Aalto-yliopisto
Perustieteiden
korkeakoulu

# Verkkosivujen toiminnallisuus (JavaScript)

CS-C1180 Verkkojulkaisemisen perusteet (5 op)

Pauli Laine

2019

*”Luennon jälkeen osaan toteuttaa verkkosivuille toiminnallisuutta JavaScript:llä.”*

# Luennon sisältö

JavaScript
- Syntaksi, käyttö, liittyminen HTML:n
- JSON-tiedonsiirtoformaatti
- DOM API
- FETCH
- Verkkosivujen optimointi

Tuntitehtävät
Yhteenveto

# Luennon 26.3.2019 sisältö

Oliot
-harjoitus

Webstorage
-harjoitus

Git

Canvas
-harjoitus

# Ohjelmoinnista yleensä

Asioita pitää kokeilla ja testata, ohjelmointia ei opi lukemalla

Pulmien ratkaisukyvyn oppiminen olennaista:

- -mikä on ongelma
- -ongelman ositus
- -ratkaisu ja testaaminen osa-alue kerrallaan
- -virheiden etsintä (debuggaus)
    - -apuilmoitukset (alert ja console.log)
- -sinnikkyys

Perustietorakenteiden hallinta

- -muuttujat, listat, oliot,oliolistat

# Ohjelmoinnista yleensä

Tietokoneohjelma tekee asioita kolmella eri 'moodilla'

- -käyttäjän syötteestä - 'Eventit'
- -ohjelman funktio kutsuu toista funktiota
- -ajastetusti

Turvallisuussyistä (koska web), Javascriptillä EI voi tallentaa tietokoneen tiedostojärjestelmään.

- -webstorage
- -palvelinpään tallennus
  - -tietokanta
  - -jquery, xmlHTTPRequest
  - -Firebase

# JavaScript

Tulkattava ***skriptikieli***

Pääasiallinen käyttö ***selaimessa***

Verkkosivun ***toiminnallisuus***

***API***:en kautta pääsy selaimeen/laitteeseen

***Prototyyppipohjainen***, dynaamisesti tyypitetty, ...

# Rajoitukset

ESIM:

-ei voi tallentaa paikallisesti

-nykyisessä istunnossa olleet tiedot katoavat

-rajoituksia voi olla myös ulkopuolelta ladattaessa (Cross Origin Resource Sharing)

https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

-huomaa, että HTML <form>:lla on yleen oletustoiminto, jolloin se yrittää lähettää kaavakkeen palvelimen PHP:lle -> ei toimi niinkuin luulisi javascriptin kanssa

# Kummallisuuksia

-vältä listojen läpikäyntiä “for item in list”-tyyppisen konstruktion avulla

-Sulkeumien aiheuttama ‘palauttaa vain viimeisen arvon’ erikoisuus:

https://dzone.com/articles/why-does-javascript-loop-only-use-last-value

Korjautuu parhaiten “LET” muuttujamäärittelyllä

# Historia

## 1995-1996

Kehittäjä ***Brendan Eich***. Ensiesiintyminen Netscape Navigator 2.0:ssa.

## 1996-1998

Standardoitavaksi Ecma International:iin. Standardoitu nimellä ***ECMAScript***.

## 1998-2004

HTML-sivujen manipuloiminen DOM Level 1, 2 ja 3 API:ien avulla.

## 2001-2002

JSON-tiedonsiirtoformaatti. Kehittäjä ***Douglas Crockford***.

## 2005

Asynkroninen kommunikointi palvelimen kanssa, AJAX. Termin kehittäjä ***Jesse James Garrett***.

## 2006

Asiakaspään JavaScript-kirjastot, kuten jQuery (kehittäjä ***John Resig***) ja Prototype.

## 2008

HTML5 ja siihen liittyvät JavaScript API:t.

## 2009

Standardoitu palvelinpään JavaScript-kirjasto, CommonJS. Kehittäjä ***Kevin Dangoor***.

## 2015

ECMAScript 2015 class oliosyntaksi selkiyttää koodausta.

## Jatkuvasti…

**Kirjastot kehittyvät mm. mobiililaitteiden antureiden käyttöä varten:**

**https://developer.mozilla.org/en-US/docs/Web/API**

# Käyttö

**.js / <script>*

# JavaScript

```
7   window.onload     = function( event ) {
8     var myVariable  = "Hello World!";
9     window.alert( myVariable );
10  }
```

# KÄYTTÖTAVAT

1) Ulkoinen skriptimäärittely (*.js –tiedosto)
2) Sisäinen skriptimäärittely (<script> –elementti)

# ULKOINEN SKRIPTIMÄÄRITTELY

Skriptit voidaan määritellä ulkoisessa tiedostossa (***.js***). Ulkoinen skriptitiedosto voidaan linkittää verkkosivuun ja näin ollen samoja skriptimäärittelyitä voidaan käyttää usealla eri verkkosivulla (helpottaa ylläpitoa). Selaimet voivat myös tallentaa ulkoisen skriptitiedoston välimuistiin (nopeuttaa verkkosivujen lataamista).

***Suositeltavin käyttötapa.***

***Käyttäminen verkkosivuilla * (Suositeltu tapa HTML-tiedoston head-tagin sisällä)***

```
<script type=”text/javascript” src=”scripts.js”></script>
```



** Perinteisesti <head> –osion sisään*

# Ulkoinen skriptimäärittely

**HTML**

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8"/>
        <script type="text/javascript" src="test.js"></script>
    </head>
    <body>
        <h1>My Heading</h1>
    </body>
</html>
```

**JavaScript**

```javascript
// test.js

window.onload = function( event ) {
  var myVariable = "Hello World!";
  window.alert( myVariable );
}
```

# SISÄINEN SKRIPTIMÄÄRITTELY

Skriptit voidaan määritellä verkkosivun sisällä (***<script>***).

Sisäinen skriptimäärittely keskittää verkkosivun skriptimäärittelyt yhteen paikkaan (helpottaa hieman ylläpitoa). Samoja skriptimäärittelyjä ei kuitenkaan voi käyttää muilla verkkosivuilla eivätkä selaimet pysty tallentamaan niitä välimuistiin.

***Käyttäminen verkkosivuilla ****

<script type=”text/javascript”>...</script>



** Perinteisesti <head> –osion sisään*

## Sisäinen skriptimäärittely

HTML

**JavaScript**

HTML

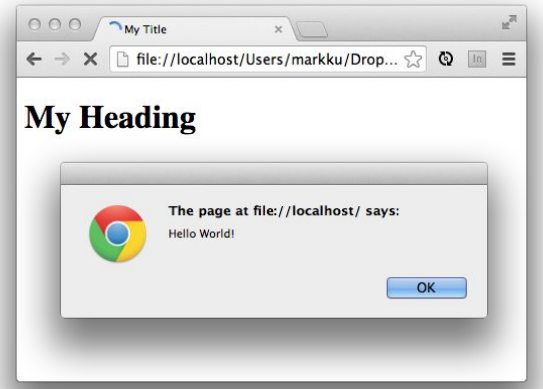
```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <title>My Title</title>
6      <script type="text/javascript">
7        window.onload     = function( event ) {
8          var myVariable  = "Hello World!";
9          window.alert( myVariable );
10       }
11     </script>
12   </head>
13   <body>
14     <h1>My Heading</h1>
15   </body>
16 </html>
```

HTML

JavaScript

HTML

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <title>My Title</title>
6      <script type="text/javascript">
7        window.onload     = function( event ) {
8          var myVariable  = "Hello World!";
9          window.alert( myVariable );
10       }
11     </script>
12   </head>
13   <body>
14     <h1>My Heading</h1>
15   </body>
16 </html>
```



