

JavaScript jatko

(JavaScript ja valittu kehyskirjasto [React])

Kokonaisuus 12 päivää

29.9.2021

Kouluttaja ja materiaali Juha-Pekka Finnilä

“The most disastrous thing that you
can ever learn is your first
programming language.”

— Alan Kay, palkittu amerikkalainen ohjelmistokehityksen tutkija.
Aiemmin myös ammatillinen jazz kitaristi.

Kurssin suoritus

Kurssin suoritus merkitään vain läsnäoloilla.

Päivittäiset läsnäolomerkinnät saa palauttamalla harjoitukset sovittuna ajanhetkenä, päivän päätteeksi tai pyytämällä lisääaikaa niiden tekemiseen.

Ennakkovaatimuksena:

HTML ja CSS –jatkokurssi (tai vastaava tietämys)
JavaScript perusteet (tai vastaava tietämys)

Tämä kurssi koostuu ”kahdesta” osa-alueesta:

1. JavaScript –kielestä
2. Valitusta kehyskirjastosta (React tai Vue)
3. Sekä tietenkin ohjelmoinnin algoritmeista ja käytännöistä

Alustava aikataulu

Päivä 1: JavaScript perusteiden kertaus

Kertausta: muuttujat ja ehtolause

Päivä 2: JavaScript perusteiden kertaus

Kertausta: silmukat ja funktiot

Päivä 3: JavaScript perusteiden kertaus

Kertausta: funktiot

Päivä 4: Taitotalolla, Oma projektityö

Päivä 5: JavaScript jatkoa

Verkkosivulle kirjoittaminen/lukeminen

Päivä 6: JavaScript jatkoa

Oliot, taulukot ja regular expression

Päivä 7: Taitotalolla, Oma projektityö

Päivä 8: JavaScript jatkoa

Verkkosivun muokkaaminen

Päivä 9: JavaScript jatkoa, React

Perusteet, projektin asentaminen,

Päivä 10: Taitotalolla, Oma projektityö

Päivä 11: JavaScript jatkoa, React

CDN, luokat, freeCodeCampia

Päivä 12: JavaScript jatkoa, React

Kertausta, kokonaisuus

Alustava aikataulu

Neljätoista (12 pv) päivää kokonaisuus.

Sisältää JavaScriptin lisäopintoja sekä React kehystä.

Suurimmalle osalle toinen ”oikea” ohjelmointikurssi.

Kurssin materiaali (PDF) löytyvät Taitoforumista:

<https://www.taitoforum.fi/course/view.php?id=5995>

Tavoitteet

- Oppia lisää ohjelmointiin liittyviä ratkaisutapoja ja käytäntöjä
- Saada harjoitusta ongelmien ratkaisuun ja hyvän koodin kirjoittamiseen
- Oppia enemmän ES6 päivityksen mukaisia rakenteita ja käytänteitä
- Oppia perusteet React kehyksen käyttämisestä
- Oppia perusteet React projektin tekemiseen ja muokkaamiseen.
- Tutustua erilaisiin ohjeisiin ja dokumentaatioihin, joiden avulla koodia voi tutkia
- Ymmärtää, että ohjelmoinnin oppimisen pitää olla itsenäistä ja jatkuvaa

Opiskeluaympäristö: Vioppe & freeCodeCamp

Vioppe

Kurssilla hyödynnetään harjoitteluympäristönä Metropolian Vioppe-ympäristöä.

Sivuston teoria on suomeksi. Tehtäviä on vähän ja ne ovat varsin ”haastavia” (nirsoja syötteen suhteen). Seuraavalla dialla kirjautumisohjeet.

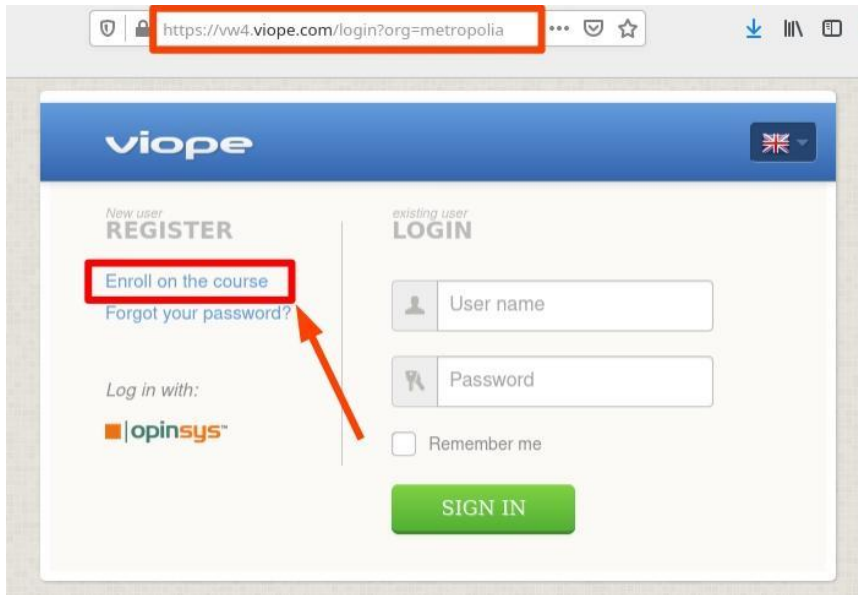
freeCodeCamp

Kurssilla hyödynnetään myös ilmaista freeCodeCamp opetussivustoa. Sieltä löytyy mm. kurssi ”**JavaScript algorithms and data structures certification**”. Tehtäviä on paljon ja ne ovat hyviä.

W3schools (+w3resource.com)

Kurssin materiaalit ja ohjeita tehtäviin kannattaa hakea W3school-sivustolta. Muista hakea **JavaScript** ohjelmointiohjeita. 😊 Tehtäviä on paljon.

Opiskeluaympäristö: Viope



vw4.viope.com/login?org=metropolia

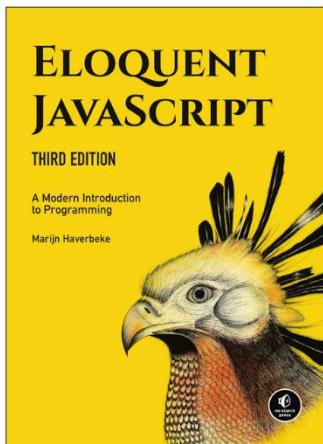
→ kurssi: **React.js fundamentals**

Mistä luodaan uusi tunnus ViOpeen
(jos haluaa luopua vanhasta)?

Avaa selain inCognito-tilaan ja
kirjoita **koko** osoiterimpu.

Sitten ”*enroll on the course*” tulee
uudelleen näkyviin.

Kirja: Eloquent JavaScript, ilmainen e-kirja



ELOQUENT JAVASCRIPT
3RD EDITION (2018)

This is a book about JavaScript, programming, and the wonders of the digital. You can read it online here, or buy your own paperback copy.

CONTENTS

Introduction	
1. Values, Types, and Operators	(Part 1: Language)
2. Program Structure	
3. Functions	
4. Data Structures: Objects and Arrays	
5. Higher-order Functions	
6. The Secret Life of Objects	
7. Project: A Robot	
8. Bugs and Errors	
9. Regular Expressions	
10. Modules	
11. Asynchronous Programming	
12. Project: A Programming Language	
13. JavaScript and the Browser	(Part 2: Browser)
14. The Document Object Model	
15. Handling Events	
16. Project: A Platform Game	
17. Drawing on Canvas	
18. HTTP and Forms	
19. Project: A Pixel Art Editor	
20. Node.js	(Part 3: Node)
21. Project: Skill-Sharing Website	

- Ilmainen ja kehuttu e-kirja.
- Rakenteeltaan melko lähellä meidän kurssimme rakennetta.

Linkki kirjaan [tässä](#).

Mitä tehdä, kun koodi/funktio ei toimi?

1. Tarkista oikeinkirjoitus

- a) Ovatko **"""-merkit** pareittain ja oikeassa järjestyksessä? Esim. `' "asia" "asia ' asia ' "`
- b) Ovatko **()**, **[]** ja **{}**-**sulkeet** pareittain ja oikeassa järjestyksessä?
- c) Onko funktion/metodin lopussa **()**-pari?

2. Tarkista funktiot ja eventit.

- a) Onko funktion nimi kirjoitettu oikein sekä funktiossa että sen kutsussa?
- b) Onko eventti oikein? Kutsuuko se oikean funktion?

Jos ei vielääkään toimi, niin

- a) yksinkertaista koodia,
- b) laita väliin `console.log()`-tarkistuksia, jotta näet mitä koodi tekee missäkin kohdassa,
- c) **Kirjoita koodi uudestaan!** Tee tämä monta kertaa. Älä säästä mitään. Kirjoita **KAIKKI** uudestaan!

Mitä tehdä, kun koodi/funktio toimii väärin?

1. Tarkista mitä funktio palauttaa.
Onko palautuslause kunnossa? Mikä on tietotyyppi?
2. Debuggaa funktion vaihteita.
3. Lisää `console.log()`-tulosteita eri vaiheisiin.

Päivä 1.

JavaScript jatko – perusasioiden kertaus

Kertaus ja päivitys muuttujista ja ehtolauseista

Ympäristöt ja kurssin rakenne

Päivä 1: Perusteet

- Tutustuminen
- Tavoitteet
- Tutustutaan vaihtoehtoihin oppimisympäristöihin: freeCodeCamp, ViOpe,...
- JavaScriptin perusrakenteet
 - Muuttujat
 - Arvojen asettaminen
 - Syntaksi
 - Ehtolauseet (If-else)

Materiaalia

PDF

freeCodeCamp

ViOpe

W3school

teoria, osa 2 tehtävät 1-3
+ monivalintatehtävät

Oppimisympäristö

- ViOpe
- freeCodeCamp
- Oma kone (Visual Studio Code)

Tehtävä 0. Millaista on hyvä koodi?

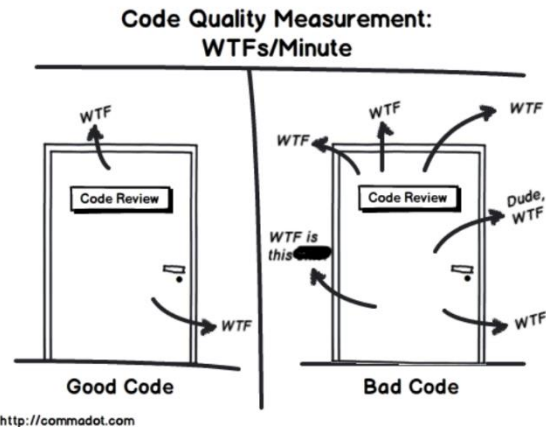
1. Lue artikkeli
<https://www.freecodecamp.org/news/clean-coding-for-beginners/>
2. Miten tiivistäisit tärkeimmät ohjeet?
3. Mitä ovat **Noise Words** ja miksi niitä ei saisi käyttää?

Aikaa tehtävään tekemiseen 20 minuuttia

Palauta vastaus **YLEISEEN** Teams-keskusteluun. Jatketaan kello 9.45

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

– Martin Fowler



Käsitteitä

Verkko

- WWW
- Skripti (script)
- Rajapinta (API)
- Palvelin (server)
- Asiakas (client)
- Skeema (schema)
- JSON

Ohjelmointi

- Muuttuja (variable, var)
- Merkkijono (string, str)
- Vakio (constant, const)
- Globaali (global)
- Paikallinen (local)
- Olio (object)
- Instanssi
- Funktio (function)
- Metodi (method)
- DOM

Sovelluksia ja teknologioita

- Node.js
- Express, AJAX,...
- PHP
- JavaScript
- SQL
- MongoDB
- React

Opetallaan keskustelemaan koodin kanssa

- Voit antaa ohjelmalle arvoja suoraan muuttujien kautta tai hyödyntämällä DOM rajapintaa.
- Opimme myöhemmin DOM-rajapinnan **document-olion** käyttöä.
- Tällä hetkellä hyvä olio on **window**.

Pyydetään käyttäjältä arvo muuttujaan **luku**.

```
let luku = window.prompt("Anna luku:");
```

An embedded page on this page says

Anna luku:

OK

Cancel

Mitä JavaScript on?

Teknisesti

- Korkean tason ohjelmointikieli
- Skriptikieli
- Tulkattava komentosarjakieli
- Ei puhdas olio-ohjelmointikieli
- Rakenteellinen
- Dynaamisesti ja heikosti tyypitetty
- Tyyliohjeeton
- Ei sukua Javalle
- Perustuu löyhästi C-kieleen
- Moniparadigmainen kieli

Käsitteellisesti

- Kieli dynaamisen sisällön tuottamiseen
- Oikealta nimeltään EcmaScript
- Verkkosivujen ohjelmointikieli
- Erittäin käytetty/suosittu kieli
- Nykyaikaisen verkon ydin teknologioita
- Nopeasti kehittyvä

JavaScript muuttujat

Muuttujat

- Seitsemän tietotyyppiä:

`undefined`, `null`, `boolean`, `string`, `symbol`, `number` ja `object`

Suomeksi: `määrittämätön`, `null`, `totuus`, `merkkijono`, `merkki`, `luku` ja `olio`

- Joskus vain merkintätapa on erona tietotyypille. Esim. 90 tai "90"
- Muuttujien nimet ovat usein matematiikasta tuttuja. Esim. x ja y. Tai i ja j.
Mutta pääasiallisesti helppolukuisia Esim. annettuLuku, tulos
- Yleinen määrittely on muotoa: `let omaArvo;`
- Arvo asetetaan lauseella: `omaArvo = 15;`
`muuttujan nimi = annettu arvo;`

Lähes kaikkia muuttujia käsitellään kuin olioita.
Eli niillä on omia **metodeja**, joilla niitä voidaan käsitellä.

Null ilmoittaa, että on selkeästi asetettu "tyhjä". `let arvo = null;`

Undefined syntyy, kun muuttujalle ei anneta sisältöä/tyyppiä. `let arvo;`

Tyhjä syntyy, kun määritellään tyhjä merkkijono. `let arvo = "";`

...koska ne ovat olioita

Muuttujilla on valmiita sisäänrakennettuja **metodeja** (eli olion funktioita), joilla niitä voidaan muokata ja niistä saadaan tietoja.

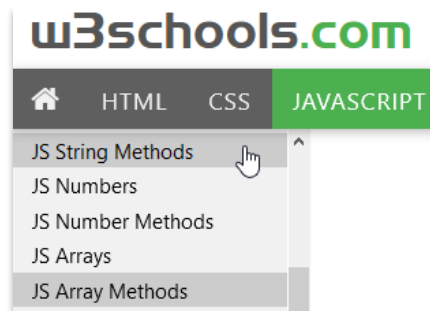
Seitsemän perustietotyypin lisäksi on myös muita **oliotietotyypppejä**, kuten

Date	-päivämäärät	
Math	-matemaattiset vakiot ja funktioita	
Error	-virheiden esitys ja ohjelman katkaisu	
RegExp	-säännönmukaisuuksien metsästys	
Array	-eli taulukko on listan kaltainen oliorakenne	var arr = ["kissa", 5, tuoli]
Object	-olio, josta lisää myöhemmin	

Joitain tärkeitä olio-muuttujien metodeja

Merkkijonojen metodeja (W3schools)

length	palauttaa merkkijonon pituuden
indexOf()	palauttaa merkkijonon paikan
charAt()	palauttaa merkin paikan avulla
toUpperCase()	suurentaa kaikki kirjaimet
replace(valittu, korvaava)	korvaa valitun merkkijonon toisella



Lukujen metodeja (W3schools)

toString()	palauttaa luvun merkkijonoksi
number()	palauttaa luvun mistä tahansa tietotyypistä

Huom. Suurin osa lukuihin liittyvistä metodeista ja funktioista löytyy Math-oliosta.



JavaScript muuttujien käyttö

- Var on alkuperäinen JS muuttuja.
- Let ja Const tulivat ES6 päivityksessä 2015.

	Päivitettävä (updatable)	Voi uudelleen luoda (re-declarable)	Käytettävissä? Globaali, lokaali/funktio tai lohko	Luonti ilman arvoa	Käyttö suositeltavaa
Var	Kyllä	Kyllä	Globaali ja lokaali	Kyllä	Ei
Let	Kyllä	Ei	Lohko	Kyllä	Kyllä
Const	Ei	Ei	Lohko	Ei	Kyllä, eniten

Esimerkki + operaatiosta eri tietotyypeillä

Merkkijono-tietotyyppi

```
let x = "56";
```

```
let y = "20";
```

```
let tulos = x + y;
```

```
console.log("Tulos on "+tulos);
```

Tulostaa merkkijonon

```
"Tulos on 5620"
```

Luku-tietotyyppi

```
let x = 56;
```

```
let y = 20;
```

```
let tulos = x + y;
```

```
console.log("Tulos on "+tulos);
```

Tulostaa merkkijonon

```
"Tulos on 76"
```

Esimerkki + operaatiosta ja tietotyypin muuttamisesta

Merkkijono-tietotyyppi

```
let x = "56";
```

```
let y = "20";
```

```
let tulos = parseInt(x) + parseInt(y);
```

```
console.log("Tulos on "+tulos);
```

Tulostaa merkkijonon

```
"Tulos on 76"
```

Aina tietotyyppiä ei voi asettaa oikeaksi **heti**, vaan se pitää muuttaa myöhemmin.

Esim. kun tieto tuodaan verkkosivulta.

parseInt()-funktio muuttaa merkkijonon numeroksi (integer), jolloin laskeminen onnistuu.

Mitä tyyppiä mitkäkin tiedot ovat?

- a) Nimi =
- b) Ikä =
- c) Syntymäaika =
- d) Postinumero =
- e) Osoite =
- f) Puhelinnumero =

Mitäs tietotyyppejä JavaScriptissä
olikaan?

- Undefined
- Null
- Boolean
- String
- Number
- Symbol
- Object
 - Johon sisältyy paljon erilaisia olioita

DOM (Document Object Model)

Jotta WWW-sivua voidaan kontrolloida, pitää olla mahdollisuus viitata tärkeisiin HTML-elementteihin. DOM on oliomalli, jonka avulla voit kohdistaa haluamasi muutokset oikein.

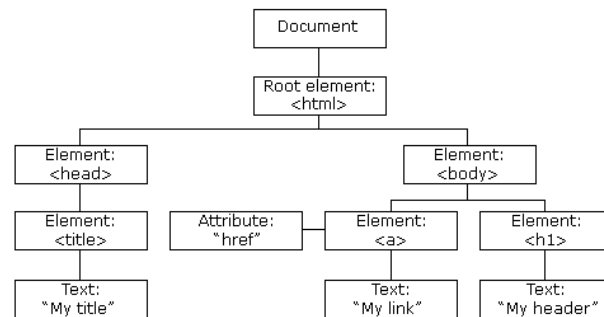
- Selainikkuna on nimeltään ***window***.

