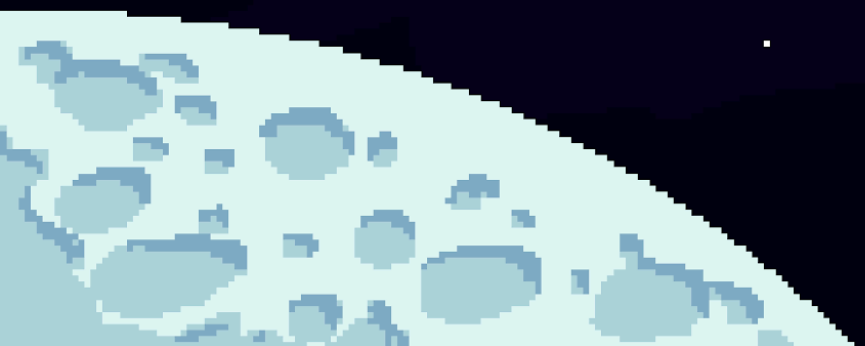
Uvod u JavaScript

O JavaScriptu

- Nastao 1995. od strane NetScapea
- Jedan od najpopularnijih programskih jezika
- Može se koristiti za:
  - frontend (Javascript + React, Vue, Angular)
  - backend (Node.js)
  - desktop aplikacije (Electron)
  - mobilne aplikacije (React Native)

Osnovna sintaksa

- Skriptni jezik, interpreter evaluira liniju po liniju
- Nema potrebe za main funkcijom
- Osnovna sintaksa bliska C-u ili Javi (zagrade, ; itd.)
- Znak `;` nije obavezan, ali se preporučuje



JavaScript ≠ Java

Primer osnovnog Hello World programa:

Java

```java
class Program {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

JavaScript

```javascript
console.log("Hello World");
```

Varijable

- U JS-u varijable se definišu pomoću jednog od 3 keywordā:
  - `let`
  - `const`
  - `var`
- `const` koristimo za konstante
- `var` se u poslednje vreme ne koristi, nasledio ga je `let`

Tipovi varijabla

- JS nije statično tipiziran
- Runtime ne brani promenu tipa tokom izvršavanja **(ali je to loša ideja)**
- `typeof` operator vraća string sa tipom varijable

Vrednosni tipovi:

- Number
- BigInt
- String
- Boolean
- Object
- Function

- **Tipovi koji ne sadrže vrednost:**
  - undefined
  - null
  - NaN

Ponekad tipovi maaaalo nemaju smisla:

- `typeof typeof 123 == "string"`
- `typeof null == "object"`
- `typeof NaN == "number"`
- `typeof class Foo {} == "function"`
- `typeof [1,2,3] == "object"`

Funkcije

Funkcije se mogu napisati na više načina:

Deklaracija funkcije

```
function greet(name) {
    return `Hello, ${name}!`;
}
```

Izraz funkcije

```
const greet = function(name) {
    return `Hello, ${name}!`;
};
```

Arrow funkcija

```
const greet = (name) => {
    return `Hello, ${name}!`;
};
```

Objekti

- Koriste se za grupisanje više različitih varijabli u jednu

