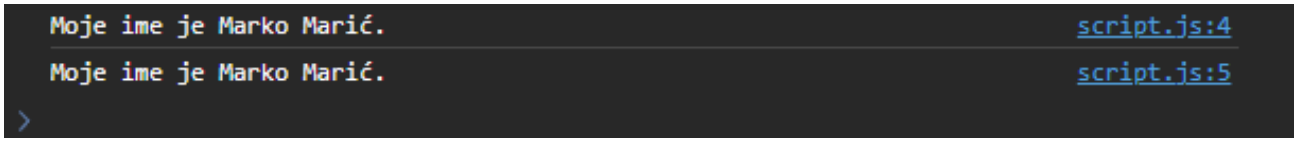
JavaScript, sa ES2020 standardom, dobiva novi tip podatka `BigInt` koji može prikazati brojeve veće od `Number.MAX_SAFE_INTEGER`, odnosno (2^{53} - 1).

```
const x = Number.MAX_SAFE_INTEGER + 1;
const y = Number.MAX_SAFE_INTEGER + 2;

console.log(Number.MAX_SAFE_INTEGER);
// Očekivani output: 9007199254740991

console.log(x);
// Očekivani output: 9007199254740992

console.log(x === y);
// Očekivani output: false ?

// Međutim, rezultat je true. x i y su isti brojevi jer ne možemo premašiti
MAX_SAFE_INTEGER

// Koristimo BigInt
const z = BigInt(Number.MAX_SAFE_INTEGER) + BigInt(2);
console.log(z == y); // false
```

Vježba 3

Deklarirajte dvije varijable `ime` i `prezime` i dodijelite im vrijednosti `Marko` i `Marić`. Ispišite dvaput u konzolu rečenicu `Moje ime je Marko Marić.`, jednom koristeći `+` operator, a drugi put koristeći `template literals`.

Rezultat:



Vježba 4

Želite si definirati nekoliko ciljeva za ovaj tjedan kako biste ispunili vaš `weekly_goal`. Vaši ciljevi definirani su sljedećim tvrdnjama, odnosno vaš `weekly_goal` je ispunjen ako:

- želim proučiti PJS1 skriptu iz JavaScripta
- želim pročitati barem 50 stranica omiljene knjige
- želim vježbati JavaScript barem 2 sata ili riješiti barem 10 zadataka
- želim se svaki dan naspavati

Za svaku izjavu definirajte po nekoliko pomoćnih varijabli, npr. jednu za ciljanu vrijednost, jednu za ostvarenu vrijednost i jednu za rezultat ostvarenja (boolean). Na primjer, za izjavu `želim pročitati barem 50 stranica omiljene knjige` deklarirajte varijable `broj_procitanih_stranica` i `ciljani_broj_stranica` te varijablu `cilj_citanje`.

Rezultat:
Napišite u obliku: `weekly_goal = cilj1 && cilj2 && cilj3 && cilj4`

Samostalni zadatak za vježbu 1

Napomena: Ne predaje se i ne boduje se. Zadatak rješavate u EduCoder aplikaciji.

EduCoder šifra: `a_new_hope`

1. Deklarirajte tri konstante i jednu promjenjivu varijablu. Konstante neka budu vaše `ime` i `prezime` i `godina_rodenja`. Promjenjivu varijablu nazovite `trenutno_vrijeme`.
    - U varijable `ime` i `prezime` pohranite svoje ime i prezime, a u varijablu `godina_rodenja` pohranite godinu rođenja kao cjelobrojnu vrijednost. U varijablu `trenutno_vrijeme` pohranite trenutno vrijeme koristeći `new Date()` objekt.
    - Dodajte novu varijablu `godine` i u nju izračunajte koliko imate godina koristeći: funkciju `getFullYear()` nad varijablom `trenutno_vrijeme` i varijablu `godina_rodenja`. Sintaksa je: `varijabla.getFullYear()`. Radi pojednostavljivanja, pretpostavljamo da je vaš rođendan već prošao ove godine.
2. Koristeći `template literals`, u konzolu ispišite "Bok moje ime je __ ** i imam ** godina.".
    - Deklarirajte dvije nove konstante `ime_duljina` i `prezime_duljina` u koje ćete pohraniti broj slova u vašem imenu i prezimenu koristeći funkciju `length` nad varijablama `ime` i `prezime`.
    - Ispišite u konzolu "Moje ime i prezime imaju ** i ** slova." koristeći `template literals`.

- Ispišite u konzolu "It is __ that my name and surname are of the same length" koristeći `template literals` i operator `"je identično"`.

3. Pohranite u novu varijablu `x` kvadrat zbroja varijabli `ime_duljina` i `prezime_duljina`. Rezultat zbrojite s vašom godinom rođenja uvećanom za 1 koristeći operator `++` ispred varijable (uočite grešku, zašto nastaje, i napravite izmjenu!) te sve skupa podijelite s `2`. Sve navedeno definirajte u obliku jednog izraza u jednoj liniji kôda.
4. Recimo da si želite definirati daily routine koji se sastoji od nekoliko ciljeva. Koristeći logičke operatore i operatore usporedbe, definirajte varijablu `daily_routine_ostvaren`, temeljem sljedećih tvrdnji. Vaš `daily_routine_ostvaren` je ispunjen ako:
   - ste pročitali više od 50 stranica vaše omiljene knjige **ili** ste vježbali JavaScript barem 1 sat
   - ste popili između litre i dvije litre vode
   - ste vježbali minimalno 30 minuta **ili** ste prošetali minimalno 3 km
   - ste naučili nešto novo
   - ste se naspavali minimalno 7 sati
   - ste se nasmijali

   Za svaki od danih izraza deklarirajte varijable za ostvarenu vrijednost i ciljanu vrijednost, te boolean varijablu koja će sadržavati rezultat ostvarenja. Na primjer, za izraz `popiti između litre i dvije litre vode` deklarirajte varijable `unos_vode` i `ciljani_dnevni_unos_vode` te varijablu `dnevni_unos_vode_zadovoljen`.
   - Definirajte varijablu `daily_routine_ostvaren` koja će sadržavati rezultat ostvarenja svih dnevnih ciljeva.