- Esim. globaalien muuttujien määrittely
- prompt, alert,...

- Dokumentti on nimeltään ***document***.

- Verkkosivukohtaisia arvoja, joita voidaan myös hakea.
- Tämän käskyn avulla etsitään elementit (tagit), joihin muutokset kohdistetaan.
- **Document** on siis sama asia kuin **html-tiedosto**

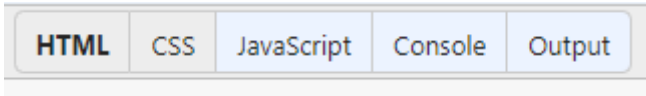
The HTML DOM Tree of Objects



DOM malliin palataan vielä
ja siihen kannattaa tutustua myös itsenäisesti.

JavaScriptien harjoittelu. JSBin

1. Kirjoita selaimeen **jsbin.com**
2. Valitse "JavaScript" ja "Console" valikot keskeltä.



3. Voit nyt kirjoittaa välittömästi ajavaa JavaScriptiä, eikä sinun tarvitse huolehtia ajoympäristöistä tai muusta.



Tehtävä 2. Perusmuuttujat ja boolean.

Tee **JSB**iniin (js ja console) scripti, jonka ajat ja saat seuraavan tilanteen.

Muuttujat yksi, kaksi, kolme ja nelja

Muuttuja yksi on numero 3.

Muuttuja kaksi on muuttuja yksi.

Muuttuja kolme on merkkijono 3.

Muuttuja nelja on totuusarvo TRUE.

Jos yksi on sama kuin kaksi

tulosta teksti **yksi on sama kuin kaksi**

Muussa tapauksessa tulosta **pieleen meni**

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun. Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 12.45

BONUS: Tee seuraava totuuslause.

Jos yksi on kaksi JA nelja TAI kolme on yksi

Tulosta **kaikki on samaa ja totta**

Muussa tapauksessa tulosta **ei mennyt niin**

Päätöksenteko-rakenteet

If – else ja switch



Ehtorakenne (if – else if – else, conditional)

Rakenteella luodaan vaiheittainen vertailu. Vertailu alkaa ylimmästä vaihtoehdosta, joka nimetään aina **if (ehto)**.

```
if (muuttuja1 == muuttuja2) {  
    //tapahtuma jos vertailu on totta  
}
```

Viimeinen vaihtoehto on määrittelemätön ja nimeltään **else**.

Kaikki muut vaihtoehdot ovat näiden kahden välissä ja kirjoitetaan **else if(ehto) –muotoon**.

Rakennetta käytetään usein tilanteissa, joissa vaihtoehtoja on monta, **järjestyksellä on väliä** ja/tai ne ovat monimutkaisia.

```
let yksi = 1;  
let kaksi = window.prompt("käyttäjän antama numero")  
  
if(yksi == kaksi){  
    console.log("Molemmat ovat nro 1");  
}else if(yksi > kaksi){  
    console.log("Annettu nro on pienempi kuin 1");  
}else if(yksi < kaksi){  
    console.log("Annettu nro on suurempi kuin 1");  
}else{  
    console.log("Jotain meni pieleen");  
}
```

Tehtävä 1. If-else

Tee vertailu

Tee muuttuja **age**

Jos ikä on alle 18 vuotta koodi ilmoittaa, että olet ala-ikäinen.

Jos yli tai yhtä suuri kuin 18, ilmoittaa, että olet täysi-ikäinen.

Tee koodistasi looginen. Ota huomioon tosielämä.

Bonus: Tee muuttuja **adult** ja käytä sitä. Mikä muuttujatyyppi?

Bonus2: Tee vastaava vertailu, joka jakaa käyttäjän lapsen, nuoreen, aikuiseen ja vanhukseen. Päätä itse iät.

Aikaa tehtävien tekemiseen 10 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun. Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 13.45

Osaamistavoite: 1. päivä

- Kerrataan ja palautetaan mieleen perusideat ja rakenteet
- Ehtorakenteet (If-else)
- Kannustetaan itseopiskeluun
- Korostetaan itsenäisen opiskelun tarvetta
- Palauttaa mieleen muutama erilainen konsoli/terminaali (selain, verkkopalvelu, VSC,...)

Päivä 2.

JavaScriptin perusasioiden kertaus

Kerrataan ja laajennetaan toistorakenteita

Päivän aiheet

- Perussyntaksia

- Ohjelmarakenteet
 - Jatketaan päätöksentekoa
 - Toistorakenteet

ViOpe luku 2.

Materiaalia

PDF

freeCodeCamp

ViOpe

W3school

teoria, osa 2 tehtävät 1-3 + monivalintatehtävät



Ehtorakenne (switch - case)

Rakenteella luodaan vaihtoehtoinen vertailu, jonka tuloksena yksi valinta toteutetaan. Vertailua aloitetaan **switch(arvo)**-komennolla.

Vaihtoehdot merkitään **case arvoX**: -muotoisina.

```
switch(arvo){
  case arvo2:
    //verrataan arvo == arvo2;
    break;
  case arvo3:
    //verrataan arvo == arvo3;
    break;
  default:
    //jos mikään vertailu ei ole totta
}
```

Viimeinen vaihtoehto on määrittelemätön ja nimeltään **default**.

Break-komento pysäyttää koodin suorittamisen switch lauseessa.

Rakennetta käytetään usein tilanteissa, joissa vaihtoehtoja on paljon, järjestyksellä ei ole väliä ja/tai ne ovat yksinkertaisia.

```
let vertaa = "käyttäjän antama numero"
```

```
switch(vertaa){
  case 1:
    console.log("Numero on 1");
    break;
  case 2:
    console.log("Numero on 2");
    break;
  default:
    console.log("Numero ei ole 1 tai 2");
}
```

Tehtävä 1. Switch

Tee vertailu (switch)

Tee muuttuja **kuukausi**

Voit ottaa sille arvon esim. promptista

Tee **switch** rakenne, joka osaa kertoa mikä vuoden aika on kyseessä kuukauden perusteella.

Bonus: huomioi suomeksi ja englanniksi

Bonus2: huomioi myös iso ja pieni kirjain erot

Bonus3: Tee erillinen funktio testaukseen. Käytä silmukkaa

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun.

Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 8.55

Tehtävä 1b. Sama tehtävä if -elsellä

Tee vertailu (if-else)

Tee muuttuja **kuukausi**

Voit ottaa sille arvon esim. promptista

Tee **switch** rakenne, joka osaa kertoa mikä vuoden aika on kyseessä kuukauden perusteella.

Bonus: huomioi suomeksi ja englanniksi

Bonus2: huomioi myös iso ja pieni kirjain erot

Bonus3: Tee erillinen funktio testaukseen. Käytä silmukkaa

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun.

Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 8.55

Toisto-rakenteet

For, while,...



Toistolauseet eli silmukat (loop)

Silmukoilla yksinkertaistetaan rakennetta ja estetään inhimilliset virheet, kun tehdään jotain toistuvaa.

Silmukka **toistaa sisältöään** niin kauan kuin **ehto on voimassa**.

Yleisiä silmukoita ovat

for	for (alustus, ehto, päivitys)
while	while (toistoehto)

Sekä niiden muunnelmät kuten

do while	do... while (toistoehto)
for in	

Ajetaan ainakin kerran

Silmukka voidaan myös keskeyttää kesken käskyllä **break**.
Silmukka voidaan keskeyttää myös ”hetkellisesti” käskyllä **continue**.

Tehtävä 2a. For-silmukka

Tee silmukka (for-lauseella)

Joka kirjoittaa seuraavat rivit:

Tämä on 1. kupillinen teetä tänään

Tämä on 2. kupillinen teetä tänään

Tämä on 3. kupillinen teetä tänään

Tämä on 4. kupillinen teetä tänään

Tämä on 5. kupillinen teetä tänään

Tämä on 6. kupillinen teetä tänään

Käytä let i:tä silmukan muuttujana.

Bonus: käytä muuttujaa Template Literaalilla.

Aikaa tehtävien tekemiseen 10 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun.

Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 10.20

Tehtävä 2b. For-silmukka

Tee silmukka (for-lauseella)

Joka laskee kuinka monta kirjainta a, on merkkijonossa

Merkkijonon voi napata promptilla.

Testaa lauseella "Kuinkahan monta a-kirjainta on tässä lauseessa?"

Hajota ongelma osiin.

Ja ratkaise osa kerrallaan.

`.length`

`muuttuja[i] == "a"`

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun.

Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 11.45

Tehtävä 3a. While-silmukka

Tee silmukka (while-silmukkana)

Joka kirjoittaa seuraavat rivit:

Tämä on 1. kupillinen teetä tänään

Tämä on 2. kupillinen teetä tänään

Tämä on 3. kupillinen teetä tänään

Tämä on 4. kupillinen teetä tänään

Tämä on 5. kupillinen teetä tänään

Tämä on 6. kupillinen teetä tänään

Käytä let i:tä silmukan muuttujana.

Bonus: käytä muuttujaa Template Literaalilla.

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun.

Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 12.15

Totuusarvot, vertailut ja ehdot

“Truth can only be found in one place: the code.”

– Robert C. Martin

Ohjaus ehdoilla

Vertailu	Selitys	Esimerkki	Tulostus (true / false)
arvo	TOTTA, jos arvo on olemassa (ei null)	arvo	true
!arvo	TOTTA, jos arvoa ei ole olemassa(null) tai tyhjä	!arvo	false
==	TOTTA, jos arvot OVAT SAMAT	'1' == 1	true
===	TOTTA, jos arvot JA tietotyypit OVAT SAMAT	'1' === 1	false
!=	TOTTA, jos arvot EIVÄT OLE SAMAT	'1' != 1	false
!==	TOTTA, jos arvot EIVÄT OLE samat tai samaa tyyppiä.	'1' !== 1	true
<	TOTTA, jos ensimmäinen arvo PIENEMPI kuin seuraava	'1' < 1	false
>	TOTTA, jos ensimmäinen arvo SUUREMPI KUIN seuraava	'1' > 1	false
<=	TOTTA, jos ensimmäinen arvo PIENEMPI TAI YHTÄ SUURI KUIN seuraava	'1' <= 1	true
>=	TOTTA, jos ensimmäinen arvo SUUREMPI TAI YHTÄ SUURI KUIN seuraava	'1' >= 1	true

Pohdinta. 5 min. Ehtorakenne ja totuusarvot

Mikä vaihtoehto tulostuu, kun ehto on...

- a) `1 < 2`
- b) `"15"`
- c) `!18`
- d) `1 !<= 2`
- e) `1 == "1"`
- f) `1 === "1"`
- g) `"hassu" <= "hassu"`
- h) `45 >= 54`
- i) `null`
- j) `!1 == 0`

```
if (ehto){  
    console.log("Tosi");  
}else {  
    console.log("Epätosi");  
}
```

Vai error?

Pohdinta. 5 min. Ehtorakenne ja totuusarvot

Mikä vaihtoehto tulostuu, kun ehto on...

a) $1 < 2$

"Tosi"

b) "15"

"Tosi"

c) !18

"Epätosi"

d) $1 \leq 2$

"error"

e) $1 == "1"$

"Tosi"

f) $1 === "1"$

"Epätosi"

g) "hassu" <= "hassu"

"Tosi"

h) $45 \geq 54$

"Epätosi"

i) null

"Epätosi"

j) $!1 == 0$

"Tosi"

```
if (ehto){  
  console.log("Tosi");  
}else {  
  console.log("Epätosi");  
}
```

Vai error?

Ohjaus vertailuilla (loogiset operaattorit)

Vertailu	Tulostus (true / false)	Selitys
ehto1 && ehto2	JA AND	Molempien ehtojen pitää olla TOSIA, jotta TOSI
ehto1 ehto2	TAI OR	Riittää että toinen ehto on TOTTA

Vertailut voivat olla pikkuisen monimutkaisempia kuin vain `ehto1 == ehto2`.

Esimerkiksi, jos `ehto1 = totta`, `ehto2 = tosi`, `ehto3 = epätosi`, `ehto4 = tosi`
`ehto1 == ehto2 && (ehto1 == ehto3 || ehto4)`

Lue
jos `ehto1` on sama kuin `ehto2` **JA**
(`ehto1` on sama kuin `ehto3` **TAI** `ehto4` on tosi)

Tehtävä 4. If else, Kirjautuminen

Tee vertailu

Tee muuttujat nimi ja salasana

Jotka käyttäjä antaa

Tee muuttujat nimiTK ja salasanaTK

Ohjelmassa annettu

Tee vertailu

- jos nimi on sama kuin nimiTKa ja
salasana on sama kuin salasanaTK

niin tulosta "Olet käyttäjä **nimi**
tervetuloa!"

- muissa tapauksissa vertailu antaa
joko "tunnus väärin" tai "salasana
väärin" tulostukset.

Aikaa tehtävän tekemiseen 20
minuuttia

Palauta kuvakaappaus opettajalle
Teams keskusteluun.

Jos koodi ei toimi, niin palauta myös
se.

Jatketaan kello 13.35

Tehtävä 4b. Kirjautuminen

Tee käytettävyyttä

Jatketaan oheista tehtävää siten että tehdään siitä käytettävämpi.

Tunnukset löytyvät taulukosta tunnukset, jossa on ainakin 3 erilaista tunnusta.

Salasanat löytyvät taulukosta salasanat, jossa saman verran salasanoja.

Täsmäytä siten että salasana ja tunnus ovat samoilla paikoilla.

Toteuta promptit siten että jos tunnus ei täsmää, niin ei pyydetä salasanaa

Aikaa tehtävän tekemiseen 20 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun.

Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 13.35

Lisätehtävä 4. Toistorakenteet / silmukat

Tee Viopen JavaScript kurssista
Osan 2 tehtävät tehtävät 1 - 3

PROGRAMMING

1. 2-1: Luvun arvon tarkistus

2. 2-2: Toistorakenne

3. 2-3: Alkulukujen tarkistaminen

Aikaa tehtävien tekemiseen 20 minuuttia

Jatketaan hetken aikaa 13.40

Ei palautusta.

Bonus: Tee osan 2 monivalintatehtävät

Kertaa: lue aiheen teoria itseksesi

Lisätehtävä 5. freecodecamp.com

Kirjaudu freecodecamp.com

- JavaScript Algorithms and Data Structures
 - Basic JavaScript

Tee tehtävät 19-

Aikaa tehtävien tekemiseen 30 minuuttia

Jatketaan huomenna

Ei palautusta.

Osaamistavoite: 2. päivä

- Ymmärtää perusteet JavaScriptin totuusarvoista.
- Ymmärtää perusteet JavaScriptin ohjelmarakenteista kuten silmukoista, ehdoista ja valinnoista.
- Osaa käyttää silmukkaa merkkijonon läpikäyntiin.
- Osaa muotoilla ehtolauseita
- Osaa käyttää valintarakennetta if-else.

Päivä 3.

JavaScriptin perusasioiden kertaus

Funktiot

Algoritminen ajattelu

***Functions should do one thing.
They should do it well.
They should do it only.***

– Clean Code

Päivän aiheet

- Kertaus

- Päätöksen teko ehtorakenteella
 - If – else if – else
 - Switch
- Toistorakenteet eli silmukat
 - For
 - While

- Funktiot

- Algoritminen ajattelu

Materiaalia

PDF

freeCodeCamp

ViOpe

W3school

osa 4 tehtävät

- Mikä on **funktio**? Miten funktiota käytetään?

- Mikä on **metodi**? Miten metodia käytetään?

- Algoritminen ajattelu.
Ongelmanratkaisu.

Tähän mennessä tuotettu koodi on vain yksi iso dokumentti, jota luetaan alusta loppuun. On tehokkaampaa pilkkoa toistuvia rakenteita ja logiikoita omiksi osaohjelmiksi.

Siksi funktiot.

Tehtävä 0. Silmukat

Lukaiskaa (ja tallentakaa itsellenne) silmukoista artikkeli.

<https://www.freecodecamp.org/news/javascript-loops-explained-for-loop-for/>

Kirjoita ylös ranskalaisilla viivoilla, mitä ovat silmukoiden yleisimmät virheet ja ongelmatilanteet.

Googlaa ja pohdi löydätkö muitakin.

Aikaa tehtävän tekemiseen 15 minuuttia

Jaa vastauksesi TEAMSin yleiseen keskusteluun.

Jatketaan kello 8.50

Jatketaan kello 9.05

Käsitteet

Funktio	function
Metodi	method
Argumentti, parametri	argument , arg, args, parameter
Muuttuja	variable
Tulos	result
Tuloste	print, (console.log())
Palautusarvo	return value

Funktio tunnetaan myös muilla nimillä... kuten aliohjelma, metodi, proseduuri, rutiini,...

//Esimerkki funktio

```
function nimi(argumentit) {  
    let muuttuja, tulos  
    console.log("Tuloste")  
    return palautusarvo;  
}
```

//funktion kutsu

```
nimi(args1, args2)
```


Funktion poikkileikkaus

//Funktio, joka palauttaa summan

```
function nimi(arvo, arvo2){  
    let tulos = arvo + arvo2;  
    return tulos;  
}
```

//Tässä funktion kutsu

```
nimi(34, 54);
```

function on avainsana, joka määrittelee uuden funktion.

nimi on funktion nimi, jonka avulla funktio kutsutaan

arvo(t) ovat **parametrejä**, jotka annetaan ()-sulkeiden sisällä funktiolle. Funktiossa ne muuttuvat **argumenteiksi**, jotka ovat alustavat **muuttujat**.

return on avainsana, joka palauttaa jotain funktiosta. Eli korvaa funktion kutsun jollain arvolla.

tulos on **muuttuja**, jonka funktio palauttaa perustuen arvoihin **arvo** ja **arvo2**.

Tehtävä 1. Funktion kirjoittaminen

Kirjoita funktio nimeltä **kerronta**, joka kertoo yhteen kaksi annettua lukua.

Aikaa tehtävän tekemiseen 20 minuuttia

Kutsu funktio eri arvoilla ainakin 5 kertaa.

Lähetä vastauksesi opettajalle Teamsiin. Tai yleiseen keskusteluun.

BONUS: voit käyttää kutsumiseen silmukoita. Huomaa että voit silmukan muuttujalla muokata myös funktion saamia arvoja.

Jatketaan kello 10.00

BONUS: KERTOTAULUTAULUKKO!

