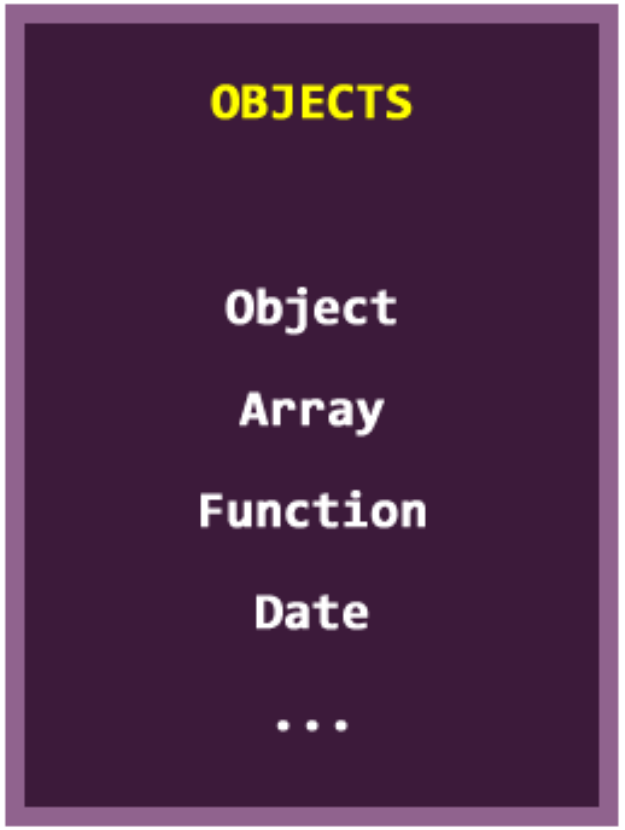
STANDARDI U PRIMJENI INTERNETSKE TEHNOLOGIJE