```
let osoba = {
    ime: 'Pera Perić',
    godine: 25,
    grad: 'Beograd'
};
```

- Pristupamo pomoću:

Notacije tačkom

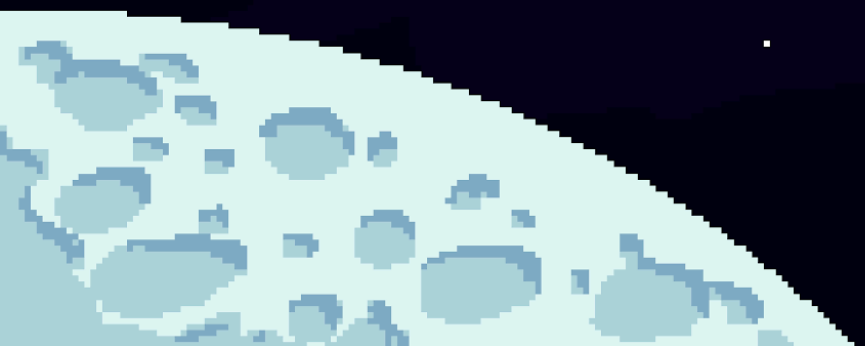
`osoba.ime`

Notacije zagradom

`osoba["ime"]`

Klase

- "Kalup" za pravljenje objekata
- Konstruktor → funkcija koja inicijalizuje objekat
- Od ECMAScript 2022 postoje privatna polja
  - Privatna polja počinju sa `#`
  - Za pristup možemo koristiti `get` i `set` operatore
- Klase se instanciraju sa `new` operatorom

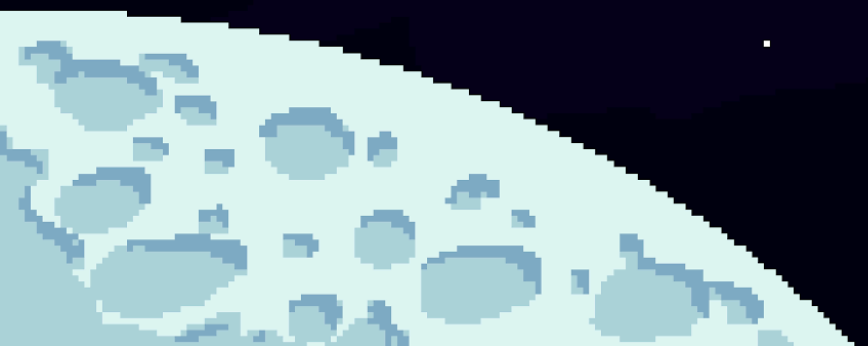


Nizovi

- Koriste se kao skup istih varijabli
- Ne funkcionišu ispod haube kao u drugim jezicima
  - Posebna, indeksirana vrsta objekta
  - Više su nalik listama

Operatori za poređenje

- Postoje **2** načina za poređenje varijabli:
  - `==` i `!=` ➜ porede samo vrednosti, ali ne i tipove
  - `===` i `!==` ➜ porede i vrednosti i tipove **(preporučljivo)**



Aritmetički operatori

- `+`, `-`, `*`, `/`, `%`, `**`
- Mogu se i assignovati: `+=`; `-=`, `*=`, `/=`, `%=`, `**=`
- Također, postoji i increment i decrement: `i++`, `i--`
- `+` i `+=` se mogu koristiti i za spajanje stringova
- Ostali operatori automatski castuju operande u Number
  - Koji je output `2 + '2' - '2'`?

Logički operatori

- `&&` (AND) → vraća `true` ako su obe vrednosti `true`
- `||` (OR) → vraća `true` ako je barem jedna vrednost `true`
- `!` (NOT) → negira vrednost, tj vraća kontra vrednost
- JS radi lazy evaluaciju
- `??` (Nullish) → vraća drugu vrednost ako je prva `null` ili `undefined`

Kondicionali

- `if...else` statement
- `switch` statement

if...else

```
if(uslov1) {
  // prva akcija
}
else if (uslov2) {
  // druga akcija
}
else {
  // treca akcija
}
```

Switch

- Ukoliko imamo više grananja za različite vrednosti varijable bolje je korisiti `switch`

```
switch(varijabla) {
  case 1:
    // prva akcija
    break;

  case 2:
    // druga akcija
    break;

  default:
    // treca akcija
}
```

Petlje

- `while` i `do..while`
- `for`, `for..in` i `for..of`

while i do..while

while

```
while(uslov) {
  // akcija
}
```

do..while

```
do {
    // akcija
} while(uslov)
```

- do..while će se garantovano barem jednom izvršiti

For petlje

`for`

```
for(init; uslov; korak) {
  // akcija
}
```

`for..in`

```
for(kljuc in iterable) {
  // akcija
}
```

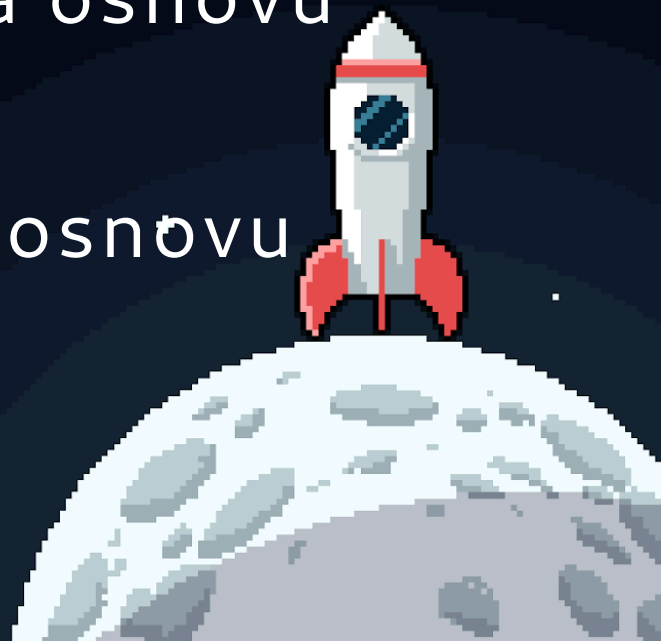
`for..of`

```
for(vrednost of iterable) {
  // akcija
}
```

- `for..in` i `for..of` služe za prolazak kroz nizove i objekte
- `for..in` vraća index ili ključ elementa
- `for..of` vraća kopiju vrednosti elementa
  - promene vrednosti elementa **NE** utiču na objekat

map, filter, reduce

- Metodi nizova
- Prolaze kroz niz, izvršavaju arrow funkciju i vraćaju neku vrednost
- `map` → vraća modifikovanu **kopiju** niza na osnovu funkcije
- `filter` → vraća filtriranu **kopiju** niza na osnovu uslova
- `reduce` → vraća akumuliranu vrednost



Pitanja?

Hvala na pažnji!