Siisti ja yksinkertainen on eleganttia - mutta aluksi vaikeaa lukea

Kommentoitu funktio

*/*nimi()-funktio saa kaksi arvoa ja laskee ne yhteen. Palauttaa yhteenlaskun tuloksen.*/*

```
function nimi(arvo, arvo2){  
  //Luodaan ja lasketaan tulos-muuttuja  
  let tulos = arvo + arvo2;  
  //Palautetaan tulos-muuttuja  
  return tulos;  
}
```

Sama funktio lyhyesti

//Palauttaa kahden arg summan

```
function nimi(arvo, arvo2){  
  return arvo + arvo2;  
}
```

Ja lyhyemmin? (arrow function)

```
nimi = (arvo, arvo2) => arvo + arvo2
```



Funktion ja metodin eroja

Funktio

jokuFunktio(arg);

- ”koodin” suorittama funktio.
- On määritelty koodissa.
- Sisältää usein/aina parametrin tai useita.
jokuFunktio(parametri).

Sekä funktio että metodi **toteuttavat** jonkin komentosarjan ja usein **palauttavat** tietoa.

Metodi

jokuOlio.jokuMetodi();

- Olion suorittama funktio.
- On tallennettu olioon olion ominaisuutena.
- Sisältää harvoin parametreja
jokuMetodi(parametri).
- Käytetään **this**-avainsanaa .

Kaikki metodit ovat funktioita, mutta kaikki funktiot eivät ole metodeja.

Algoritminen ajattelu

Miten ohjelmointiin liittyviä ongelmia voi ratkaista? Mihin kiinnittää huomio?

Algoritminen ajattelu (ongelman ratkaisu)

Kuinka ajatella kuin koodari?

Mozillan esitys aiheesta

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/A_first_splash

Youtube (10 osainen sarja), Ted talk

<https://www.youtube.com/watch?v=KFVdHDMcepw&list=PLJicmE8fK0EgogMqDYMgcADT1j5b911or>

Youtube, How To Think Like a Programmer, 6:15 min

- <https://www.youtube.com/watch?v=rWMuElcdJP4>

freeCodeCamp.org: How to learn coding, blogikirjoitus

- <https://www.freecodecamp.org/news/how-to-learn-coding-approach-is-everything/>

Miten ajatella ja löytää ratkaisu?

1. Ymmärrä mitä kysytään.

Lue kysymys/ongelma **ainakin kolmesti**. Kysy selventäviä kysymyksiä.

2. Työstä ratkaisuja

Ratkaise paperilla ainakin kolmella eri tapauksella.

Tämä on ratkaisun löytämisen tärkein vaihe. Eli oletko ymmärtänyt ongelman oikein?

3. Yksinkertaista ja optimoi

Pura tehtävän erillisiin loogisiin osiin. Esim. verkkosivun koodi, funktio 1 ja funktio 2 sekä testausosio.

4. Kirjoita pseudo-ratkaisu

Pseudokoodi on syntaksitonta koodia, jolla voit kirjoittaa ratkaisun ensimmäisen vaiheen.

Jokainen rivi on yksi tapahtuma tai tapahtumasarja (kuten silmukka).

5. Kirjoita oikea ratkaisu

Kirjoita ratkaisu oikealla koodilla. Tee se osa kerrallaan ja varmista, että kaikki osat toimivat ennen kuin liität ne yhteen.

6. Testaa oikeaa ratkaisua

Ainakin kolmesti.

Joka kerta erilaisilla lähtöarvoilla.

Toimiiko koodi **jokaisessa** tapauksessa? Ottaako se kaikki virheet huomioon?

Pseudokoodi

Vaiheittainen ratkaisutekniikka.
Kirjoita vaiheet kuin opastaisit
konetta.

Esimerkkitehtävä

Etsi annetusta **numerolistasta** parilliset
luvut

```
// life motto  
if (sad() === true) {  
  sad().stop();  
  beAwesome();  
}
```

Pseudokoodina

funktio etsiParillisetLuvut
luo tyhjä **tuloslista** tuloksia varten

Jokaisella **numerolistan** alkiolla/paikalla
tarkasta jos numero on parillinen
 jos numero on parillinen
 kirjoita numero **tuloslistaan**
 jos numero ei ole parillinen
 älä tee mitään

Palauta tuloslista

Tehtävä 2. Ajattelua

Etsi netistä itsellesi **kaksi erilaista** ohjetta ***algoritmisen ongelman*** ratkaisuun.

Silmäile ja tee ohjeista tiivistelmä omaan käyttöäsi.

Aikaa tehtävien tekemiseen 25 minuuttia

Lisää tiivistelmä ja lähdelinkki **yleiseen Teams**-keskusteluun.

Jatketaan kello 12.20

Ajattelun kehittäminen

1. Opettele formaalia logiikkaa

Formaali logiikka on mm. matemaattisen ajattelun muoto, joka antaa sinulle vinkkejä miten ongelmia puretaan osiin ja ratkotaan merkkikielellä.

2. Hanki lisää tietoa

Lue kirjoja. Kuuntele podcasteja. Katso dokumentteja. Kehitä mielikuvitustasi ja alitajuntaasi.

3. Hanki elämäkokemusta

Elä ja opi. Juttele erilaisten ihmisten kanssa. Kysy asioita. Opi erilaisia ammatteja ja taitoja.





Funktionaalinen ohjelmointi

- Lähestymistapa, jossa hyödynnetään funktioita.
- Yhdistämällä yksinkertaisia ja lyhyitä funktioita rakennetaan monimutkaisia ratkaisuja ja toteutuksia.
- Koodissa pyritään matemaattiseen puhtauteen ja sivuvaikutusten minimoimiseen.
- Tärkeimmät ideat:
 - Funktiot ovat itsenäisiä ohjelman tilasta ja globaaleista muuttujista.
 - Saavat **aina** argumentteina/parametreina arvonsa
 - Funktiot eivät tee muutoksia ohjelman tilaan ja globaaleihin muuttujiin, ellei välttämätöntä. Tällä vähennetään sivuvaikutuksia.
 - Funktiot vaikuttavat mahdollisimman vähän itse ohjelmaan.

Funktionaalinen ohjelmointi

Miksi? Mitä hyötyä tästä kaikesta on?

Paljonkin, sillä...

- a) Funktiot ovat siirrettävissä eri ohjelmien välillä.
- b) Ohjelmat pysyvät paremmin hallinnassa.
- c) Muutokset tapahtuvat vain funktion sisällä, jolloin virheet eivät eskaloitu.
- d) Virheiden hallinta on huomattavasti helpompaa

Funktioiden hyvät käytännöt

Hyviä käytäntöjä

- Nimen tulee **kuvata funktion tehtävää/ehtoa**.
- Funktion pituus aina alle 20 riviä.
- Yksi ja **vain** yksi tehtävä.
- Arvo(t) annetaan parametreinä/argumenttina. Arvoja vain vähän.
- Ei lippuargumentteja. (totuusarvo, joka annetaan funktiolle parametrina)

Hyviä käytäntöjä määrittelevät eri yritysten ja organisaatioiden tyyliohjeet. Näiden avulla pyritään yhtenäistämään koodia sekä tekemään siitä helpommin luettavaa

Tyyliohjeita

- [Googlen JavaScript tyyliohje](#)

Tehtävä 1. Funktio

Tee **JSBin**:issä yksinkertainen funktio nimeltä **tervehdys**, joka saa argumentiksi nimen.

Testaa funktiota

```
tervehdys("Simo");  
tervehdys("omanimi");
```

Funktio toteuttaa seuraavat tulosteet:

```
Hyvää päivää Simo.  
Hyvää päivää "omanimi".
```

Bonus: tee funktio, joka saa argumentiksi nimen ja numeron. Funktio toistaa tervehdyksen numeron monta kertaa.

Bonus2: Ja viimeisellä kerralla nimi on kirjoitettu isoilla kirjaimilla.

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun. Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 13.05

Tehtävä 2. Funktioita

Tee **JSBin**:issä yksinkertainen funktio nimeltä **yhteenlasku**, joka saa argumentiksi kaksi lukua ja laskee ne yhteen. Eli palauttaa niiden summan.

Bonus: Tee toinen funktio, joka huolehtii tulostamisesta.

Ja yhdistä funktiot.

Aikaa tehtävien tekemiseen 10 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun. Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 13.30

Tehtävä 3. Funktioita

Tee **JSBin**:issä yksinkertainen funktio nimeltä **sekalasku**, joka saa argumentiksi kaksi lukua.

Jos molemmat ovat positiivisia, palautetaan niiden erotus.

Jos molemmat ovat negatiivisia, palautetaan niiden kertolasku.

Jos toinen on negatiivinen, palautetaan niiden jakolasku.

Bonus: Onkohan tässä kaikki?
Puuttuuko jotain?

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun. Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 13.00

Tehtävä 4. Funktioita

Tee **JSBin**:issä funktio nimeltä **tulostus**, joka saa argumentiksi luvun.

Funktio tulostaa luvun määrän verran lauseita "I will not do anything bad again." Ja lopuksi yhden kerran "Bart Simpson".

Bonus: Onkohan tässä kaikki?
Puuttuuko jotain?

Bonus2: Funktio saa argumenttina myös lauseen ja allekirjoituksen.

Aikaa tehtävien tekemiseen 20 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun. Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 8.45

Tehtävä 5. Funktioita

Tee **JSBin**:issä funktio nimeltä **nopeampi**. Funktio saa neljä arvoa.

elain1, nopeus1, elain2, nopeus2

Jos eläin1 on nopeampi se tulostaa tiedon. Jos eläin 2 on nopeampi se tulostaa tiedon. Jos eläimet ovat yhtä nopeita se tieto tulostetaan.

Mahdollisia tulosteita ovat:

"Kissa on nopeampi kuin koira."

"Papukaija on nopeampi kuin kissa."

"Koiraa ja susi ovat yhtä nopeita."

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta kuvakaappaus opettajalle Teams keskusteluun. Jos koodi ei toimi, niin palauta myös se.

Jatketaan kello 9.35



Mistä funktio/olio saa muuttujat?

Scope (eli laajuus) määrittelee muuttujan käytettävyyden ja näkyvyyden koodissa.

Aiemmin vain kaksi (global ja local), nyt myös **lohkolaajuus** (block).

Paras käytäntö on käyttää **lohkolaajuuksia**, mutta poikkeuksia on.

Globaalit (global variable)

Koko ohjelman laajuiset muuttujat. Vaarallisia muuttaa. Muuttuvat samaan aikaan kaikkialla ohjelmassa.

Paikalliset (local/function variable)

Jonkin koodin/funktion/metodin sisäiset muuttujat. Eivät säilytä arvoaan funktion ulkopuolella.

Lohkot (block variable)

Käytettävissä { }-lohkon sisällä.

Osaamistavoite: 3. päivä

- Ymmärtää perusteet JavaScriptin funktioista ja metodeista.
- Ymmärtää funktionaalisen ohjelmoinnin tarkoituksen ja idean.
- Osaa tehdä funktion.
- Osaa asettaa muuttujia funktioon.
- Osaa rakentaa funktiolle tehtävän silmukoilla, ehdoilla ja valinnoilla.
- Osaa palauttaa funktiolla halutun tuloksen.

Päivä 4.

Itseopiskelu, ohjaus, projektitehtävät Taitotalolla

Päivä 5.

Verkkosivun muokkaamista JavaScriptillä.

Tapahtumanhallintaa ja toiminnallisuutta

Olio-ohjelmointi (pieni vilkaisu)

PDF sivut 33-41

Päivän aiheet ja tavoitteet

- Visual Studio Coden käyttöönotto (kertaus jos tarpeen)
 - Scriptien liittäminen HTML sivuun
 - Tapahtumanhallinta ja –kuuntelu
- Tutustutaan aiheen alkeisiin.
- Lukeminen ja kirjoittaminen HTML sivuun.
 - Olio-ohjelmointi

Materiaalia

PDF

sivut 33 – 41

freeCodeCamp

ViOpe

teoria, osa 3 tehtävät 1-3 + monivalintatehtävät

W3school

HTML DOM osio ja siitä eteenpäin + hyviä tehtäviä ja esimerkkejä

Vaihdetaan Visual Studio Codeen

Vaihdetaan Visual Studio Codeen.

Tehtäviä ja muita voi vielä testaila JSBinissä, mutta pyritään liittämään jatkossa tehdyt koodit osaksi verkkosivua.

Tee itsellesi pohjatiedosto, jota käytät jatkossa koodin ”testaukseen”.

HUOM. `console.log()` –komennot **EIVÄT** tulostu verkkosivulle **VAAN** sen konsoliin/terminaaliin. ***CTRL + SHIFT + i***

Tehtävä 0. Alustus

Tarkasta, että VS Code toimii.
Lisää siihen haluamasi lisäosat.

Suositus on:

Babel JavaScript
Live Server
(joku *lint)

Tee itsellesi kansio, lataa se VS Codeen ja tee sinne HTML-tiedosto.

Muokkaa HTML-tiedosto haluamaasi muotoon.

Esim.

pari div-rakennetta,
pari p-elementtiä ja niissä tekstiä.
Yksi lomakkeen aloitus.

Aikaa tehtävien tekemiseen 15 minuuttia

Laita peukku keskusteluun, kun olet tyytyväinen tilanteeseesi.

Jatketaan kello 10.10

Pidetään heti perään tauko.

Eli jatketaan kello 10.25

Tehtävä 0b. VS Coden testaus

Käytä html-tiedostoa ja lisää sinne script-osio, jossa teet tulostuksen konsoliin.

Tarkasta Live Serverin kautta, että tulostus onnistuu.

Aikaa tehtävien tekemiseen 15 minuuttia

Laita peukku keskusteluun, kun olet tyytyväinen tilanteeseesi.

Jatketaan kello 9.15

Tehtävä 1. Konsoliin tulostus

Tulosta JS:llä selaimen konsoliin

Aikaa tehtävien tekemiseen 10 minuuttia

Kirjoita yksinkertainen JavaScripti verkkosivuun ja aseta se tulostamaan jotain konsoliin.

Palauta tehtävä opettajalle Teams-keskusteluun

Missä se näkyy VSC:ssä?

Jatketaan kello 12.05

Tarkista tuleeko sen näkyviin verkkosivulle.

CTRL + SHIFT + i

Scriptin lisääminen sivuun

- Yhtä oikeaa paikkaa ei ole, vaan sijoittaminen riippuu tarkoituksesta.
- Scripti voi olla yhtä hyvin ulkoinen (oma tiedosto) kuin sisäinen pätkä koodia.
- Scripti voidaan ladata yhtä hyvin head osiossa kuin body:n lopussa
 - Jos scriptissä ei ole tapahtumankäsittelijää, pitää se ladata elementin jälkeen.

Perussäännöt ovat:

- Pitkät koodit aina erilliseen ulkoiseen tiedostoon.
- Pitkät ja raskaat koodit (jotka eivät ole sivun toiminnan kannalta välttämättömiä) aina body:n loppuun.

HTML DOM malli ja muokkaaminen

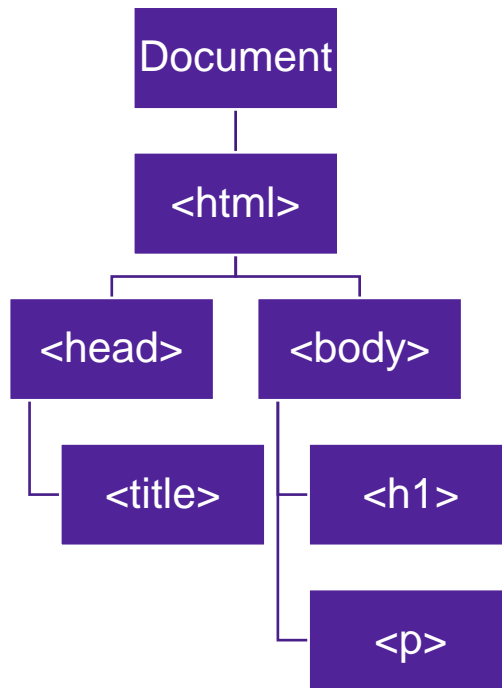
W3Schools [HTML DOM](#)

Mistä JS tietää mihin muutokset kohdistetaan?

Valitun elementin tunnistaminen tehdään useimmiten HTML elementin **id**-ominaisuudella.

`<p id="tunniste"></p>`

Seuraavalla kalvolla lisää....



Kirjoitetaan elementin sisäosa uudelleen

`.innerHTML` –komento kirjoittaa elementin sisälle

Esimerkiksi `<p>Tänne</p>` Halutun tekstin ja/tai HTML komentoja.

```
<div id="tunniste"></div>
```

```
document.getElementById("tunniste").innerHTML = "Tässä teksti";
```

Pieni huomio: Ole tarkka lainausmerkkien kanssa. Joskus `"tunniste"` on hyvä ja joskus `'tunniste'`. Lainausmerkeillä on väliä. Erityisesti sisäkkäisissä tapauksissa.

Tässä vielä osat eroteltuna riveille

document

tarkoittaa, että luetaan koko HTML dokumentti (käytännössä nykyinen verkkosivu) kun toteutetaan seuraava käsky.

getElementById("tunniste")

tarkoittaa, että etsitään kaikki elementit, joissa on id="tunniste"

innerHTML = "Tässä teksti";

tarkoittaa, että kirjoitetaan kaikkien valittujen elementtien **sisään** teksti "Tässä teksti"

Luetaan elementin sisältö

```
<div id="tunniste">Luetaan tämä</div>
```

```
let teksti = document.getElementById("tunniste").innerHTML;
```

Nyt teksti **Luetaan tämä** on tallennettu muuttujaan **teksti**.

Näiden lisäksi on paljon käskyjä, joilla voidaan lukea ja kirjoittaa eri osia HTML elementeistä sekä niitä lisää.

Tutustu kunnolla W3Schoolin aiheeseen [DOM Elements](#)



Tapahtumat ja tapahtumankäsittelijät

Kohdistetaan haluttu toiminto valittuun tapahtumaan.

Kolme erilaista tapaa.

Esim. Napin klikkaus:

onclick – event/tapahtuma on ominaisuus, joka annetaan HTML elementille, johon haluamme kohdistaa muutoksen sitä painettaessa.

Ei ole pakko kohdistaa aina nappiin.

Vain yksi tapahtumankäsittelijä per tapahtuma.

1. HTML elementin ominaisuudella

```
<button type="button" onclick="funktio()">Tässä se on</button>
```

Jatkuu

2. Tapahtumankäsittelijäominaisuuksilla (event handler properties)

```
let nappi = document.getElementById('ohjaavaID');
```

```
nappi.onclick = seFunktio();
```

Tässä on sama tapahtuma kuin yläpuolella, mutta lyhyemmin.

```
document.getElementById('ohjaavaID').onclick = seFunktio();
```

Tutustu kunnolla W3Schoolin aiheeseen [DOM Events](#)

Nykyään suositaan....

