Rad sa objektima

Objekti su na svojevrstan nacin “kopija” objekata iz stvarnog svijeta. Objekti se sastoje od **key: value** parova, pri cemu se key odnosi na naziv polja, a value sadrzi vrijednost, koja moze biti primitivna, funkcija, niz, drugi objekat…  
  
Uzmimo za primjer prekidac. Prekidac ima svoje neke osobine i stanja, na primjer:   
- boja,   
- dimenzije,  
- broj tastera,  
- ukljucen/iskljucen.  
  
U programiranju ovo zovemo karakteristike objekara (properties ili fields). Ove karakteristike mogu (ali i ne moraju) imati vrijednosti - mogu im se dodijeliti naknadno.   
  
Objekti takodje imaju da vrse ili da se nad njima vrsi neka akcija. U programiranju to zovemo metode (methods, functions - metodama obicno nazivamo funkcije koje su definisane u nekom objektu.   
  
U javascript-u objekte mozemo kreirati “on-the-fly”, to jest, ne moramo nuzno imati klase iz koji kreiramo objekte. Najjednostavniji primjer objekta:  
  
const obj = {};

Za ovaj objekat kazemo da je prazan, tj, nema nikakve vrijednosti. Medjutim, za njega je alociran odredjeni memorijski prostor i u ovaj objekat mozemo dinamciki dodavati polja (npr: obj.myFied = “Test”;)

Primjer kompleksnijeg objekta:  
  
const person = {

firstName: "John",

lastName: "Doe",

age: 35,

occupation: "Driver"

};

Citanje vrijednosti iz ovog objekta radimo na sledeci nacin:  
  
console.log(person.firstName);

ili

console.log(person[“firstName”]);

s tim da se notacija sa tackom cesce koristi.

Kao sto smo naveli, objekti mogu imati odredjene akcije/metode. Na primjer:

const person = {

firstName: "John",

lastName: "Doe",

age: 35,

occupation: "Driver",

getFullName: function() {

return this.firstName + " " + this.lastName;

}

};

Ako izvrsimo person.getFullName() nazd cemo dobiti puno ime i prezime.

Sta je **this** u kontekstu objekta? To je referenca ka trenutnoj instanci konkretnog objekta. Dakle, this pokazuje na trenutni objekat u memoriji i napreko this kljucne rijeci mozemo pristupiti poljima i metodama trenutnog objekta. Svaki objekat ima samo jedno this - svaki objekat ima svoje this.  
  
U kontekstu funkcije/metode this se odnosi na objekat “kreator” te metode. Svaki objekat je unikatan na neki nacin (nesto sto ga razlikuje od drugih objekata). To je property “this” koji pokazuje na unikatan objekat u memoriji.

Drugi nacini kreiranja objekata:

* iz Object ugradjene metode create -> const obj = Object.create();
* instanciranjem ugradjenog Object-a: const obj1 = new Object();
* upotrebnom new kljucne rijeci -> const date = new Date();

Za upotrebu kljucne rijeci new, moramo naprije imati definisanu klasu. Klase su “sabloni” iz kojih kreiramo objekte. Ovome ce biti posvecena posebna sekcija.

Za prolazak kroz polja u objektu koristimo **for…in** sintaksu.

Na primjer:

const person = {

firstName: "John",

lastName: "Doe",

age: 35,

occupation: "Driver"  
};

for(let key in person) {  
 console.log(`${key}: ${person[key]}`);  
}

Brisanje polja iz objekta mozemo izvrsiti koristeci delete kljucnu rijec:  
  
delete person.age

Nakon ovoga i kljuc i vrijednost su nestali iz objekta i pozivanje person.age rezultirace sa undefined.