# Syntaksin perusteet

# Muuttujat

- Paikalliset muuttujat määritellään *var* –sanalla, globaalit ilman. Nyrkkisääntö: käytä useimmiten paikallisia muuttujia.
  - Muuttujien näkyvyys
  - paikalliset muuttujat näkyvät esim. funktion sisällä
- Tyyppi jätetään määrittelemättä
  - Määrittyy *dynaamisesti*
- ```
  var courseCode = "ME-C2300"; // String
  var isWonderful = true; // Boolean
  ```
  - undefined, null, totuusarvo, merkkijono, numero, olio
  ```

# Primitiivityypit

- undefined
  - Ei määritelty
- null
  - Ei viittausta mihinkään
- Totuusarvo (boolean)
  - *true* tai *false*

# Primitiivityypit

- Numero
  - Kokonais- tai desimaaliluku
  - Laskuoperaattorit: +, -, *, /, jne.
  - isNaN( muuttuja ) testaa onko *luvaton* numero (Not-a-Number)
  - Merkkijonosta numeroksi

```
1   var string  = "1234";
2   var number  = Number( string );
```

# Primitiivityypit

- Merkkijono (string)
  - Arvo ympäröidään joko *lainausmerkein (”…”)* tai *puolilainausmerkein (’…’)*
  - Merkkijonojen yhdistäminen + –merkillä
  - Myös olio, jolla useita ominaisuuksia ja metodeita, esim. length
  - Numerosta merkkijonoksi

```
1  var number  = 1234;
2  var string  = number.toString();
```

  - http://www.quirksmode.org/js/strings.html

# Taulukko

- Taulukko (array)
  - Lista arvoja
    - Alkioiden arvot voivat olla eri ja/tai mitä tahansa tyyppiä
  - Taulukko on olio, jolla on useita ominaisuuksia ja metodeita, esim. length

Alkioiden arvojen käsittely indeksinumeroiden kautta

- Indeksointi alkaa *nollasta*

# Kolme tapaa luoda taulukko

```
1  // Method 1
2  var array1  = [ true, 1234, "String" ]; // Preferred
3
4  // Method 2
5  var array2  = new Array();
6  array2[ 0 ] = true;
7  array2[ 1 ] = 1234;
8  array2[ 2 ] = "String";
```

- var kissat = [ ]
- kissat.push("Miisu");

# FUNKTIO

- Funktio (function)
  - Lohko ohjelmakoodia, joka eriyttää *tietyn* ja/tai *usein suoritettavan* toiminnon
  - Määrittely koostuu *function* –sanasta sekä funktion nimestä, parametreista, ja paluuarvosta
    - Funktion nimi, parametrit ja paluuarvo ovat valinnaisia
  - Funktiot voivat myös olla muuttujien arvoina
  - Esimerkki

```
1  // String to number converter
2  function stringToNumber( stringToBeConverted ) {
3      var number  = stringToBeConverted * 1;
4      return number;
5  }
```

```
8  var stringToNumber = function( stringToBeConverted) {
9    /*...*/
10 }
```

# Muita tieto/oliotyyppejä

- Date
  - Päivämäärään ja aikaan liittyvät metodit
  - http://www.w3schools.com/jsref/jsref_obj_date.asp
- Math
  - Matemaattisiin operaatioihin liittyvät ominaisuudet ja metodit
  - http://www.w3schools.com/jsref/jsref_obj_math.asp
- RegExp
  - Säännöllisiin lausekkeisiin liittyvät toiminnot
  - http://www.w3schools.com/jsref/jsref_obj_regexp.asp

# Näkyvyys (scope)

- Muuttujat ilman *var* –sanaa *(global)* näkyvät kaikkialla, esim. funktion ulko- ja sisäpuolella
  - Ohjenuora: määrittele muuttujat aina *var* –sanaa käyttäen
- Funktion ulkopuolella *var* –sanalla määritellyt muuttujat näkyvät sekä funktion ulko- että sisäpuolella
- Funktion sisäpuolella *var* –sanalla määritellyt muuttujat näkyvät vain funktion sisäpuolella

```
1  var course = "ME-C2300";
2
3  var createCredits = function() {
4    var credits = 5;
5  }
6  createCredits();
7
8  console.log(course);  // ME-C2300
9  console.log(credits); // credits is not defined
```

```
11 var course = "ME-C2300";
12
13 var createCredits = function() {
14   credits = 5;
15 }
16 createCredits();
17
18 console.log(course);  // ME-C2300
19 console.log(credits); // 5
```

# Input

-Syötteen tarkastaminen (input validation)

-Siksi, että esim. tarkastamattomaan teksti kenttään voidaan syöttää mitä tahansa, esim. HTML-koodia

-Yksinkertainen ja kohtuuhyvä tapa on käyttää ‘pattern’ suojausta:

```
<input type="text" name="country_code"
   pattern="[A-Za-z]{3}" title="Three letter code">
```

# Laskuoperaattorit

- + lisäys
- - vähennys
- * kerto
- / jako
- % jakojäännös
- ++ yhden lisäys
- -- yhden vähennys

# Vertailuoperaattorit

- == arvot vastaavat toisiaan
- === arvot ja tyypit vastaavat toisiaan
- != arvot eivät vastaa toisiaan
- !== arvot tai tyypit eivät vastaa toisiaan
- > suurempi kuin
- < pienempi kuin
- >= suurempi tai yhtä suuri kuin
- <= pienempi tai yhtä suuri kuin

# Kommentit

- Yksirivinen kommentti

```
1  // A single line comment
```

- Monirivinen kommentti

```
1  /*
2      A
3      multi-line
4      comment
5  */
```

# Logiikkaoperaattorit

- && ja
- || tai
- ! vastakohta

# Valintalauseet

```
1   var gradeString = null;
2   var grade       = "B";
3
4   if ( grade === "A" ) {
5       gradeString = "Grade A: Excellent";
6   }
7   else if ( grade === "B" ) {
8       gradeString = "Grade B: Very good";
9   }
10  ...
11  else {
12      gradeString = "Unknown grade";
13  }
```

- if –lause
  - Suorita lohko, mikäli
    lauseen ehto toteutuu
- if…else –lause
  - Suorita lohko, mikäli
    lauseen ehto toteutuu *tai*
    muussa tapauksessa toisen lauseen lohko
- if…else if…else –lause
  - Suorita *yksi (ensimmäinen)* lohko, joka toteuttaa lauseen ehdon *tai* muussa tapauksessa toisen lauseen lohko

# Valintalauseet

- switch –lause
  - Suorita *yksi* *(ensimmäinen)* lohko, joka toteuttaa lauseen ehdon *tai* vaihtoehtoisesti vakiolauseen lohko *(default)*

```
1   var gradeString     = null;
2   var grade           = "B";
3
4   switch ( grade ) {
5       case "A":
6           gradeString = "Grade A: Excellent";
7           break;
8       case "B":
9           gradeString = "Grade B: Very good";
10          break;
11      ...
12      default:
13          gradeString = "Unknown grade";
14          break;
15  }
```

  - *break* –sana estää suoritusta jatkamasta seuraavaan lohkoon (tästä lisää *Hyppylauseet* –kohdassa)

# Toistolauseet

- while –lause
  - Suorita lohko *nolla tai useampi* kertaa, niin kauan kuin toistolauseen ehto toteutuu

```
1  var index   = 0;
2
3  do {
4      index++;
5  } while ( index < 5 );
```

- do…while –lause
  - Suorita lohko *yksi tai useampi* kertaa, niin kauan kuin toistolauseen ehto toteutuu

# Javascript ja CSS

Voit muuttaa CSS-parametrejä javascriptin avulla:

```
<html>
<p id=”p2”>kappale kaksi </p>
<script>
document.getElementById("p2").style.color ="blue";
</script>
</html>
```

# Queryselector ja radiobuttonit

Kuinka löytää valittu radio button?

-

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"  >
    </head>
<body>

<p> Kysymys 1) Onko harjoituksia tarpeeksi?</p>
  <input type="radio" name="k1" value="1">Aivan eri mieltä<br>
  <input type="radio" name="k1" value="2">Jonkin verran eri mieltä <br>
 <input type="radio" name="k1" value="3">En osaa sanoa<br>
  <input type="radio" name="k1" value="4">Jonkin verran samaa mieltä<br>
    <input type="radio" name="k1" value="5">Aivan samaa mieltä<br>
<p> Kysymys 2) Onko koodausta tarpeeksi?</p>

  <input type="radio" name="k2" value="1">Aivan eri mieltä<br>
  <input type="radio" name="k2" value="2">Jonkin verran eri mieltä <br>
 <input type="radio" name="k2" value="3">En osaa sanoa<br>
  <input type="radio" name="k2" value="4">Jonkin verran samaa mieltä<br>
    <input type="radio" name="k2" value="5">Aivan samaa mieltä<br>

  <br>

  <input type="button" onclick="showSelection()" value="Lähetä  vastaukset">

  <br><br>
  <input type="text" id="tulos" size="50">
  <input type="text" id="tuloska" size="50">

<script>
function showSelection() {
    var a,b;
    a= 0;
```

# Javascript kehitysympäristöt verkossa

Aalto-yliopisto
Perustieteiden korkeakoulu

# Näillä voit tehdä pikakokeita

https://js.do/
https://jsfiddle.net/
https://codepen.io

https://htmlcheatsheet.com/js/

# GIT

# Miksi Git

- Helpottaa versionhallintaa
- Automatisoi varmuuskopiointia
- Helpottaa tiimityöskentelyä

Git-voi olla alkuun hieman hankala oppia, mutta sen käyttö kannattaa opetella - vaikkapa vain siksi, että sen osaaminen on välttämätöntä hyvin monissa työpaikoissa.

# Miten Git

Git projekti kannattaa perustaa Githubiin/Gitlabiin. Sen jälkeen:

Omalla koneella projekti/kansio linkitetään Git:iin.

On kenties hyödyllistä ajatella git-projektia **‘kansiona jolla on versionhallintaan liittyvät supervoimat’**

Kun asioita on muokattu, niistä tehdään ‘Commit’ eli ‘uusi versio’ - tämä commit viedään palvelimelle komennolla Push.

# Käsitteistöä

Repository

Commit

Stage

Push

Branch

# Github ja Gitlab

Gitlab on laajempi ohjelmiston ja projektinhallintajärjestelmä. Se on Aallon opiskelijoiden käytössä (Aalto Gitlab)  ja tuettu (ja siksi suositeltavampi)

https://usersnap.com/blog/gitlab-github/

# Git omalla koneella

Git-ohjelma pitää asentaa omalle koneelle, jotta se osaa huolehtia yhteydestä Githubiin ja ylläpitää projektia.

https://www.linode.com/docs/development/version-control/how-to-install-git-on-linux-mac-and-windows/

# Linkkejä ja tutoriaaleja

https://www.elegantthemes.com/blog/resources/git-and-github-a-beginners-guide-for-complete-newbies

https://scicomp.aalto.fi/aalto/git.html

# Github Pages

https://pages.github.com/

# CANVAS

# Canvas

Mahdollistaa piirtämisen HTML-sivulle

Perusalkioita mm. suorakulmio, ellipsi jne.

Mahdollistaa kuvien manipuloinnin pikselitasolla

Runsaasti kirjastoja (mm. pelinkehitykseen, fysiikkamallinnukseen yms.)

# Canvas - HTML - JS

Lisää HTML:n canvas tagi:

```
<canvas id="myCanvas">Your browser does not support the canvas tag.</canvas>
```

Ota 'kiinni' tagista ja tee siihen 2D tai 3D piirtopinta/paperi getContext-käskyllä:

```
var canvas=document.getElementById('myCanvas');
var ctx=canvas.getContext('2d');
```

Piirrä neliö:

```
ctx.fillStyle='#FF0000';
//fillRect(x,y,x_koko,y_koko)
ctx.fillRect(20,20,5,5);
```

# Animointi

Animointi tapahtuu piirtämällä elementti, odottamalla, putsaamalla elementin paikka ja piirtämällä se uudelleen.

# Canvas koodiesimerkki

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas">Your browser does not support the canvas tag.</canvas>

<script type="text/javascript">
var canvas=document.getElementById('myCanvas');
var ctx=canvas.getContext('2d');
for(var bigluuppi = 0; bigluuppi < 200;bigluuppi += 10){
	ctx.fillStyle='#FF0000';
	ctx.fillRect(20,bigluuppi,5,5);
}
</script>
</body>
</html>
```

# Eventit, hiiren sijainnin lukeminen

```
function hiirifunktio(myEvent) {
var r = myEvent.pageX % 223;
var ry = myEvent.pageY % 223;
cxt2.fillStyle = "rgb( "+r+", "+r+", "+ry+")";
//cxt2.fillRect(myEvent.pageX, myEvent.pageY, 134, 134);
for( k = 0; k < 60;k+= 2){
xpaikka = (myEvent.pageX + Math.sin(k / 10.0) * 45) - 100;
ypaikka = (myEvent.pageY + Math.cos(k / 10.0) * 45) - 100;
cxt2.drawImage(img2, xpaikka ,ypaikka,koko,koko);
            }
}
document.onmousemove = hiirifunktio;
```

# RGBA värit

```
var r_a = 0.3;

ctx.fillStyle = "rgba(32, 45, 21, " + r_a + ")";
```

# Kuvan liikuttaminen animoiden - setInterval()  - funktio