3. addEventListener:iä, johon voi mm. yhdistää monta funktioita yhteen tapahtumaan.

```
let nappi = document.getElementById('ohjaavaID');
```

```
nappi.addEventListener('click', itseFunktio);
```

Käydään tätä läpi myöhemmin. Aloitetaan harjoittelu aiemmilla menetelmillä.

Tehtävä 1. Tapahtumatulostus

Tulosta JS:llä HTML-elementtiin tekstiä

Tulosta `<p>` elementtiin, jonka id on "tulosta" teksti "Moikka vaan!", kun painetaan nappia.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Vinkki: w3school

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun.
Koodina/kuvakaappauksena.

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 9.55

Vastaus:

Lyhyesti:

```
<p id="testaus">Tässä on yksinkertaisempaa tekstiä. </p>
```

```
<button type="button" onclick="getElementById('testaus').innerHTML = 'Moikka vaan';">Testaa</button>
```

Pitkästi:

```
<form>
  <button type="button" onclick="testiFunktio()">Lisäys</button>
</form>

<p id="teksti">Tässä on yksinkertaisempaa tekstiä. </p>

<script>
  function testiFunktio(){
    let elementti = document.getElementById('teksti');
    elementti.innerHTML = "Moikka vaan!"
  }
</script>
```

Esimerkki: muutetaan funktio kerta-ajoksi

```
<h1>Tässä on otsikko</h1>
```

```
<p id="tulosta"></p>
```

```
<button id="btn">Tulosta</button>
```

```
<script>
```

```
    document.getElementById("btn").addEventListener("click", function(){
```

```
        document.getElementById("tulosta").innerHTML = "Moikka vaan!"
```

```
    }
```

```
</script>
```

Etuna

- Koodi tiiviimpää
- Ei kerrannaisvaikutuksia

Haittoina

- Ei uudelleen käytettävää
- Vaikeampi lukea

Tehtävä 2. Lomakkeen tiedonhaku ja tulostus

Lue ja tulosta JS:llä HTML-elementtiin tekstiä

Lue lomakekentästä nimi ja tulosta se <p>-elementin sisään, jonka id on "tervehdys".
Tulosta teksti "Hei, sinun nimesi on: *"nimi"*"

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Vinkki: input type="text"
.value

Vinkki2:

https://www.w3schools.com/jsref/tryit.asp?filena me=tryjsref_text_value

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun.

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 10.50

Vastaus:

```
<form>
  <label >nimi</label>
  <input id="annettuNimi" type="text">
  <button type="button" onclick="lisaaNimi()">Lähetä</button>
</form>

<p id="lisaaNimi">Tähän tulee lisää toinen</p>

<script>
  function lisaaNimi(){
    let nimiKaksi = document.getElementById('annettuNimi').value;
    document.getElementById('lisaaNimi').innerHTML = "Moikka, sinun nimesi on "+nimiKaksi+"!";
  }
</script>
```


Tehtävä 3. Lomakkeen tiedonhaku ja tulostus

Lue, laske ja tulosta JS:llä HTML-elementtiin summa

Lue lomakkeesta kaksi lukua ja tulosta se <p>-elementin sisään, jonka id on "summa", niiden yhteenlaskettu summa.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Vinkki: input type="text" (vai number?)

.value

parseInt()

Vinkki2:

https://www.w3schools.com/jsref/tryit.asp?filena me=tryjsref_text_value

Aikaa tehtävien tekemiseen 15 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 12.10

Vastaus:

```
<input type="text" id="eka"></input>
<input type="text" id="toka"></input>
<p id="tulosta">Tähän tulos: </p>
<button onclick="summaa()">Summaa</button>

<script>
  function summaa(){
    let eka = document.getElementById("eka").value;
    let toka = document.getElementById("toka").value;

    let tulos = parseInt(eka) + parseInt(toka);

    document.getElementById("tulosta").innerHTML = tulos;
  }
</script>
```

Tehtävä 4. Lomakkeen tiedonhaku, silmukka ja tulostus

Tulosta JS:llä HTML-elementtiin tekstiä ja käsittele tietoa silmukassa

Lue lomakekentästä numero ja tulosta se `<p>`-elementin sisään, jonka id on "laskelma".

Tee silmukka, joka tulostaa kirjoitetun luvun perään sen kertotaulun 0-10.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Vinkki:

kirjoita tulokset sarjaksi uuteen muuttujaan.
Tai käytä `+=` -menetelmää.

Aikaa tehtävien tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 13.00

Jatketaan kello 13.15

Vastaus:

```
<input type="text" id="nro"></input>
```

```
<p id="tulosta">Tähän tulos: </p>
```

```
<button onclick="kertotaulu()">Summaa</button>
```

```
<script>
```

```
    function kertotaulu(){
```

```
        let x = parseInt(document.getElementById("nro").value);
```

```
        document.getElementById("tulosta").innerHTML = x + ":n kertotaulu on: ";
```

```
        for(let i = 0; i <= 10; i++){
```

```
            let tulos = i * x;
```

```
            document.getElementById("tulosta").innerHTML += tulos+", ";
```

```
        }
```

```
    }
```

```
</script>
```

Olio-ohjelmointi

Object-oriented programming offers a sustainable way to write spaghetti code. It lets you accrete programs as a series of patches

- Paul Graham, tietojenkäsittelyn tohtori, roskapostisuodatuksen kehittäjä

Olio-ohjelmointi (Object Oriented Programming, OOP)



JavaScript tukee olioajattelua, mutta ei ole täysi oliokieli.

Olio on *tavallaan* **moniarvoinen muuttuja**, joilla on **ominaisuuksia**, niiden **arvoja** ja **metodeja** (eli funktioita).

Olion tärkein ominaisuus on se, että siitä voidaan tehdä **useita kopioita helposti ja nopeasti**. Jokaisella oliolla on oma tilansa.

JavaScriptissa olio luodaan (alustetaan)

```
let auto =  
  {malli: "lada",  
    ika: 7.5,  
    hinta: function() {return 10000/this.ika}}
```

- ← olion nimi
- ← olion 1. ominaisuus ja sen arvo
- ← olion 2. ominaisuus ja sen arvo
- ← olion 3. ominaisuus, joka on metodi

Isompi esimerkki

Hauki on olio.

Hauki-olion ominaisuudet (muuttujat) ovat

- ikä (ika),**
- syvyys (syvyys) ja**
- evien määrä (evat).**

Hauki-oliolla on metodeja (sisäisiä funktioita), jotka ovat:

- kerro ikä,**
- kerro evien määrä,**
- kerro nopeus,**
- sukella ja**
- nouse pintaan.**

```
let hauki = {  
  ika: 2,  
  syvyys: 3,  
  evat: 6,  
  
  kerrolka: function() {return ika},  
  kerroEvat: function() {return evat},  
  kerroNopeus: function() {return evat*10},  
  sukella: function() {this.syvyys =  
    this.syvyys -1},  
  nousePintaan: function() {this.syvyys = 0}  
}
```

Olio voidaan luoda ainakin ”kolmella” tapaa

1. ”Muuttujaluenti”

(olio literaali, object literal)

```
let olio1 = {  
  ominaisuus1: arvo1,  
  ominaisuus2: arvo2,  
  ominaisuus3: function() {return tulos1}}
```

2. ”Olioluonti”

(piste notaatio, dot notation)

```
let olio1 = new Object();  
olio1.ominaisuus1 = arvo1;  
olio1.ominaisuus2 = arvo2;  
olio1.ominaisuus3 = function() {return tulos1}
```

3. Konstruktori

(helppo käyttää monen olion luomiseen)

```
function olio(ominaisuus1,ominaisuus2){  
  this.ominaisuus1 = arvo1;  
  this.ominaisuus2 = arvo2;  
  this.ominaisuus3 = function() {return tulos1}}  
let olio1 = new olio(arvo1,arvo2);
```

← tämä on konstruktori

← tämä on luontitapahtuma (konstruktorin aktivointi)

Tehtävä 1. Olion luonti ja käsittely

Luo olio ja anna sille arvoja.

Luo lemmikki-olio.

Anna lemmikki oliolle ainakin 5 järkevää muuttujaa ja ainakin 2 metodia.

Testaa, että olio toimii

Kirjoita ~~HTML, CSS~~ ja JS yhteen tiedostoon.

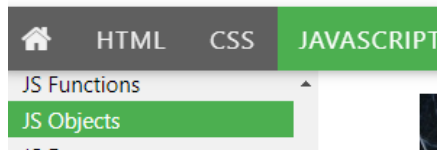
Aikaa tehtävien tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

~~Lisää tehtävä omaan
portfoliosivuusi.~~

Jatketaan kello 9.25

w3schools.com



Vastaus:

//Luodaan lemmikki olio. 5 ominaisuutta ja 2 metodia.

```
let lemmikki = {nimi: "rekku",  
  laji: "koira",  
  rotu: "mäyris",  
  ika: 7.5,  
  paino: 4.2,  
  aantele: function() {return "Wuf wuf!"},  
  syo: function(a) {this.paino = this.paino + a/2}  
}
```

//Tutkitaan ja testataan oliota

lemmikki.syo(0.5)

console.log(lemmikki.nimi + " painaa nyt " + lemmikki.paino + " kiloa.")

Tehtävä 2. Usean olion luonti ja käsittely

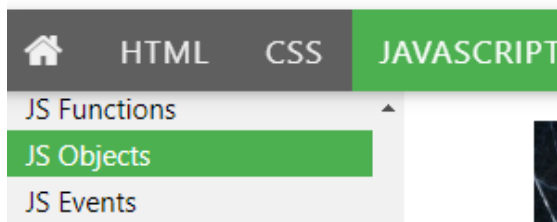
Luo olioita ja testaa niitä

Tee lemmikille **konstruktori**.

Tee ainakin kolme erilaista lemmikki-oliota.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

w3schools.com



Aikaa tehtävien tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 9.55

Vastaus:

//Luodaan lemmikki konstruktori. 5 ominaisuutta ja 2 metodia.

```
function lemmikki(name, species, race, age, weight){  
  this.nimi = name,  
  this.laji = species,  
  this.rotu = race,  
  this.ika = age,  
  this.paino = weight,  
  this.aantele = function() {return "Wuf wuf!"},  
  this.syo = function(a) {this.paino = this.paino + a/2}  
}
```

//Luodaan konstruktorilla viisi lemmikkiä (koiraa)

```
let koiraYksi = new lemmikki("rekku", "koira", "mäyris", 7.4, 4.2)  
let koiraKaksi = new lemmikki("wuffe", "koira", "saksis", 1.5, 6.7)  
let koiraKolme = new lemmikki("musti", "koira", "sekarotuinen", 2.5, 5.6)
```

//Tutkitaan oliota

```
koiraYksi.syo(0.5)  
console.log(koiraYksi.nimi+" painaa nyt "+koiraYksi.paino+" kiloa.")  
console.log(koiraKaksi.nimi+" sanoo "+koiraKaksi.aantele())  
console.log(koiraKolme.nimi+" on rodultaan "+koiraKolme.rotu)
```

Taulukko (array)

On alkiojoukon luomisen ja käsittelyyn tarkoitettu **olio-tyyppi**.

Sen arvoihin viitataan kokonaisluvuilla ja sillä on useita valmiita metodeja.

Indeksi eli paikka

0

1

2

Huom. Pituus on kuitenkin 3

Se luodaan esim. **let kalat = ["ahven", "hauki", "kuha"];**

Console.log(kalat[1]);

← tulostaa tiedon "hauki"

Tärkeimpiä metodeja ovat mm:

length	pituus
push()	lisää uusi alkio loppuun
pop()	poista viimeinen alkio
unshift()	lisää uusi alkio alkuun
shift()	poista ensimmäinen alkio
toString()	kirjoittaa kaikki alkiot merkkijonoina, joissa pilkku erottimena

Tehtävä 3. Array olion luonti ja käsittely

Tulosta JS:n Array -oliolla asioita

Tee hedelmakori-taulukko. Tallenna taulukkoon ainakin 5 hedelmä-oliota. Yksi hedelmien ominaisuus on hinta.

Tee funktio, joka kertoo kuinka paljon koko hedelmakori maksaa.

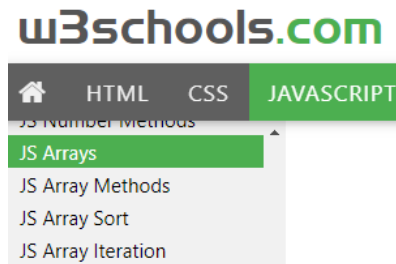
Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Aikaa tehtävien tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 12.10



Vastaus:

//Luodaan hedelmät. Yhteensä neljä hedelmä oliota, joissa on eri sisällöt.

```
let hedelma = {nimi: "banaani", hinta: 5}
```

```
let hedelmaKaksi = {nimi: "litsi", hinta: 8}
```

```
let hedelmaKolme = {nimi: "kumkvatti", hinta: 15}
```

```
let hedelmaNelja = {nimi: "turian", hinta: 12}
```

//Luodaan hedelmäkori taulukko (array)

```
let hedelmakori = [hedelma, hedelmaKaksi, hedelmaKolme, hedelmaNelja]
```

//console.log(hedelmakori[0].hinta)

Tätä voidaan käyttää suunnitteluun ja tarkistukseen.

//Lasketaan hedelmäkorin hinta funktiolla.

```
function laskeHinta(taulukko){
```

```
  let tulos = 0
```

```
  for(let i = 0; i < taulukko.length; i++){
```

```
    tulos = tulos + taulukko[i].hinta
```

```
  }
```

```
  return tulos
```

```
}
```

//Funktio ajetaan

```
console.log("Hedelmäkorin hinta on kokonaisuudessa "+ laskeHinta(hedelmakori) +" euroa.")
```

Lisää aiheesta

Merkittävä laajennus aiheeseen ja käsitteisiin.

30 minuutin pikakurssi olio-ohjelmoinnista (youtube)

<https://www.freecodecamp.org/news/object-oriented-programming-crash-course/>

ViOpe tehtävä.

JavaScriptin yhdistäminen sivuun

Tee Viopen JavaScript kurssista
Osan 3 tehtävät tehtävät 1 - 3

PROGRAMMING

1. 3-1: Elementin sisällön muokkaaminen
2. 3-2: Funktion kutsu HTML-sivulta
3. 3-3: Elementin luominen

Aikaa tehtävien tekemiseen 20 minuuttia

Jatketaan huomenna

Ei palautusta.

Bonus: Tee osan 3 monivalintatehtävät

Kertaa: lue aiheen teoria itseksesi

Osaamistavoite: 4. päivä

- Osaa käyttää Visual Studio Codea koodaamiseen
- Osaa muokata verkkosivun HTML-elementtejä JavaScriptillä
- Osaa liittää JavaScriptiä eri tavoilla sivuun.
- Osaa lukea ja kirjoittaa sivulle JavaScriptillä
- Osaa aktivoida tapahtuman onclick-ominaisuudella
- On tutustunut DOM malliin

Aiheiden lisäopiskelut ja harjoitukset

Olioista ja taulukoista

ViOpe tehtävä. Olioita ja taulukkoja

Tee Viopen JavaScript kurssista
Osan 5 tehtävät tehtävät 1 - 3

PROGRAMMING

1. 5-1: Oliot

2. 5-2: Taulukon sisältö

3. 5-3: Taulukon metodeja

Aikaa tehtävien tekemiseen 20 minuuttia

Jatketaan kello 13.30

Ei palautusta.

Bonus: Tee osan 5 monivalintatehtävät

Kertaa: lue aiheen teoria itseksesi

Lisää olioista

Olioiden määrittely

https://www.w3schools.com/js/js_objects.asp

this. –sanan käytöstä (erityisesti olioiden kanssa)

https://www.w3schools.com/js/js_this.asp

Oliota voidaan käyttää esim. tietokantayhteyden luomiseen

Tietokanta yhteyden luominen ja hallinta olion avulla

```
const mysql = require("mysql");
```

```
let conn = mysql.createConnection({  
    host: "localhost",  
    user: "käyttäjätunnus",  
    password: "salasana"  
});
```

← localhost-serveri, muista portti

← tietokannan käyttäjätunnus

← tietokannan salasana

Lisää aiheesta:

https://www.w3schools.com/nodejs/nodejs_mysql.asp

<https://www.hamrodev.com/en/tutorials/storing-objects-database-javascript-webservice>

Päivä 6.

Olio-ohjelmointi

Pieni kertaus taulukosta (array)

Regular Expression

Päivän aiheet ja tavoitteet

- Regular expression vertailujen perusteet ja rakentaminen
- Tapahtumanhallinta ja –kuuntelu
 - onclick
- Lukeminen ja kirjoittaminen HTML sivuun.
 - `querySelector()` ← viittaa ENSIMMÄISEEN id, class,.. ominaisuuden omaaviin
 - `querySelectorAll()` ← viittaa KAIKKIIN id, class,... ominaisuuden omaaviin

Materiaalia

PDF	s. 19
freeCodeCamp	oma osio tehtäviä
ViOpe	
W3school	<u>Regex</u>
Mozilla Dev	<u>Form validation</u> ,

freecodecamp.org: Array

Kirjaudu freecodecamp.com

- JavaScript Algorithms and Data Structures
 - Basic JavaScript

▼ JavaScript Algorithms and Data Structures Certification (300 hours)

▼ Basic JavaScript

Tee tehtävät

- ✔ Store Multiple Values in one Variable using JavaScript Arrays
- ✔ Nest one Array within Another Array
- ✔ Access Array Data with Indexes
- ✔ Modify Array Data With Indexes
- ✔ Access Multi-Dimensional Arrays With Indexes
- ✔ Manipulate Arrays With push()
- ✔ Manipulate Arrays With pop()
- ✔ Manipulate Arrays With shift()
- ✔ Manipulate Arrays With unshift()
- ✔ Shopping List

Aikaa tehtävien tekemiseen 30 minuuttia

Jatketaan huomenna

Ei palautusta.

freecodecamp.org: Object - oliot

Kirjaudu freecodecamp.com

- JavaScript Algorithms and Data Structures
 - Basic JavaScript

▼ JavaScript Algorithms and Data Structures Certification (300 hours)

▼ Basic JavaScript

Tee tehtävät

- ✓ Build JavaScript Objects
- ✓ Accessing Object Properties with Dot Notation
- ✓ Accessing Object Properties with Bracket Notation
- ✓ Accessing Object Properties with Variables
- ✓ Updating Object Properties
- ✓ Add New Properties to a JavaScript Object
- ✓ Delete Properties from a JavaScript Object
- ✓ Using Objects for Lookups
- ✓ Testing Objects for Properties
- ✓ Manipulating Complex Objects
- ✓ Accessing Nested Objects
- ✓ Accessing Nested Arrays
- ✓ Record Collection

Aikaa tehtävien tekemiseen 30 minuuttia

Jatketaan huomenna

Ei palautusta.