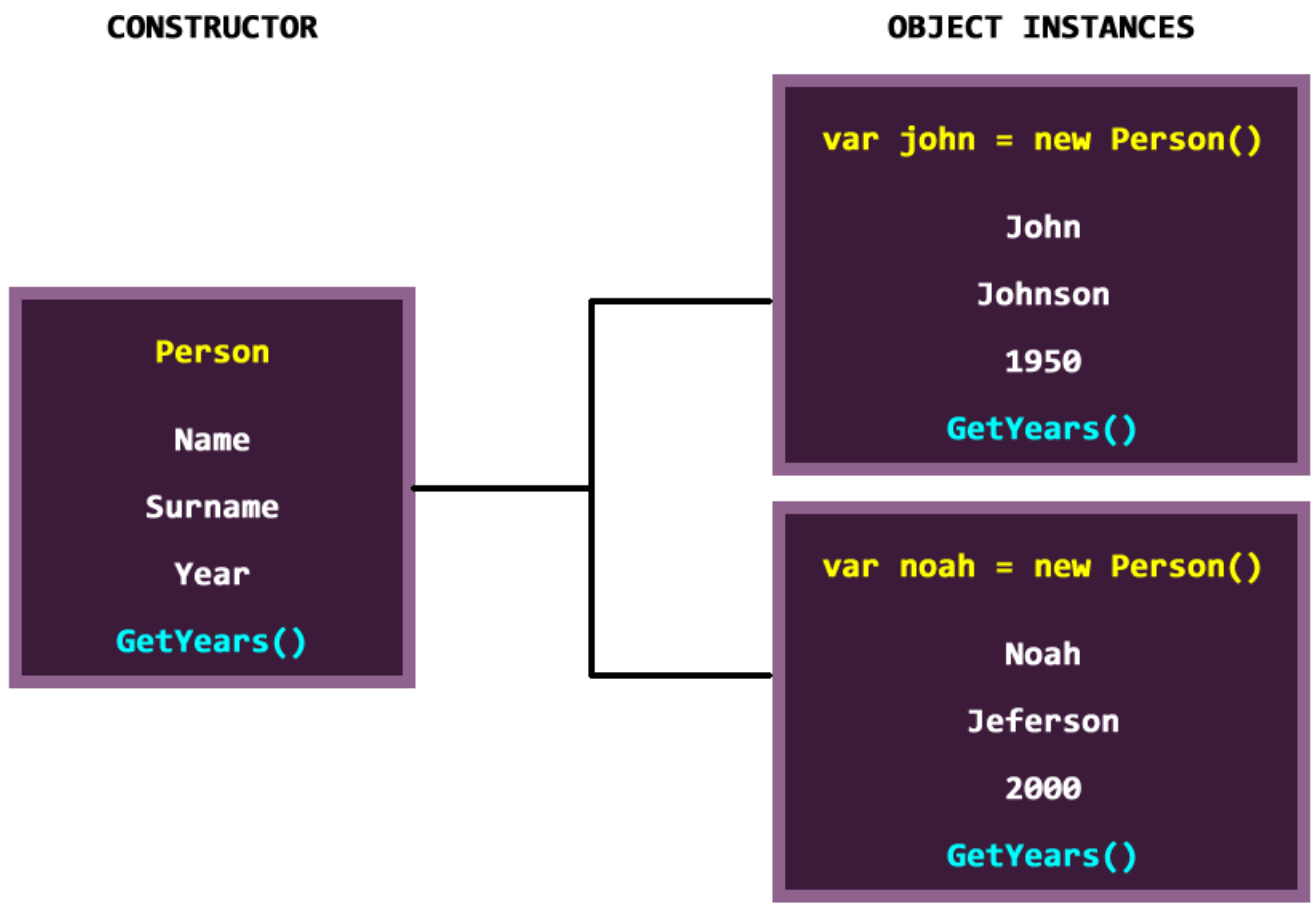
Predavanje 11

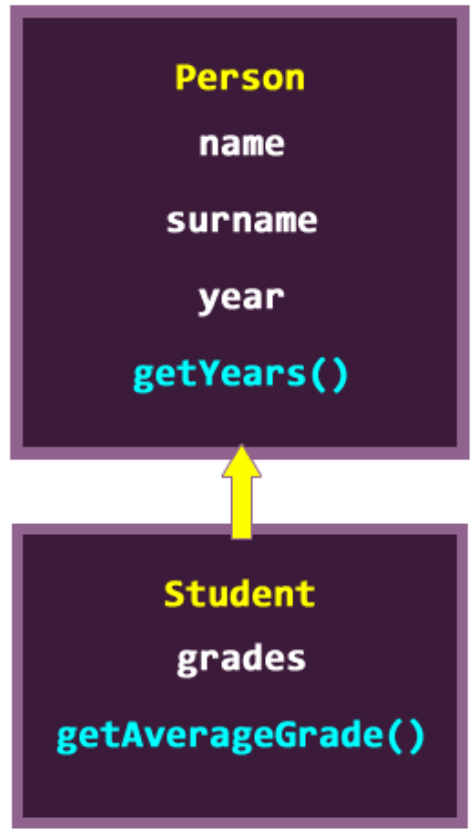
Objekti

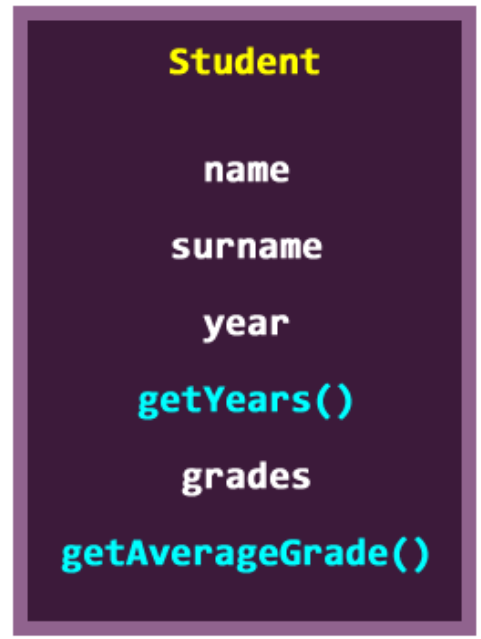
- Predstavljaju složeni tip podataka koji sadržava parove svojstvo-vrijednost kojima se pristupa preko naziva svojstva
- **Struktura objekata**
  - Objekti u JavaScriptu su sastavljeni od ključeva i vrijednosti. Vrijednosti mogu biti bilo koji tip podataka.
- **Pristupanje vrijednostima**
  - Vrijednosti unutar objekta mogu se pristupiti koristeći notaciju točke ili notaciju uglatih zagrada.
- **Metode**
  - Objekti mogu sadržavati funkcije, koje se nazivaju metodama objekta.
- **this**
  - Unutar metoda objekta this se odnosi na sam objekt.
- **Prototipovi**
  - Svaki JavaScript objekt ima prototip, objekt od kojeg nasljeđuje metode i svojstva.
- **Konstruktori i klase**
  - JavaScript koristi funkcije konstruktora i klase (uvedene od ES6) za stvaranje instanci objekata.