```
setInterval ( "timedrawpic()", 30 );
var tick = 0;
function timedrawpic ( )
{
	tick += 1;
	sx = (Math.sin(tick / 12) * 0.03);
	sy = (Math.cos(tick / 6) * 0.03);
	cxt.transform(1, sx, sy, 1, 0, 0);
cxt.drawImage(img2,-200,-200,wkoko + 400 ,wkoko + 400);
		cxt.fillStyle = "rgba(23, 23, 23, 0.1)";
		//cxt.fillStyle = "rgb(5,5,5)";
		cxt.fillRect (-770, -770, 770 + wkoko, 770 + wkoko);
}
```

# Lisätietoa - Canvakseen perustuvat pelinkehitysympäristöt

Pelinkehitys on hauska ja tehokas tapa oppia ohjelmointia. Javascriptille on tarjolla monia ilmaisiakin pelikirjastoja, joilla voi tehdä helposti näyttäviä pelejä, esim:

http://www.pixijs.com/

https://pixijs.io/examples/#/demos/tinting.js

Huomaa, että pixi.js ja monet muut vastaavat kirjastot vaativat, että ne toimivat palvelimelta. Tämän voit toteuttaa paikallisesti esim. käyttämällä Chromen server lisäosaa. Osoitat tälle lisäosalle vain, mistä kansiosta ‘palvellaan’.

# Lisätietoa - Canvas ja kokeellinen ohjelmointi

Kaikki ohjelmointi ei ole ennaltamääriteltyä ja tekniikka- tai bisnesorientoitua. Ohjelmoimalla voidaan tehdä myös taidetta ja kokeilla uusia käyttötapoja tms.

P5.js on ilmainen Processing-kieleen perustuva yhteisöllinen Javascript kirjasto:

https://p5js.org/

Muita ‘creative coding’ Javascript sivustoja:

https://ptsjs.org/

http://soulwire.github.io/sketch.js/

# Canvas

- “Piirtoalusta”
- Määritellään HTML:ssä, ‘otetaan kiinni’ javascriptissä
- Mahdollistaa mm. pelit
- Varsin tehokas
- Nykyään saatavilla pelikirjastoja, kokeilevaa ohjelmointia ja fysiikkamallinnusta
  - P5.js
    - https://p5js.org/
  -

# Canvas-kirjastot

- https://ihatetomatoes.net/guide-to-html5-canvas-javascript-libraries/
  - http://paperjs.org/
  - https://p5js.org/libraries/

- Pelikirjastot
  - graafiset
  - fysiikkamallinnus
  -

# Canvas harjoitus 1

Tee piirrä canvakselle punainen neliö.

# Canvas harjoitus 2

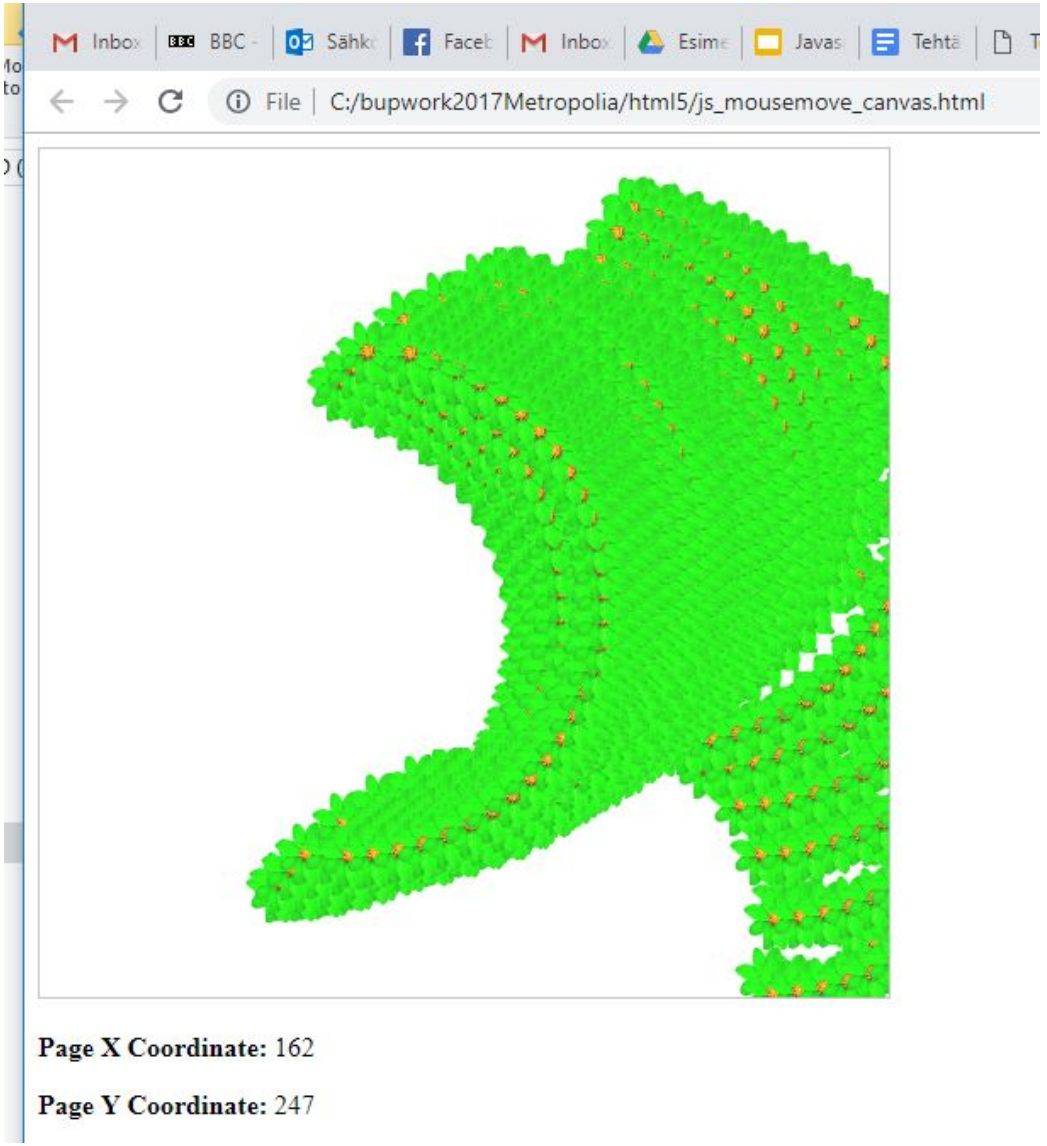
Tee piirrä canvakselle erikokoisia ja erivärisiä suorakulmioita satunnaisiin paikkoihin

# Canvas harjoitus 3

Tee yksinkertainen piirto-ohjelma, joka piirtää pallon/palloja siihen missä hiiri on.

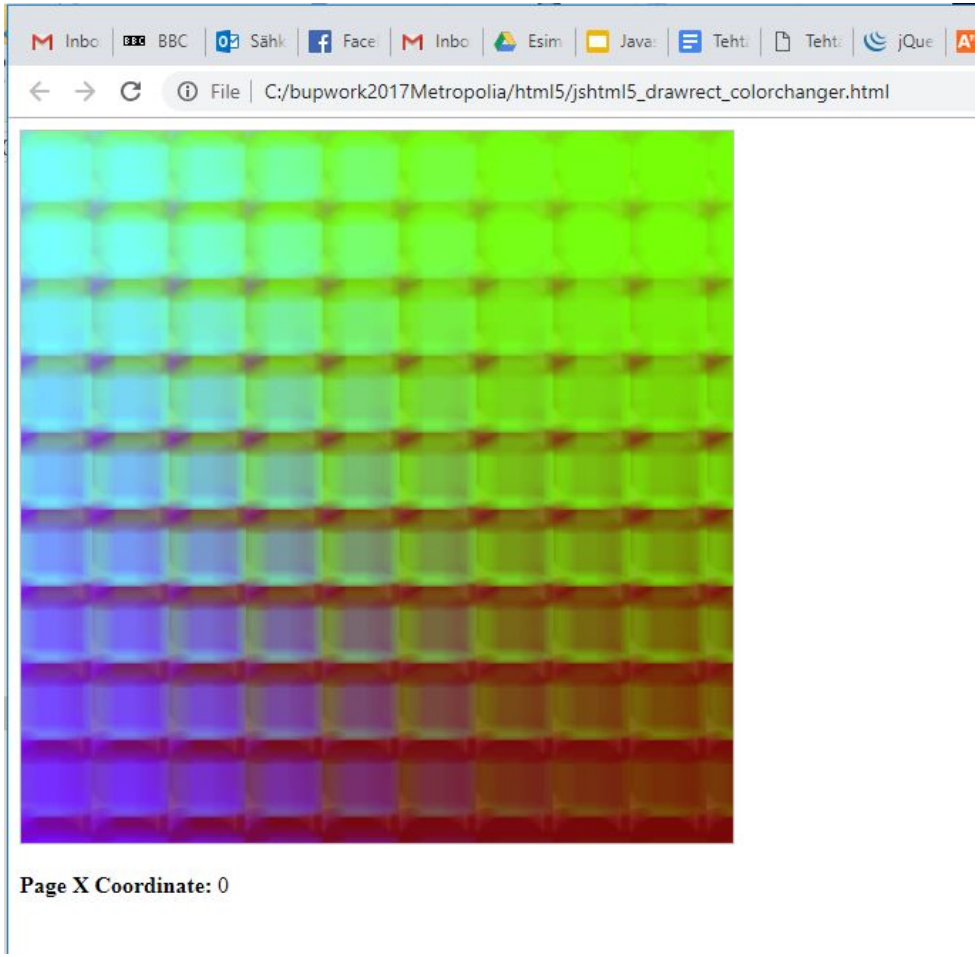
# Canvas harjoitus 4

Tee yksinkertainen piirto-ohjelma, joka piirtää kuvan  siihen missä hiiri on.



# Canvas harjoitus 5

Kuinka oheinen kuvio on tehty? Lisää siihen ajastettu toiminnallisuus.



# Toistolauseet

# Toistolauseet

- for –lause
  - Suorita lohko *nolla tai*
    *useampi* kertaa, niin kauan
    kuin toistolauseen ehto
    toteutuu
  - Toistojen määrä yleensä etukäteen tiedossa