Vrijednosti iz nekog objekta mozemo pretvoriti u niz koristeci Object.values(). Npr:  
  
const values = Object.values(person); // [“John”, “Doe”, “Driver”]

Sve kljuceve iz nekog objekta mozemo izvuci koristeci u niz koristeci Object.keys(). Npr:

const values = Object.keys(person); // [“firstName”, “lastName”, “occupation”]

Koristeci Object.assign(), mozemo “spajati” objekte u nove. Object.assign tehniku mozemo iskoristiti za kreiranje “kopija” objekta. Npr:

const obj = {a: “a”};

const objCopy = Object.assign({}, obj); // objCopy ce biti novi objekat u memoriji sa {a: “a”} vrijednostima

Obratite paznju: U JS funkcije su objekti u sustini! Razlika u odnosu na ostale objekte je u tome so se funkcije mogu izvrsiti/pozvati. Validna sintaksta za kreiranje Function objekta je:  
  
const fo = new Function(params);

**Zastita objekata**

Podrazumijevano - objekti u JS su mutabilni, sto znaci da mozemo bez problema dinamicki dodavati kljuceve i vrijednosti ili mijenjati iste nad nekig objektom. Postoje mehanizmi da se ovo sprijeci. Najsesce koriscene metode su freeze i seal.

const obj = Object.freeze({}) - je krajnje striktan i ne dozvoljava promjenu kljuceva/vrijednosti u budnocsti

const obj = Object.seal({}) - je fleksibilniji metod i dozvoljava promenju vrijednosti, ali ne i kljuceva.

Kompletna lista metoda nad objektima:   
<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object>

**JSON**

JSON je skracenica od Javascript Object Notation. To je u sustini specijalna vrsta stringa naprisana u odredjenom formatu. Taj string prodjeca na JS objekte, s tim da je zaista string. Na primjer:  
  
const jsonString = ‘{

“firstName”: "John",

“lastName”: "Doe",

“age”: 35,

“occupation”: "Driver"  
}’;

Ovdje mozemo vidjeti neke razlike u odnosu na JS objekte:

* pocinje i zavrsava se znacima navoda,
* svaki kljuc ima znake navoda (obavezno),

Kao vrijednosti, JSON podrzava sledece tipove podataka:

* string,
* number,
* array,
* object,
* boolean.

Javascript ima specijalno ugradjeni objekat koji se koristi za manipulaciju json stringovima - JSON. JSON objekat nosi sa sobom dvije vrlo korisne metode:

* JSON.parse(string) - metoda koja pretvara json string u JS objekat,
* JSON.stringify(obj) - metoda koja radi suprotno - pretvara neki objekat u json string.

Vazno je napomenuti da se json format koristi vrlo cesto kao osnovi metod kod kreiranja API (Application Programming Interface) kao sredstvo za razmjenu podataka izmedju razlicitih dijelova sistema (npr. Frontend/Backend).

Zadaci:

1. Napisati funkciju koja provjerava da li su dva objekta ista, tj. da li imaju jednake kljuceve i vrijednosti.
2. Napisati “rucnu” implementaciju Object.values() metode - dakle funkciju koja vraca sve vrijednosti iz nekog objekta kao niz.
3. Napraviti objekat koji sadrzi firstName, lastName i funkciju abbrivatedName() koja vraca inicijale. Npr. ako je fristName Branimir, lastName Dragicevic, abbrivatedName treba da vrati “B. D.”.
4. Napisati funkciju spawnPlayer(name, type) koja vraca objekat, koji sadrzi polja:

* name - naziv igraca,
* type - tip igraca - moze biti: human, ork, elf i undead - ovo ima uticaj na akcije koje igrac moze da vrsi:

1. movementSpeed:

* human: nasumicno generisan broj izmedju 100 i 120,
* ork: nasumicno generisan broj izmedju 90 i 100,
* elf: nasumicno generisan broj izmedju 110 i 130
* undead:: nasumicno generisan broj izmedju 90 i 120

2. damage:

* human: nasumicno generisan broj izmedju 40 i 45,
* ork: nasumicno generisan broj izmedju 45 i 60,
* elf: nasumicno generisan broj izmedju 35 i 45
* undead:: nasumicno generisan broj izmedju 40 i 50