

OHJELMOINNIN PERUSTEET

# JAVASCRIPT MUUTTUJAT JA NÄKYVYYDSALUE



**JS**

## **MUUTTUJAT**

Muuttujat ovat yksi ohjelmoinnin peruskäsitteistä ja ne ovat teille jo tuttuja Pythonista.

Muuttujaan voidaan tallettaa aina yksi tieto kerrallaan ja tiedon vaihtuessa edellinen arvo unohdetaan. Tätä arvoa voidaan käyttää monessa eri paikassa ja yleensä muuttujia on samassa ohjelmassa useita.

# NÄKYVYYSALUE JA LOHKOT

- Näkyvyysalue eli scope on kaikissa ohjelmointikielissä tärkeä käsite
- Määrittelee, missä luomasi **muuttujat** ovat olemassa ja mistä kaikkialta siihen päästään käsiksi.

- Lohkot ovat tärkeä osa näkyvyysalueita sillä ne rajaavat näkyvyyttä
- JavaScriptissä lohkojen määrittelyyn käytetään aaltosulkeita
- Aaltosulkeita käytetään mm. funktioissa, ehtolauseissa, silmukoissa ja olion

```
{ aaltosulkeet  
määrittelevät  
JavaScript  
lohkon  
}
```

```
Pythonissa  
sisennys  
määrittelee  
lohkon
```

# NÄKYVYYSALUEET

Javascriptissä on kolme eri tyyppistä näkyvyysalueetta:

## 1. Globaali näkyvyysalue

- Määritelty täysin aaltosulkujen ulkopuolella
- Muuttujat näkyvät kaikissa, myös funktioiden sisällä

## 2. Funktiokohtainen näkyvyysalue

- Määritelty funktion sisällä
- Muuttujat näkyvät vain kyseisen funktion sisällä

## 3. Lohokohtaisella näkyvyysalueella

- Määritelty muiden kuin funktiolle kuuluvien aaltosulkujen sisällä
- Näkyvät vain kyseisen lohkon sisällä
- Näkyvät myös sen alilohkoissa ja sen sisällä olevissa funktioissa

“—  
WHAT HAPPENS IN A BLOCK  
STAYS IN THE BLOCK

JS

# JAVASCRIPT MUUTTUJAT

**JavaScriptissä on kolme muuttujatyyppiä: const, let ja var.**

## **Const eli vakiomuuttuja (eng. constant)**

- Kun siihen on kerran laitettu arvo, tätä arvoa ei voi enää ylikirjoittaa
- Sopii hyvin asioille, joita ei halutakaan muuttaa kuten esimerkiksi piin ( $\pi$ ) lukuarvo
- Huom, jos teet esimerkiksi taulukon const-muuttujana, sen sisältöä voidaan silti muuttaa. Mutta sitä ei voida muuttaa taulukosta vaikkapa merkkijonoksi
- Näkyvyysalue on lohkokohdainen, funktiokohtainen tai globaali riippuen siitä, missä muuttuja on määritelty

```
const pi = 3.14159;
```

*pi* on globaali  
vakiomuuttuja

JS

# JAVASCRIPT MUUTTUJAT

## Let -muuttuja

- Tämän muuttujan arvoa ja sisällön tyyppiä voi vaihdella vielä jälkeenpäin
- Periaatteessa kaikki muuttujasi voivat olla tätä tyyppiä
- Voi olla lohkokohtainen, funktiokohtainen tai globaali, riippuen missä muuttuja on määritelty

```
if (onko == true) {  
  let viesti = "Tämä on lohkossa! Eikä viestiä pääse muualta muuttamaan";  
  console.log(viesti);  
}
```

*viesti* on lohkokohtainen muuttuja

JS

# JAVASCRIPT MUUTTUJAT

## Var -muuttuja

- Vanha muuttuja, jota ei kannata enää käyttää.
- Käytä tämän sijasta let-muuttujaa.
- Voi olla funktiokohtainen tai globaali.
- Ei tue lohkokohtaisuutta, mikä tuottaa ongelmia ja siksi sitä ei enää suositella.

```
var huonoEsimerkki = "Älä käytä näitä"
```

JS

# JAVASCRIPT AVAINSANAT

Nämä ovat varattuja sanoja, joita ei voi käyttää muuttujan nimenä

await	debugger	extends	import	package	static	void
break	default	false	in	private	this	while
case	delete	finally	instanceof	protected	throw	with
catch	do	for	interface	public	try	yield
class	else	function	let	return	True	
const	enum	if	new	super	typeof	
continue	export	implements	null	switch	var	

JS

# JAVASCRIPT SYNTAKSI

```
let x, y, z;    // Määritellään muuttujat
x = 5; y = 6;  // Annetaan muuttujalle arvo
z = x + y;    // Lasketaan muuttujan arvo
```

**Huom:** JavaScript on kirjainkoon tarkka (**case-sensitive**), eikä väliviivoja (-) sallita muuttujien nimissä.

**Esimerkiksi:**

**first-name** → **first\_name** tai **FirstName** tai **firstName**

JS

# GLOBAALIEN MUUTTUJIEN ONGELMA MIKSI NIITÄ KANNATTAA VÄLTTÄÄ

Globaalia muuttuja voi muokata missä tahansa vaiheessa koodia. Lyhyissä harjoituksissa, joita me alkuun teemme, se ei ole ongelma.

Projektien kasvaessa on hyvä, jos muuttuja voi muokata vain tietystä lohkossa tai funktionna, johon se kuuluu. Eikä vahingossa uudelleen rivillä 120.

Jos muuttujalle ei anna tyyppiä (const, let / var) se on automaattisesti globaali muuttuja, vaikka sen määrittelisi lohkon sisällä. Onneksi 'use strict' estää tämän.

