JavaScript ohjelmointi

Funktiot jatkuu , Ehtolauseen alkeet

MARGIT TENNOSAARIN MATERIAALISTA
LAURA JÄRVISEN MUOKKAAMA

Ennen kertausta:
Ohjelmoinnin peruspilarit

Ohjelmoinnin tärkeimmät konseptit ovat:

Tietotyypit

Muuttujat

Funktiot

Ehtolauseet

Silmukat

Näihin pohjautuu suurin osa ohjelmoinnista. Kohta(!) on kaikki käyty ja jatkossa niitä yhdistellään haastavammin ja haastavammin.

Eli kaikkea tähän asti opittua pääsee kertaamaan tällä kurssilla ja koko lopun koodaajauransa.

Kertaus eiliseltä

- Uudelleen käytettävää koodia
- Käytetään jonkin tietyn toiminnon tekemiseen toistuvasti tai reaktiona johonkin tapahtumaan (esim. klikkaus - tämä opetellaan pian!)
- Funktiota täytyy kutsua nimellä ja suluilla ()

```
function sum(x, y) {
  return x + y;
}

sum(3, 4);
console.log(sum(3, 4)); // 7
```

Kertaus eiliseltä

Jos funktiosta halutaan tietoa ulos, käytetään lopussa **return**-sanaa.

Funktion sisälle voidaan lähettää tietoa -kuten arvoja tai muuttujia- funktiokutsun yhteydessä.

```
function multiply(a, b) {
  return a * b;
  console.log("This will not run.");
}

console.log(multiply(2, 3)); // 6
```

Funktiot jatkuvat

https://github.com/bc-web-ohjelmistokehitys/WP25K-JS/blob/main/01_viikko/06_funktioiden_perusteet.md

Nuolifunktiot

```
// Standard function
const sum = (a, b) => {
  return a + b;
};

// Implicit return
const add = (a, b) => a + b;

const isEven = (num) => num % 2 === 0;
```

- Nuolifunktiot voivat palauttaa ilman **return** -sanaa
- Suositellaan yksirivisille ilmauksille
- Selkeys ennemmin kuin tiiviys!

Funktiot vs. Nuolifunktiot

Ominaisuus	Tavallinen funktio	Nuolifunktio
Tiiviys	Pidempi syntaksi	Lyhyempi syntaksi
this-konteksti	Oma this	Perii ympäröivän this
Soveltuu konstruktoreihin	✅	❌ (ei käytetä new-kanssa)
Lyhyet funktiot	OK	Erinomainen valinta

- Eli ilman nuolifunktioitakin pärjää
- Nuolifunktiot eivät käyttäydy 100% samalla tavalla kuin tavalliset funktiot

Oletusparametrit

- Oletusarvo, joka annetaan funktiolle, jos funktiokutsussa ei anneta arvoa

```
function greet(name = "Guest") {
  return `Hello, ${name}!`;
}

console.log(greet()); // Hello, Guest!
```

Funktioiden yhdistäminen

- On mahdollista, usein myös ihan järkevää
- Vältä liian monimutkaisia rakenteita!

```
const trimText = str => str.trim();
const toLowerCase = str => str.toLowerCase();

function cleanText(str) {
  return toLowerCase(trimText(str));
}
console.log(cleanText("   JavaScript!   ")); // javascript!
```

Tee tehtäviä

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

ps. funktiotehtävien vastausvaihtoehdot julkaistu GitHubiin. Tee ensin ja tutki sitten miten muuten ne olisi voitu toteuttaa.

Tutustutaan ehtolauseeseen

https://github.com/bc-web-ohjelmistokehitys/WP25K-JS/tree/main/02_viikko/05_ehtolauseet.md

Ehtolauseet

Ehtolauseita käyttämällä ohjelma voi tehdä päätöksiä määriteltyjen ehtojen perusteella.

Eri koodilohkot suoritetaan sen mukaan onko ehto tosi vai epätosi.

if -lause

- Tarkistaa onko ehto tosi.
- Jos tosi, suoritetaan lohkon koodi.

```
if (age >= 18) {
  console.log("You are an adult");
}
```

if … else -lause

- Antaa kaksi vaihtoehtoa:
  - Yksi, jos ehto on tosi
  - Toinen, jos ehto ei ole tosi

```
if (age >= 18) {
  console.log("You are an adult");
} else {
  console.log("You are a minor");
}
```

if … else if … else -lause

- Mahdollistaa useamman ehdon tarkistamisen.
- Antaa useita vaihtoehtoja.

```
if (score >= 90) {
  console.log("Grade: A");
} else if (score >= 80) {
  console.log("Grade: B");
} else {
  console.log("Grade: F");
}
```

Mikä olisi vastaus käytössä ei ole *if … else if … else* vaan pelkkiä if-lauseita ja muuttujan *score* arvo on 91?

Tee tehtäviä

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