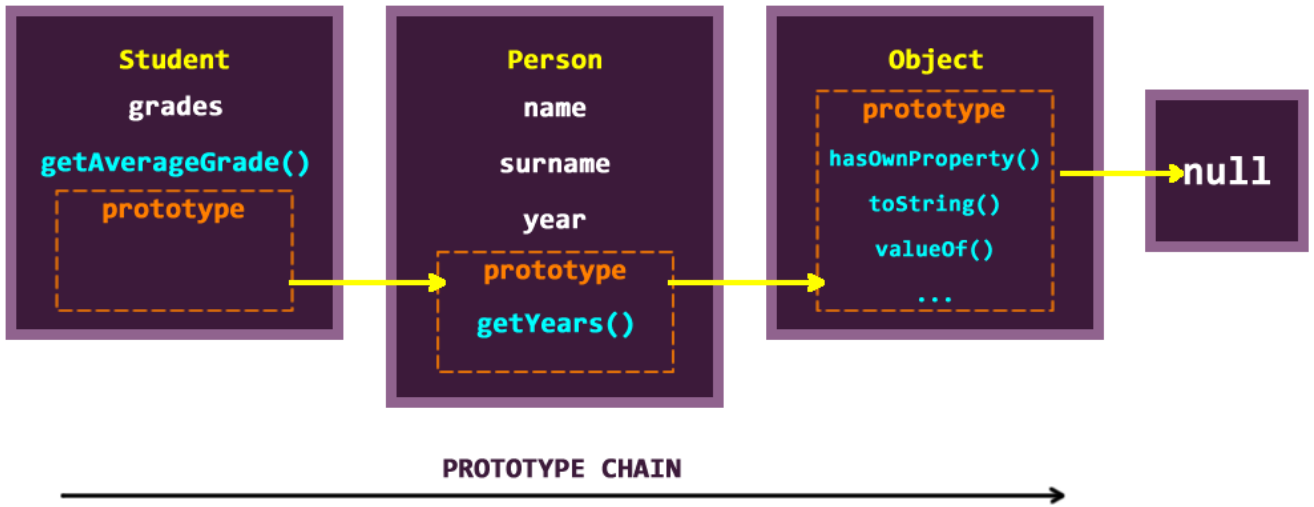
Izuzmemo li primitivne tipove sve su objekti




CONSTRUCTOR
OBJECT INSTANCES
Person
Name
Surname
Year
GetYears()
var john = new Person()
John
Johnson
1950
GetYears()
var noah = new Person()
Noah
Jeferson
2000
GetYears()


Person
name
surname
year
getYears()
Student
grades
getAverageGrade()


Student
name
surname
year
getYears()
grades
getAverageGrade()




```
▼ Student {grades: Array(8)}
  ▶ grades: (8) [5, 4, 5, 5, 4, 5, 3, 5]
  ▼ __proto__: Person
    ▶ constructor: ƒ Student(grades)
    ▶ getAverageGrades: ƒ ()
    ▼ __proto__:
      ▶ getYears: ƒ ()
      ▶ constructor: ƒ Person(name, surname, yea
      ▼ __proto__:
        ▶ constructor: ƒ Object()
        ▶ hasOwnProperty: ƒ hasOwnProperty()
        ▶ isPrototypeOf: ƒ isPrototypeOf()
        ▶ propertyIsEnumerable: ƒ propertyIsEnum
        ▶ toLocaleString: ƒ toLocaleString()
        ▶ toString: ƒ toString()
        ▶ valueOf: ƒ valueOf()
        ▶ __defineGetter__: ƒ __defineGetter__()
        ▶ __defineSetter__: ƒ __defineSetter__()
        ▶ __lookupGetter__: ƒ __lookupGetter__()
        ▶ __lookupSetter__: ƒ __lookupSetter__()
```

throw i try-catch-finally

Mehanizam za upravljanje i rukovanje greškama

- **throw** se koristi za "bacanje" iznimke. Kada se iznimka baci, izvršavanje trenutnog koda se zaustavlja, i kontrola se prebacuje na **catch** blok.
- Blok **try...catch** omogućuje testiranje bloka koda na greške, dok **catch** blok omogućuje rukovanje greškom.
- **finally** blok se izvršava nakon što se **try** i **catch** blokovi izvrše, bez obzira na to je li došlo do greške ili ne.
  - Ako kod unutar try bloka izlazi sa return
  - Ako se desi greška unutar catch bloka

```
function Kvadriraj(broj) {
    if (typeof broj !== 'number') {
        throw new TypeError("Pogrešan tip podataka");
    }
    return broj * broj;
}

try {
    console.log(Kvadriraj('4'));
}catch (exception) {
    console.log(exception);
    throw new Error('catch error');
}finally {
    console.log('finally');
}
```

throw i try-catch-finally

Osim generičkog **Error** objekta postoji još nekoliko specifičnih:

- **InternalError**
  - greška se podiže ako je došlo interne greške unutar JavaScript interpretera (npr. beskonačne petlje ili rekurzije)
- **RangeError**
  - greška se podiže ako su numeričke varijable izvan dozvoljenih vrijednosti
- **ReferenceError**
  - greška se podiže ako se pokuša referencirati nepostojeći objekt ili varijabla
- **TypeError**
  - greška se podiže ako su varijable neodgovarajućeg tipa