7. Verkkosivut 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

Jakson projektitehtävä

Tässä jaksossa toteutetaan laajempi projektitehtävä, jota työstetään koko jakson aikana. Tehtävä rakentuu siten, että jokaisessa luvussa opiskeltu uusi asia lisätään tehtävään eli tehtävä rakentuu pala palalta opiskeltujen asioiden myötä.

Jakson projektitehtävänä on jaksossa 3 tehdyn ansioluettelon muuttaminen sähköiseen muotoon eli omaksi verkkosivustokseen.

1. HTML-perusteet (2 oppituntia)

<kommentti>

Tavoitteet

Tässä luvussa

- opit luomaan yksinkertaisen ja validin verkkosivun.
- opit HTML-merkintäkielen syntaksin: aloitus- ja lopetustagi, elementit, attribuutit.
- opit käyttämään div-elementtiä.
- opit viittaamaan verkkosivulta toiselle.

Johdanto (otsikkoa ei tekstiin)

Internetin käytetyin osa, **World Wide Web (WWW)**, koostuu verkkosivuista. Tässä luvussa opiskellaan verkkosivujen tekemisestä **HTML-merkintäkieltä** kirjoittamalla. Itse tehtyjä verkkosivuja voidaan katsoa omalta tietokoneelta verkkoselaimella. Jotta sivut kuitenkin näkyisivät kaikille internetissä, tarvitaan verkkosivuille **palvelintilaa**. Palvelintilaa tarjoavat useat yritykset, muun muassa teleoperaattorit. Useat yritykset (esimerkiksi nettihotelli.fi) tarjoavat melko edullisesti myös nettihotellipalveluita. Niissä on mahdollista saada verkkosivujen palvelintilan lisäksi oma **domain eli verkkotunnus**, jolloin verkkosivun **WWW-osoitteeksi eli verkko-osoitteeksi** voidaan valita omavalintainen osoite, esimerkiksi www.etunimisukunimi.net.

Tässä jaksossa käsitellään kolmea verkkosivujen tekemiseen liittyvää osa-aluetta: HTML:ää, CSS:ää ja JavaScriptiä. **HTML (HyperText Markup Language)** on verkkosivujen hypertekstin merkintäkieli eli ns. rakennustyökalu verkkosivujen tekemiseen. **CSS (Cascading Style Sheet)** tarkoittaa verkkosivun tyyliohjeita eli sen avulla määritellään verkkosivun ulkoasu. Toisin sanoen HTML-merkintäkielellä tehdään verkkosivuille rakenne ja sisältö, mutta CSS:n avulla verkkosivujen muotoilu ja asettelu saadaan halutunlaiseksi. **JavaScript** on sen sijaan ohjelmointikieli, jolla voidaan lisätä verkkosivuille toiminnallisuutta.

HTML

HTML:n uusin versio on **HTML5**, jota myös tässä oppimateriaalissa käytetään. **HTML-tiedoston** tiedostopääte on .html. Kun HTML-tiedostoa tarkastellaan verkkoselaimella, tiedostosta tulee verkkosivu. HTML-tiedostoja voidaan tarkastella millä tahansa verkkoselaimella. HTML-tiedoston nimessä ei saa olla ääkkösiä, erikoismerkkejä tai välilyöntejä, sillä ne voivat aiheuttaa ongelmia selailussa joillakin selaimilla.

Syntaksi

Syntaksi tarkoittaa HTML-merkintäkielen ns. kielioppia. Koska tietokoneet ymmärtävät rajallisesti kirjoitettua tekstiä, täytyy aluksi määritellä yhteiset säännöt, joiden mukaan tiettyjä asioita merkitään ja ilmaistaan. Jotta tietokone ymmärtäisi kieltä, tulee kielen syntaksin olla oikein. Koodin syntaksin oikeellisuus voidaan tarkistaa validaattorilla eli syntaksin tarkastajalla.

Elementit

HTML-tiedosto koostuu tekstistä, jossa dokumentin rakenne merkitään **elementeillä**. Elementit merkitään **tagien** väliin. Elementeillä on aina **aloitus- ja lopetustagi**, joiden välissä on varsinainen elementin sisältö. Esimerkiksi ensimmäisen tason otsikko merkitään h1-elementillä seuraavasti:

```
<h1>Ensimmäisen tason otsikko</h1>
```

<h1> on aloitustagi ja se avaa elementin. </h1> on lopetustagi ja se sulkee elementin. 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 tekstiin halutaan esimerkiksi enemmän kuin yksi välilyönti peräkkäin, täytyy ylimääräiset välilyönnit merkitä erikoismerkeillä (esimerkiksi välilyönnin koodi on).

