Prototip Nasljeđivanje

Antonija Barić

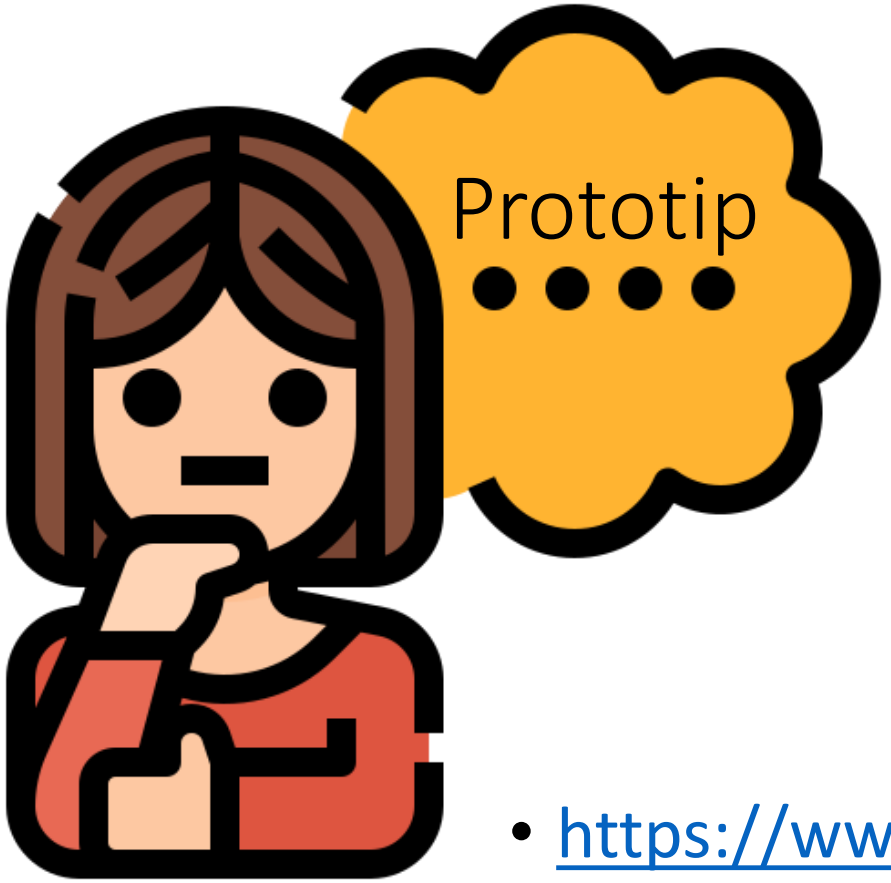
26.4.2023.

Prototype

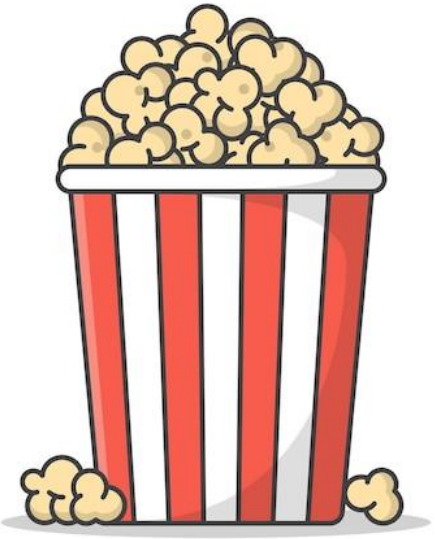
- prototype je objekt koji se koristi kao predložak za stvaranje drugih objekata
- svaki objekt u JavaScriptu ima prototip, što znači da objekt nasljeđuje svojstva i metode prototipa
- kada se traži svojstvo ili metoda na objektu, JavaScript prvo provjerava ima li ga sam objekt. Ako objekt ne posjeduje traženo svojstvo ili metodu, JavaScript će tražiti u njegovom prototipu i u svim drugim prototipovima

Zašto ga koristimo?

1. **Nasljeđivanje**
   - prototipni model omogućuje nasljeđivanje svojstava i metoda iz drugih objekata. To omogućuje programerima da stvore nove objekte koji nasljeđuju svojstva i metode postojećih objekata, čime se smanjuje količina ponovno napisanog koda.
2. **Efikasnost**
   - prototipni model je vrlo učinkovit jer smanjuje količinu memorije potrebne za stvaranje novih objekata. Svojstva i metode se ne kopiraju za svaki novi objekt, već se dijele između svih objekata koji nasljeđuju prototip.
3. **Dinamičnost**
   - prototipni model omogućuje programerima da dinamički mijenjaju svojstva i metode objekta u bilo kojem trenutku. Ovo je korisno kada programer treba proširiti funkcionalnost postojećeg objekta bez izmjene samog objekta.
4. **Jednostavnost**
   - prototipni model je jednostavan za korištenje i razumijevanje, čak i za početnike u JavaScriptu. Koncept prototipova nije složen, što olakšava programerima da brzo započnu s pisanjem funkcija i objekata.



