7. Nettisivut ja ohjelmointi

Jaksossa opetetaan html-koodin perusteet, CSS-muotoilun perusteet ja ohjelmoinnin perusteet. Jakson alussa esitellään tehtäväkokonaisuus, jota lähdetään työstämään esimerkkien kautta.

Päivitetään vanhan oppimateriaalin tekstiä ja otetaan uudenlainen lähestymistapa HTML-koodin muokkaamiseen. Esim. WYSIWYG-editorit voidaan mainita, mutta niiden käyttöön uudessa materiaalissa ei keskitytä. Ns. online-editointisivut on hyvä pitää idealtaan samana uudessa materiaalissa, jolloin oppilas saa välittömän palautteen kirjoittamastaan koodista ja näkee muutokset heti. Oppilas lähtee työstämään koodia valmiilta pohjalta.

Oletetut esitiedot ennen jaksoa

Oppilas

- ymmärtää ohjelmoinnin ja algoritmisen ajattelun perusteet.
- on ohjelmoinut graafisessa ohjelmointiympäristössä.

Luvut ja lukukohtainen priorisointi

1. HTML perusteet (2 oppituntia)
    a. www-sivustorakenne
    b. elementit
    c. muotoilut (siirretään CSS-muotoiluun)
2. CSS perusteet (1 oppitunti)
    a. keskeisimmät muotoilut
3. JavaScript perusteet (3 oppituntia)

Jakson läpikäyminen vie yhteensä 6 oppituntia.

Jakson aikana tehtävä tehtäväkokonaisuus:

Oppilas saa aikaiseksi jakson päätteeksi verkkosivujen kokonaisuuden, joka käsittelee häntä itseään ja omia mielenkiinnon kohteita. Jakson sisältö linkitetään Teksinkäsittely-jaksoon tekemällä sähköinen versio Tekstinkäsittely -luvussa tehdyssä ansioluettelosta.

Tavoitteet

Tämän jakson jälkeen

- osaat luoda itsenäisesti yksinkertaisia verkkosivuja HTML-koodilla
    - ymmärrät HTML-koodin syntaksin: aloitus- ja lopetustagi, attribuutit, elementit ja osaat tarkistaa HTML-koodin oikeellisuuden
    - ymmärrät div-elementtien käyttämisen perusidean
- osaat muokata ulkoasua yhden ulkoisen CSS-tiedoston avulla
    - ymmärtää CSS:n syntaksin
- pystyt tuottamaan yksinkertaisen javascript-komennon ja muokkaamaan sivuja javascriptiä käyttäen

1. HTML-perusteet (2 oppituntia)

Tavoitteet

Tämän luvun jälkeen

- osaat luoda itsenäisesti yksinkertaisia HTML-sivuja validilla koodilla
    - ymmärrät HTML-koodin syntaksin: aloitus- ja lopetustagi, attribuutit, elementit
    - ymmärrät div-elementtien käyttämisen perusidean

Johdanto (otsikkoa ei tekstiin)

Internetin käytetyin osa, World Wide Web (WWW), koostuu verkkosivuista. Tässä luvussa opiskellaan periaatteet verkkosivujen tekemisestä koodia kirjoittamalla. Itse tekemiään verkkosivuja voi katsoa omalta tietokoneelta verkkoselaimella. Jotta sivut näkyisivät internetissä, tarvitaan palvelintilaa verkkosivuille, jota tarjoavat useat firmat, muun muassa teleoperaattorit. Nykyään useat firmat (esimerkiksi nettihotelli.fi) tarjoavat melko edullisesti myös nettihotellipalveluita. Niissä saa usein WWW-sivujen palvelintilan lisäksi sivut omaan domainiin, jolloin sivujen WWW-osoitteeksi voi valita esimerkiksi www.etunimisukunimi.net.

Tässä jaksossa puhutaan kolmesta käsitteestä: HTML:stä, CSS:stä ja JavaScriptistä. HTML (HyperText Markup Language) on verkkosivujen koodauskieli eli rakennustyökalu. CSS (Cascading Style Sheet) on taas tyyliohjeiden määritystyökalu. HTML:n avulla tehdään verkkosivujen raamit, mutta CSS:n avulla nämä raamit voidaan muotoilla ja asetella halutunlaiseksi. JavaScript on taas ohjelmointikieli, jolla voidaan lisätä toiminnallisuuksia verkkosivuille.

HTML

HTML:n uusin versio on HTML5 ja tässä oppimateriaalissa käytetään myös sitä. HTML:n tiedostopääte on .html. HTML-tiedostonimessä ei saa olla ääkkösiä, erikoismerkkejä tai välilyöntejä, sillä ne voivat aiheuttaa ongelmia sivujen selailussa joillakin selaimilla. Verkkosivuja voidaan katsella millä tahansa verkkoselaimella.

Syntaksi

Syntaksi tarkoittaa lauseoppia tai kielioppia. Koska tietokoneet ymmärtävät hyvin rajallisesti kirjoitettua tekstiä, on aluksi täytynyt määritellä yhteiset säännöt, miten tiettyjä asioita merkitään ja ilmaistaan. Jotta tietokone ymmärtää koodia, tulee koodin syntaksin (eli kieliopin) olla oikein. Koodin syntaksin voi tarkistaa validaattorilla, eli syntaksin tarkastajalla.

Elementit

HTML-dokumentti koostuu tekstistä, jossa merkitään dokumentin rakenne elementeillä (tageilla), jotka kirjoitetaan < > -merkkien väliin. Esimerkiksi ensimmäisen tason otsikko on merkitty seuraavasti seuraavassa esimerkissä: <h1>. Elementeillä on aina **aloitus- ja lopetustagi**, joiden välissä on varsinainen elementin sisältö. Esimerkiksi edellä mainitun otsikon tapauksessa sisältö esitetään seuraavasti: <h1>Ensimmäisen tason otsikko<h1>. <h1> on **aloitustagi**. Vastaavasti </h1> on **lopetustagi**. Lopetustagissa on aina /-merkki. Elementtien välillä ja tekstin keskellä saa olla vapaasti välilyöntejä, sarkaimia ja rivinvaihtoja. Niillä ei ole merkitystä elementin sisällön ulkoasuun. Jos siis esimerkiksi halutaan tekstiin enemmän kuin yksi välilyönti peräkkäin, täytyy ylimääräiset välilyönnit koodata erikoismerkeillä (esimerkiksi välilyönnin koodi on).