Jos elementillä ei ole tekstimuotoista sisältöä, lopetustagi voidaan lyhentää suoraan aloitustagiin. Esimerkiksi pakotettu rivinvaihto voidaan merkitä kokonaisuudessaan
. Myös kuvaelementti voidaan päättää suoraan aloitustagiin .

Useampi sisäkkäinen elementti tulee aloittaa ja lopettaa siten, että sisempi avattu elementti suljetaan ennen ulomman elementin sulkemista. Moi on siis oikein, mutta Hei on taas väärin, sillä ulompi p-elementti on suljettu ennen sisempää strong-elementtiä.

Yksinkertainen HTML-dokumentti

HTML-merkintäkieltä voidaan kirjoittaa **tekstieditoriohjelmalla** (ei kuitenkaan tekstinkäsittelyohjelmalla). Käyttöjärjestelmiin valmiiksi asennetuista ohjelmista on yleensä karsittu ylimääräiset ominaisuudet pois, kuten syntaksiväritys ja automaattinen sisennys. Syntaksiväritys tarkoittaa, että elementtimerkintöjä selkeytetään erilaisin värimerkinnöin ja automaattisessa sisennyksessä elementit sisennetään koodin lukemisen helpottamiseksi. Nämä ovat merkintäkielen kirjoittamista helpottavia ominaisuuksia, jotka ovat yleensä Internetistä saatavissa ilmaisissa tekstieditoriohjelmissa.

<huomiolaatikko 1>

Valmiiksi asennettuja tekstieditoriohjelmia

- Notepad
- TeXturi

<huomilaatikko 2>

Ilmaisia tekstieditoriohjelmia

- Notepad++
- Sublime Text
- Vim
- Atom

<kuva 1>

Pakolliset ominaisuudet omaava HTML-tiedoston peruspohja on seuraavanlainen:

```
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 1 kerrotaan tietokoneelle, että tiedosto on HTML-tyyppinen dokumentti. Seuraavalla rivillä 2 on html-aloitustagi. Verkkosivun koko sisältö on html-elementin sisällä. Html-elementin lopetustagi on tiedoston viimeinen tagi rivillä 16. Riviltä 3 alkaa head-elementti, joka sisältää tekstin merkistön koodaustavan (utf-8) ja <title> -tageilla merkityn verkkosivun otsikon. Otsikko näkyy verkkoselaimen välilehdellä. Otsikko kannattaa valita informatiiviseksi, sillä pidemmät otsikot eivät näy kokonaan välilehdellä. Lisäksi head-elementissä voidaan kertoa tietokoneelle html-tiedoston käyttämä tyylitiedosto, erilaiset muut linkit ja verkkosivulla olevat skriptit. Head-elementti päättyy riville 6.
Riviltä 7 alkaa varsinaisen verkkosivun rakentaminen body-tagilla. Kaikki verkkosivuilla näkyvät elementit sijoitetaan body-tagien väliin.

HTML-merkintäkieltä kirjoittaessa on suositeltavaa, että omat elementit eli 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.

Hyödyllisimmät elementit HTML5:ssä: laatikot, online-kokeilu

h1	Ensimmäisen tason otsikko eli pääotsikko, yleensä vain yksi per verkkosivu.
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 verkkosivuston sisälle tai toiseen internetosoitteeseen. Linkkielementti ei voi esiintyä itsekseen, vaan sen tulee olla jonkun lohkotason elementin (esimerkiksi p, li, td tai div) sisällä.

img Kuva. 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 lihavoitua tekstiä. Strong-elementti tulee olla jonkun lohkotason elementin (esimerkiksi p, li, td tai div) sisällä.

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

hr Vaakatasoinen viiva, syntaksi <hr /> eli lopetustagi voidaan lyhentää suoraan aloitustagiin

br Pakotettu rivinvaihto, syntaksi
 eli lopetustagi voidaan lyhentää suoraan aloitustagiin. 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 sivun osaan, jolloin 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- elementtejä löytyy esimerkiksi Mozillan kirjoittamasta html-dokumentaatiosta.

Attribuutit

Elementeillä voi olla **ominaisuuksia eli attribuutteja**, joilla määritellään elementtiin liittyviä ominaisuuksia. Esimerkiksi kuva lisätään img-elementillä, jolle voidaan määritellä osoite, kuvan vaihtoehtoinen teksti ja kuvan otsikko. Osoite määritellään src-attribuutilla, vaihtoehtoinen teksti alt-attribuutilla ja kuvan otsikko title-attribuuteilla seuraavasti:

```
<img src="kuvat/kuva.jpg" alt="Kuva tietokoneesta" title="Tietokone" />.
```

Attribuuttien arvot siis merkitään lainausmerkkeihin = -merkin jälkeen. Elementtien ominaisuuksien lukumäärää ei ole rajoitettu.

Luokka ja tunniste

Luokka eli class on attribuutti. HTML-elementit voidaan määrittää kuuluvaksi tiettyyn luokkaan. Luokka-attribuutin avulla jokainen saman luokan elementti voi esimerkiksi saada saman muotoilun. Luokka-attribuutti voidaan antaa mille tahansa elementille. Esimerkiksi

tekstikappale-elementille p voidaan antaa luokka, jonka nimi on “isompiteksti”, asettamalla attribuutiksi class=””:

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

Luokan nimeä määritettäessä isoilla ja pienillä kirjaimilla on väliä. Tästä johtuen “isompiteksti”, “ISOMPITEKSTI” ja “IsompiTeksti” ovat kaikki eri luokkia. Luokkia käyttäviä elementtejä voidaan muotoilla CSS:n avulla. CSS:stä kerrotaan enemmän seuraavassa luvussa.

<huomiolaatikko 3>

Ääkkösten käyttö ei ole suositeltavaa luokkien nimissä.

<video 1 luokan ominaisuuksista: ylikirjoittaminen>

Videolla toteutetaan esimerkkinä “peruspohjainen” HTML-tiedosto, jossa käytetään joitakin tärkeimpiä elementtejä ja luokka-attribuuttia.

Tehtävät

<kommentti>

Aloitetaan CV:n verkkoversion tekeminen.

1. Avaa tekstieditori ja liitä sinne HTML-tiedostopohja **täältä**. Tallenna tiedosto html-nimiseen kansioon nimellä etusivu.html. Tallennettuasi tiedoston, voit avata sen verkkoselaimella.