- https://www.youtube.com/wa
?v=1UTqFAjYx1k



Prototype u konstruktor funkciji i klasi

- *Kada se poziva osoba1.pozdrav(), JavaScript traži pozdrav() u osoba1 objektu, koji ga ne posjeduje.*
- *Zatim JavaScript traži pozdrav() u prototipu Osoba.prototype, gdje je metoda definirana.*
- *Kada se pronađe pozdrav(), poziva se s osoba1 kao kontekstom.*
- ***prototype.js***

Zdk.

- Napišite funkciju konstruktora Automobil koja ima svojstvo brzina postavljeno na 0. Dodajte metodu prototipa ***ubrzaj()*** koja će povećati brzinu automobila za 10 km/h. Također, dodajte metodu prototipa ***uspori()*** koja će smanjiti brzinu automobila za 10 km/h.
- Napravite novu instancu objekta auto i pozovite metode prototipa ***ubrzaj() i uspori()*** nekoliko puta.
- Ispišite trenutnu brzinu automobila nakon svakog poziva metode.

Razlika prototipa u konstr. funkciji i klasama

- U konstruktor funkciji, prototipovi se definiraju korištenjem **function** ključne riječi i dodavanjem svojstava i metoda na **prototype** objekt konstruktora.
- Također, kada se stvara nova instanca objekta, koristi se ključna riječ **new**.

```
function Osoba(ime, prezime) {
  this.ime = ime;
  this.prezime = prezime;
}

Osoba.prototype.pozdrav = function() {
  console.log("Pozdrav, ja sam " + this.ime + " " +
}

const osoba1 = new Osoba("John", "Doe");
osoba1.pozdrav(); // Output: Pozdrav, ja sam John Do
```

Razlika prototipa u konstr. funkciji i klasama

- U klasi, prototipovi se definiraju korištenjem **class** ključne riječi i dodavanjem metoda i svojstava u bloku deklaracije klase.
- Kod kreiranja nove instance, koristi se ključna riječ **new.**
- https://medium.com/@parsyval/javascript-prototype-vs-class-a7015d5473b
- https://www.freecodecamp.org/news/a-beginners-guide-to-javascripts-prototype/

```
class Osoba {
  constructor(ime, prezime) {
    this.ime = ime;
    this.prezime = prezime;
  }

  pozdrav() {
    console.log(`Pozdrav od ${this.ime} ${this.prezime}`);
  }
}

// Definiramo podklasu (dijete) koja nasljeđuje osobine od klase Osoba
class Student extends Osoba {
  constructor(ime, prezime, fakultet) {
    // Pozivamo konstruktor nadklase da bi inicijalizirali svoje osobine
    super(ime, prezime);
    this.fakultet = fakultet;
  }
```

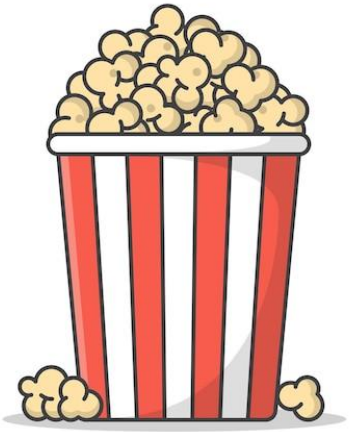
Razlika prototipa u konstr. funkciji i klasama

- U suštini, konstruktor funkcija i klase se koriste za istu svrhu - definiranje prototipova objekata.
- Klase su međutim noviji način definiranja objekata u JavaScriptu i pružaju sintaksu koja je sličnija drugim objektno-orijentiranim programskim jezicima

Nasljeđivanje

- Nasljeđivanje se postiže korištenjem prototipa objekata. Svaki objekt u JavaScriptu ima prototip, koji se može naslijediti od drugog objekta.
- Nasljeđivanje se postiže tako da se jedan objekt postavi kao prototip drugog objekta. Na taj način, objekt koji nasljeđuje dobiva sve metode i svojstva definirana u prototipu.

Nasljeđivanje



- https://www.youtube.com/watch?v=1UTqFAjYx1k

Primjer

- *prototype_inheritance.js*

Call()

- Funkcija call() u prethodnom primjeru se koristi za pozivanje konstruktora Osoba na objektu Student.
- Konstruktor Student prima tri argumenta: ime, prezime i fakultet. Međutim, konstruktor Osoba prima samo dva argumenta: ime i prezime.
- Pozivom Osoba.call(this, ime, prezime) u konstruktoru Student, pozivamo konstruktor Osoba i prosljeđujemo mu dva argumenta, ime i prezime.
- Ako ne bismo koristili call, već bismo pokušali proslijediti tri argumenta konstruktoru Osoba, konstruktor bi primio samo prva dva argumenta i ignorirao treći.
- Korištenjem call omogućujemo da se konstruktor Osoba pozove i da se postave svojstva ime i prezime na objektu Student, što je potrebno da bismo kasnije mogli koristiti metodu pozdrav koja je definirana u prototipu Osoba.

Object.create()

- ***Student.prototype = Object.create(Osoba.prototype);***

- Object.create() je metoda koja kreira novi objekt i postavlja ga kao nasljednika (prototip) drugog objekta.
- Object.create(Osoba.prototype) kreira novi objekt koji nasljeđuje prototip Osoba.prototype. Taj novi objekt postavlja se kao prototip objekta Student.prototype.