Useampi sisäkkäinen elementti tulee aloittaa ja lopettaa siten, että sisempi avattu elementti lopetetaan ennen ulomman elementin lopettamista: Moi on siis oikein, mutta Hei on taas väärin, sillä ulompi elementti (p-elementti) on suljettu ennen avattuaa strong-elementtiä.

HTML-koodia kirjoittaessa on suositeltavaa, että omat elementit - blokit - sisennetään omalle tasolleen (em. esimerkissä siis otsikko ja tekstiosio ovat sisennettynä, koska ne ovat samantasoisia elementtejä eli omissa blokeissaan). Tämä tekee koodista helpommin luettavampaa.

Yksinkertainen HTML-dokumentti

HTML:ää voi kirjoittaa tekstieditorilla (huom, ei tekstinkäsittelyohjelmalla). Windows-käyttöjärjestelmässä on valmiina tekstieditorina Notepad ja Mac OsX -käyttöjärjestelmässä TextEdit (suom. TeXturi). Valmiiksi asennetuista ohjelmista on yleensä karsittu kaikki ylimääräiset ominaisuudet pois: syntaksiväritys (elementtimerkinnät väritetään syntaksin ollessa oikea) ja automaattinen sisennys (rivinvaihdon yhteydessä elementit järjestellään sisemmäksi koodin lukemisen helpottamiseksi) ovat koodin kirjoittamista helpottavia ominaisuuksia, jotka ovat yleensä

Internetistä saatavissa ilmaisissa tekstieditoriohjelmissa (esim. Notepad++, Sublime Text, Vim, Atom).

<kuva>

<kommentti>

Pakolliset ominaisuudet omaava peruspohja HTML-sivusta on seuraava:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8”>
5      <title>Otsikko</title>
6  </head>
7  <body>
8
9      <h1>Ensimmäisen tason otsikko</h1>
10     <p>
11          Tämä on tekstikappale. <br />
12          Tähän tulee myös kuva myöhemmin.
13          </p>
14
15 </body>
16 </html>
```

Rivillä yksi kerrotaan tietokoneelle, että tiedosto on html-tyyppinen dokumentti. Rivillä 2 ja 16 on html-tagit. Riviltä kolme alkaa head-elementti, joka sisältää verkkosivun mm. otsikkotiedot ja tekstin merkistön koodaustavan. Head-elementissä kerrotaan tietokoneelle html-tiedoston tyylitiedosto, erilaiset muut linkit ja sivulla olevat skriptit. Head-elementti päättyy riville 6.

Riviltä seitsemän alkaa varsinainen html-sivun rakentaminen body-tagilla. Kaikki verkkosivuilla näytettävät elementit sijoitetaan body-tagien väliin.

Hyödyllisimmät elementit HTML5:ssä: laatikot, online-kokeilu

html	HTML-sivun koko sisältö on html-elementin sisällä. HTML-sivu siis alkaa html-aloitustagilla ja päättyy html-lopetustagiin.
head	Head-elementin sisään tulevat HTML-sivun otsikkotiedot, mm. title.
title	Sivun otsikko, joka näkyy Internet-selaimen otsikkorivillä. Sijoitetaan head-elementin sisälle.
body	Kaikki HTML-sivulla näkyvät elementit tulevat body-elementin sisään. Siispä esimerkiksi kaikki tässä taulukossa seuraavaksi esiteltävät elementit tulevat body-elementin sisään.
h1	Ensimmäisen tason otsikko, yleensä vain 1 kpl per sivu, eli käytetään sivun pääotsikkoon
h2-h3	Toisen ja kolmannen tason otsikko
p	Normaali tekstikappale

ul Järjestämätön lista (merkitty luettelo, listamerkit tulevat automaattisesti)

ol Järjestetty lista (numeroitu luettelo, listanumerot tulevat automaattisesti)

li Järjestämättömän tai järjestetyn listan lista-alkio (sijoitetaan ul- tai ol-elementin sisälle)

a Linkki (joko sivun sisälle tai toiseen Internet-osoitteeseen), elementin rakenne on seuraava: Linkin teksti. Osoitteessa ei saa olla ääkkösiä, erikoismerkkejä tai välilyöntejä. Linkin otsikko (title) näkyy, kun Internet-selaimessa viedään hiiren kursori linkin päälle. Linkkielementti ei voi esiintyä itsekseen, vaan sen tulee olla jonkun lohkotason elementin (esimerkiksi p, li, td tai div) sisällä.

img Kuva, elementin rakenne on seuraava: . Kuvan osoite (src) annetaan joko suhteellisena tai absoluuttisena viittauksena. Kuvan vaihtoehtoinen teksti (alt) näytetään selaimessa, jos kuvan lataaminen ei onnistu. Kuvan otsikko (title) näytetään, kun Internet-selaimessa viedään hiiren kursori kuvan päälle. Kuvaelementti tulee olla jonkun lohkotason elementin (esimerkiksi p, li, td tai div) sisällä.

table Taulukko

tr Taulukon rivi (sijoitetaan table-elementin sisälle)

th Taulukon otsikkosolu (sijoitetaan tr-elementin sisälle)

td Taulukon solu (sijoitetaan tr-elementin sisälle)

strong Korostuselementti, jolla saadaan lihavointi tekstille. Strong-elementti tulee olla jonkun lohkotason elementin (esimerkiksi p, li, td tai div) sisällä.

em Korostuselementti, jolla saadaan kursivointi tekstille. Em-elementti tulee olla jonkun lohkotason elementin (esimerkiksi p, li, td tai div) sisällä.

hr Vaakatasoinen viiva, syntaksi <hr />

br Pakotettu rivinvaihto, syntaksi
. Rivinvaihto tulee olla jonkun lohkotason elementin (esimerkiksi p, li, td tai div) sisällä.

div Yleinen lohkoelementti, jota käytetään esimerkiksi merkkaamaan sivu lohkoihin. Sen avulla saadaan esimerkiksi tehtyä marginaalit kerralla tietylle osalle sivusta, eikä marginaaleja tarvitse laittaa erikseen jokaiselle elementille.

span Span-elementtiä käytetään merkkaamaan tietty muotoilu (esimerkiksi tekstin väri) jollekin tekstin osalle esimerkiksi p-elementin sisällä. Span-elementti tulee olla aina jonkun lohkotason elementin (esimerkiksi p, li, td tai div) sisällä.

Lisää HTML5:n elementtejä löytyy esimerkiksi Mozillan kirjoittamasta html-dokumentaatiosta.

Attribuutit

Elementeillä voi olla ominaisuuksia eli *attribuutteja*, joilla määritellään elementtiin liittyviä ominaisuuksia. Esimerkiksi kuvan tapauksessa kuvan osoite, kuvan vaihtoehtoinen teksti ja kuvan otsikko määritellään img-elementtiin (kuva) tulevilla src (kuvan osoite), alt (kuvan vaihtoehtoinen teksti) ja title (kuvan otsikko) -attribuuteilla seuraavasti: . Attribuutin arvo siis syötetään lainausmerkkeihin = -merkin jälkeen. Ominaisuuksien lukumäärää ei ole rajoitettu.

Luokka ja tunniste

Yksi attribuutti, jota käsitellään tässä luvussa, on *class*, eli luokka. Luokka-attribuutin antamalla jokainen saman luokan omaava HTML-elementti saa saman muotoilun. Esimerkiksi tekstikappaleelle voidaan antaa luokka asettamalla attribuutiksi class="", tässä luokkanimi on “isompiteksti”:

```
<p class=”isompiteksti”>Tämä teksti on kirjoitettu isommalla fontilla. </p>
```

Huomaa, että luokkanimeä määritettäessä isoilla ja pienillä kirjaimilla on väliä. Siis “isompiteksti” ja “ISOMPITEKSTI” ja “IsompiTeksti” ovat kaikki eri luokkia. Luokka-attribuutti voidaan antaa mille tahansa HTML-elementille. Luokan muotoilu tapahtuu CSS:n avulla. CSS:stä kerrotaan seuraavassa luvussa.

Elementille voidaan myös määrittää tunniste eli id, tässä esimerkissä tunnisteeksi asetetaan “otsikko1”:

```
<h1 id=”otsikko1”>Tervetuloa!</h1>
```

Ero luokan ja id:n käytössä on se, että samaan luokkaan voi kuulua useampi elementti, mutta id on aina yksilöivä attribuutti. Esimerkiksi yläkoulussa usea(mpi?) oppilas voi kuulua saman luokan oppilaisiin, mutta jokaisella oppilaalla on oma henkilöturvatunnus, joka on yksilöllinen ominaisuus. <kommentti>

Jos elementillä ei ole tekstimuotoista sisältöä, niin lopetustagi lyhennetään suoraan aloitustagiin. Esimerkiksi pakotetun rivinvaihdon tapauksessa onkin HTML-tiedoston peruspohjan esimerkissä merkitty:
. Toinen esimerkki on kuvaelementistä, joka myös päätetään suoraan aloitustagiin: .

<tehtävä>

Aloitetaan sähköisen ansioluettelon tekeminen.

Avaa tekstieditori ja liitä sinne HTML-dokumenttipohja. Tallenna tiedosto html-nimiseen kansioon nimellä etusivu.html. Tallennettuasi tiedoston, voit avata sen selaimella.

Muokkaa tämän jälkeen HTML-tiedostoasi <title> -tagin (näyttää selaimen työkalupalkissa dokumentin nimen), otsikon ja tekstikappaleen osalta. Täydennä esittelyteksti itsestäsi ja lisää kaksi vapaavalintaista kuvaa etusivulle. Muista päivittää sivu aina muutosten jälkeen.

Lisää myös kaksi muuta sivua: koulutus.html ja vapaa-aika.html. Täydennä tekstisisällöt näille sivuille sivun nimen mukaisiksi.

</tehtävä>

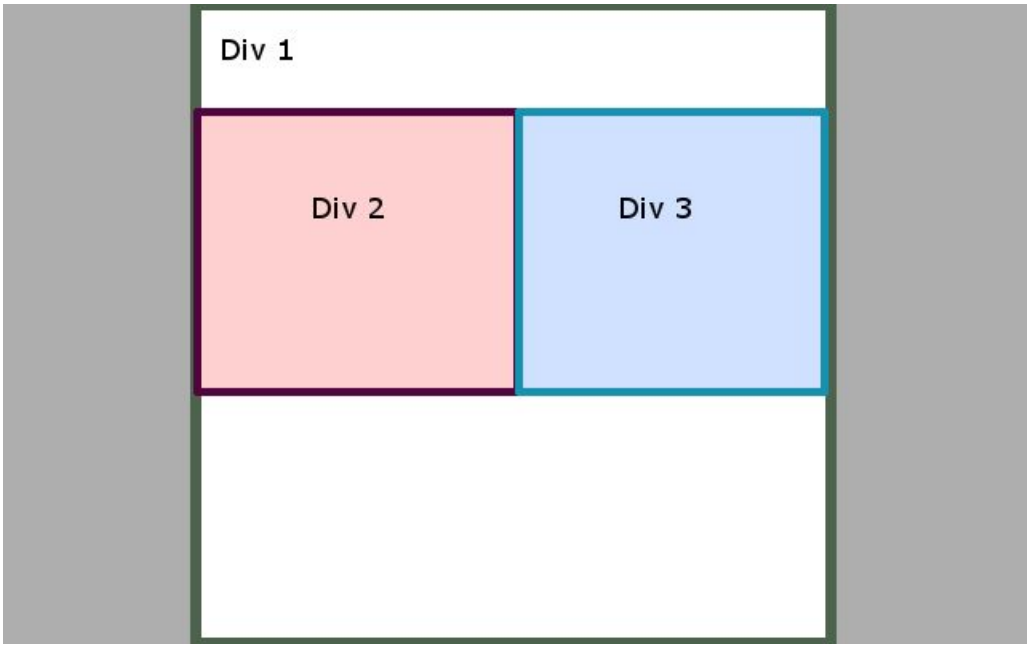
<kuva1>

<kuva2>

Div-elementit

Termi *div* tulee sanasta division (osa / osa-alue) ja div-tageja käyttämällä HTML-sivu voidaan jakaa erilaisiin osiin, joka helpottaa esimerkiksi muotoilujen ryhmittelyä. Div-elementtejä voidaan sisällyttää sisäkkäin.

<kuva3>



HTML-*sivu (harmaa), johon sisällytetty kolme diviä: Ensimmäisen divin (vihreät reunukset ja valkoinen pohja) sisällä on rinnakkaiset divit 2 ja 3.*

Esimerkki peräkkäin oleveista diveistä:

```
<div class=”sisalto">
      <h3>Otsikko</h3>
      <p>Ja tähän tulee tekstiä.</p>
</div>
<div class=”sivupalkki">
      <h3>Toinen otsikko</h3>
      <p>Ja tähän tulee myös tekstiä.</p>
</div>
```

<huomiolaatikkoi>Ääkköset eivät ole suositeltavia luokkien nimissä</huomiolaatikko>

Sisäkkäin olevat divit merkitään seuraavalla tavalla:

```
<div class=”kokonaisuus">
      <div class=”alkupuhe">
            <h1>Pääotsikko</h1>
            <p>Alkuteksti kotisivuilleni</p>
      </div>
      <div class=”tarkempitieto">
            <h2>Toinen otsikko</h2>
            <p>Tähän kirjoitan lyhyesti, kuka olen.</p>
      </div>
</div>
```

Linkittäminen

Verkkosivut koostuvat harvoin vain yhdestä HTML-sivusta; itse asiassa HTML nimensä mukaisesti on tarkoitettu *hypertekstin*, eli joukon toisiinsa linkitettyä tekstiä, merkitsemiseen. Verkkosivut voidaan ajatella hypertekstinä, sillä ne sisältävät linkkejä toisiin sivuihin. Verkkosivusto rakennetaankin tekemällä useita HTML-sivuja, joista viitataan toisiin sivuihin linkeillä (a-elementti). Sivuston pohjalle on järkevää luoda myös hakemistorakenne, eli kaikki sivut eivät sijaitse samassa hakemistossa, vaan loogisesti jaoteltuina eri hakemistoihin. Eri hakemistoihin ja hakemistoissa sijaitseviin tiedostoihin viittaaminen tapahtuu seuraavalla periaatteella:

Absoluuttiseen viittaukseen merkitään viitattavan kohteen koko verkko-osoite. Sitä kannattaa käyttää vain oman sivuston ulkopuolelle menevissä viittauksissa. Absoluuttinen viittaus on riippumaton sivusta tai sijainnista, jossa viittaus tehdään. Absoluuttinen viittaaminen voisi vertautua paikannukseen, esimerkiksi maantieteellisiksi koordinaateiksi 62°14′ 25″ N, 025°44′ 40″ E, sillä koordinaatit eivät ole riippuvaisia siitä, missä ne mainitaan. Absoluuttinen viittaus verkkosivuilla näyttää seuraavalta:
“http://www.example.com/” viittaa example.com-domainin juurihakemistoon “/” (käytännössä tämä palauttaa etusivun, esimerkiksi index.html).
“https://www.google.fi/search” viittaa google.fi-domainin “www”-nimiseen isännän “search”-sivuun sivuston juurihakemistossa “/”.
“/etusivu.html” on absoluuttisen viitteen erikoistapaus ja viittaa **nykyisen sivuston** juurihakemistosta “/” löytyvään “etusivu.html”-nimiseen tiedostoon.

Toisin kuin absoluuttinen viittaus, *suhteellinen viittaus* on riippuvainen sijainnista, jossa viittaus tehdään: käyttäen aiempaa paikannusvertauskuvaa, suhteellinen sijainti voisi olla esimerkiksi “viisisataa metriä etelään sieltä, missä nyt olet”. Verkkosivulla suhteellisella viittauksella tarkoitetaan kohteen sijaintia hakemistorakenteessa omalla sivustolla.

Suhteellinen viittaus hakemistoon tai tiedostoon toimii seuraavilla periaatteilla: <kuva?>

"etusivu.html" viittaisi saman hakemiston etusivu.html-tiedostoon.

"kuvat/" viittaisi saman hakemiston kuvat-nimiseen alihakemistoon (oletuksena hakemistosta näytettäisiin sivu index.html, jos sellainen löytyisi).

"kuvat/logo.jpg" viittaa saman hakemiston kuvat-alihakemistossa olevaan logo.jpg-tiedostoon.

"../" viittaa ylempään hakemistoon (hakemisto, jossa aktiivinen hakemisto sijaitsee).

"../sivut/" viittaisi ylemmässä hakemistossa sijaitsevaan sivut-nimiseen alihakemistoon.

Suhteellisten viittausten hyötyjä ovat viittausten toimivuus millä tahansa palvelimella ja viittausten pituus, sillä ne ovat lyhyempiä kuin absoluuttiset viittaukset.

<tehtävä>

Luo työstämääsi html-tiedostoon neljä diviä, joilla on luokat navigointi, teksti, kuvat ja alaviite. Sijoita divit siten, että navigointi on heti ensimmäisenä body-aloitustagin jälkeen. Tämän jälkeen divit “teksti” ja “kuvat” ja “alaviite”.

Navigointi: luo lista, johon teet kohdiksi “Esittely”, “Koulutus” ja “Vapaa-ajan harrastukset”. Laita jokainen listan teksti linkiksi suhteellisella viittauksella.

<huomiolaatikko?>

Numeroimattoman listan luonti käy seuraavasti:

Ensimmäinen kohta

Toinen kohta

</huomiolaatikko?>

Teksti: aiemmin luomasi teksti tulee tähän.

Kuvat: siirrä aiemmin lisäämäsi kuvat tähän div-elementin sisään.

Alaviite: Kirjoita tähän © ja oma nimesi, lisäksi vuosiluku.

</tehtävä>

Seuraavaksi lisätään CSS-muotoiluja sivulle.

Kuvat

Kuva: syntaksiväritys ja rivinumerointi Notepad++-ohjelmalla

Kuva: HTML-tiedosto kansiossa

kuva 2: html-tiedosto selaimessa

kuva 3: div-elementtien jaottelu sivulla

Huomiolaatikot

ääkköset

a. www-sivustorakenne
b. elementit

2. CSS-perusteet (1 oppitunti)

CSS tulee sanoista Cascading Style Sheets, ja sillä muotoillaan HTML-elementtejä. HTML-tiedostoon kirjoitetaan siis pelkästään sivun elementit, sisältö ja tarvittaessa attribuutteja.

CSS:n syntaksi on helppo:

```
ominaisuus: arvo;
```

HTML-elementin muotoilua voitaisiin tehdä suoraan **upottamalla** ominaisuus-arvo-parit CSS-syntaksin mukaisesti. Mikäli CSS:ää upotettaisiin HTML:ään, se tulisi muodossa

```
<h1 style="ominaisuus: arvo;">Ensimmäisen tason otsikko</h1>
```

ja upotuksia voisi tehdä useamman:

```
<h1 style="ominaisuus: arvo; ominaisuus2: arvo;">Ensimmäisen tason otsikko</h1>
```

Elementin ominaisuutta määriteltäessä siis ensin määritellään mikä on **ominaisuus**, ja kaksoispisteen jälkeen sille asetetaan **arvo**. Esimerkki ominaisuudesta ja arvosta on tekstielementin p väri **color**, jonka arvoksi asetetaan musta eli **black**. Arvoa seuraa puolipiste (;), joka erottaa ominaisuudet toisistaan.

Tässä luvussa tyylimuotoilut kuitenkin opetetaan tekemään erilliseen tiedostoon, sillä muotoilujen määritteleminen jokaiselle elementille on aikaavievää ja raskasta.

HTML-tiedostosta viitataan CSS-tiedostoon, eli sieltä haetaan ulkoasumääritykset WWW-sivulle. Hyötynä on HTML-koodin pysyminen selkeänä sekä se, että samaa tyylitiedostoa voivat käyttää useat HTML-sivut. Tällä tavalla esimerkiksi ison WWW-sivuston tyylimääritykset voidaan kirjoittaa vain yhteen paikkaan. CSS-tiedoston pääte on .css .

Yksinkertainen CSS-tiedosto

```
body{
    margin: 0;
    padding: 0;
}
```

```
p {
	font-family: Arial;
	color: black;
	line-height: 1.7;
}
```

Body-elementin määrittelyt tekevät dokumentin kattavat yleiset, eli globaalit muotoilut koko sivulle. Huomaa aaltosulut, joiden sisään muotoilut sijoitetaan. Edellisessä esimerkissä määritellään yleisten muotoilujen lisäksi tekstikappaleiden fontiksi Arial, fonttiväriksi musta ja tekstikappaleen riviväliksi 1.7. CSS-tiedostossa voi määrittää muotoiluja tarvittavan määrän, eli mikään elementti ei ole CSS-tiedostossa pakollinen. <laatikko1> Yleisimmät css-muotoilut

Mikäli useampi elementti määritellään samoilla arvoilla, ne voidaan ryhmitellä pilkuilla eroteltuina:

```
h2, h3{
	color: red;
	text-decoration: underline;
	text-align: center;
}
```

Linkittäminen HTML-tiedostoon

CSS-tiedosto linkitetään HTML-tiedostoon sijoittamalla seuraava koodi HTML-tiedostoon <head> </head>-tagien väliin:

```
<link rel="stylesheet" type="text/css" href="tyyli.css" />
```

<tehtävä

Tee siis nyt oma CSS-tiedosto tekstieditorilla ja linkitä se html-tiedostoosi. Lisää yleiset muotoilut (body), muotoilut tekstikappaleille (p), eri otsikkotasoille (h1-h3).

</tehtävä>

<kommentti>

Luokkien ja id:n muotoilu

<kommentti>

Luokkien muotoilu tapahtuu helpoiten CSS:llä.

Esimerkiksi Tekstikappale -elementtiä ja <h1 id=”ensimmainen”>Otsikko</h1> -otsikkoa voitaisiin muokata seuraavasti:

.sisalto{

```
        float: left;
        text-align: center;
        max-width: 75%;
}
#ensimmainen{
        text-align: center;
}
```

Luokka-attribuutti ilmoitetaan CSS-tiedostossa laittamalla luokan nimen eteen piste, id:n tyylimäärityksissä id ilmoitetaan merkitsemällä sitä risuaidalla (#). Jos p-elementille määrätyissä tyylimäärityksissä olisi päällekkäisyyksiä sisalto-luokalle määrättyjen tyyliominaisuuksien kanssa, voimaan jäisivät sisalto-luokan tyylimääritykset. Luokalle määrätyt tyyliominaisuudet siis kumoavat (ylikirjoittavat) elementeille määrätyt tyyliominaisuudet.

Esimerkki ylikirjoittavista tyyliominaisuuksista:
Asetetaan h2-elementin yleiseksi fontiksi Tahoma:

```
h2{
font-family: “Tahoma”;
}
```

Tämän jälkeen asetetaan h2-elementin valiotsikko -luokan fontiksi Arial:

```
h2.valiotsikko{
font-family: “Arial”;
}
```

ja HTML-koodissa luodaan kaksi eri otsikkoa:

