06. JS Object

Justina Balsė

Object

- Object allow us to store related data in a single place.
- When calling functions, we can pass an object as an argument. This lets you pass all that related data to a function without having to pass in many individual arguments.

Object

```
let myBook = {
   title: "1984",
   author: "George Orwell",
   pageCount: 326
};

// "1984 by George Orwell"
console.log(`${myBook.title} by ${myBook.author}`);

// Change an object property value using dot notation
myBook.title = "Animal Farm";

// "Animal Farm by George Orwell"
console.log(`${myBook.title} by ${myBook.author}`);
```

Object

```
let myBook = {
    title: "1984",
    author: "George Orwell",
    pageCount: 326
}

let getSummary = function (book) {
    console.log(`${book.title} by ${book.author}`);
}

getSummary(myBook);
```

Object

```
let person = {
   name: ['Bob', 'Smith'],
   age: 32,
   gender: 'male',
   interests: ['music', 'skiing'],
   bio: function () {
      alert(this.name[0] + ' ' + this.name[1] + ' is ' + this.age + ' years old. He
      likes ' + this.interests[0] + ' and ' + this.interests[1] + '.');
   },
   greeting: function () {
      alert('Hi! I\'m ' + this.name[0] + '.');
   }
};

person.bio();
person.greeting();
```

Objects as function return value

```
let convertFahrenheit = function (fahrenheit) {
    return {
        fahrenheit: fahrenheit,
        kelvin: (fahrenheit + 459.67) * (5 / 9),
        celsius: (fahrenheit - 32) * (5 / 9)
    }
}
let temps = convertFahrenheit(74);
console.log(temps);
```

Object | Sub-namespaces

```
let person = {
    name: {
        first: "Bob",
        last: "Smith"
    },
    age: 32
};

console.log(person.name.first);
```

Setting object members

```
let person = {
   name: {
      first: "Bob",
      last: "Smith"
   },
   age: 32
};

person.age = 45;
person.name.last = 'Cratchit';

// "45, Cratchit"
console.log(person.age + ", " + person.name.last);
```

Creating new members

```
let person = {
    name: {
        first: "Bob",
        last: "Smith"
    },
    age: 32
};

person.eyes = "green";
person.farewell = function() {
    alert("Bye everybody!");
};
```

in operator

- Returns **true** if the specified property is in the specified object.

```
let car = {make: 'Honda', model: 'Accord',
year: 1998};

// true
console.log('make' in car);
```

delete operator

```
let Employee = {
    firstname: "Tom",
    lastname: "Smit"
}

delete Employee.firstname;

// undefined
console.log(Employee.firstname);
```

for...in

```
let string = "";
let object = {a: 1, b: 2, c: 3};

for (let property in object1) {
   string += object[property];
}

console.log(string);
```

Constructor function

```
function Student() {
   this.name = "John";
   this.gender = "Male";
   this.sayHi = function () {
      alert('Hi');
    }
}
var student1 = new Student();
console.log(student1.name);
console.log(student1.gender);
student1.sayHi();
```

Attach new properties

```
function Student() {
   this.name = 'John';
   this.gender = 'Male';
}

var studObj1 = new Student();
studObj1.age = 15;
alert(studObj1.age); // 15

var studObj2 = new Student();
alert(studObj2.age); // undefined
```

Attach new properties

```
function Student() {
   this.name = 'John';
   this.gender = 'Male';
}

var studObj1 = new Student();
studObj1.age = 15;

var studObj2 = new Student();

console.log(studObj1);
console.log(studObj2);
```

- [object Object] {
  age: 15,
  gender: "Male",
  name: "John"
  }
- [object Object] {
  gender: "Male",
  name: "John"
  }

Prototype

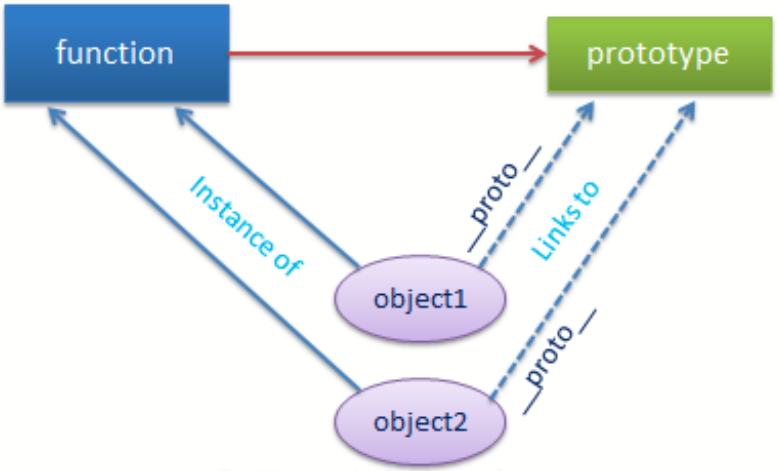
- The prototype is **an object** that is associated with every functions and objects by default in JavaScript

Prototype

```
function Student() {
   this.name = 'John';
   this.gender = 'M';
}

Student.prototype.age = 15;

var studObj1 = new Student();
alert(studObj1.age); // 15

var studObj2 = new Student();
alert(studObj2.age); // 15
```

Prototype

- Every object which is created using literal syntax or constructor syntax with the new keyword, includes **__proto__** property that points to prototype object of a function that created this object.



Date object

- The JavaScript Date object provides a way to work with dates and times.
- You can instantiate it in a number of different ways depending on the desired results.

Date object

- getDate
- getDay
- getFullYear
- getHours
- getMilliseconds
- getMinutes
- getMonth
- getSeconds
- getTime
- getTimezoneOffset

- setDate
- setFullYear
- setHours
- setMilliseconds
- setMinutes
- setMonth
- setSeconds
- setTime

Date object

```
// using built-in methods
let start = new Date();

// doSomethingForALongTime();

let end = new Date();
// elapsed time in milliseconds
let elapsed = end.getTime() - start.getTime();
```

Praktika (1) Restoranas

- Sukurti restorano objektą:
  - Restorano pavadinimas;
  - Vietų skaičius;
  - Rezervuotų vietų skaičius;
  - Funkcijos:
    - Tikrina ar yra vietų;
    - Rezervuoja vietas;
    - Atšaukia rezervaciją.

Praktika (2) E-Bankas

- HTML dokumente, pateikti įvedimo lauką, pasirenkamąjį sąrašą (Papildyti, Pasiimti) ir mygtuką.
- Įrašų istorija su data: raudona – išimta, žalia – papildyta.
- Įrašų filtras (pagal sumą)
- Įrašų rūšiavimas

Praktika (3) E-knygynas

- Knyga - objektas, kuris turi laukus:
  - laukus: pavadinimas, autorius, leidimo metai, puslapių skaičius, liko knygų, kaina.
- Sukurti kelias knygas, sudėti į masyvą.
- Sukurti funkcijas: ieškoti pagal pavadinimą, pagal autorių, pagal metus.