JavaScript ohjelmointi
Oliot (eng. objects*)*

MARGIT TENNOSAARIN MATERIAALISTA
LAURA JÄRVISEN MUOKKAAMA

Mikä olio?

- JavaScript-olio on kokoelma **avain–arvo -pareja**, joissa avain on merkkijono ja arvo mikä tahansa (numero, merkkijono, funktio jne.). Nämä ovat olion **ominaisuuksia**.
- JavaScriptin oliot mahdollistavat **datakokoelmien** tallentamisen ja **todellisten olentojen mallintamisen**.
- Oliot kokoavat yhteen toisiinsa liittyvän datan (**ominaisuudet**) ja toiminnot (**metodit**)

Olion luominen

- Helpoin tapa luoda yksittäinen olio on käyttää olion literaalia:

```
const animal = {
  name: "Kettu",
  species: "Koiraeläin",
  age: 4,
};
```

Olion ominaisuuksien käsittely

- Ominaisuuksiin pääsee käsiksi kahdella tavalla:
  - Piste-notaatio → Yleisin ja selkein tapa.
  - Hakasulkunotaatio → Tarvitaan, jos ominaisuuden nimi sisältää erikoismerkkejä tai on tallennettu muuttujaan.

```
console.log("Eläin on:", animal.name);
```

Ominaisuuksien lisääminen ja poistaminen

- Oliot ovat dynaamisia eli ominaisuuksia voi lisätä tai poistaa milloin tahansa.

```
animal.habitat = "metsä";

console.log("eläimen asuinpaikka:", animal.habitat);
```

```
delete animal.age;

console.log(animal);
// tulostaa: { name: 'Kettu', species: 'Koiraeläin', habitat: 'metsä' }
```

Olion metodit

- Metodit ovat olion sisällä olevia funktioita
- Niiden avulla oliot voivat suorittaa tehtäviä

```
const car = {
  brand: "Toyota",
  start() {
    console.log("Auto käynnistyy...");
  },
};
car.start(); // Tulostaa: Auto käynnistyy...
```

This -avainsana

- **this** viittaa kyseiseen olioon
- Sitä käytetään, kun metodin pitää päästä käsiksi olion ominaisuuksiin

Vanha tapa

```
const animal = {
  name: "Kettu",
  speak: function () {
    console.log(`Olen ${this.name}!`);
  },
};
animal.speak(); // Tulostaa: Olen Kettu!
```

ero

Uudempi tapa

```
const animal = {
  name: "Fox",
  speak() {
    console.log(`Olen ${this.name}!`);
  },
};
animal.speak(); // Tulostaa: Olen Kettu!
```

- ilman **this**-sanaa emme pääsisi dynaamisesti käsiksi nimeen

Olion metodit vs. tavalliset funktiot

```
const animal = {
  name: "Kettu",
  sound() {
    console.log(`${this.name} sanoo: Ring-ding-ding!`);
  },
};

animal.sound(); // "Kettu sanoo: Ring-ding-ding!"
```

- **Olioiden sisällä** käytä **this**-avainsanaa viitataksesi ominaisuuksiin.
- **Olioiden ulkopuolella** funktiot toimivat normaalisti.
- Jos **this** puuttuu, se ei viittaa oikeaan olioon.

Olion läpi käyminen silmukalla

- Käytä **for...in**-rakennetta käydäksesi kaikki olion ominaisuudet läpi.

```
const bike = { brand: "Helkama", model: "Jopo", year: 2023 };

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

// tulostaa
// brand: Helkama
// model: Jopo
// year: 2023
```

Oliot taulukoissa

- Oliot tallentaan usein taulukoihin, varsinkin tosielämän toteutuksissa

```
const zoo = [
  { name: "Fox", species: "Canine" },
  { name: "Eagle", species: "Bird" },
  { name: "Bear", species: "Mammal" },
];
console.log(zoo[1].name); // Tulostaa: Eagle
```

Monta samanlaista oliota

- Usein luodaan monta oliota, joilla on sama rakenne.
- Ne voidaan luoda tavalliseen tapaan yksi kerrallaan tai käyttämällä **konstruktoria** ja **new-**avainsanaa.
- Voit myös käyttää **luokkaa** ja **new-**avainsanan avulla tehdä uuden olion tähän luokkaan.
- Kontstruktori-funktioiden ja luokkien nimet aloitetaan isolla alkukirjaimella, jotta ne erotetaan tavallisista funktioista. Tämä on tapa eli konventio, eikä vaikuta koodin toimimisen.

Konstruktori-funktio

- Konstruktori-funktion avulla voit luoda useita olioita, joilla on sama rakenne

```
function Animal (name, species, age){
    this.name = name,
    this.species = species,
    this.age = age
}

const animal1 = new Animal ("Susi", "koiraeläin", 5)
```

- *Näiden nimet kirjoitetaan isolla alkukirjaimella!*

ES6 luokka eli class syntaksi

- JavaScript toi luokat (classes) käyttöön 2016 tarjoten jäsennellymmän tavan luoda olioita.

```
class Animal {
  constructor(name, species, age) {
    this.name = name,
    this.species = species,
    this.age = age;
  }
}

const animal1 = new Animal("Susi", "koiraeläin", 5);
```

- *Näidenkin nimet kirjoitetaan isolla alkukirjaimella!*

ES6 luokka eli class syntaksi

- Prettier muuttaa luokan usein tähän muotoon:

```
class Animal {
  constructor(name, species, age) {
    (this.name = name), (this.species = species), (this.age = age);
  }
}

const animal1 = new Animal("Susi", "koiraeläin", 5);
```

```
class Henkilo {
  constructor(etunimi, sukunimi, ika) {
    this.etunimi = etunimi;
    this.sukunimi = sukunimi;
    this.ika = ika;
  }
  // Metodi koko nimen palauttamiseksi
  getFullName() {
    return `${this.etunimi} ${this.sukunimi}`;
  }

  // Metodi tarkistamaan, onko henkilö täysi-ikäinen
  isAdult() {
    return this.ika >= 18;
  }
}

const henkilo2 = new Henkilo("Börje", "Smålland", 28);
console.log(henkilo2.getFullName()); // Tulostaa: Börje Smålland
console.log(henkilo2.isAdult()); // Tulostaa: true
```

Harjoitellaan

Löydät olio-harjoitukset viikon 6 kansiosta

https://github.com/bc-web-ohjelmistokehitys/WP25K-JS

Huom! Opettele itse - älä pyydä tekoälyä ratkaisemaan tehtävää puolestasi.