```
<h2>Nisäkkäät</h2>
<h2 class=”valiotsikko”>Linnut</h2>
```

Nyt selaimella verkkosivua tarkastellessa otsikoida huomataan, että Nisäkkäät-otsikko on Tahoma-fontilla, mutta Linnut-otsikko puolestaan Arialilla, vaikka sekin on h2-elementti.

<lisätietoa>Validointi</lisätietoa>

Verkkosivuja tehdessä koodiin voi helposti tulla virheitä. Esimerkiksi tietyn elementin lopputagi voi unohtua melko helposti. Jos koodissa on syntaksivirheitä tai päättämättömiä tageja, voi ulkoasu

näyttäytyä virheellisenä. Verkkosivun "kieliopin" oikeellisuuden eli validisuuden voi onneksi tarkistaa helposti Internetissä olevan palvelun (validaattorin) avulla. Tarkistaminen tapahtuu seuraavasti:

Mene WWW-osoitteeseen http://validator.w3.org/.
Syötä sen verkkosivun verkko-osoite, jonka validisuuden haluat tarkistaa, kohtaan Address ja paina painiketta Check. Voit myös ladata tiedoston tarkistettavaksi omalta koneeltasi, se tapahtuu välilehden Validate by File Upload kautta.
Hetken kuluttua avautuu verkkosivu, joka kertoo englanniksi oliko sivullasi virheitä. Validaattori kertoo virheiden lukumäärän, millaisia mahdolliset virheet ovat ja millä koodirivillä virheet sijaitsevat.
Saadun informaation perusteella pystyt tarvittaessa korjaamaan verkkosivusi validiksi. Tarkista korjausten jälkeen sivusi validaattorilla vielä uudestaan.

Edellä mainittu validaattori tarkistaa sivun HTML-koodin, eikä se puutu mahdollisiin CSS-tyylivirheisiin. Niiden tarkistamiseen on myös oma validaattori Internetissä, jolla tarkistaminen tapahtuu seuraavasti:

Mene osoitteeseen http://jigsaw.w3.org/css-validator/.
Syötä sen verkkosivun WWW-osoite, jonka tyylimääritysten validisuuden haluat tarkistaa, kohtaan Address ja paina painiketta Check. Voit myös ladata tiedoston tarkistettavaksi omalta koneeltasi, se tapahtuu välilehden By File Upload kautta. Jos olet tehnyt tyylimääritykset erilliseen CSS-tyylitiedostoon, voit syöttää osoitteeksi myös kyseisen tiedoston osoitteen tai ladata kyseisen tiedoston omalta koneeltasi tarkistettavaksi. Validaattori toimii samalla tavalla kuin HTML-validaattori.

<tehtävä>

Lisää tämän CSS-tiedoston määrittelyt: <tiedosto.css> työstämääsi HTML-sivuusi
Muotoile tämän jälkeen CSS:ää käyttäen HTML-luvussa tekemäsi divit seuraavalla tavalla:

- ul-elementille
  - taustaväri #ADD8E6
  - marginaali 0, täyte 1em
  - lisää rivit:
    - list-style-type: none;
    - overflow: hidden;
- li-elementeille

    - kellunta vasemmalle
- Linkeille
    - fontti arial
- Asemoi teksti-div vasemmalle ja kuvat-div oikealle
    - Aseta teksti-divin leveydeksi 60% ja kuvat-divin leveydeksi 40%
- Alaviitteelle
    - taustaväri vihreä
    - tekstin koko oletusarvoa pienemmäksi.

</tehtävä>

Kuvat

Huomiolaatikot / laajenevat laatikot

- **Yleisimmät CSS-muotoilut**

- **Taustaväri**
- background-color: väri;
- esim. background-color: white;

- **Fontti ja tekstin asemointi**
- font-family: fontti;
- color: väri;
- text-align: left|right|center|justify;
- font-size: <huomiolaatikko>fonttikoot</huomiolaatikko>

- **Taulukon reunat**
- border: leveys tyyli väri;
- esim. border: 2px solid #400222;

- **Korkeus ja leveys**
- width: auto|arvo|initial|inherit;
- height: auto|arvo|initial|inherit;

- **Marginaali ja täyte**
- Määritykset marginaalille ja täytteelle tapahtuvat samalla tavalla. Marginaali ja täyte ovat siitä erilaisia määriteltäviä, että ne lähtökohtaisesti asettavat määritellyn arvon jokaiselle neljälle sivulle. Tämän takia ei tarvitse määritellä haluttua arvoa jokaiselle erikseen, esim.
    - margin-top
    - margin-left

    - margin-right
    - margin-bottom
- Joten jos määrittelet yhden arvon, asetat sen jokaiselle neljälle sivulle. Jos asetat arvoja kaksi kappaletta, esim.
    - padding: 0 2em;
- Tällöin ensin asetetaan elementin ylä- ja alapuolen arvot, tämän jälkeen vasemman ja oikean puolen arvot. Tässä tapauksessa padding-top ja padding-bottom olisivat 0, padding-left ja padding-right taas 2em.
- Jokainen arvo voidaan kuitenkin määritellä erikseen yhdellä rivillä, tällä tavalla:
    - padding: ylä oikea ala vasen;
    - padding: 0 2em 1em 3em;

- **Elementin asemointi / kellunta**
- Voidaan käyttää esim. kuvan, divin tai navigointilinkkien asemointiin
- float: none|left|right|initial|inherit;

Ruutukaappausvideot

Video: yleisimmät html+css-muotoilut (esim. padding/margin erojen havainnollistaminen)

3. JavaScript-perusteet (3 oppituntia)

JavaScript on ohjelmointikieli. Sitä käytetään yleisesti HTML:n ja CSS:n kanssa yhdessä, sillä sen avulla voidaan lisätä helposti toiminnallisuutta verkkosivulle. JavaScriptiä voidaan käyttää myös pelien ohjelmoinnissa ja mobiilisovellusten luomisessa. Nimensä takia sitä ei tule sekoittaa Java-ohjelmointikieleen.

JavaScriptiä voi kirjoittaa suoraan HTML-dokumenttiin joko sijoittamalla koodin <head></head> -tagien väliin, tai varsinaisen sivun sekaan merkitsemällä skriptiä <script></script> -tageilla, mutta helpon hallinnoimisen takia se kannattaa kirjoittaa omaan tiedostoon. JavaScript -tiedoston tunnistaa .js-tiedostopäätteestä.

Ohjelmoinnin peruspalikat

Muuttujat ja tyypit

Muuttuja on muistista varattu alue, johon voidaan tallentaa tietoa. Muuttuja julistetaan (eli kerrotaan tietokoneelle, että muistista on varattava alue) tunnisteella *var*. Esimerkiksi muuttujaan *ika* voitaisiin heti muuttujan julistamisen yhteydessä tallentaa numero 25:

```
var ika = 25;
```

<huomiolaatikko>
Kannattaa lukea tiedon tallennusta ikään kuin oikealta vasemmalle: “sijoitetaan arvo 25 muuttujaan ika”. Muuttujaan ei JavaScriptissä tarvitse aluksi tallettaa mitään tietoa, vaan se voidaan vain alustaa (var ika;)
</huomiolaatikko>

Muuttujia voi myös määrittää aiempien muuttujien perusteella operaattoreilla:

```
var vanhempi = ika + 20;
```

Tällöin muuttujalla *vanhempi* arvo olisi 25+20 eli 45.

<lisätietoa>Joissain muissa ohjelmointikielissä muuttujan tyyppi tulee julistaa muuttujan määrittelyssä, esim. C#:

int ika = 25;

Jolloin ika muuttujalle määritetään tyyppi int, kokonaisluku. Muita muuttujatyyppejä on kirjaimet (char), kirjainjonot (string), totuusarvo (boolean), long (liukuluku). JavaScriptsissä ei tyyppimäärityksiä tarvitse tehdä, sillä muuttuja julistetaan tunnisteella var (sanasta variable).</lisätietoa>

Silmukka

Silmukoita käyttämällä voidaan automatisoida toistuvien koodirivien kirjoittaminen ja esittää ne tiiviimmin.

<huomiolaatikko>eri silmukoista ja niiden käyttökohteista</huomiolaatikko>

For-silmukan syntaksi on seuraava

```
for (alustuslauseke; ehtolauseke; lauseke joka suoritetaan
suorituksen jälkeen) {
    koodi, joka suoritetaan
}
```

Alustuslausekkeita voi olla useampi. Ehtolauseke tulee muotoilla siten, että silmukan suoritus päättyy jossain vaiheessa, muuten tuloksena on ikuinen silmukka. Suoritettujen koodilausekkeiden jälkeen yleensä kasvatetaan tai vähennetään alustuslausekkeessa määritettyä muuttujaa. Silmukan syntaksi selvenee parhaiten katsomalla esimerkkiä:

While-silmukan syntaksi on seuraava:

Do-while -silmukan syntaksi on seuraava:

--

Sen sijaan, että lisäisit listan kuvia useaan kertaan HTML-koodina:

```
<div id=”kuvat”>
    <img scr=”kuva1.jpg” alt=”kuva1” /><br />
    <img scr=”kuva2.jpg” alt=”kuva2” /><br />
    <img scr=”kuva3.jpg” alt=”kuva3” /><br />
    <img scr=”kuva4.jpg” alt=”kuva4” /><br />
    <img scr=”kuva5.jpg” alt=”kuva5” /><br />
</div>
```

voidaan lisääminen tehdä silmukan avulla. Koska näemme lisättäviä kuvia olevan viisi, voimme tehdä silmukan, joka toistuu viisi kertaa:

```
for (var i = 1; i < 6; i++){ [alustetaan silmukka, asetetaan muuttujan arvoksi 1, ehtona on että muuttujan arvo on pienempi kuin 6, ja jokaisella kierroksella kasvatetaan muuttujaa yhdellä.]
      var kuvat += ‘<img scr=”kuvat”’ + i + ‘.jpg /><br />’
[kirjoitetaan jokainen rivi yksitellen silmukan avulla, numero muuttuu aina välissä.]
      document.getElementById.innerHTML(“kuvat”) = kuvat;
      }
```

<tehtävä>

Tee silmukka, jossa lasket yhteen numerot yhdestä kymmeneen (1+2+3+ ... +10). Toteuta erikseen for, while ja do-while -silmukat. Tulosta silmukan tulos HTML-elementtiin sivullesi.

</tehtävä>

Ehtolauseet

<kommentti>

if, else, if-else

Erilaisilla ehdoilla voidaan määrittää erilaisia toimintoja ja täten säädellä koodin suorittamista. Esimerkiksi HTML-sivulle voitaisiin lisätä otsikko, jossa toivotetaan vierailija tervetulleeksi senhetkisen kellonajan mukaan. Tällöin algoritmi voisi mennä

- “Katso kelloa. **Jos** kello on vähemmän kuin iltakuusi, toivota “hyvää päivää” <kommentti>

```
if (kello < 18:00){
      tervehdys = “Hyvää päivää”;
}
```

Jos tarkistettava ehto on epätosi arvoltaan ja toiselle vaihtoehdolle halutaan toimenpide, voidaan asettaa if-else -rakenne:

- “Katso kelloa. **Jos** kello on vähemmän kuin iltakuusi, toivota “hyvää päivää”
- **Muuten**, toivota “hyvää iltaa”.

```
if (kello < 18:00){
     tervehdys = “Hyvää päivää”;
}
else {
     tervehdys = Hyvää iltaa”;
}
```

Else-if -lauseella voimme lisätä tarkistettavan ehdon sille tapaukselle, että ensimmäinen ehtolause on epätosi. Tällöin esimerkki voisi muuttua hieman:

- “Katso kelloa. **Jos** se on vähemmän kuin aamuyksitoista, toivota “hyvää huomenta”.
- **Muuten jos** kello on vähemmän kuin iltakahdeksan, toivota hyvää päivää.
- **Muuten** toivota hyvää iltaa.

```
if (kello < 11:00){
     tervehdys = “Hyvää huomenta”;
}
else if (kello < 20:00){
     tervehdys = “Hyvää päivää”;
}
else {
     tervehdys = “Hyvää iltaa”;
}
```

<kommentti>

Funktiot

Funktioita käyttämällä voidaan useampi kerrallaan suoritettava komentojen sarja nimetä yhdelle nimelle. Tätä komentosarjaa voidaan myös käyttää aina vain kun sitä tarvitaan, eikä välttämättä siis aina ohjelmakoodia läpi käytäessä.

Funktiota kutsutaan JavaScriptissä:

```
funktionNimi(parametri1, parametri 2, .. parametriN);
```

Esimerkiksi suorakulmion pinta-alan laskemiseen voimme määritellä funktion, joka ottaa syötteinä, eli *parametreinä*, suorakulmion pituuden ja leveyden ja lopuksi tulostaa vastaavan suorakulmion pinta-alan.