Regular expression

Give a man a regular expression and he'll match a string...
teach him to make his own regular expressions and you've got a man with problems.



Regular Expression

- Regular Expression (RegExp) eli **säännöllinen lauseke** on ohjelmoinnissa merkkien ja/tai merkkijonojen etsintään käytetty olio.
- Sen avulla voidaan asettaa vaatimuksia hyväksyttävälle merkkijonolle.
- Olio sisältää valmiita metodeja, jotka on tärkeä tuntea, jotta sen käyttö on luontevaa. Sekä tietenkin oikea tapa kirjoittaa (syntaksi) ehdot ja käskyt.

```
let merkkijono= "Tämä merkkijono tutkitaan";  
merkkijono.search(/tämä/i);
```

Tutustu aiheeseen myös [Wikipediassa](#). [Suomeksi](#).

RegExp

Olion metodeja

search(/etsi/i)

replace(/etsi/)

(**huom** merkkijonolla on samoja metodeja)

palauttaa merkkijonon alkupaikan (**case-insensitive**)

korvaa merkkijonon (**case-sensitive, täsmälleen sama**)

Vain regexpillä

test()

palauttaa tosi, jos merkki löytyy

palauttaa epätosi, jos merkkiä ei löydy

exec()

palauttaa merkkijonon oliona, jos löytyy.

palauttaa null, jos ei löydy

RegExp



Haun muokkaus (modifier)

- i **case-insensitive**, vertailu tapahtuu piittaamatta **merkkiriippuvuudesta**
(iso kirjain == pieni kirjain)
- g **globaali vertailu**, eli käydään läpi **koko merkkijono**. (Esim. `/[Aa]nd/g`)
Muuten valitaan vain ensimmäinen toteutuma
- m monen rivin haku

Merkkijonon valinta (sulkeet)

- `[abc]` etsii minkä tahansa merkin sulkeiden sisältä a, b tai c
- `[^abc]` etsii minkä tahansa merkin, joka EI OLE sulkeiden sisällä EI a, b tai c
- `(a|bc)` etsii | erottimen vaihtoehdot a TAI bc
- `[a-e]` etsii merkit jotka ovat välillä a ja e
- `[A-E]` etsii merkit jotka ovat välillä A ja E

Lisäksi käytössä on mm. metakirjaimia/ryhmiä

Ryhmistä ja metakirjaimista käyttökelpoisia ovat mm.

- . Mikä tahansa yksi kirjain
- \w Mikä tahansa kirjain
- \W Mikä tahansa **EI** kirjain
- \d Mikä tahansa numero
- \D Mikä tahansa **EI** numero

Ryhmistä ja määrittelyistä käyttökelpoisia ovat mm.

- n**⁺ Joukko jossa ainakin yksi **n**
- n**^{*} Joukko, jossa ei yhtään tai useampi **n**
- n**? Joukko, jossa 0 tai yksi **n**
- ^n** Joukko **n**, joka on ALUSSA
- n**\$ Joukko **n**, joka on LOPUSSA
- n**{8} Joukko **n**, jonka pituus tasan 8 merkkiä
- n**{3,5} Joukko **n**, jonka pituus 3-5 merkkiä

Kaksi tapaa rakentaa lauseke

Poissulkien

- Helpompi tapa, jossa suljetaan pois/kielletään tietyt merkit ja merkkijonot.
- Pitää etukäteen tietää kaikki tapaukset, jotka halutaan hylätä.
- Vaikeampi testata.
- Sopii pääasiassa yksinkertaisiin tilanteisiin.

Merkkijonossa saa olla vain kirjaimia. → Ratkaisu: poissuljetaan numerot.

Sallien ainoastaan

- Tehokkaampi tapa, jossa sallitaan vain tietyt merkit ja merkkijonot sekä niiden järjestys.
- Pitää etukäteen tietää kaikki tapaukset, jotka halutaan sallia.
- Helpompi testata.
- Sopii monimutkaisiinkin tilanteisiin, mutta vaatii aina tarkkuutta ja vaivaa.

Merkkijonossa saa olla vain kirjaimia. → Ratkaisu: sallitaan vain isot ja pienet kirjaimet.

Tehtävä 1. RegEx

Tee oheinen tehtävänanto. Voit tehdä tehtäviä myös JSBinillä.

Tutki onko merkkijonossa "Tämä on varsin pitkä merkkijono, ja siinä on eri asioita." vain kirjaimia.

Tee ehtolause, joka kertoo onko merkkijonossa vain kirjaimia.

Vai myös muita merkkejä?

Bonus: Korvaa numerot kirjaimilla.

Vinkki: käytä match-metodia, myös test-metodi on hyvä

Aikaa tehtävän tekemiseen 10 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 13.38

Vastaus

```
let merkkijono = "Tämä on varsin pitkä merkkijono";
```

```
let tulos = "";
```

```
if(merkkijono.match(/^[A-Öa-ö ]+$/)){  
    tulos = "Merkkijonossa on vain kirjaimia."  
}else{  
    tulos = "Merkkijonossa ei ole pelkästään kirjaimia."  
}
```

```
console.log(tulos);
```

Mitä äskeisen tehtävän regexp tekee?



/^[A-Öa-ö]+\$/

lyhyesti:

hyväksyy merkkijonon, jonka pituus on ainakin kaksi merkkiä, vain isoja ja pieniä kirjaimia sekä välilyönnit.

^

merkkijonon aloittajamerkki (merkitys voi muuttua paikan mukaan)

\$

merkkijonon päättäjamerkki

[A-Öa-ö]

merkkijonossa saa olla mitä tahansa merkkejä, jotka ovat:

välistä A – Ö (isot kirjaimet scandinavisesti)

välistä a – ö (pienet kirjaimet scandinavisesti)

(välilyönti eli space)

+

edeltävästä elementistä yksi tai useampi toteutuma

Vielä vähän tarkemmin

Tehtävä 2. RegExp

Tehdään tehtäviä W3resources ympäristössä. Voit tehdä tehtäviä myös JSBinillä.

Osiossa **Validation with Regular Expression** on **21 tehtävää**. Tee tehtäviä niin paljon kuin ehdit.

Tehtävät:

<https://www.w3resource.com/javascript-exercises/javascript-regexp-exercises.php>

Käytetään aikaa tehtävien tekemiseen noin 30 minuuttia.

Käydään jotain tehtäviä yhdessä läpi, jotta aihe kirkastuu.

Jatketaan kello 10.30

HTML5:n oma ”merkkijonojen” tarkistus



HTML5 sisältää myös oman sisäänrakennetun **merkkijonojen tarkistuksensa (validation)**. Se on osa form- ja input-tageihin liittyviä määrittelyitä.

Sen avulla voidaan:

- Vaatia kenttä täytetyksi
- Vaatia minimejää ja maximejää
- Vaatia kenttään tietyn tyyppinen teksti
- Vaatia kenttään tietynlaisen merkkijono

ominaisuus

- (required)
- (minlength, maxlength, min, max)
- (type="??")
- (pattern="??")

Validointia (**:valid**) korostetaan usein CSS-muotoiluilla (kuten tekstin tai elementin väri) ja kuvilla (kuten punainen rasti tai peukku ylös)

Lisäksi validointia tuetaan **value-ominaisuudella** ja/tai **placeholder-ominaisuudella** tarjotulla esimerkillä

Tehtävä 3. HTML5 validaatio

Tee oheinen tehtävänanto.

Toteuta lomake, jossa pyydetään ja valitoidaan ainakin:

- Sähköpostiosoite
- Luotokorttinumero
- Merkkijono, jonka pitää loppua "&a"

Käytä ainakin **required**, **type** ja **pattern** ominaisuuksia.

Bonus: Muotoile valitut elementit punaisiksi, kun validointi ei ole oikein. Muotoile elementit vihreiksi, kun muotoilu on oikein.

Aikaa tehtävän tekemiseen 30 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 13.00

freecodecamp.com: Regular Expressions

Kirjaudu freecodecamp.com

- JavaScript Algorithms and Data Structures
 - Regular Expressions

▼ Regular Expressions

○ 0/33

- Introduction to the Regular Expression Challenges
 - Using the Test Method
 - Match Literal Strings
 - Match a Literal String with Different Possibilities
 - Ignore Case While Matching
 - Extract Matches
 - Find More Than the First Match
 - Match Anything with Wildcard Period
 - Match Single Character with Multiple Possibilities
 - Match Letters of the Alphabet
 - Match Numbers and Letters of the Alphabet
 - Match Single Characters Not Specified
 - Match Characters that Occur One or More Times
 - Match Characters that Occur Zero or More Times
 - Find Characters with Lazy Matching
 - Find One or More Criminals in a Hunt
 - Match Beginning String Patterns
 - Match Ending String Patterns
 - Match All Letters and Numbers
 - Match Everything But Letters and Numbers

Aika-arvio tehtävien tekemiseen 2 tuntia.

Tehkää nyt noin 30 minuuttia.

Lopetellaan 14.00

Ei palautusta.

Osaamistavoite: 6. päivä

- Ymmärtää perusteet JavaScriptin olioista, olio-muuttujista ja niiden käytöstä.
- Ymmärtää olio-ohjelmoinnin tarkoituksen ja idean.
- Osaa tehdä olion ja antaa sille ominaisuuksia.
- Osaa tehdä olion eri tavoilla.
- Ymmärtää konstruktorin idean.
- Osaa tehdä taulukko-olion ja varastoida sinne muuttujia sekä olioita.
- Ymmärtää perusteet JavaScriptin regular expression vertailuista ja niiden käytöstä.
- Osaa käyttää HTML5 validointiominaisuuksia.
- Osaa tehdä monipuolisen lomakkeen palveluun rekisteröitymistä varten.

Päivä 7.

Ohjauspäivä Taitotalolla

Päivä 8.

Sivulle kirjoittaminen

Sivulta lukeminen

Sivun muokkaaminen

Tapahtumat (hallinta ja kuuntelu)

Kertausta verkkosivun käsittelyyn

Kertaus

Kertaus 1

1. Hedelmäoliot ja hedelmäkori taulukko
2. Hinnan laskeminen kaikista hedelmäolioista

Kertaus 2

1. Määrän lisääminen hedelmäolioihin
2. Ja uuden hinta-funktion käyttäminen

Kertaus/demo

- Yksinkertaisen hedelmäkorin muokkaaminen merkkijonoilla (push, pop, shift, unshift)
- Sort- is fun? Mitä tapahtuu, jos järjestetään numeroita sort-metodilla?

```
var arr = ["Rasmus", "ACDC", "CD", "Abba", "Metallica"]
```

```
arr.sort()
```

```
Tulos → ["Abba", "ACDC", "CD", "Metallica", "Rasmus"]
```

```
var arrKaksi = [9, 80, 100001, 110, 222, 345, 34000]
```

```
arr.sort()
```

```
Tulos → [100001, 110, 222, 34000, 345, 80, 9]
```

Tehtävä 0. Kertaus + lisää ominaisuuksia hedelmäkoriin

Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua

Muistele edellistä hedelmäkoria.

Toteuta kolme oliota, joilla nimi ja hinta.

Toteuta funktio, joka saa luvun. Arpoo luvun monta kertaa oliot taulukkoon. Palauttaa täytetyn taulukon.

Toteuta funktio, joka saa taulukon. Laskee taulukon olioista hinta muuttujat yhteen. Palauttaa tuloksen.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Vinkki: Math-olio, random ja floor

Ylimääräinen array rakenne helpottaa

Aikaa tehtävien tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 10.10

ONGELMA
miten välttää circular object

Päivän aiheet ja tavoitteet

- Tapahtumanhallinta ja –kuuntelu
 - Laajennetaan osaamista onclick + muuta
- Lukeminen ja kirjoittaminen HTML sivuun.
 - CSS style muokkaus
 - `querySelector()` ← viittaa ENSIMMÄISEEN id, class,.. ominaisuuden omaaviin
 - `querySelectorAll()` ← viittaa KAIKKIIN id, class,... ominaisuuden omaaviin
Usean elementin lukeminen arrayksi

Materiaalia

PDF

[freeCodeCamp](#)

[ViOpe](#)

[W3school](#)

[Mozilla Dev](#)

[Event Listener](#),

Päivän aiheet ja tavoitteet

- Kertaus esimerkein:
 - Olio-ohjelmointi, teoriaa ja käsitteitä.
 - Taulukot (array)
- Tapahtumanhallinta ja –kuuntelu
 - onclick
- Lukeminen ja kirjoittaminen HTML sivuun.
 - `querySelector()` ← viittaa ENSIMMÄISEEN id, class,.. ominaisuuden omaaviin
 - `querySelectorAll()` ← viittaa KAIKKIIN id, class,... ominaisuuden omaaviin

Materiaalia

PDF	-
freeCodeCamp	-
ViOpe	-
W3school	-

JavaScriptillä HTML:n ja sisällön muokkaus

//Valitse testi-elementti. Kirjoita niiden tagi JA sisältö uudestaan
`document.getElementById("testi").outerHTML = "<h5>Testi</h5>";`

Tämä ylikirjoittaa HTML-elementin koko sisällön.

JavaScriptillä CSS:n muokkaus

//Värin asettaminen tulosta-elementtiin. Kirjoittaa style-arvoja.

```
document.getElementById("tulosta").style.color = "green";
```

Lisää aiheesta:

https://www.w3schools.com/Jsref/dom_obj_style.asp

JavaScriptillä CSS:n muokkaus

//Värin asettaminen tulosta-elementtiin. Kirjoittaa style-arvoja.

```
document.getElementById("tulosta").style.color = "green";
```

//Taustavärin asettaminen tulosta-elementtiin. Kirjoittaa style-arvoja.

```
document.getElementById("tulosta").style.backgroundColor = "red";
```

Lisää aiheesta:

https://www.w3schools.com/Jsref/dom_obj_style.asp

JavaScriptillä CSS:n muokkaus

//Määrittele elementti, johon muutokset kohdistuvat

```
const elem = document.querySelector("#kohde");
```

//Määrittele halutut tyylimuokkaukset

```
const tyyli = {  
  color: "white",  
  border: "3px solid black"  
};
```

//Yhdistä tyylit kerralla kohteeseen

```
Object.assign(elem.style, tyyli);
```

Elementin etsiminen querySelectorilla

Tehokkaampi ja helpompi tapa etsiä elementtejä

`querySelector("#id");`

← kuin `getElementById("id");`

Palauttaa ensimmäisen esiintymän

`querySelectorAll(".class");`

← sisältää useita olioita, vaatii läpikäynnin
silmukan avulla

Palauttaa KOKOELMAN lapsielementtejä

Viittauksen elementtiin, id:hen tai luokkaan pitää olla samoin kuin CSS:sä.

Elementille **elem-nimi**, luokalle **.class** ja identiteetille **#id**

Lisää aiheesta

[W3school querySelector\(\)](#)

[W3school querySelectorAll\(\)](#)

Tehtävä 1. Lomakkeen tiedonhaku, tarkistus, tulostus sekä muotoilu.

Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua

Lue lomakekentästä numero. Jos se on välillä $-10 + 10$, tulosta **ok ja numero**. Tulosta **liian pieni** tai **liian suuri** muissa tapauksissa.

Väritä annettu luku vihreällä.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Bonus: Värjää vain luku

Aikaa tehtävien tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 9.40

Vastaus:

```
function vertaaJaMuotoile(){
```

//Tallenna muuttuja. Parsettaminen integeriksi (parseInt) ei ole pakollinen

```
let arvo = document.querySelector("#eka").value;
```

//Vertailu ja tulostaminen

```
if(-10 <= arvo && arvo <= 10){  
    document.getElementById("tulosta").innerHTML = "Tulos: just ok <p id='nro'>"+arvo+"</p>";  
}  
else if(arvo < -10){  
    document.getElementById("tulosta").innerHTML = "Tulos: liian pieni luku <p id='nro'>"+arvo+"</p>";  
}  
else if(arvo > 10){  
    document.getElementById("tulosta").innerHTML = "Tulos: liian suuri luku <p id='nro'>"+arvo+"</p>";  
}  
else{  
    document.getElementById("tulosta").innerHTML = "Jotain meni pieleen!" + num + " Onko tää edes luku?";  
}  
}
```

//Muotoilun lisääminen "lukuun" (eli koko kohtaan)

```
document.getElementById("nro").style.color = "green";  
//  
}
```

Vastaus pikkuisen paremmin

```
<script>
  function testaus(){
    let arvo = document.querySelector("#annettuNimi").value
    let tulos
    let vari
```

```
    if(arvo < -10){
      tulos = "liian pieni!"
      vari = "blue"
```

Alle sovitun rajan
olevat tapaukset

```
    }else if(arvo >= -10 && arvo <= 10){
      tulos = "ok. Annoit arvon "+arvo
      vari = "green"
```

Sovitun rajan
tapaukset

```
    }else if(arvo > 10){
      tulos = "Liian suuri!"
      vari = "red"
```

Yli sovitun rajan
olevat tapaukset

```
    }else{
      tulos = "Syötit jotain muuta. Tsut tsut!"
      vari = "black"
```

Kaikki muut
outoudet

```
    document.querySelector("#lisaaNimi").innerHTML = tulos
    document.querySelector("#lisaaNimi").style.color = vari
```

```
  }
</script>
```


Tehtävä 2. Lomakkeen tiedonhaku, tarkistus, tulostus sekä muotoilu.

Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua

Lue kaksi numeroa lomakkeesta ja laske ne yhteen. Jos tulos on välillä 25-75, tulosta ok. Jos pienempi kuin 25, tulosta **liian pieni**. Jos suurempi kuin 75, tulosta **liian suuri**.

Muotoile CSS:llä teksti **siniseksi**, jos liian pieni. **Punaiseksi**, jos liian suuri.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Aikaa tehtävien tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 10.55

Tehtävä 3. Tulostus sekä muotoilu.

Tulosta JS:llä HTML-elementtiin tekstiä, muotoilua ja silmukka

Aikaa tehtävien tekemiseen 20 minuuttia

Toteuta sivulle nappi, josta käynnistyy silmukka, joka käy läpi luvut 1-10. Jokaiselle luvulle kirjoitetaan "Tässä on luku x."

Palauta tehtävä opettajalle Teams-keskusteluun

Muotoile jokainen tekstifontin koko +3.