```
1. parametri = käsiteltävä elementti
2. parametri = ehto, jonka täyttyessä jatketaan
3. parametri = jokaisen loopin yhteydessä suoritettava funktio
```

```
1  var index   = 0;
2
3  for ( index; index < 5; index++ ) {
4      ...
5  }
```

# While

Toista kunnes ehto täyttyy:

```
text = “Alku:”
while (i < 10) {
  text += "The number is " + i;
  i++;
}
console.log(text);
```

# Toistolauseet - taulukon läpikäynti

```
var mundata = ['a','b','c','d'];
function toimi(){
for(var k = 0;k<mundata.length;k++ ){
    console.log(mundata[k])
    }
  }
```

# Kummallisuuksia

-vältä listojen läpikäyntiä “for item in list”-tyyppisen konstruktion avulla

-Sulkeumien aiheuttama ‘palauttaa vain viimeisen arvon’ erikoisuus:

https://dzone.com/articles/why-does-javascript-loop-only-use-last-value

Korjautuu parhaiten “LET” muuttujamäärittelyllä

# Pohdinta

Miten yhdistetään kaksi taulukkoa A ja B kolmanteen taulukkoon?

Tulos:

a1,b1,a2,b2…..

# Hyppylauseet

- return –lause
  - Poistu metodista *tai* silmukasta
  - Voi palauttaa arvon
- break –lause
  - Poistu toistolauseesta *tai* silmukasta
- continue –lause
  - Poistu toistolauseen lohkosta

```
1  var index   = 0;
2
3  while ( index < 5 ) {
4      if ( index === 3 ) {
5          break;
6      }
7      index++;
8  }
```

# Virhetilanteiden käsittely

- try...catch...finally –lause
  - Suorita *catch* –lohko, mikäli *try* –lohkon suorituksessa tapahtuu poikkeus
    - Suorita lopuksi vielä valinnainen *finally* –lohko
  - Esimerkki

```
try {
    if ( courseCredits < 1 ) {
        aaaaalert( "This function call should throw an exception." );
    }
} catch ( ex ) {
    errorMessage    = ex.name + ": " + ex.message;
} finally {
    courseCredits   = 1;
}
```

# Virhetilanteiden käsittely

- throw –lause
  - Virheviestit voidaan räätälöidä *throw* –lauseen avulla
  - Esimerkki

```
try {
    if ( courseCredits < 1 ) {
        throw {
            "name"      : "MinimumValueError",
            "message"   : "Credits must be more than zero."
        };
    }
} catch ( ex ) {
    errorMessage    = ex.name + ": " + ex.message;
} finally {
    courseCredits   = 1;
}
```

# Varatut sanat

| | | | |
|---|---|---|---|
| abstract | const | enum | function |
| boolean | continue | export | goto |
| break | debugger | extends | if |
| byte | default | false | implements |
| case | delete | final | import |
| catch | do | finally | in |
| char | double | float | instanceof |
| class | else | for | int |

# Varatut sanat

| | | | |
|---|---|---|---|
| interface | public | throw | void |
| long | return | throws | while |
| native | short | transient | with |
| new | static | true | |
| null | super | try | |
| package | switch | typeof | |
| private | synchronized | var | |
| protected | this | volatile | |

# Javascript ja lisäkirjastot

MOBIILAITTEISIIN mm:

Paikannus

https://www.w3schools.com/html/html5_geolocation.asp

Anturit

-liiketunnistin, kiihtyvyysanturi, kompassi, lämpötila

Kamera

https://developers.google.com/web/fundamentals/media/capturing-images/

Yms

# OLIO-OHJELMOINTI

# This

- Viittaa käytettävissä olevaan kontekstiin *(itseviite)*
  - Määräytyy kutsuja(olio)n mukaan
  - Käytetään esim. olion sisällä viittaamaan sen ominaisuuksiin
- Esimerkki

```
1   // Define an object and its properties and methods
2   var course  = {
3       "name"      : "Digitaalisen median työvälineet",
4       "code"      : "T-111.1100",
5       "credits"   : 3,
6       "toString"  : function() {
7           return this.code + " " + this.name + " (" + this.credits + "cr)";
8       }
9   };
10
11  // Access object methods
12  var toString    = course.toString();
```

# Olio-ohjelmointi JavaScriptillä

- JavaScript on *oliopohjainen* kieli
- JavaScript ei kuitenkaan tarjoa Javan / Scalan kaltaista tukea olio-ohjelmoinnille
  - Ei suoranaisesti tue luokkia, perintää, jne.
- Javascriptin oliomäärittelyyn on useita tapoja. Tällä kurssilla käytetään seuraavassa määriteltyä tapaa:

Linkki:

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

# Oliomäärittely

```
class Rectangle {
constructor(height, width) {
    this.height = height;
          this.width = width;
    }
calculate_area(){
          return = this.height * this.width;
    }
}

var r = new Rectangle()
r.height = 200;
r.calculate_area();
```

# Oliolista

- Listaan voi tallentaa erilaisia olioita, esim ’asiakas’
- Hyvin monenlaisia ilmiöitä voi esittää varsin selkeästi oliolistoilla. (esim. musiikissa tahti-oliossa voi olla nuottilista, jossa on nuotti-olioita)
- On tärkeää osata käydä läpi oliolistoja ja muuttaa niiden alkioiden arvoja.

## Olioanalyysi - ryhmätuntitehtävä

Mahdollisia kohteita

- Sähköpyörien vuokrausjärjestelmä
  - Pyörä, lainaaja, pyörän varaus jne.
- Lämpötila-anturiverkosto
  - anturi, tila/huone,säätöjärjestelmä, anturin antaman tiedon lähetys

Mitä olioita on?

Mitkä kuuluvat mihinkin?

Mitä Metodeja?

Mitä olioille pitää voida tehdä?

Tekoaika 15 min, purku 5 min/ryhmä.

Ryhmät palauttavat ja esittelevät tulokset.

# TUNTIHARJOITUKSIA - Javascript ja oliot

# Tuntiharjoitus 1

Harjoittelemme olioden määrittelya ja niiden käyttöä.

1) Laadi olio jolla on ominaisuuksia ja toimintoja. Esim kissa, jolla on nimi ja paino ja joka voi tehdä jotain. Luo yksi kissa :

```
class Kissa
        ominaisuus: nimi
        toiminto: naukaisee

minunkissa = new Kissa()
```

2) Kokeile, että edellisessä tehdyn olion toiminto toimii

3) Tee toinen olio, esim. Kissatarha, jossa on taulukko, jonne teet useita Kissa-luokan olioita ja jossa on toiminto, joka kokoaa kaikista olioista jotain tietoa (esim. yhteispaino).

```
class Kissatarha
        tarhankissat = taulukko
        yhteispaino = luku

        toiminto: LaskeYhteispaino
```

4) Muuta edellinen tehtävä (1-3) niin, että se käsittelee jotain toista aihepiiriä.

# Tuntiharjoitus 2

Lue sivun input kentästä nimi muuttujaan.

Luo uusi 'Asiakas' luokan olio 'asiakas_a', jonka nimi-ominaisuudeksi laitat muuttujassa olevan nimen.

Tee Asiakkaat taulukko

Lisää 'asiakas_a' taulukkoon

# Tuntiharjoitus 3

Muuta asiakas-luokkaa siten, että sillä on myös ominaisuus 'Raha'. Oletusarvo 100.

Tee uusi luokka 'Pankki. Laita Asiakkaat-taulukko siihen. Lisää ominaisuus 'Kokonaisvarallisuus'.

Muuta jokaisen Asiakkaat-taulukon jäsenen Raha-parametria satunnaisesti +-20

Käy läpi Asiakkaat taulukko ja laske rahat yhteen. Sijoita tulos Pankki-olion Kokonaisvarallisuus kenttään.

# Tuntiharjoitus 4 - Metodi

Suunnittele metodi, jossa asiakas voi siirtää rahaa toiselle asiakkaalle

Onko metodi Asiakas-luokassa vai Pankki-luokassa?

# Pieni harjoitus sorttauksesta

Täytä lista olioilla Kissa, joiden painoksi arvotaan satunnaisluku.

Tee kaksi nappia: Isoin ja Pienin

Kun painetaan Isoin, tulostuu listan painavimman kissan paino (ja/tai nimi).

Kun painetaan Pienin, tulostuu listan kevyimmän kissan paino (ja/tai nimi).

# Sort function

```
//members on oliolista
// a ja b ovat olioita
this.members.sort(function(a,b){
    return b.combined_suitability - a.combined_suitability;
                }
            )
```

# Emo ja lapsi - olioiden välinen viittaus

Joskus voi kahden luokan välillä tarvita suoraa kommunikaatiota. Tällöin voi viittauksen toiseen luokkaan laittaa parametriksi.

