

# JS - prototype vs \_\_proto\_\_

Monday, May 1, 2023 4:56 PM

## 1. \_\_proto\_\_ svojstvo

Svaki novi objekt ima \_\_proto\_\_ svojstvo koje pokazuje na Object objekt na osnovu kojeg je kreiran kreiran novi objekt

```
> const noviObjekt = {};  
< undefined  
-----  
> noviObjekt  
< {}  
  ▾ {}  
    ▾ [[Prototype]]: Object  
      ▶ constructor: f Object()  
      ▶ hasOwnProperty: f hasOwnProperty()  
      ▶ isPrototypeOf: f isPrototypeOf()  
      ▶ propertyIsEnumerable: f propertyIsEnumerable()  
      ▶ toLocaleString: f toLocaleString()  
      ▶ toString: f toString()  
      ▶ valueOf: f valueOf()  
      ▶ __defineGetter__: f __defineGetter__()  
      ▶ __defineSetter__: f __defineSetter__()  
      ▶ __lookupGetter__: f __lookupGetter__()  
      ▶ __lookupSetter__: f __lookupSetter__()  
      ▶ __proto__: Object  
        ▶ constructor: f Object()  
        ▶ hasOwnProperty: f hasOwnProperty()  
        ▶ isPrototypeOf: f isPrototypeOf()  
        ▶ propertyIsEnumerable: f propertyIsEnumerable()  
        ▶ toLocaleString: f toLocaleString()  
        ▶ toString: f toString()  
        ▶ valueOf: f valueOf()  
        ▶ __defineGetter__: f __defineGetter__()  
        ▶ __defineSetter__: f __defineSetter__()  
        ▶ __lookupGetter__: f __lookupGetter__()  
        ▶ __lookupSetter__: f __lookupSetter__()  
        ▶ __proto__: null  
        ▶ get __proto__: f __proto__()  
        ▶ set __proto__: f __proto__()  
        ▶ get __proto__: f __proto__()  
        ▶ set __proto__: f __proto__()  
    >
```

## 2. Vrsta i sadržaj (svojstva i metode ) prototipa ovise o vrsti objekta kojeg kreiramo

Primjer za Array

```

> const boje = ['crveno', 'zeleno'];
< undefined
> boje
< ▼ (2) ['crveno', 'zeleno'] 0
  0: "crveno"
  1: "zeleno"
  length: 2
  ▼ [[Prototype]]: Array(0)
    ▶ at: f at()
    ▶ concat: f concat()
    ▶ constructor: f Array()
    ▶ copyWithin: f copyWithin()
    ▶ entries: f entries()
    ▶ every: f every()
    ▶ fill: f fill()
    ▶ filter: f filter()
    ▶ find: f find()
    ▶ findIndex: f findIndex()
    ▶ findLast: f findLast()
    ▶ findLastIndex: f findLastIndex()
    ▶ flat: f flat()
    ▶ flatMap: f flatMap()
    ▶ forEach: f forEach()
    ▶ includes: f includes()
    ▶ indexOf: f indexOf()
    ▶ join: f join()
    ▶ keys: f keys()
    ▶ lastIndexOf: f lastIndexOf()
    ▶ length: 0
    ▶ map: f map()
    ▶ pop: f pop()
    ▶ push: f push()
    ▶ reduce: f reduce()
    ▶ reduceRight: f reduceRight()
    ▶ reverse: f reverse()
    ▶ shift: f shift()
    ▶ slice: f slice()
    ▶ some: f some()
    ▶ sort: f sort()
    ▶ splice: f splice()
    ▶ toLocaleString: f toLocaleString()
    ▶ toReversed: f toReversed()
    ▶ toSorted: f toSorted()
    ▶ toSpliced: f toSpliced()
    ▶ toString: f toString()
    ▶ unshift: f unshift()
    ▶ values: f values()
    ▶ with: f with()
    ▶ Symbol(Symbol.iterator): f values()
    ▶ Symbol(Symbol.unscopables): {at: true, copyWithin: true, entries: true, fill: true, find: true, ...}
    ▶ [[Prototype]]: Object
  >

```

Primjer za string

```

> const ime = 'Ivan';
< undefined
> ime.__proto__
< String {} constructor: f, anchor: f, at: f, big: f, ...}
  ▶ anchor: f anchor()
  ▶ at: f at()
  ▶ big: f big()
  ▶ blink: f blink()
  ▶ bold: f bold()
  ▶ charAt: f charAt()
  ▶ charCodeAt: f charCodeAt()
  ▶ codePointAt: f codePointAt()
  ▶ concat: f concat()
  ▶ constructor: f String()
  ▶ endsWith: f endsWith()
  ▶ fixed: f fixed()
  ▶ fontcolor: f fontcolor()
  ▶ fontsize: f fontsize()
  ▶ includes: f includes()
  ▶ indexOf: f indexOf()
  ▶ isWellFormed: f isWellFormed()
  ▶ italics: f italics()
  ▶ lastIndexOf: f lastIndexOf()
  ▶ length: 0
  ▶ link: f link()
  ▶ localeCompare: f localeCompare()
  ▶ match: f match()
  ▶ matchAll: f matchAll()
  ▶ normalize: f normalize()
  ▶ padEnd: f padEnd()
  ▶ padStart: f padStart()
  ▶ repeat: f repeat()
  ▶ replace: f replace()
  ▶ replaceAll: f replaceAll()
  ▶ search: f search()
  ▶ slice: f slice()
  ▶ small: f small()
  ▶ split: f split()
  ▶ startswith: f startswith()
  ▶ strike: f strike()
  ▶ sub: f sub()
  ▶ substr: f substr()
  ▶ substring: f substring()
  ▶ sup: f sup()
  ▶ toLocaleLowerCase: f toLocaleLowerCase()
  ▶ toLocaleUpperCase: f toLocaleUpperCase()
  ▶ toLowerCase: f toLowerCase()
  ▶ toString: f toString()
  ▶ toUpperCase: f toUpperCase()
  ▶ toWellFormed: f toWellFormed()
  ▶ trim: f trim()
  ▶ trimEnd: f trimEnd()
  ▶ trimLeft: f trimStart()
  ▶ trimRight: f trimEnd()
  ▶ trimStart: f trimStart()
  ▶ valueOf: f valueOf()
  ▶ Symbol(Symbol.iterator): f [Symbol.iterator]()
  ▶ [[Prototype]]: Object
  ▶ [[PrimitiveValue]]: ""