Lisää tehtävä omaan portfoliosivuusi.

Kirjoita HTML,CSS ja JS yhteen tiedostoon.

Jatketaan kello 12.30

Bonus: lisää myös jokin muu muuttuva muotoilu.

Seuraavassa tarvitsee

...muokata väriarvoja joustavasti.

...esittää vain osa arvoista

...tehdä arvojen tarkistus

Tehtävä 4. Lomakkeen tiedonhaku, tarkistus, tulostus, silmukka sekä muotoilu.

Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua

Lue kaksi numeroa. Ensimmäisen pitää olla väliltä 20-100. Ja toisen väliltä 2-5.

Tee silmukka, joka tulostaa nollasta alkaen luvut ensimmäiseen annettuun lukuun asti. Luvut tulostetaan toisen annetun luvun välein.

Tarkista, että arvot ovat annetuissa rajoissa.

esim. Syötteet 24 ja 3

Tuloste olisi 0, 3, 6, 9, 12, ...24

Bonus: Muotoile CSS:llä jokainen teksti pikkuisen vaaleammaksi aloittaen mustasta.

Aikaa tehtävien tekemiseen 20 minuuttia

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 13.35

Tapahtumien kuuntelu

addEventListener on kehittyneempi tapa aktivoida funktioita HTML-sivulla. Sen avulla useita tapahtumia voidaan kohdistaa samaan elementtiin.

```
kohde.addEventListener("tapahtuma", funktio, vapaaehtoinenKaappaus);
```

Esimerkki

```
elementti.addEventListener("click", function() {alert("Moikka!"); });
```



Tapahtumien kuuntelun tyypit

`addEventListener("tyyppi", funktio);`

<code>click</code>	<code>onclick</code>
<code>mousedown</code>	<code>onmousedown</code>
<code>mouseup</code>	<code>onmouseup</code>
<code>mouseover</code>	<code>onmouseover</code>
<code>mouseout</code>	<code>onmouseout</code>

<code>change</code>	<code>onchange</code>
<code>load</code>	<code>onload</code>
<code>unload</code>	<code>onunload</code>

Vinkkejä: Event types [Mozilla](#)

Tee sivu, jossa käytät **jokaista** kuuntelijaa. Käytä myös ainakin kahta kuuntelijaa yhteen elementtiin.

Tarkasta miten kuuntelijatyypin pitää kirjoittaa.

Kirjoita sivulle miten click ja onclick eroavat toisistaan.

Aikaa 30 minuuttia.

Jatketaan kello 9.30

Palauta tehtävä Teamsiin opettajalle.
Tai lisää omaan portfolio-sivuusi

Tehtävä 1. Taulukko (array)

Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua

Luo taulukko, jossa on 5 kpl numeroita.
Kerro taulukon kaikki numerot 3:lla. Tulosta uudet numerot, kun painetaan nappia.
Kirjoita luvut <p> tagin sisään.

Bonus: Käytä forEach()ia.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Aikaa tehtävän tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 13:45

Perinteinen silmukka-ratkaisu

```
<body>
```

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>JS Bin</title>
</head>
<body>

  Tässä on tulostusalue.
  <button id="nappi">Kerro</button>
  <p id="tulosta">Tähän tulostus</p>

</body>
</html>
```

JavaScript ▾

```
var taulukko = [1, 2, 3, 4, 5];

var nappi = document.querySelector("#nappi");
var tulosta = document.querySelector("#tulosta");

//Tapahtumankuuntelija
nappi.addEventListener("click",toimi);

//Funktio joka kertoo kaikki taulukon luvut kolmella
//ja tulostaa ratkaisun haluttuun elementtiin
function toimi(){
  for(var i = 0; i<taulukko.length; i++){
    taulukko[i] = taulukko[i] * 3;
  }
  tulosta.innerHTML = taulukko.toString();
}
```

```
</script>
```

```
</body>
```


Vastaus: yksinkertainen forEach() - ratkaisu

JavaScript ▾

```
let arr = [1,3,333,-8,3643461]
```

```
arr.forEach(kerro)
```

```
function kerro(item, index){  
  arr[index] = arr[index] * 3  
}
```

```
console.log(arr.toString())
```

Vastaus: forEach() ratkaisu

```
<body>

<h1>forEach()-toteutus taulukon muokkaamiselle</h1>

<button id="btn">Toteuta</button>

<p>Joka kerta kun nappia painetaan niin taulukko mutatoidaan ja tulostetaan uudelleen</p>

<p id="vastaus"></p>

<script>

  //HTML elementtien tuonti olioksi JS:ään

  const nappi = document.querySelector("#btn")

  const vastausAlue = document.querySelector("#vastaus")

  //Alusta taulukko ja näytä se

  const arr = [5, 10, 30, -10, 0]

  vastausAlue.innerHTML = `${arr}<br>`

  //Napin tapahtumankuuntelija

  nappi.addEventListener("click", napinToiminta)

  //Funktio napinToiminta, joka käynnistää funktiot

  function napinToiminta(){

    //Mutatoi arrayta forEach()-metodilla

    arr.forEach(kerronta)

    //Kirjoita verkkosivulle

    vastausAlue.innerHTML += `${arr}<br>`

  }

  //Kerronta funktio joka saa parametrinsa suoraan forEachin kautta taulukosta

  function kerronta(item, index){

    arr[index] = item * 3

  }

}
```

Vastaus: täydellisempi forEach()-ratkaisu

```
<body>

  <button id="btn">Kirjoita rivejä</button>

  <p id="tulostaVastaus"></p>

<script>

  const arr = [5, 10, 15, 30, 50]

  //Luodaan nappia varten elementti ja kuuntelija
  const nappi = document.querySelector("#btn")
  nappi.addEventListener("click", nimi)

  //Luetaan vastausalueelle elementti
  const vastausAlue = document.querySelector("#tulostaVastaus")

  //Napin käynnistämä funktio, joka muokkaa/mutatoi taulukon
  function nimi(){
    arr.forEach(kertoma);

    //Tulostetaan koko array sivulle
    vastausAlue.innerHTML = arr
  }

  //Varsinainen funktio joka suorittaa mutatoonnin
  function kertoma(item, index){
    arr[index] = arr[index] * 3
  }

</script>
```

Tehtävä 2. Taulukko (array)

Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua

Tee 5 peräkkäisen divin rakenne. Kirjoita jokaiseen elementtiin teksti.

Muuta jokaista diviä siten, että kaikkien taustaväri on punainen.

Anna jokaiselle elementille sama id tai class.

Käytä **querySelectorAll**:ia elementtien hakemiseen.

Bonus: Vaihda jokaiselle elementille eri väri.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Aikaa tehtävän tekemiseen 30 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 12.15

Tehtävä 3. Merkkijonon tarkistus ja muutos

Tulosta JS:llä HTML-elementtiin tekstiä

Tarkasta alkaako lomakkeen syötteen ensimmäinen kirjain isolla kirjaimella.

Jos ei, korjaa se alkavaksi isolla kirjaimella ja tulosta se sivulle <p> tagiin.

Bonus: Tulosta sama syöte alapuolelle myös **KOKONAAN ISOILLA** kirjaimilla kirjoitettuna.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.


Aikaa tehtävän tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 12.50

Ratkaisu – muutetaan eka kirjain aina isoksi

 File ▾ Add library Share

HTML CSS JavaScript Console Output

Login or Register Blog ¹ Help

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>JS Bin</title>
</head>
<body>
  <br><br>
  Anna tähän merkkijono isolla alkukirjaime
  <input id="lue" type="text"></input>
  <button id="nappi">Tarkista</button>
  <p id="tulosta">Tähän tulostus</p>
</body>
</html>
```

JavaScript ▾

```
var nappi = document.querySelector("#nappi");
nappi.addEventListener("click", toimi);

function toimi(){
  //Lukee lomakkeesta merkkijonon
  var merkkijono = document.querySelector("#lue").value;
  var tulos = "";

  //Jos eka kirjain pieni, muutetaan se isoksi.
  var eka = merkkijono.charAt(0);

  tulos = merkkijono.charAt(0).toUpperCase() + merkkijono.slice(1);

  //Tulosta
  document.querySelector("#tulosta").innerHTML = tulos;
}
```

Output

Run with JS Auto-run JS ☒

Anna tähän merkkijono isolla alkukirjaimella

Tarkista

Tähän tulostus

Ratkaisu - yksinkertaisempi

```
<body>
  <input id="syöte" type="text">
  <button onclick="testaus()">Testaa</button>
  <p id="vastaus">Vastaus tulee tähän</p>

  <script>
    function testaus(){
      var x = document.querySelector("#syöte").value
      if(x[0] === x[0].toUpperCase()){
        document.querySelector("#vastaus").innerHTML = "Syöte on ok. Ensimmäinen merkki on iso."
      }else{
        //Eka kirjain pieni
        x = x.replace(x[0],x[0].toUpperCase())
        document.querySelector("#vastaus").innerHTML = "Syöte on viallinen. Ensimmäinen merkki ei ole iso. " +x
      }
    }
  </script>
</body>
```

Tehtävä 4. Kumpi luku lähempänä?

Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua

Pyydä kaksi lukua lomakkeen avulla ja tarkasta kumpi on lähempänä lukua 1000. Tulosta lähempi `<p>` tagin sisään.

Mitä jos luvut ovat samat? Jääkö muitakin tapahtumia käsittelemättä?

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Aikaa tehtävän tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 13.40

Ratkaisu

```
<body>
  <input id="syöte1" type="number">
  <input id="syöte2" type="number">
  <button onclick="testaus()">Testaa</button>
  <p id="vastaus">Vastaus tulee tähän</p>

  <script>
    function testaus(){
      var x = 1000 - parseInt(document.querySelector("#syöte1").value)
      var y = 1000 - parseInt(document.querySelector("#syöte2").value)
      var tulos = ""
      if(x < y){
        tulos = "Ensimmäinen luku on lähempänä tuhatta"
      }else if(y < x){
        tulos = "Toinen luku on lähempänä tuhatta"
      }else{
        tulos = "Luvut ovat sama"
      }
      document.querySelector("#vastaus").innerHTML = tulos;
    }
  </script>
```

Tehtävä 4. Arvaa numero

Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua

Nappia painamalla (Funktio) arpoo satunnaisen numero väliltä 0-100.

Käyttäjä voi arvailla numeroa syöttämällä sen sivulle. Toisen napin painallus tarkastaa sen.

Mikäli arvaus on oikein, ilmoitetaan käyttäjälle selkeästi, että vastaus on löytynyt sekä kuinka monta yritystä siihen meni.

Mikäli vastaus on väärin, käyttäjälle kerrotaan kuinka mones arvaus oli sekä oliko numero arvatun luvun yli vai ali.

Kirjoita HTML, CSS ja JS yhteen tiedostoon

Bonus: väri vihjeet, lopussa katoava nappi.

Aikaa tehtävän tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 12.05

Vastaus:

```
* <body>
*
*   <div>
*
*     <div>
*
*       <form>
*
*         <button id="btnKaksi" type="button">Arvo uusi</button><br>
*
*         <label>Arvaa lukua 1-100 välillä: </label><br>
*
*         <input type="number" id="lueYksi" min=0 max=100><br>
*
*         <button id="btn" type="button">Testaa arvaus</button>
*
*       </form>
*
*     </div>
*
*   <div>
*
*     <p>Käyttäjän vastaus:</p>
*
*     <p id="vastaus">Tähän tulee vastaus</p>
*
*   </div>
*
* </div>
*
* <script>
*
*   //Tapahtuman kuuntelijat
*
*   const kuunteleArvontaa = document.querySelector("#btnKaksi")
*
*   kuunteleArvontaa.addEventListener("click", arvoUusi)
*
*   const kuunteleArvaus = document.querySelector("#btn")
*
*   kuunteleArvaus.addEventListener("click", uusiArvaus)
*
*
*   //Määritellään "globaali" muuttuja, johon talletetaan arvattava luku
*
*   let arvattava
```

Tehtävä x. Vertailua

Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua

Lue lomakkeesta kaksi numeroa. Jos jompikumpi numeroista on 50, tai numeroiden summa on 50, kirjoita <p>tagiin teksti true, muuten false.

Kirjoita HTML,CSS ja JS yhteen tiedostoon.

Aikaa tehtävän tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 12.05

Tehtävä x. Olio

Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua

Luo lomake, joka luo **auto-olio**.

Sen ominaisuuksia ovat väri, paino ja omistaja.

Sillä on funktio autoTiedot, joka tulostaa "Auton omistaja on omistaja. Se painaa paino ja väriltään väri.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Aikaa tehtävän tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 12.05

Osaamistavoite: 8. päivä

- Tutustutaan tarkemmin verkkosivujen muokkaukseen
- `querySelector`in käyttämisen oppiminen
- Harjoitellaan verkkosivujen muokkausta
- Tärkeimmät työkalut ja oliot tulevat tutuiksi
- DOM mallin sisäistämistä
- Osaa käyttää myös muita funktioiden aktivointeja kuin `onclick`iä ja `click`iä
- Tapahtumankuuntelijat

Päivä x - lisäjutut.

EcmaScript 6 lisäykset

Muutama looginen kokonaisuus

Päivän aiheet ja tavoitteet

- Kertausta ja vinkkejä
- Isompi kokonaisuus verkkosivulle
- Peli

Materiaalia

PDF

freeCodeCamp

ViOpe

W3school

Mozilla Dev

Tutustu freeCodeCampin artikkeliin

5 JavaScript tips That'll Help You Save Time

Lue artikkeli

<https://www.freecodecamp.org/news/time-saving-javascript-tips/>

Tutustu viiteen vinkkiin ja testaa niistä ainakin yhtä.

Käytä esim. JSBiniä

Aikaa tehtävien tekemiseen 30 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun. Tai kokouksen yleiseen keskusteluun.

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 8.50

Merge ...

- Ennen kahden tai useamman olion yhdistäminen tehtiin `Object.assign(uusi, vanha1, vanha2,...)` metodilla.
- Nyt yhdistäminen voidaan tehdä heti luonnissa.
- ```
const uusi = {
 ...vanha1,
 ...vanha2
}
```
- Lopputulos on molemmissa **muuttujien** ja niiden **arvojen** kopiointi uuteen olioon.



# Ehtolause (ternary operation)

"Kolmivaihe" operaatio on tapa kirjoittaa if-else rakenteita nopeammin ja yksinkertaisemmin.

//Ehto ? Jos totta : Jos epätosi

```
let tulos = vertailtava >= 50 ? "yli
puolet" : "alle puolet"
```

```
Console.log(`tulos on ${tulos}`)
```

- Voidaan kirjoittaa myös haastavampia ja pidempiä if-else rakenteita.

```
let arvosana = koetulos >= 95 ? 5
: koetulos >= 90 ? 4
: koetulos >= 80 ? 3
: koetulos >= 70 ? 2
: koetulos >= 60 ? 1
: 0
```

```
console.log(`Arvosanasi on
${arvosana}`)
```

# Tehtävä A. Ternary operaatio

**Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua**

Aikaa tehtävien tekemiseen 10 minuuttia

Toteuta tämä koodi ternary rakenteella

```
let ikä = window.prompt("Anna ikäsi")
if(ikä < 18){
 console.log("Olet alaikäinen")
}else{
 console.log("Olet täysi-ikäinen")
}
```

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 12.30

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

# Tehtävä 0. Olio

**Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua**

Luo lomake, joka luo **auto-olio**.

Sen ominaisuuksia ovat ainakin väri, paino ja omistaja.

Sillä on funktio **autoTiedot**, joka tulostaa "Auton omistaja on omistaja. Se painaa paino ja väriltään väri."

**Bonus:** voit tehdä myös napin, joka tulostaa funktion tulosteen sivulle.

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

Aikaa tehtävän tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 9.50

# Yksi vastaus

```
<form>
 <label>Omistaja:</label>
 <input type="text" id="eka" placeholder="Jussi">

 <label>Paino:</label>
 <input type="text" id="toka" placeholder="1500">

 <label>Väri:</label>
 <input type="text" id="kolmas" placeholder="Musta">

</form>
<button onclick="teeAuto()">Auto</button>
<div id="tulostus">Tähän vastaus</div>
```

```
<script>
 //Konstruktori autolle
 function auto(omistaja, paino, vari){
 this.omistaja = omistaja
 this.paino = paino
 this.vari = vari
 this.autoTiedot = function(){
 return "Auton omistaja on "+omistaja+". Se painaa "+paino+" kiloa. Ja sen väri on "+vari+"."
 }
 }

 function teeAuto(){
 var omistaja = document.querySelector("#eka").value
 var paino = document.querySelector("#toka").value
 var vari = document.querySelector("#kolmas").value

 var uusiAuto = new auto(omistaja, paino, vari)

 document.querySelector("#tulostus").innerHTML = uusiAuto.autoTiedot()
 }
</script>
```

# Tehtävä 1. Arvauspeli

**Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua**

Toteutua numerokenttä ja kaksi nappia.

Ensimmäinen arpoo numeron 0-100 välillä.

Syötä toiseen kenttään arvauksesi, ja nappia painamalla saat tiedon oliko valitsemasi numero suurempi vai pienempi. Kirjoita annettu numero, ~~väritä se~~ ja sen suuruudesta kertova tieto sivulle.

Numerokenttään voi antaa vain lukuja välillä 0 - 100.

**Bonus:** Kaikki arvatut luvut jäävät sivulle näkyviin.

**Bonus2:** Käyttäjälle näytetään kuinka monta kertaa on arvattu.

**Bonus3:** Onnistunut arvaus aktivoi onnitteluanimaation.

Kirjoita HTML,CSS ja JS yhteen tiedostoon.

Aikaa tehtävien tekemiseen 40 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 12.30

# Arvauspelin alustava ratkaisu

```
<form>
 <label>Tähän arvaus: </label>
 <input type="text" id="arvaus">
</form>
<button onclick="uusiPeli()">Arvo uusi numero</button>
<button onclick="testaa()">Tarkista arvaus</button>
<div id="tulosta">Tähän tulos</div>
```

```
<script>
 var arvattava;

 function uusiPeli(){
 arvattava = Math.floor(Math.random()*101)
 }

 function testaa(){
 var arvattu = document.querySelector("#arvaus").value

 let tulos = "";
 if(arvattu > arvattava){
 tulos = "hakemasi luku on pienempi"
 }else if(arvattu < arvattava){
 tulos = "hakemasi luku on suurempi"
 }else if(arvattu == arvattava){
 tulos = arvattu+" on "+arvattava+" eli hakemasi luku. Onnittelut!"
 }else{
 tulos = "Jotain meni pieleen."
 }

 //Tulostus
 document.querySelector("#tulosta").innerHTML = tulos
 }
</script>
```



# Tehtävä 2. 1337 5p34k generaattori

**Tulosta JS:llä HTML-elementtiin tekstiä ja muotoilua**

Toteutua tekstikenttä ja nappi, joiden avulla annettu merkkijono muutetaan leet speakiksi.