```
class Kaappi{
        constructor(){
                this.lippaat = []
                }
        }
munKaappi = new Kaappi(); //Kaapissa on taulukko lippaat, eli se tietää, mitkä lippaat sillä on


for(var k = 0;k < 10;k++){
                var tmplipas = new Lipas(k);
                tmplipas.omistaja = munKaappi; //Lipas tietää mihin Kaappiin se kuuluu
                Kaappi.lippaat[k] = tmplipas;
        }
```

# Pieni harjoitus tallentamisesta

Tallenna edellisessä harjoituksessa tekemäsi lista webstorageen/localstorageen.

# JSON

# JavaScript Object Notation (JSON)

- Kevyt, tekstipohjainen (string), kieliriippumaton tiedonsiirtoformaatti
  - Erittäin suosittu webissä, kuten XML
  - Toimii erittäin hyvin yhteen JavaScriptin kanssa
- Avoin teknologia (ts. ei ole standardoitu)
  - Pohjautuu ECMAScript 3:een
  - http://tools.ietf.org/html/rfc4627
- XML:ään verrattuna teknisesti hieman rajoittuneempi
- Alkaa olla standardi syötteissä, esimerkiksi Twitter- ja Facebook ovat siirtyneet xml:stä json:iin

# JavaScript Object Notation (JSON)

- *Olio* tai *taulukko*
  - Olio kuvaa jotakin *käsitettä* ja sen *ominaisuuksia*, esim. kurssi
  - Taulukko listaa arvoja
  - Arvojen sallitut tyypit
    - null, boolean, numero, merkkijono, taulukko, olio
- Helppokäyttöinen sekä ihmisen että koneen näkökulmasta
  - Natiivituki tullut selaimiin vasta 2009

```
1  // Stringify JSON (object)
2  var jsonString  = JSON.stringify( course );
3
4  // Parse JSON (object)
5  var json        = JSON.parse( jsonString );
```

  - Käytä erillisiä kirjastoja vanhojen selaimien kanssa

# Esimerkki JSON

```
{
    "name"      : "Digitaalisen median työvälineet",
    "code"      : "T-111.1100",
    "credits"   : 3
}
```

# JSON:n käyttö

- Olion ominaisuuksien lukeminen
  - Kaksi eri notaatiota: piste- ja sulkumerkkinotaatio
  - Esimerkki
    - course[ "credits" ]
    - course.credits
- Taulukon arvojen lukeminen
  - Haetaan indeksin perusteella (alkaa nollasta)
  - Esimerkki
    - courses[0]
- Syvähierarkisen JSON:n lukeminen
  - Tarvittaessa käytä molempia tapoja useampaan kertaan ja/tai yhdistele niitä keskenään
  - Esimerkki
    - courses[0].credits

# TAPAHTUMAT

# Kaksi päätyyppiä

- Käyttäjän tuottama ‘event’
  - Hiiri
  - Näppäimistö
  - Kosketus
- Järjestelmän tuottama tapahtuma (esim. ajastettu)
  - window.onload
  - setinterval

# Tapahtumat

- Suuri osa selainpään JavaScript-ohjelmakoodista liittyy *tapahtumien kuunteluun* ja niihin *reagoimiseen*
  - HTML-sivun latauksen valmistuminen
  - Linkin painaminen
  - Hiiren liikkeiden seuraaminen
- Tapahtumasta välittyy muun muassa sen *tyyppi* ja *kohde*
  - http://www.quirksmode.org/js/events_properties.html

# Tapahtumat

- DOM Level 3 Events
  - W3C:n työluonnos, syyskuu 2012
  - Määrittelee tapahtumat, niiden kuuntelun ja etenemisen, jne.
  - http://www.w3.org/TR/DOM-Level-3-Events/
- Yleisimmät tapahtumatyypit
  - Käyttöliittymätapahtumat, esim. load
  - Kohdistustapahtumat, esim. focus
  - Hiiritapahtumat, esim. click, mouseover ja mousemove
  - Näppäimistötapahtumat, esim. keyup ja keydown
- Lista tapahtumatyypeistä
  - http://www.w3.org/TR/DOM-Level-3-Events/#event-types-list

# Tapahtumien käyttö

- Tapahtumakuuntelijoiden lisääminen
  - addEventListener( tyyppi, kuuntelija )
  - https://developer.mozilla.org/en/DOM/element.addEventListener
- Tapahtumakuuntelijoiden poistaminen
  - removeEventListener( tyyppi, kuuntelija )
  - https://developer.mozilla.org/en-US/docs/DOM/element.removeEventListener
- Tapahtumien lähettäminen
  - dispatchEvent( tapahtuma )
  - https://developer.mozilla.org/en-US/docs/DOM/element.dispatchEvent



*http://www.w3.org/TR/DOM-Level-3-Events/#events-EventTarget*

# Tapahtumien käyttö

```
1   var eventType       = null;
2   var eventKeyCode    = null;
3   var eventKey        = null;
4
5   // Add keydown event listener
6   document.addEventListener( "keydown", function( event ) {
7       eventType       = event.type;
8       eventKeyCode    = event.keyCode;
9       eventKey        = String.fromCharCode( eventKeyCode );
10  });
```

# Tiedon tallennus

# Tiedon tallennustavat verkossa (selainpäässä)

- Web Storage
  - **Web Storage, http://www.w3.org/TR/webstorage/**
  - Tallentaa käyttäjän selaimen muistiin tietoa joko kyseisen session ajaksi (kunnes selainikkuna suljetaan, sessionstorage) tai “pysyvästi” (kunnes muisti tyhjennetään, localstorage)
- Kolmannen osapuolen palvelut
  - Firebase, https://www.firebase.com/
    - Reaaliaikainen tiedonsiirto
    - Hyödyllinen esimerkiksi moninpeleissä
  - https://jsonbin.io/
  - http://myjson.com/

# Web Storage:n käyttö

- Varmista aluksi että käyttäjän selain tukee web storagea