```

### 3. Lanac nasljeđivanja

Primjer sa funkcijom / vitičastim zagradama

```

> const Zivotinja = {
  vrsta: 'pas',
};
< undefined
> const fido = Object.create(Zivotinja);
< undefined
> fido.rep = 'kratak';
< 'kratak'
> const rex = Object.create(fido);
< undefined
> rex.dlaka = 'crna';
< 'crna'
> rex
< {dlaka: 'crna'}
  dlaka: "crna"
  [[Prototype]]: Object
    rep: "kratak"
    [[Prototype]]: Object
      vrsta: "pas"
      [[Prototype]]: Object
        constructor: f Object()
        hasOwnProperty: f hasOwnProperty()
        isPrototypeOf: f isPrototypeOf()
        propertyIsEnumerable: f propertyIsEnumerable()
        toLocaleString: f toLocaleString()
        toString: f toString()
        valueOf: f valueOf()
        __defineGetter__: f __defineGetter__()
        __defineSetter__: f __defineSetter__()
        __lookupGetter__: f __lookupGetter__()
        __lookupSetter__: f __lookupSetter__()
        __proto__: (...)
        get __proto__: f __proto__()
        set __proto__: f __proto__()
> rex.dlaka
< 'crna'
> rex.rep
< 'kratak'
> rex.vrsta
< 'pas'
>

```

Primjer sa class sintaksom

```

> class Osoba {
  govor() {
    return 'Govorim...';
  }
}
< undefined
> class SuperOsoba extends Osoba {
  let() {
    return 'Letim...';
  }
}
< undefined
> let ivica = new SuperOsoba();
< undefined
> ivica.let();
< 'Letim...'
> ivica.govor();
< 'Govorim...'
> ivica
< ▼ SuperOsoba {} ⓘ
  ▾ [[Prototype]]: Osoba
    ▶ constructor: class SuperOsoba
    ▶ let: f let()
    ▾ [[Prototype]]: Object
      ▶ constructor: class Osoba
      ▶ govor: f govor()
      ▾ [[Prototype]]: Object
        ▶ constructor: f Object()
        ▶ hasOwnProperty: f hasOwnProperty()
        ▶ isPrototypeOf: f isPrototypeOf()
        ▶ propertyIsEnumerable: f propertyIsEnumerable()
        ▶ toLocaleString: f toLocaleString()
        ▶ toString: f toString()
        ▶ valueOf: f valueOf()
        ▶ __defineGetter__: f __defineGetter__()
        ▶ __defineSetter__: f __defineSetter__()
        ▶ __lookupGetter__: f __lookupGetter__()
        ▶ __lookupSetter__: f __lookupSetter__()
        ▶ __proto__: (...)
        ▶ get __proto__: f __proto__()
        ▶ set __proto__: f __proto__()
  >

```

### 3. \_\_proto\_\_ vs prototype

`__proto__` je svojstvo svakog objekta - pokazuje na prototip objekt na osnovu kojeg je novi objekt kreiran  
`prototype` je svojstvo konstruktor funkcije ili klase - pokazuje na sva svojstva/metode koje nasljeđuje novi objekt

Primjer:

```

> function Osoba(ime) {
    this.ime = ime;
}
< undefined
> const ivan = new Osoba('Ivan');
< undefined
> ivan
< ▼ Osoba {ime: 'Ivan'} ⓘ
    ime: "Ivan"
    ▾ [[Prototype]]: Object
      ► constructor: f Osoba(ime)
      ► [[Prototype]]: Object
> ivan.__proto__
< ▼ {constructor: f} ⓘ
    ► constructor: f Osoba(ime)
    ► [[Prototype]]: Object
> Osoba.prototype
< ▼ {constructor: f} ⓘ
    ► constructor: f Osoba(ime)
    ► [[Prototype]]: Object
> ivan.__proto__ === Osoba.prototype
< true
> |

```

Ako želimo objektu ivan dodati novu metodu, to radimo pomoću prototype svojstva na objektu Osoba:

```

> Osoba.prototype.govor = function() {
    return 'Govorim...';
};
< f () {
    return 'Govorim...';
}
> ivan.govor();
< 'Govorim...'
> |

```