Jos et tunne leet speak:iä, niin googleta.

**Vinkki:** tee ensin sellainen funktio joka kääntää vain yhden kirjaimen.

Esim i:t korvataan 1-merkillä.

Kirjoita HTML,CSS ja JS yhteen tiedostoon.

Aikaa tehtävien tekemiseen 20 minuuttia

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

Jatketaan kello 13.40

Ja käydään silloin vaiheet 1-4

# Pilkotaan tehtävä helpompiin osatehtäviin

- Oma toteutus: Verkkosivutoteutus = nappi ja tekstikenttä
- Varsinainen funktio on aika ”monimutkainen”. Siinä paljon tapauksia, joten yksinkertaistetaan sitä.
  - Tulkitaan/oletus tehtävänantoa siten, että leet speakissä korvataan osa kirjaimista numeroilla.
  - Toteutetaan aluksi sellainen funktio, joka muuttaa vain yksittäisen kirjaimen numeroksi

**Tehtävä tee funktio, joka saa merkkijonon josta kaikki e-kirjaimet korvataan 3:lla.**

Palautukset joko yleiseen keskusteluun tai suoraan mulle.

Aikaa 15 minuuttia.

Jatketaan kello 12.05

# Leet speak generaattorin osittaisratkaisu

Versio 1. muuttaa osan kirjaimista numeroiksi, ei huomioida verkkosivu vielä

Luodaan funktio joka muuttaa annetun merkkijonon leet speakiksi

Saadaan merkkijono nimeltä merkkis

Käydään merkkis merkki kerralla läpi (for-silmukka)

vertailu: jos kirjain voidaan muuttaa numeroksi,

tehdään näin

Palautetaan muutettu merkkis-merkkijono

# Tehtävä 3. Uuden elementin liittäminen sivuun

**Tulosta JS:llä uusi HTML-elementti sivulle ja muokkaa sitä.**

Aikaa tehtävien tekemiseen 40 minuuttia

Toteuta sivulle pari elementtiä ja yksi nappi. Painamalla nappia sivulle luodaan uusi nappi toiseen sijaintiin.

Palauta tehtävä opettajalle Teams-keskusteluun

**Bonus:** Anna uudelle napille jokin funktio siten, että käyttäjä voi nyt muokata jotain muutakin.

Lisää tehtävä omaan portfoliosivuusi.

Esim. "tuhota" eka nappi

Jatketaan kello 14.00

Kirjoita HTML, CSS ja JS yhteen tiedostoon.

# Päivä 9.

*ES6 kertausta*

*React tausta ja tarkoitus*

*React käytön perusteet*

# Kertaus: nuolifunktiot

## Anonyymit nuolifunktiot

```
//Parametritön
() => a+b
```

```
//Parametrillinen
(a,b) => a+b
```

”Kertakäyttöisiä.”

## Nimetyt nuolifunktiot

```
//Nimetään ja luodaan funktio
let kerro = (a,b) => console.log(a*b)
```

```
//Kutsutaan fnktio
kerro(5,6)
```

Uudelleenkäytettäviä.

# Uusi asia: callback funktiot

Funktio, joka siirtyy toiseen funktioon parametrina. Hyödynnetään usein osana asynkronista ohjelmointia.

```
function tervehdys(nimi) {
 alert(`Moikka ${nimi}`);
}
```

Erittäin käytettyjä.

**Tehtävä:** kirjoita tämä uusiksi nuolifunktiolla. Edes yksi funktio.

Aikaa 15 minuuttia

```
function syötteenLukeminen(takaisinkutsu) {
 let nimi = prompt('Kerro nimesi');
 takaisinkutsu(nimi);
}
```

Palauta kello 10.05 mennessä opettajan Teams-keskusteluun

```
syötteenLukeminen(tervehdys);
```

# Demo: muokataan edellistä funktiota ”siistimmäksi”

```
function tervehdys(nimi) {
 alert(`Moikka ${nimi}`);
}
```

```
let lukeminen = (takaisinkutsu) => {
 let nimi = prompt('Kerro nimesi');
 takaisinkutsu(nimi);
};
```

```
lukeminen(tervehdys)
```





# Ehtolause: kolmivaihe operaatio - ternary operation

"Kolmivaihe" operaatio on tapa kirjoittaa if-else rakenteita nopeammin ja yksinkertaisemmin.

//Ehto ? Jos totta : Jos epätosi

```
let tulos = vertailtava >= 50 ? "yli
puolet" : "alle puolet"
```

```
console.log(`tulos on ${tulos}`)
```

- Voidaan kirjoittaa myös haastavampia ja pidempiä if-else rakenteita.

```
let arvosana = koetulos >= 95 ? 5
: koetulos >= 90 ? 4
: koetulos >= 80 ? 3
: koetulos >= 70 ? 2
: koetulos >= 60 ? 1
: 0
```

```
console.log(`Arvosanasi on
${arvosana}`)
```

# Tehtävä A. Ternary operaatio

**Voit käyttää Jsbinia. Tai VSC.**

Aikaa tehtävien tekemiseen 10 minuuttia

Toteuta tämä koodi ternary rakenteella

```
let ikä = window.prompt("Anna ikäsi")
if(ikä < 18){
 console.log("Olet alaikäinen")
}else{
 console.log("Olet täysi-ikäinen")
}
```

Palauta tehtävä opettajalle Teams-keskusteluun

Lisää tehtävä omaan portfoliosivuusi.

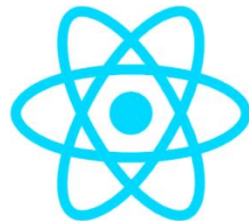
Jatketaan kello 10:50

**Bonus:** Lisää muitakin ikäryhmiä.  
Esim. taapero, lapsi, nuori, aikuinen, vanhus,...



# Mikä on React?

- React on avoimen lähdekoodin JavaScript kirjasto, joka auttaa käyttöliittymien (UI) toteuttamisessa.
  - Facebookin kehittämä 2011
- Koostuu komponenteista, jotka ovat helposti käytettäviä ja opittavia.
- React kirjoitetaan JSX-kielellä, joka on JavaScriptin ja HTML:n sekoitus. Tämä käännetään puhtaaksi JavaScriptiksi ja HTML:ksi.
- Reactin oppiminen sisältää
  - npm (Node Package Manager)
  - Komponenttiarkkitehtuurin
  - Modernin JavaScriptin ymmärtäminen (ES6)
    - Let, const, literaalit, nuolifunktiot, ternary operaattori, array metodit,...



# Miksi React?

- Avoin, ilmainen
- Helppokäyttöinen
- Yksinkertainen kehitysympäristö
- Itsenäinen, uudelleenkäytettävä, nopea

## Yksinkertainen kehitysympäristö

- Node ja sen npx/npm
- Selain (Chrome)
- Visual Studio Code (editori)
- React Developer Tools
- Komentokehoite/terminaali

# Mikä on MERN (pino/stack)?

- Yhteisnimi suositellulle ”teknologiapinolle” (tech stack), jonka avulla voit rakentaa koko palvelun; serverin tietokannasta verkkosivun käyttöliittymään.
- MERN tulee teknologioista
  - **MongoDB** (tietokanta)
  - **Express.js** (yhteys palvelimeen HTTP-pyyntöjen kautta)
  - **ReactJS** (Front End:in rakentaminen selaimeen)
  - **Node.js** (Back End palvelin puoli)

# Huomioita React JavaScriptin kirjoittamisesta

## Perusohjeita JavaScriptin ja Reactin käyttämiseen ([freeCodeCamp](https://www.freecodecamp.org/))

1. Käytä vain `let` ja `const` muuttujia. Älä `var` muuttujaa.
2. Suosi nuolifunktiota lyhyissä funktioissa.  
Suosi ternary operaatiota lyhyissä vertailuissa.
3. Literaalit. Käytä template literaaleja muuttujien käsittelyyn ja olio literaalia olioiden luomiseen. Muuttujien käyttäminen lähempänä luonnollista kieltä. ``${userName}, olet tervetullut ${serviceName}-palveluumme``
4. Moduulien tuominen ja vieminen (`import` ja `export`)

# Muita suosittuja kehityskirjastoja Front Endille?

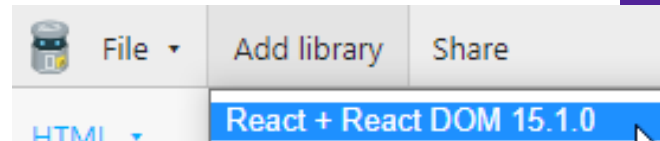
**Angular 2** (ei siis AngularJS) on TypeScript pohjainen mobiili- ja verkkokehityskirjasto.

- Vähän eriävä toimintafilosofia verrattuna muihin suosittuihin kirjastoihin.
- Angularin teknologiapino tunnetaan nimellä **MEAN** (MongoDB, Express.js, Angular, Node.js)

**Vue.js** on JavaScript pohjainen käyttöliittymien ja yhden sivun verkkosovellusten kehittämiskirjasto.

- Vue:n teknologiapino on vähemmän tunnettu, mutta joskus käytetään lyhennettä **MEVN**

# React ajaminen JSbinissä



1. Avaa HTML, JavaScript ja Output osiot

Lisää uusi JSX elementti ja renderöi se osaksi koodia.

2. Lataa React + ReactDOM 15.1.0 kirjasto

```
const elementti = <h1>Ensimmäinen oma elementti</h1>
```

3. Käynnistä JavaScriptiin JSX (React)-kääntäjä

```
const kohta =
document.querySelector("#tässä")
```

```
ReactDOM.render(elementti, kohta)
```

*"root" ei jostain syystä toimi tässä ympäristössä?*



# Vastaus:



The screenshot shows a web development environment with three main panels. The left panel displays HTML code, the middle panel displays JSX (React) code, and the right panel displays the rendered output.

**HTML Panel:**

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width">
 <title>JS Bin</title>
</head>
<body>
 <div id="paikka"></div>
 <script src="https://fb.me/react-15.1.0.js"></script>
 <script src="https://fb.me/react-dom-15.1.0.js"></script>
</body>
</html>
```

**JSX (React) Panel:**

```
const elementti = <h1>Ensimmäinen oma</h1>
const kohta = document.getElementById("paikka")
ReactDOM.render(elementti, kohta)
```

**Output Panel:**

Ensimmäinen oma

# Tehtävä 1. JSX kirjoittaminen

Toteuta monimutkaisempi rakenne:

H1 "Teen pääluokat"

p "Tee voidaan jakaa esim. kuuteen pääluokkaan teeuutoksen värin perusteella. Tämäkään ei ole kuitenkaan täysin toimiva luokittelu."

Lista valkoinen tee, keltainen tee, vihreä tee, oolong tee, musta tee, tumma tee

**Bonus:** lisää joukkoon myös kuva

Aikaa tehtävien tekemiseen 15 minuuttia

Jatketaan kello 13.30

Palauta tehtävä opettajalle Teamsiin. Tai yleiseen keskusteluun.

**Teen pääluokat.**

Teen voidaan jakaa esim. kuuteen pääluokkaan teeuutoksen värin perusteella. Tämäkään ei ole kuitenkaan täysin toimiva luokittelu.

- valkoinen tee
- keltainen tee
- vihreä tee
- oolong tee
- musta tee
- tumma tee

# Kehyksen asentaminen

Asenna React

# Kehyksen asentaminen

1. Asennetaan node.js, jotta npm on käytettävissä.

<https://nodejs.org/en/>

Viimeisin LTS

Testaa, että **node -v** toimii kaikkialla!

2. Asenna paketti/Lataa paketti [reactjs.org](https://reactjs.org) –sivustolta

Valitse kansio minne haluat asentaa projektin

```
npx create-react-app mun-eka
```

```
cd mun-eka
```

```
npm start
```

3. Yhdistä paketti projektiin

- Kehyksen asentaminen tarkoittaa käytännössä uuden React projektin aloittamista.

- Aloittaminen tehdään npm käskyllä ja asennuksella.

# npm komennot projektin aikana

## npm start

Aja app:i kehitystilassa

## npm test

Aja testausseuraaja vuorovaikutustilassa

## npm run build

Rakenna ja optimoi app:i tuotantoa varten.

Nyt se on valmis

## Tehtävä 2. Node.js asentaminen

1. Lataa LTS asennuspaketti  
nodejs.org -sivustolta
2. Hyväksy kaikki paketin  
vaatimukset ja asenna se  
haluamaasi kansioon.
3. Varmista että Node toimii  
globaalisti avaamalla  
komentokehoite ja kirjoittamalla  
missä tahansa sijainnissa komento  
**node -v**

Aikaa tehtävien tekemiseen 30  
minuuttia

Jatketaan kello 11.55

Ei palautusta.

Jos asennus on onnistunut, node  
kertoo versionsa.

# freecodecamp.com: React

## Kirjaudu freecodecamp.com

- Front End Development Libraries
  - React

### React

React is a popular JavaScript library for building reusable, component-driven user interfaces for web pages or applications.

React combines HTML with JavaScript functionality into its own markup language called JSX. React also makes it easy to manage the flow of data throughout the application.

In this course, you'll learn how to create different React components, manage data in the form of state props, use different lifecycle methods like `componentDidMount`, and much more.

▼ Collapse courses

○ 22/47

- ✔ Create a Simple JSX Element
- ✔ Create a Complex JSX Element
- ✔ Add Comments in JSX
- ✔ Render HTML Elements to the DOM
- ✔ Define an HTML Class in JSX
- ✔ Learn About Self-Closing JSX Tags

Aika-arvio tehtävien tekemiseen 2 tuntia. Tee ainakin 11 tehtävää.

Tehkää nyt noin 30 minuuttia.

Lopetellaan 14.00

Ei palautusta.

# Päivä 10.

*Taitotalolla, hybridiopetusta*



# Päivä 11.

*React jatkoa*

*React projekti*

# Kertaus: käynnistä projekti

- Avaa projekti VSC:hen
  - Avaa projektista terminaali
- Käynnistä projekti uudelleen
  - Kirjoita komento npm start kun olet projektin kansiossa

# Omaa työskentelyä

- Tehkää app.js (ja muihin app.css) tiedostoihin muutoksia.
- Tehkää esim. lista tai kuva.
- Aikaa 15 minuuttia
- Jatketaan kello 9.45

# Käyttö CDN:n avulla

- Etsi uusin CDN linkki Reactjs.com sivulta

```
<script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
```

```
<script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
```

- Muista, että kun haluat julkaista projektin, niin nämä linkit pitää korjata
  - Korvaa **development** → **production** sanalla

# Tehtävä 1. Tutustu Reactin ohjeisiin

- Tutustu Reactin ohjesivuun
- Etsi ohje CDN:n käyttöönotolle
- Tee yksinkertainen HTML sivu, jolle luot ainakin yhden React komponentin.
  - Samanlainen kuin se meidän ensimmäinen
  - `Const jsx = <h1>Otsikko</h1>`
  - `Const paikka = ....`
  - `ReactDOM.render( )`
- Aikaa tehtävän tekemiseen 15 minuuttia
- Jatketaan kello 10.30
- Palauta kuvakaappaus/ratkaisusi opettajalle Teamsiin

HTML ja JS/JSX siis samassa tiedostossa.

# Vastaus:

```
<!-- Reactin lataus ja ReactDOM lataus-->
<script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
<!-- Vielä babel jotta react kääntyy-->
<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-standalone/6.26.0/babel.min.js"></script>

<!-- Oma React koodi-->
<script type="text/babel">
 const elementti = <p>Tähän tulee Reactin tekemää tekstiä</p>
 const paikka = document.querySelector("#paikka")

 ReactDOM.render(elementti,paikka)
```

# React projektin aloitus

- Kertausta



# Ohjelmointikäsité: Luokka

- **Luokka** on (olio-)ohjelmoinnissa käytetty olion tyyppi.
  - Tyylikkäämpi ja yksityiskohtaisempi kuin pelkkä olio.
- Luokkien tärkein piirre on **periytyminen**, jonka avulla koodia yksinkertaistetaan. Käytännössä luokka voi periä toisten luokkien ominaisuuksia.

## Esimerkki

Luokka on kuin pohjapiirros talolle, josta voidaan tehdä useita eri värisiä ja pintaisia fyysisiä taloja.

Esimerkki React luokasta, joka perii React.components luokan ominaisuudet

```
class tehoNappi extends React.Component {
 render() {
 return (
 <elementOrStuff />
)
 }
}
```



# Demo: staattinen luokka nappi

```
class Button extends
 React.Component {
 render() {
 return (
 <button>42</button>
);
 }
}
```

```
ReactDOM.render(<Button />,
 mountNode);
```



# Props ja State

- **Props (properties)** on tapa siirtää tietoa komponenttien välillä. Annetaan vain kerran projektin alussa.
  - Data siirtyy vain yhteen suuntaan. Read-only.
  - Props luodaan ja kohdistetaan liitettävään vanhempi-elementtiin. Parent → Child
  - Kohdistetaan komponentin uudella arvolla. `<Component arvo="">`
  - Toimii kuin funktio
- **State** on tietomalli, joka tulee muuttumaan.
  - Näissä hyödynnetään **useState** koukkuja (hooks).
  - Asynkroninen. Päivittää vain itsensä.
  - Aiemmin state toimi vain class (eli luokka) komponenteissa.
  - Tietoa voidaan luoda ja hallita.

# Props:in (eli ominaisuuden) luominen

## Luominen

Tarvitsee konstuktorin (tai muun starttaajan)

## Renderöinti

- tapahtuu return-lauseella, joka käyttää literaalimuotoa (dot notation)

# Demo: this ja props luokan kanssa

```
class Button extends React.Component {
 constructor(props) {
 super(props);
 this.state = { counter: 3 };
 }
 render() {
 return (
 <button>{this.state.counter}</button>
);
 }
}

const mountNode = document.querySelector("#paikka")
ReactDOM.render(<Button />, mountNode);
```

# State:n (eli tilan) luominen

# Demo: this ja state luokan kanssa

## Mitä tarvitaan?

- Tuodaan useState koukku
- Luodaan tilaa ylläpitävät muuttujat
- Tehdään nappi
- Ja lisätään siihen laskurifunktio