```
suorakulmionPintaAla (pituus, leveys){
     ala = pituus * leveys
     tulosta (ala)
}
```

<kommentti>

Objekteista ja metodeista

JavaScript on oliopohjainen ohjelmointikieli. Tämä tarkoittaa sitä, että ohjelmoidessa voidaan luoda erilaisia olioita eli objekteja (*Object*), ja näille eri ominaisuuksia sekä ominaisuuksista riippuvaisia funktioita eli metodeja. Otetaan esimerkiksi olio nimeltä *henkilo*:

```
var henkilo = {
nimi: [‘Matti’, ‘Meikalainen’],
syntymaaika: date(1995, 4, 28),
paino: 67,
pituus: 175,
tervehtii: function() {
    return ‘Moi! Olen ' + this.name[0] + '.';
},
vaatteet: {
     paita: “t-paita”,
     housut: “farkut”
     }
}
```

Nyt siis kokoelma erilaisia ominaisuuksia (kuten *ika*, *paino* ja *pituus*) on tallennettu oliomuotoisesti muuttujaan henkilo. Lisäksi henkilon ominaisuuteen *vaatteet* on tallennettu sisäkkäinen olio, jolla on ominaisuudet *paita* ja *housut*, ja nimi on tallennettu taulukkona (Array), jossa ensimmäisellä paikalla on etunimi ja toisella sukunimi. Huomaa, että taulukossa indeksointi alkaa nollasta, toisin sanoen ensimmäinen taulukon solu löytyy kohdasta 0, toinen kohdasta 1, jne.

Olion ominaisuuksiin pääsee käsiksi esimerkiksi seuraavasti:

henkilo.nimi[1] *// palauttaa Meikalainen*

henkilo.paino *// palauttaa arvon 67*

henkilo.tervehtii() *// palauttaa “Moi! Olen Matti.”*

henkilo.vaatteet.paita *// palauttaa arvon t-paita*

<Tehtävä>

Ota uusi HTML-tiedosto tästä: <tiedosto>

Kirjoita HTML-tiedostoosi tyhjä tekstikappale, jonka **id** on “terve”.

Tee uusi tiedosto testi.js, ja tallenna se uuteen kansioosi *skriptit*. Kirjoita testi.js-tiedostoon:

```
document.getElementById("terve").innerHTML = "Tämä toimii
JavaScriptillä!";
```

Lisää HTML-tiedostoosi seuraava rivi, juuri ennen </body> -tagia

```
<script src="skriptit/testi.js"></script>
```

Avaa nyt tekemäsi uusi html-tiedosto. Löytyykö JavaScriptillä luotu teksti sivuilta?

</tehtävä>

Scriptin kutsurivi sijoitetaan HTML-koodiin sijoitettaessa mahdollisimman loppuun, mutta ennen body-tagia, jotta HTML-sivu on luonut kaikki tarvittavat elementit sivulle, ennen kuin komento suoritetaan.

Edellisessä esimerkissä skripti hakee dokumentista (objekti!) elementin id:den perusteella, ja tämä haettava id on määritelty suluissa terve-id:ksi. Tämän elementin innerHTML-ominaisuus on kytköksissä kyseisen elementin sisältöön, ja tähän uuden arvon sijoittamalla elementin esitys muuttuu sivulla.

JavaScriptille on tyypillistä, että *komennon kohteen tarkentaminen* tapahtuu komennon edetessä.
Esim.

```
document.getElementById("terve").style.fontSize = "3em";
```

Tällöin jokaisen pisteen vasemmalla puolella olevalta objektilta kysytään pisteen oikealla puolella olevaa ominaisuutta. Tässä tapauksessa dokumentilta pyydetään elementti id:n perusteella, jonka tyyliä muutetaan fonttikoon osalta.

Eventit eli tapahtumat

JavaScriptiä käytettäessä HTML:n kanssa sivuille voidaan lisätä toiminnallisuutta. Tämä tapahtuu erilaisten event-käsittelijöiden avulla.
Esimerkiksi voidaan luoda button -elementti, johon sisällytetään event-käsittelijä: kun nappia painetaan, tapahtuu päivämäärä.

<button onclick="document.getElementById('demo').innerHTML=Date()">Näytä päivämäärä ja kellonaika</button>

<kommentti>

<huomilaatikko>Yleisiä tapahtumankäsittelytapoja</huomiolaatikko>

Huomiolaatikot

Silmukoita on kolmea eri tyyppiä: for, while, ja do-while.

- For-silmukalla toistetaan lukumäärältään vakiosti määritellyt kerrat. Aluksi määritellään alkuasetukset, jossa muuttuja toimii laskurina. Toistoja tehdään määrätty määrä toistoehdon täyttyessä.
- While-silmukalla aina aluksi tarkastetaan toistoehto, ja toistetaan sitä niin kauan, kuin ehto on tosi. While-silmukalle tulee määrittää toimenpide, joka vaikuttaa toiston lopetukseen (esim. numeerisen muuttujan arvon muuttaminen), muuten tuloksena on ikuinen silmukka.
- Do-while -silmukalla aluksi suoritetaan komennot, jotka on määritelty alussa. Tämän jälkeen siirrytään tarkistamaan ehtoa, jonka ollessa tosi, siirrytään taas suorittamaan alussa määritettyjä komentoja. Kun ehto on epätosi, silmukasta poistutaan.

Silmukoita voidaan laittaa myös sisäkkäin.

<tähän silmukan syntaksit esimerkkeinä>

Metodeista

Tapahtumat

onchange HTML-dokumentti on muuttunut
onclick Käyttäjä klikkaa HTML-elementtiä
onmouseover Käyttäjä kuljettaa hiiren elementin päälle
onmouseout Käyttäjä kuljettaa hiiren elementin päältä pois
onkeydown Käyttäjä painaa näppäimistöstä näppäintä
onload Kun selain on ladannut näkymän täysin

Kesken! - Lopullinen tehtävä - tarkistimen tekeminen

HTML-pohja <tästä> ...

Lisätehtävä - CSS-tiedoston vaihtaminen nappia painamalla

HTML-pohja <tästä>

Tee kaksi css-tiedostoa: toisessa taustaväri on XXX, fontti Arial, fontin koko 1.2em, toisessa taustaväri YYY, fontti Verdana, fonttikoko 0.8em ....