```
if(typeof(Storage) !== "undefined") {
      // Code for localStorage/sessionStorage.
} else {
      // Sorry! No Web Storage support..
}
```

- Tiedon tallentaminen avain-arvo pareilla
  - `localStorage.setItem("lastname", "Smith");`
- Tiedon hakeminen avaimen avulla
  - `localStorage.getItem("lastname"); //Smith`
- Tarkempi dokumentaatio
  - http://www.w3schools.com/html/html5_webstorage.asp

# Webstorage+json+oliolistat

- Funktioiden tallennus ei mahdollista
- Mutta olioiden sisällön tallennus on
- Kun lataa oliolistan, niin voi konstruoida uudet oliot ja täyttää niiden sisällöt

# Tuntiharjoitus 1

Tee taulukko, jonne tallennetaan text-input kentässä olevat sanat erillisinä:
input kenttä: Kissa koira lintu
-> taulukko[“Kissa”, “koira”, “lintu”]

Tee HTML-sivu, jossa on kaksi painiketta:

Tallenna (tallentaa taulukon alkiot local storageen
Lataa (lataa storagesta) ja näyttää

# Tuntiharjoitus 2

Tallennamme edellisessä tehtävässä tehdyt oliot localstorageen. Ota huomioon, miten Javascriptissa voi tallentaa olioita (tai miten ei voi). Kun osaat tallentaa olioita, voit myöhemmin tallentaa ne palvelimelle ja tietokantaan (tai esim. json-tallentimeen). On myös olemassa localstorage-tallentimia, jotka pitävät yllä palvelinkopiota local storagesta.

1) Lue seuraavat lyhyet kuvaukset

2) Tallenna ja lataa (kokeile toimivuus) edellisessä harjoituksessa tehdyt oliot.

Javascript hukkaa olioiden funktiomääritykset (metodit) kun se muunnetaan Jsoniksi ja takaisin, siksi oliot pitää rekonstrudoida seuraavalla kalvolle esitetyllä tavalla:

# Stringify/Stringifaijatun olion rekonstruktio

Koska json stringify hukkaa funktiot (!), niin localstoragesta ladatut oliot pitää rekonstruoida, käytämme tähän Object.assign funktiota, näin:

var uusi_instanssi = new MunLuokka()

Object.assign(uusi_instanssi, localstoragesta_tulevainstanssi)

uusi_instanssi sisältää kaiken datan/properties, jotka on ladattu storagesta, mutta lisäksi funktiot!

# Rekonstruktio-koodi

```
function localstorage_store(){
	localStorage.setItem('Kissaoliot', JSON.stringify(Kissat));
	console.log(JSON.stringify(Kissat)) //kone hoitaa Jsonin teon
}

function localstorage_load(){
   Kissat_fromstorage = JSON.parse(localStorage.getItem('Kissaoliot')); //kone
parsaa
	Kissat = [ ] /tyhjennetaa lista
	for (var Kissaolio of Kissat_fromstorage){
		var Kissaolio_withfunc = new Kissaolio();
		//nyt ehja ja myos metodifunktiot toimii, koska oliot rekonstruoidaan
		Object.assign(Kissaolio_withfunc,Kissaolio);
		Kissat.push(Kissaolio_withfunc);

	// rikki ilman object.assign-juttua, eli Kissojen metodit ei toimi funtioina
		//Kissat.push(Kissaolio);
		}
```

# Verkkosivujen optimointi

# Kuvien optimointi

- Myös kuvat kannattaa optimoida käyttötarkoitukseen sopivaksi
  - Pienennä kuvat tarvittavan kokoisiksi
  - Käytä optimaalisinta kuvaformaattia
  - Poista ylimääräinen tieto kuvista
- Työkaluja
  - http://addyosmani.com/blog/image-optimization-tools/
  - http://www.picresize.com/

# Muu optimointi (good to know, mutta ei tämän kurssin kannalta oleellista)

- Olennaista käytettävyyden ja saavutettavuuden kannalta
  - Nopeasti latautuva ja optimoitu sivusto sijoittuu korkeammalle hakutuloksissa
  - Jopa muutaman sekunnin latausajat karkottaa suuren osan käyttäjistä https://blog.kissmetrics.com/loading-time/
  - https://developers.google.com/speed/pagespeed/
- Hakukoneoptimiointi
  - Semanttinen html-koodi (meta-tiedot, oikeanlaiset elementit oikeissa tarkoituksissa)
  - Responsiivisuus (pienelle näytölle optimoitu sivusto sijoittuu korkealle mobiililaitteilla haettaessa)
  - https://www.google.com/webmasters/tools/mobile-friendly/

# FETCH

# FETCH

The `fetch()` method takes one mandatory argument, the path to the resource you want to fetch. It returns a `Promise` that resolves to the `Response` to that request, whether it is successful or not.

# PROMISE

The **`Promise`** object represents the eventual completion (or failure) of an asynchronous operation, and its resulting value.

Objekti joka on vastaus asynkroniselle pyynnölle (eli voi tulla heti tai joskus).

# REQUEST

```
Palvelupyyntö

const myRequest = new
Request('https://www.mozilla.org/favicon.ico');

fetch(myRequest) .then(response =>
response.blob()) .then(blob => { image.src =
URL.createObjectURL(blob); });
```

# RESPONSE

The **`Response`** interface of the Fetch API represents the response to a request.

Tämä on varsinainen vastaus. Sisältää monia funktioita, mm. json-funktion:

**`Body.json()`**

Takes a `Response` stream and reads it to completion. It returns a promise that resolves with the result of parsing the body text as `JSON`.

# Open Weather

https://openweathermap.org/current

# API

# API ja miten sitä luetaan

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data

# Studio Ghibli API

https://www.taniarascia.com/how-to-connect-to-an-api-with-javascript/

https://ghibliapi.herokuapp.com/films

# API avain ja rekisteröityminen

https://openweathermap.org/appid

# GIT

# Miksi Git

- Helpottaa versionhallintaa
- Automatisoi varmuuskopiointia
- Helpottaa tiimityöskentelyä

Git-voi olla alkuun hieman hankala oppia, mutta sen käyttö kannattaa opetella - vaikkapa vain siksi, että sen osaaminen on välttämätöntä hyvin monissa työpaikoissa.

# Miten Git

Git projekti kannattaa perustaa Githubiin/Gitlabiin. Sen jälkeen:

Omalla koneella projekti/kansio linkitetään Git:iin.

On kenties hyödyllistä ajatella git-projektia **‘kansiona jolla on versionhallintaan liittyvät supervoimat’**

Kun asioita on muokattu, niistä tehdään ‘Commit’ eli ‘uusi versio’ - tämä commit viedään palvelimelle komennolla Push.

# Käsitteistöä

Repository

Commit

Stage

Push

Branch

# Github ja Gitlab

Gitlab on laajempi ohjelmiston ja projektinhallintajärjestelmä. Se on Aallon opiskelijoiden käytössä (Aalto Gitlab)  ja tuettu (ja siksi suositeltavampi)

https://usersnap.com/blog/gitlab-github/

# Git omalla koneella

Git-ohjelma pitää asentaa omalle koneelle, jotta se osaa huolehtia yhteydestä Githubiin ja ylläpitää projektia.

https://www.linode.com/docs/development/version-control/how-to-install-git-on-linux-mac-and-windows/

# Linkkejä ja tutoriaaleja

https://www.elegantthemes.com/blog/resources/git-and-github-a-beginners-guide-for-complete-newbies

https://scicomp.aalto.fi/aalto/git.html

# Github Pages

https://pages.github.com/

# Linkkejä yms.

# Linkkejä

- http://www.codecademy.com/tracks/javascript
- http://www.codecademy.com/tracks/jquery
- https://developer.mozilla.org/en-US/docs/DOM/DOM_Reference
- http://www.gotapi.com/jsdomw3s
- http://www.w3schools.com/jsref/default.asp

# Yhteenveto

# Yhteenveto

- JavaScriptistä on kehittynyt vuosien varrella *johtava* asiakaspään ohjelmointikieli
  - HTML5:n ja siihen liittyvien JavaScript API:ien myötä JavaScriptin suosio *kasvaa entisestään*
- JavaScript tarjoaa *ohjelmointirajapinnat* niin tapahtumien kuunteluun kuin DOM-puun käsittelyyn
- JSON tarjoaa *yksinkertaisen* ja *toimivan* tiedonkuvaus- ja tiedonsiirtoformaatin
- JavaScript-kirjastot (esim. jQuery) saattavat *helpottaa* ja *nopeuttaa* asiakaspään ohjelmointia, jos osaa käyttää niitä.
- Verkkosivujen optimointi nopeuttaa sivulatauksia sekä parantaa käyttökokemusta

# Kiitos!

# Tuntiharjoitukset

# Tuntiharjoitukset

Dom-elementin sisällön vaihto
-taustaväri
-tekstikenttä

Nappulaan reagoiminen
-taustaväri vaihtuu
-tekstikenttä vaihtuu

Ehdot ja toistot
Listasta otettava elementti
Listan läpikäynti
Ajastettu elementti
NAV palkilla valittava sisältö
Funktiot

# FETCH

1) Fetch

Tee yksinkertainen sivu jolla haet Studio Ghibbli kokeilusivustolta elokuvien nimet ja niiden päähenkilöt (1/elokuva).

Tulosta tulokset sivulle  kenttään, jonka id on “tulos”.

# Ajastetut tapahtumat - Timed events

2) Timed events

Tee yksinkertainen sivu, jossa käynnistät napilla jonkin toiminnon ja toisella napilla pysäytät tämän toiminnon. Toiminto voi olla esim. yksinkertainen laskuri.

Käytä “setinterval”-funktiota

# Haku radiolistasta

3) Queryselector from radiobuttons

Harjoitellaan valintojen kokoamista javascriptillä.

Tee sivusto, jossa voit tehdä monia pasta-aterioita. Käyttäjä valitsee seuraavista kategorioista radionappuloiden avulla. Kunkin kategorian nappuloiden “name” on kategorian nimi ja “value” on itse aines. Checked arvolla löydetään, mikä on valittu.

Ohjelma näyttää kolmen kategorian yhdistelmän, esim.:
Spagetti+Tomaattikastike+Lihapullat

1) Pastalaji
2) Kastike
3) Pääaines

# Tuntiharjoituksia  - jatkoa

1) Tee nappula, joka vaihtaa  Nimetyn Dom-elementin (esim. div tai p) tekstikentän sisällön:

"Kissa" -> nappulaa painetaan -> "Koira"

2) Lisää toiminto edelliseen, joka vuorottelee edellisen tehtävän kahden tekstin välillä:

"Kissa" -> nappulaa painetaan -> "Koira" ->nappulaa painetaan -> "Kissa" jne...

3) Lisää toiminto edelliseen, joka vuorottelee edellisen tehtävän kolmen eri tekstin välillä:

"Kissa" -> nappulaa painetaan -> "Koira" ->nappulaa painetaan -> "Lintu "  ->"Kissa"

4) Tee nappula, joka painettaessa muuttaa id:llä valitun elementin (esim. div) taustavärin siniseksi.

5) Tee nappula, joka painettaessa muuttaa id:llä valitun elementin (esim. div) taustavärin satunnaiseksi.

sini:

```
var red = Math.floor(Math.random() * 255);
tee sama siniselle ja vihreälle
var alpha = (Math.random());
taustavari = "rgba(" + red + "," + green + "," + blue + "," + alpha + ")"
document.getElementById("myDIV").style.backgroundColor = taustavari;
```

# Tekstikentän sisällön vaihtaminen

Tee painike, joka vaihtaa  Nimetyn Dom-elementin (esim. div tai p) tekstikentän sisällön:

"Kissa" -> nappulaa painetaan -> "Koira"

# Taulukkoharjoitus 2

Lisää toinen taulukko jossa on atribuutteja 4
kiva, ihana jne

Ja tulosta ne eläinten kanssa vuorotellen

kissa kiva, koira ihana ….

# Tuntiharjoitus 2

1) Tee nappula, joka ottaa input-kentästä sinne syötetyn tekstin ja sijoittaa sen nimettyyn 'tulos' kenttään

2) Tee nappula, joka ottaa input-kentästä sinne syötetyn tekstin ja sijoittaa sen nimettyyn 'tulos' kenttään lisäten sen jo olemassaolevaan tekstiin:

kissa -> kissakoira ->kissakoiralintu

3) Tee nappula, joka ottaa  kahdesta input-kentästä niihin syötetyn tekstin, yhdistää ne ja sijoittaa tuloskenttään:

etunimi+sukunimi

4) Tee nappula, joka ottaa  kahdesta input-kentästä tekstin ja sijoittaa sen viisialkioiseen  taulukkoon, kasvattaen taulukon indeksiä. Tee toinen nappula, joka tulostaa taulukon sisällön 'tulos' kenttään.

5) Lisää edelliseen 'satunnaisvalinta'-nappula, joka tulostaa toiseen kenttää taulukosta satunnaisesti valitun elementin.

# Tuntiharjoitus 3

1) Tee funktio, joka tulostaa seuraavat luvut:

    1 2 3 4
    1 2 3 4
    1 2 3 4
    1 2 3 4

2) Tee funktio, joka tulostaa seuraavat luvut. (rivillä '0 1' käytä jakojäännös operaattoria '%')

    1 2 3 4
    0 1 0 1
    1 2 3 4
    0 1 0 1

3) Tee funktio, joka tulostaa seuraavan:

    1 2 kolme 1 2 kolme

4) Tee funktio, joka tulostaa seuraavan:

    1 2 kolme 4 suurempi_kuin_4 suurempi_kuin_4

# Tuntiharjoitus 4

1) Tee HTML-sivu, jossa elementin esim <p> sisältö muuttuu, kun sitä klikataan:

VINKKI (tässä on liian paljon helpotusta, mutta kun 'this'):

```
<p id = "secondpara" onclick="changeText_para(this)"> OLEN eka kappale <p>

function changeText(id) {
id.innerHTML = 'Olen muutettu teksti'
}
```

2) Tee HTML sivu, jossa on navigointilista (esim. kaupungit). Jotain kaupunkia klikattaessa, ei aukea uusi sivu vaan sivun keskelle tulostuu ko. kaupunkia kuvaava teksti.

VINKKI:

Navigointipalkin listaelementit kutsuvat muutosfunktiota klikattaessa, siten että funktion parametrina on kaupungin nimi

```
<li id="firstpara" onclick="changeText('London')"> Tietoa Lontoosta </li>

function changeText(cityname){
    ----muuttaa kuvaustekstin
```

# Tuntiharjoitus 5

1) Tee yksinkertainen kyselysivu radiobuttoneiden avulla, jossa käyttäjä valitsee mieluisat vaihtoehdot. Valinnat tulostetaan.

Onko tarpeeksi tehtäviä (1 = ei  5 = kyllä)

1

2 x

3

4

5

Ovatko tehtävät sopivan vaikeita (1 = liian vaikeita  5 = liian helppoja)

1

2

3

4 x

5

VALINTASI: 2 ja 4

VINKKI: queryselector

```
tulos1 = document.querySelector('input[name=kysymys1]:checked').value;
```

# Koodiesimerkkejä

# Radiobutton - queryselector

Seuraavan sivun koodissa näytetään kuinka ns radiopainikkeista löydetään valittu elementti.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"  >
    </head>
<body>

<p> Kysymys 1) Onko harjoituksia tarpeeksi?</p>
  <input type="radio" name="k1" value="1">Aivan eri mieltä<br>
  <input type="radio" name="k1" value="2">Jonkin verran eri mieltä <br>
 <input type="radio" name="k1" value="3">En osaa sanoa<br>
  <input type="radio" name="k1" value="4">Jonkin verran samaa mieltä<br>
    <input type="radio" name="k1" value="5">Aivan samaa mieltä<br>
<p> Kysymys 2) Onko koodausta tarpeeksi?</p>

  <input type="radio" name="k2" value="1">Aivan eri mieltä<br>
  <input type="radio" name="k2" value="2">Jonkin verran eri mieltä <br>
 <input type="radio" name="k2" value="3">En osaa sanoa<br>
  <input type="radio" name="k2" value="4">Jonkin verran samaa mieltä<br>
    <input type="radio" name="k2" value="5">Aivan samaa mieltä<br>

  <br>

  <input type="button" onclick="showSelection()" value="Lähetä  vastaukset">

  <br><br>
  <input type="text" id="tulos" size="50">
  <input type="text" id="tuloska" size="50">

<script>
function showSelection() {
    var a,b;
    a= 0;
```