```
class Button extends React.Component {
 state = { counter: 1 };

 handleClick = () => {
 this.setState({
 counter: this.state.counter + 1
 });
 };

 render() {
 return (
 <button onClick={this.handleClick}>{this.state.counter}</button>
);
 }
}

const mountNode = document.querySelector("#paikka")
ReactDOM.render(<Button />, mountNode);
```

# Tehtävä 1. Kahden napin tilamuutokset

- Toteuta kahdella napilla siten, että nappi – poistaa yhden arvon ja nappi + lisää yhden arvon.
- Nappien välissä on elementti mihin tila tulee näkyviin.
- Toteuta Reactilla miten haluat (app tai CDN)
- Aikaa tehtävän tekemiseen 30 minuuttia
- Jatketaan kello 12.30
- Palauta kuvakaappaus/ratkaisusi opettajalle Teamsiin



# Tauko

- Jatketaan 13.15



## Tehtävä 2: Kuvan liittäminen

- Etsi netistä ilmainen kuva.
- Liitä kuva polun avulla verkkosivuun käyttäen React elementtiä.
- Aikaa tehtävän tekemiseen 15 minuuttia
- Jatketaan kello 8.30
- Palauta kuvakaappaus/ratkaisusi opettajalle Teamsiin

# Tehtävä 3. Kolme nappia

Tee samaan tehtävään kolme nappia, joista jokainen ylläpitää omaa laskuriaan.

Bonus:

Aikaa tehtävien tekemiseen 30 minuuttia

Jatketaan kello 11.55

Palauta kuvakaappaus tai koodi opettajalle Teamsiin. Tai yleiseen keskusteluun

**Demo: luodaan ehto, joka kirjoittaa eri asian.**

# Demo: Reac dev toolsin käyttö

- Tutki projektia react dev toolsilla.
  - Avaa tarkastelutila ja sieltä **Components** ja **States**

# freecodecamp.com: React

## Kirjaudu freecodecamp.com

- Front End Development Libraries
  - React

### React

React is a popular JavaScript library for building reusable, component-driven user interfaces for web pages or applications.

React combines HTML with JavaScript functionality into its own markup language called JSX. React also makes it easy to manage the flow of data throughout the application.

In this course, you'll learn how to create different React components, manage data in the form of state props, use different lifecycle methods like `componentDidMount`, and much more.

▼ Collapse courses

○ 22/47

- ✔ Create a Simple JSX Element
- ✔ Create a Complex JSX Element
- ✔ Add Comments in JSX
- ✔ Render HTML Elements to the DOM
- ✔ Define an HTML Class in JSX
- ✔ Learn About Self-Closing JSX Tags

Aika-arvio tehtävien tekemiseen 2 tuntia. Tee ainakin 11 tehtävää.

Tehkää nyt noin 30 minuuttia.

Lopetellaan 14.00

Ei palautusta.

# Päivä 12.

*React jatkoa*

*React projekti*



# Koukut (hooks)

- Rajoitteet:
  - Toimii vain function-komponentin sisällä.
  - Pitää aina kutsua samassa järjestyksessä. Esim. ei voi olla if-lauseessa.

## Esim. useState on yksi koukuista

Palauttaa aina taulukon, jossa on kaksi arvoa. [nykyinen tila, tilan päivittävä funktio]

```
const [laskuri, laskeLaskuria] = useState()
```

# Tehtävä koukku. Katso video ja tee perässä

- Katso video ja tee perässä
  - [Video \(15 min\)](#)
- Toteuta kahdella napilla toimiva laskuri. Toinen nappi vähentää, toinen lisää.
- Keskity siihen miten koukku (hook) muuttaa koodin kirjoittamista ja muuttujan käyttämistä.

Aikaa tehtävien tekemiseen 30 minuuttia

Jatketaan kello 13.50

Palauta kuvakaappaus tai koodi opettajalle Teamsiin. Tai yleiseen keskusteluun





# Lomakkeen teko Reactilla

- Ohjeet yläreunassa



# Todo-lista Reactilla

- Luo uusi projekti  
npx create-react-app
- Kirjoita HTML-elementti

# Hyviä React projekteja

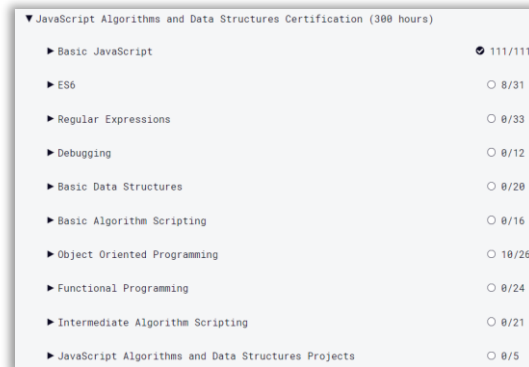
- <https://www.freecodecamp.org/news/how-to-make-2048-game-in-react/>

# Ylimääräistä asiaa

# freeCodeCamp – sertifiointi aiheesta

## Nimi: JavaScript Algorithms and Data Structures Certification

- Arvio suoritusajasta 300 tuntia. Ja voi osua aika lähelle.  
Sertifiointi on **TYÖLÄS**, mutta **opettavainen ja hyödyllinen**.
- Koostuu 10 osa-alueesta,
  - joissa yhteensä **299** tehtävää ja projektia
  - Sisältäen myös 5 projektia ja 26 + 24 + 21 ohjelmointihaastetta
- Kattaa varsin erinomaisesti JavaScriptin perusteet
- Vahva suositus itsenäiseen läpikäymiseen!



▼ JavaScript Algorithms and Data Structures Certification (300 hours)	
▶ Basic JavaScript	111/111
▶ ES6	8/31
▶ Regular Expressions	8/33
▶ Debugging	8/12
▶ Basic Data Structures	8/28
▶ Basic Algorithm Scripting	8/16
▶ Object Oriented Programming	18/26
▶ Functional Programming	8/24
▶ Intermediate Algorithm Scripting	8/21
▶ JavaScript Algorithms and Data Structures Projects	8/5

# Tee itsenäisesti ainakin

freeCodeCamp:in osioita

# freecodecamp.com: Data Structures

Kirjaudu freecodecamp.com

- JavaScript Algorithms and Data Structures
- Basic Data Structures

▼ Basic Data Structures

0/20

- Introduction to the Basic Data Structure Challenges
- Use an Array to Store a Collection of Data
- Access an Array's Contents Using Bracket Notation
- Add Items to an Array with push() and unshift()
- Remove Items from an Array with pop() and shift()
- Remove Items Using splice()
- Add Items Using splice()
- Copy Array Items Using slice()
- Copy an Array with the Spread Operator
- Combine Arrays with the Spread Operator
- Check For The Presence of an Element With indexOf()
- Iterate Through All an Array's Items Using For Loops
- Create complex multi-dimensional arrays
- Add Key-Value Pairs to JavaScript Objects
- Modify an Object Nested Within an Object
- Access Property Names with Bracket Notation
- Use the delete Keyword to Remove Object Properties
- Check if an Object has a Property
- Iterate Through the Keys of an Object with a for...in Statement
- Generate an Array of All Object Keys with Object.keys()
- Modify an Array Stored in an Object

Aikaa tehtävien tekemiseen x minuuttia

Ei palautusta.

# freecodecamp.com: ES6

Kirjaudu freecodecamp.com

- JavaScript Algorithms and Data Structures
  - ES6

Tee tehtävät 1-8

Aikaa tehtävien tekemiseen 30 minuuttia

Jatketaan huomenna

Ei palautusta.

▼ JavaScript Algorithms and Data Structures Certification (300 hours)

► Basic JavaScript

🕒 111/111

▼ ES6

🕒 8/31

- 🕒 Introduction to the ES6 Challenges
- 🕒 Explore Differences Between the var and let Keywords
- 🕒 Compare Scopes of the var and let Keywords
- 🕒 Declare a Read-Only Variable with the const Keyword
- 🕒 Mutate an Array Declared with const
- 🕒 Prevent Object Mutation
- 🕒 Use Arrow Functions to Write Concise Anonymous Functions
- 🕒 Write Arrow Functions with Parameters
- 🕒 Set Default Parameters for Your Functions



# freeCodeCamp – sertifiointi aiheesta

**Nimi:** Front End Development Libraries

Arvio suoritusaajasta 300 tuntia. Ja on jonkin verran vähemmän.

Sertifiointi on **TYÖLÄS**, mutta **opettavainen ja hyödyllinen**.

- Koostuu 7 osa-alueesta,
  - joissa yhteensä **132** tehtävää ja projektia
  - Sisältäen myös 5 projektia
  - Kattaa varsin erinomaisesti JavaScriptin perusteet
- Vahva suositus itsenäiseen läpikäymiseen!

## Front End Development Libraries



# Tee itsenäisesti ainakin

freeCodeCamp:in osioita

# freecodecamp.com: React

Kirjaudu freecodecamp.com  
Front End Development Libraries  
• React

▼ Collapse courses

○ 22/47

- ✔ Create a Simple JSX Element
- ✔ Create a Complex JSX Element
- ✔ Add Comments in JSX
- ✔ Render HTML Elements to the DOM
- ✔ Define an HTML Class in JSX
- ✔ Learn About Self-Closing JSX Tags
- ✔ Create a Stateless Functional Component
- ✔ Create a React Component
- ✔ Create a Component with Composition
- ✔ Use React to Render Nested Components
- ✔ Compose React Components

Aikaa tehtävien tekemiseen x  
minuuttia


Ei palautusta.

# Kertaus ja loppujutut

*Vielä vähän*

# Oppimistavoitteiden saavuttaminen

Aseta itsellesi oppimistavoite. Ja arvioi kurssin lopussa saavutitko sen. Kirjoita noin 5 asiaa, jotka haluat oppia/tietää/ymmärtää.



Tarkasta oppimistavoitteesi. Saitko tavoitettua ne? Vai jäikö jotain puuttumaan? Miten jatkat tästä eteepäin?

# Palaute kurssista



Vastaa kyselyyn. Aikaa menee noin 5 minuuttia.

**Kiitokset!**

Käytän vastauksia opetuksen kehittämiseen ja parantamiseen.

**Mukavaa päivänjatkoa!**

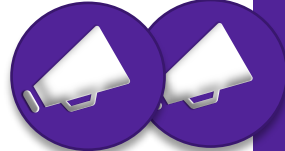
# Lisää vinkkejä ja apuja

*Oppiminen on jatkuva prosessi.*

*Perustutkinnossa käymme läpi vain perusasiat, ja suurin osa jää itseopiskeluksi.*

*Ole kiinnostunut, ole innostunut ja tutki itsenäisesti.*

*Osallistu kursseihin, webinaareihin ja verkostoihin. Kysy ja jaa.*



# Visual Studio Code ja sen lisäosat

Visual Studio Code sopii hyvin myös JavaScriptin koodaamiseen.

Hyviä lisäosia ovat:

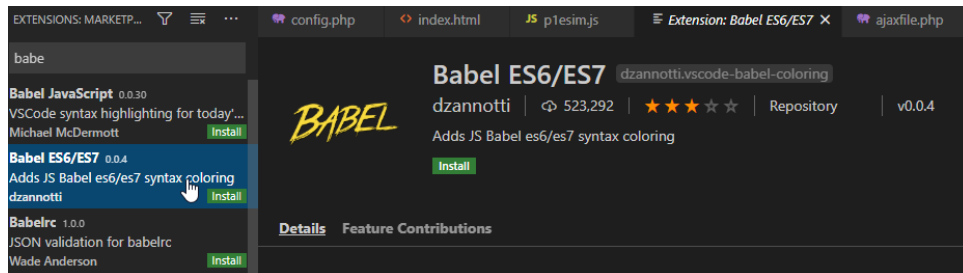
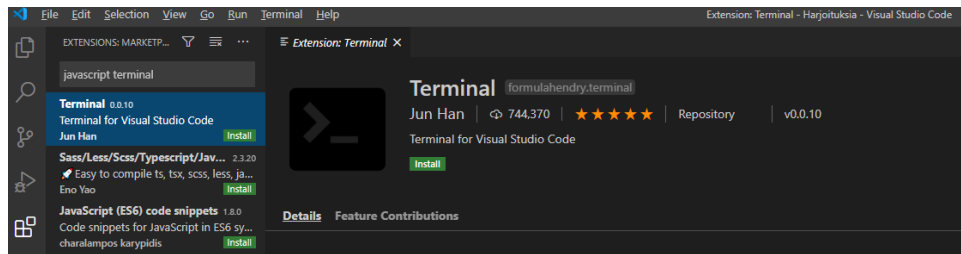
Babel	tuo väriykset ja muotoilut
Terminal	tuo komentokehötteen suoraan osaksi VSC:tä
Live Server	kääntää HTML-sivua suoraan selaimeen

Voit käyttää VSC:tä kahdella eri tavalla:

1. Asentaa Node.js:n, jolloin voit ajaa/testata JavaScriptiä suoraan komentokehöttestä.
2. Kirjoittaa JS osaksi verkkosivua, jolloin selain huolehtii sen kääntämisestä.



# VSC lisäosat



# Lisää opiskelua

## CodeAcademy

Ilmainen kurssi JavaScriptistä.

<https://www.codecademy.com/learn/introduction-to-javascript>

## Youtube

The Net Ninja –kanava. Useita 3- 15 minuutin videoita aiheesta.

<https://www.youtube.com/channel/UCW5YeuERMmlnqo4oq8vwUpg>

Traversy Media. JavaScript Crash Course for Beginners. (1h 40min)

<https://www.youtube.com/watch?v=hdl2bqOjy3c>

# Lisää opiskelua

## Coursera (ilmaisia, isoja kursseja)

[https://www.classcentral.com/course/html-css-javascript-for-web-developers-4270?utm\\_source=fcc\\_medium&utm\\_medium=web&utm\\_campaign=cs\\_programming\\_september\\_2020](https://www.classcentral.com/course/html-css-javascript-for-web-developers-4270?utm_source=fcc_medium&utm_medium=web&utm_campaign=cs_programming_september_2020)

[https://www.classcentral.com/course/javascript-jquery-json-9568?utm\\_source=fcc\\_medium&utm\\_medium=web&utm\\_campaign=cs\\_programming\\_september\\_2020](https://www.classcentral.com/course/javascript-jquery-json-9568?utm_source=fcc_medium&utm_medium=web&utm_campaign=cs_programming_september_2020)

## Ohjelmointikurssi (suomeksi, perusasiat, selkeä)

<https://ohjelmointikurssi.github.io/>

# JS projekteja

21 projektia ja niiden koodit

<https://skillcrush.com/blog/projects-you-can-do-with-javascript/>

freecodecamp (40 projektia aloittelijalle)

<https://www.freecodecamp.org/news/javascript-projects-for-beginners/>

freeCodeCamp (2 h, youtube)

[Accessible web app](#)

# Cheat Sheets ja muut vinkit

## Yleiset

Kaikki oliot: <https://overapi.com/javascript>

Perusteet + esimerkkejä: <https://htmlcheatsheet.com/js/>

Perusteet + selitykset: <https://websitesetup.org/javascript-cheat-sheet/>

Perusteet + piirroksia: <https://ilovecoding.org/blog/js-cheatsheet>

Tärkeimmät asiat: <https://www.codecademy.com/learn/introduction-to-javascript/modules/learn-javascript-introduction/cheatsheet>

## Regular Expression (regex)

<https://www.debuggex.com/cheatsheet/regex/javascript>

# Clean Code ja Best Practices

Hyvä koodi on selkeää, johdonmukaista ja kaunista. Huono ja sekava koodi haittaa kaikkia. Ammattilaisen on tärkeä tutustua hyvän koodin kirjoittamiseen ja tärkeimpiin käytäntöihin.

Lyhyt selostus	<a href="https://www.freecodecamp.org/news/clean-coding-for-beginners/"><u>https://www.freecodecamp.org/news/clean-coding-for-beginners/</u></a>
Parhaat käytännöt	<a href="https://www.w3.org/wiki/JavaScript_best_practices"><u>https://www.w3.org/wiki/JavaScript_best_practices</u></a>
Tyyliohje	<a href="https://google.github.io/styleguide/jsguide.html"><u>https://google.github.io/styleguide/jsguide.html</u></a>

# Tietorakenteista

Laaja johdatus erilaisiin tietorakenteisiin, joita ohjelmointikielissä käytetään.

freeCodeCamp – Data Structures (3 h, youtube-video)

<https://www.freecodecamp.org/news/learn-all-about-data-structures-used-in-computer-science/>

# Mitä kaikkia työkaluja Front End koodari tarvitsee?

Lue freeCodeCampin artikkeli aiheesta

- <https://www.freecodecamp.org/news/front-end-development-tools-you-should-know/>



# Muita työkaluja: Google Web Designer

Ilmainen ja asennettava laaja työkalu HTML5 ja JS sisällön tuottamiseen. Helpottaa monimutkaisten rakenteiden kehittämistä, tarjoamalla siihen visuaalisen työkalun.

Google Web Designer (ilmainen, ladattava, JS ”banner”työkalu)

<https://webdesigner.withgoogle.com/>

Download Web Designer

Google Web Designer



# Kuinka oppia ajattelemaan kuin koodaaja?

Algoritminen ajattelu

# Luentoja siitä miten koodaamista voi/kannattaa ajatella

Luento alkaa hitaasti, mutta siinä käydään läpi monta koodaamiseen liittymää väärinkäsitystä ja tärkeää asiaa. Erittäin hyödyllinen kuunnella.

[How to think like A Programmer](#) (1 h)

Uncle Bob Clean Code – luentosarja. Useita aiheita, vinkkejä ja esimerkkejä

[Clean Code – Uncle Bob](#) (~1,5 h per esitys)



# Loogiset virhepäätelmät

Ilmaisutaitoihin kytkeytyvät logiikan osa-alue.

Jakautuu kahteen osaan:

1. **Muodollisessa ongelma** on se **kuinka** rakennat väitteesi ja korostat asioita. Puhut totta, mutta rakennat väitteesi huonosti.
2. **Epämuodollisessa ongelma** on se **mitä väität** ja väitteesi on epätosi tai harhaanjohtava

Ohjelmistokehittäjän näkökulmasta on tärkeää tietää mitä ollaan tekemässä ja ymmärtää onko asia se mitä varsinaisesti halutaan.

→ asiakaspalvelu, ryhmätyö, esittäminen, keskustelu, asiakkaan aitojen tarpeiden määrittely,...

Myös ongelmanratkaisussa olennaisen ongelman löytäminen on joskus haastavaa. Usein keskitytään aivan vääriin asioihin.

→ ryhmätyö, projektityö, ongelmanratkaisu, ohjelmointi, tekninen ratkaisu,...

Oheisessa artikkelissa useita tärkeitä virhepäätelmiä, mitkä vaikuttavat helposti ajatteluumme sekä sen ilmaisemiseen.