2. Muokkaa etusivu.html-tiedostoa tekstieditorissa <title> -tagin, otsikko-elementin ja tekstikappale-elementin osalta.
3. Täydennä esittelyteksti itsestäsi ja lisää kaksi vapaavalintaista kuvaa etusivulle (tekijänoikeudet huomioiden). Tallenna kuvat samaan hakemistoon etusivu.html-tiedoston kanssa.
4. Luo samasta pohjasta koulutus.html-niminen tiedosto. Lisää tiedostoon ansioluettelosi Koulutus-osio. Tälle sivulle voit lisätä myös aiemman työkokemuksesi.

5. Luo harrastukset.html-niminen tiedosto. Lisää tiedostoon ansioluettelosi Harrastukset-osio.

malliesimerkki - etusivu

malliesimerkki - työkokemus

<huomiolaatikko 4>

Muistaa tallentaa HTML-tiedosto aina muutosten jälkeen ja päivittää verkkosivu nähdäksesi muutokset myös verkkoselaimessa.

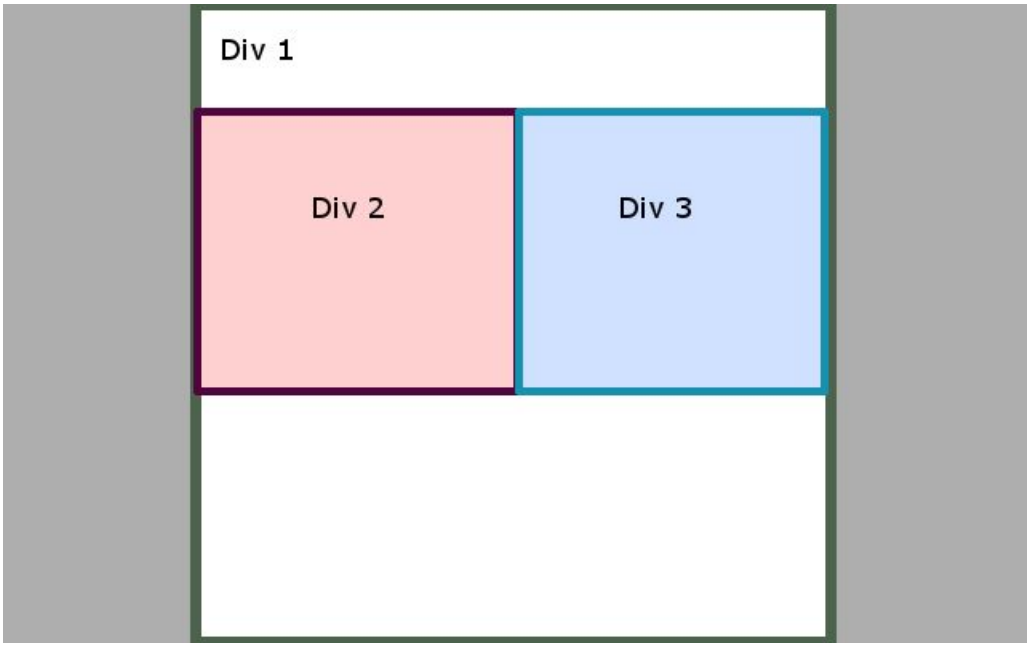
<kuva 2>

<kuva 3>

Div-elementit

Div-elementin nimi tulee sanasta division (osa / osa-alue). Div-elementtejä käyttämällä verkkosivu voidaan jakaa erilaisiin osiin, mikä helpottaa esimerkiksi muotoilujen ryhmittelyä. Div-elementtien avulla voidaan tehdä esimerkiksi kappalemuotoiluja ja tekstikehyksiä. Div-elementtejä voidaan laittaa myös sisäkkäin. <kommentti>

<kuva 4>



HTML-*sivu (harmaa), johon sisällytetty kolme div-elementtiä: Ensimmäisen div-elementin (vihreät reunukset ja valkoinen pohja) sisällä on rinnakkaiset div-elementit 2 ja 3.*

Esimerkki peräkkäin oleveista div-elementeistä:

```
<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>
```

Esimerkki sisäkkäin olevista div-elementeistä:

```
<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>
```

<video 2>

Videolla havainnollistetaan div-elementtien käyttöä.

Linkittäminen

Verkkosivut koostuvat harvoin vain yhdestä HTML-sivusta. HTML eli HyperText Markup Language on nimensä mukaisesti on tarkoitettu **hypertekstin** eli toisiinsa linkitetyn tekstin merkitsemiseen. Verkkosivut voidaankin ajatella hypertekstinä, sillä ne sisältävät linkkejä toisiin verkkosivuihin. Verkkosivusto rakennetaan tekemällä useita verkkosivuja, joista **viitataan toisiinsa linkeillä** (a-elementti). Useasta verkkosivusta koostuvan verkkosivuston pohjaksi on järkevää luoda hakemistorakenne. Kaikkien HTML-tiedostojen ei ole hyvä sijaita

samassa hakemistossa, vaan loogisesti jaoteltuina eri hakemistoihin. Eri hakemistoihin ja hakemistoissa sijaitseviin HTML-tiedostoihin viittaaminen tapahtuu seuraavalla periaatteella:

<lisätieto>
Linkkielementin rakenne on seuraavanlainen:
Linkin teksti. Osoitteessa ei saa olla ääkkösiä, erikoismerkkejä tai välilyöntejä. Linkin otsikko (title) näkyy, kun verkkoselaimessa hiiren kursori viedään linkin päälle.
</lisätieto>

Absoluuttiseen viittaukseen merkitään viitattavan kohteen koko verkko-osoite. Absoluuttista viittausta kannattaa käyttää vain oman sivuston ulkopuolelle kohdistuvissa viittauksissa. Absoluuttinen viittaus on riippumaton sivusta tai sijainnista, jossa viittaus tehdään. Absoluuttista viittaamista voidaan verrata esimerkiksi paikannukseen, jossa maantieteellisiksi koordinaateiksi määritellään 62°14′ 25″ N, 025°44′ 40″ E. Koordinaatit eivät ole riippuvaisia siitä, missä ne mainitaan.

Absoluuttisia viittauksia:

- http://www.example.com/ viittaa example.com-verkkotunnuksen juurihakemistoon “/”. Juurihakemistoon linkittäminen vie yleensä sivuston etusivulle.
- https://www.google.fi/search viittaa google.fi-verkkotunnuksen “search”-nimiseen sivuun juurihakemistossa “/”.
- /etusivu.html on absoluuttisen viittauksen erikoistapaus ja viittaa nykyisen sivuston juurihakemistosta “/” löytyvään “etusivu.html”-nimiseen sivuun.

Toisin kuin absoluuttinen viittaus, **suhteellinen viittaus** on riippuvainen sijainnista, jossa viittaus tehdään. Käyttäen aiempaa paikannusvertausta, suhteellinen viittaus sijaintiin liittyen voisi olla esimerkiksi “viisisataa metriä etelään sieltä, missä nyt olet”. Verkkosivustolla suhteellisella viittauksella tarkoitetaan verkkosivun sijaintia hakemistorakenteessa omalla sivustolla. Suhteellisten viittausten hyötyjä ovat viittausten toimiminen millä tahansa palvelimella sekä viittausten pituus, sillä ne ovat lyhyempiä kuin absoluuttiset viittaukset.

Suhteellisia viittauksia:

- etusivu.html viittaa saman hakemiston etusivu.html-sivuun.
- kuvat/ viittaaa saman hakemiston kuvat-nimiseen alihakemistoon (oletuksena näytetään alihakemistosta etusivu, jos sellainen löytyy).
- kuvat/logo.jpg viittaa saman hakemiston kuvat-alihakemistossa olevaan logo.jpg-tiedostoon.
- ../ viittaa ylempään hakemistoon (hakemisto, jossa aktiivinen alihakemisto sijaitsee).
- ../sivut/ viittaa ylemmässä hakemistossa sijaitsevaan "sivut"-nimiseen alihakemistoon.

<video 3>

Videolla havainnollistetaan linkittämistä ja viittaamista.

Tehtävät

<tehtävä>

1. Luo aloittamaasi etusivu.html-tiedostoon neljä div-elementtiä, joille määritä luokat "navigointi", "teksti", "kuvat" ja "alaviite".
2. Sijoita navigointi-luokan div-elementti ensimmäiseksi heti body-aloitustagin jälkeen. Sijoita tämän jälkeen teksti-, kuvat- ja alaviite-luokan div-elementit.
3. Luo navigointi-luokan div-elementtiin lista, johon teet kohdiksi "Esittely" (tämä on etusivu.html-tiedostosi), "Koulutus" ja "Harrastukset". Laita jokainen listan teksti linkiksi suhteellisella viittauksella.
4. Lisää teksti-luokan div-elementin sisälle aiemmin kirjoittamasi esittelyteksti.
5. Lisää kuvat-luokan div-elementin sisälle aiemmin lisäämäsi kuvat.
6. Lisää alaviite-luokan div-elementtiin tekstinä ©, oma nimesi ja vuosiluku.

</tehtävä>

esimerkkiratkaisu

<huomiolaatikko 5>

Numeroimattoman listan luonti käy seuraavasti:


```
    <li>Ensimmäinen kohta</li>
    <li>Toinen kohta</li>
</ul>
```

Yhteenveto

- Kun HTML-tiedostoa tarkastellaan verkkoselaimella, tiedostosta tulee verkkosivu.
- HTML-tiedosto koostuu elementeistä, joita merkitään tageilla. Elementit voivat sisältää attribuutteja.
- Yleisimpiä elementtejä ovat esimerkiksi p (tekstikappale), h1-h4 (otsikot), table (taulukko), ul (järjestämätön lista), ol (järjestetty lista), a (linkki), img (kuva) ja br (rivinvaihto).
- Div-elementtejä käyttämällä verkkosivu voidaan jakaa erilaisiin osiin, mikä helpottaa esimerkiksi muotoilujen ryhmittelyä.
- Verkkosivusto koostuu useista verkkosivuista. Sivuilta viitataan toisiinsa linkeillä. Viittaukset oman sivuston sisälle kannattaa tehdä suhteellisina viittauksina.

Kuvat ja videot

Kuva 1: syntaksiväritys ja rivinumerointi Notepad++-ohjelmalla

Kuva 2: HTML-tiedosto kansiossa

Kuva 3: HTML-tiedosto selaimessa

Kuva 4: div-elementtien jaottelu sivulla

Video 1: Video HTML-tiedoston luomisesta ja elementtien ja attribuuttien käytöstä

Video 2: Video div-elementtien käytöstä

Video 3: Video viittausten käytöstä

Huomiolaatikot

Huomiolaatikko 1: Valmiiksi asennettuja tekstieditoriohjelmia

Huomiolaatikko 2: Ilmaisia tekstieditoriohjelmia

Huomiolaatikko 3: Huomio ääkkösten käytöstä

Huomiolaatikko 4: Muistutus tallentamisesta ja päivittämisestä

Huomiolaatikko 5: Ohjeistus numeroimattoman listan luomisesta

2. CSS-perusteet (1 oppitunti)

Tavoitteet

Tässä luvussa

- opit muotoilemaan verkkosivuja CSS:n avulla.
- opit CSS:n syntaksin.
- opit hyödyntämään muotoilussa erilaisia yksiköitä.
- opit hyödyntämään muotoilussa luokka-attribuuttia.

Teksti

Johdantoteksti

Laajempia verkkosivustoja luodessa verkkosivun **tyylimääritykset eli muotoilut** on järkevää pitää erillään verkkosivun rakenteesta ja sisällöstä. Muotoilut kannattaakin yleensä toteuttaa **CSS-tyylitiedostojen** avulla.

CSS

CSS-tyylitiedostoja käytettäessä HTML-tiedostoon merkitään pelkästään verkkosivun elementit, sisältö ja tarvittaessa attribuutteja. Kaikki muotoilut merkitään erilliseen CSS-tiedostoon (tiedoston pääte .css). HTML-tiedostosta viitataan CSS-tiedostoon eli CSS-tiedostosta haetaan muotoilut verkkosivulle. Erillisen tyylitiedoston hyötynä on HTML-tiedoston pysyminen selkeänä sekä se, että samaa tyylitiedostoa voivat käyttää useat HTML-tiedostot. Tällä tavoin esimerkiksi laajan verkkosivuston muotoilut voidaan merkitä yhteen tiedostoon.

Linkittäminen HTML-tiedostoon

CSS-tiedosto linkitetään HTML-tiedostoon sijoittamalla link-elementti HTML-tiedostoon head-elementin sisälle seuraavasti:

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

Href-ominaisuuden arvoksi määritellään käytettävän tyylitiedoston nimi, joka on esimerkissä tyyli.css. Linkittämisen jälkeen tyyli.css-tiedoston tyylimääritykset tulevat voimaan ja verkkosivulle saadaan halutunlaiset muotoilut.

Syntaksi

CSS:n syntaksi eli ns. kielioppi on seuraavanlainen:

```
elementti {
        ominaisuus: arvo;
        ominaisuus: arvo;
 }
```

HTML-elementin muotoilua määritettäessä ensin määritellään, mikä **ominaisuus** on kyseessä ja kaksoispisteen jälkeen ominaisuudelle asetetaan **arvo**. Arvon jälkeen merkitään puolipiste (;), joka erottaa eri ominaisuudet toisistaan. Elementin muotoilut sijoitetaan aaltosulkeiden sisälle. CSS-tiedostossa voi määrittää muotoiluja tarvittavan määrän eli mikään elementti tai muotoilu ei ole CSS-tiedostossa pakollinen.

Esimerkki tekstielementin väriominaisuuden määrittämisestä mustaksi:

```
p {
   color: black;
}
```

Mikäli useamman elementin samat ominaisuudet halutaan määritellä samoilla arvoilla, elementit voidaan ryhmitellä pilkuilla eroteltuina:

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

Body-elementin ominaisuuksien määrittelyt tekevät **yleiset eli globaalit** muotoilut koko verkkosivulle:

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

<Huomiolaatikko 1>

HTML-elementtiä voidaan muotoilla myös **sisällyttämällä eli upottamalla** ominaisuus ja sen arvo HTML-tiedostoon elementiin CSS-syntaksin mukaisesti:

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

Ominaisuuksia voidaan upottaa myös useampia:

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

Yleisimmät CSS-muotoilut

Taustaväri (background-color)

- esim. background-color: white;

Fontti (font-family)

- esim. font-family: Arial;

Fontin väri (color)

- esim. color: red;

Fontin asettelu (text-align)

- esim. text-align: left|right|center|justify;
    - Tekstin asemointi tapahtuu arvoilla left (vasen reuna), right (oikea reuna), center (keskitetty) tai justify (tasaus kumpaankin reunaan).

Fontin koko (font-size)

- esim. font-size: 120%;

Taulukon reunat (border)

- esim. border: 2px solid #400222;
    - Reunoille voidaan määritellä leveys, tyyli ja väri.

- Leveys voidaan määritellä arvoilla medium (keskipaksu), thin (ohut), thick (paksu) tai jollain yksiköllä (esim. pikseleinä).
- Reunan tyyli voidaan määritellä arvoilla dotted (pisteinä), dashed (viivoitettu), solid (yhtenäinen) tai double (tuplareunus).

Korkeus (height) ja leveys (width)

- height: auto|arvo;
- width: auto|arvo;

Marginaali (margin) ja täyte (padding)

- esim. padding: 4em;
  - Määritykset marginaalille ja täytteelle tehdään samalla tavalla.
  - Asetetut määrittelyt koskevat automaattisesti jokaista neljää sivua.
  - Jos halutaan määritellä vain yhden sivun arvo, voidaan käyttää tarkempia ilmauksia:

    margin-top / padding-top

    margin-left / padding-left

    margin-right / padding-right

    margin-bottom / padding-bottom
  - *Jos määritellään vain yksi arvo, asetetaan se automaattisesti jokaiselle neljälle sivulle. Jos taas määritellään kaksi arvoa (esim. padding: 0 2em;), asetetaan ensin elementin ylä- ja alasivun arvot ja sitten vasta vasemman ja oikean sivun arvot. Esimerkin tapauksessa padding-top ja padding-bottom olisivat 0 ja padding-left ja padding-right taas 2em.*
  - *Jokaisen sivun arvo voidaan myös määritellä erikseen:*

    esim. padding: 0 2em 1em 3em;

Elementin asemointi / kellunta (float)

- esim. float: right;
  - Voidaan käyttää esim. kuva-, div- ja linkkielementtien asemointiin.
  - Asemointi tapahtuu arvoilla left (vasemmalle) tai right (oikealle).

<online kokeilu yleisistä muotoiluista>

Yksiköt

CSS-tiedostossa voidaan määritellä esimerkiksi fontin kokoa, marginaalia tai täytettä erilaisilla yksiköillä. Mittayksiköitä on kahdenlaisia: absoluuttisia ja suhteellisia. **Absoluuttiset mittayksiköt** ovat aina tietyn mittaisia riippumatta siitä, missä niitä käytetään. **Suhteelliset mittayksiköt** määrittyvät tapauskohtaisesti eli sen mukaan, missä niitä käytetään. Suhteelliset mittayksiköt määrittyvät siis aina suhteessa johonkin muuhun. Yleensä suhteellisia mittayksiköitä kannattaa suosia.

Fonttikoko määritellään yleensä suhteessa oletusfontin kokoon em-yksikkönä tai prosentteina. Marginaali ja täyte taas määritellään suhteessa oletusfontin kokoon pikseleinä tai em-yksikköinä tai vastaavasti suhteessa selainikkunan kokoon prosentteina. Yksikköjä määriteltäessä luku ja yksikkö kirjoitetaan yhteen ilman välilyöntiä.

Esimerkki tekstielementin fontin ja täytteen koon määrittelystä:

```
p {
    font-size: 0.8em;
    padding: 10px;
}
```

<huomiolaatikko 2>

Hyödyllisimmät mittayksiköt:

Absoluuttisia mittayksiköitä

- pikseli (px)

Suhteellisia mittayksiköitä

- kirjasinlajin korkeus (em)
- prosentti (%)

<online kokeilu yksiköistä>

Luokkien muotoilu

<huomiolaatikko 3>

Luokka-attribuuttia käsiteltiin edellisessä luvussa. Voit käydä kertaamassa asiaa **luvusta 1**.

CSS:n avulla yksittäistä luokka-attribuuttia ja siihen kuuluvia elementtejä voidaan muotoilla helposti samanlaisiksi ja samalla kertaa. Esimerkiksi tekstielementtiä, joka kuuluu sisalto-nimiseen luokkaan (HTML-tiedostoon merkitty Tekstikappale), voidaan muotoilla seuraavasti:

```
.sisalto {
	float: left;
	text-align: center;
	max-width: 75%;
}
```

Luokka-attribuutti ilmoitetaan CSS-tiedostossa laittamalla luokan nimen eteen piste. Jos p-elementille määritellyissä muotoiluissa on päällekkäisyyksiä sisalto-luokalle määriteltyjen muotoilujen kanssa, voimaan jäävät sisalto-luokan muotoilut. Luokalle määritellyt muotoilut siis **ylikirjoittavat eli kumoavat** elementeille määritellyt muotoilut.

<online kokeilu luokista>

Esimerkki kumoavista muotoiluista:

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

HTML-merkintäkieltä kirjoittaessa syntaksiin voi helposti tulla virheitä. Jos HTML-tiedostossa on syntaksivirheitä, voi verkkosivun ulkoasu näyttää virheelliseltä. Verkkosivun syntaksin oikeellisuuden eli validisuuden voi tarkistaa helposti internetissä löytyvän **validaattorin** avulla. Tarkistaminen tapahtuu seuraavasti:

1. Mene WWW-osoitteeseen http://validator.w3.org/.
2. Syötä tarkistettavan verkkosivun verkko-osoite kohtaan Address ja napauta Check-painiketta. Voit myös ladata HTML-tiedoston tarkistettavaksi omalta koneeltasi, mikä tapahtuu Validate by File Upload -välilehden kautta.

Syötettyäsi verkko-osoitteen tai ladattuasi tiedoston avautuu verkkosivu, joka kertoo englanniksi, oliko verkkosivullasi virheitä. Validaattori kertoo virheiden lukumäärän, millaisia virheet ovat ja millä rivillä HTML-tiedostossa virheet sijaitsevat. Saadun informaation perusteella pystyt tarvittaessa korjaamaan HTML-tiedostosi **validiksi**. Tarkista korjausten jälkeen HTML-tiedostosi validaattorilla vielä uudelleen.

Edellä kuvattu validaattori tarkistaa vain verkkosivun HTML-tiedoston sisällön, mutta ei puutu mahdollisiin CSS-tiedoston syntaksivirheisiin. CSS-tiedoston tarkistamiseen löytyy myös oma validaattori internetistä. Tarkistaminen tapahtuu seuraavasti:

1. Mene osoitteeseen http://jigsaw.w3.org/css-validator/.
2. Syötä tarkistettavan verkkosivun verkko-osoite kohtaan Address ja napauta Check-painiketta. Voit myös ladata CSS-tiedoston tarkistettavaksi omalta koneeltasi, mikä tapahtuu By File Upload -välilehden kautta.

Validaattori toimii samalla tavalla kuin HTML-validaattori.

<VIDEO syntaksin tarkistamisesta>

Yhteenveto

Tehtävät

1. Tee oma CSS-tiedosto tekstieditorilla ja linkitä se edellisessä luvussa työstämääsi HTML-tiedostoon. Lisää CSS-tiedostoon yleiset muotoilut body-elementille, muotoilut tekstikappaleille (p) ja eri otsikkotasoille (h1-h3).
2. Muotoile HTML-tiedoston div-elementit seuraavalla tavalla:
    a. Määritä ul-elementille
        i. taustaväriksi #ADD8E6.
        ii. marginaaliksi 0 ja täytteeksi 1em.
    b. Määritä li-elementti kellumaan vasemmalle.
    c. Määritä linkeille fontiksi Arial.
    d. Asemoi teksti-luokan div-elementti vasemmalle ja kuvat-luokan div-elementti oikealle.
    e. Aseta teksti-luokan div-elementin leveydeksi 60% ja kuvat-luokan div-elementin leveydeksi 40%.
    f. Määritä alaviitteen
        i. taustaväriksi vihreä.
        ii. tekstin koko oletusarvoa pienemmäksi.

Kuvat

Huomiolaatikot / laajenevat laatikot

Huomiolaatikko 1: Muotoilujen upottaminen HTML-tiedostoon

Huomiolaatikko 2: Absoluuttisia mittayksiköitä

Huomiolaatikko 3: Suhteellisia mittayksiköitä

Huomiolaatikko 4: Huomio luokka-attribuutin kertaamisesta

Ruutukaappausvideot

Yleisimmät html+css-muotoilut (esim. padding/margin erojen havainnollistaminen)

Validointi

3. JavaScript-perusteet (3 oppituntia)

27.11. TIM-ympäristö luotu: https://tim.jyu.fi/view/users/jokasavo/tvtdok/js/testi-js

<kommentti>

Tavoitteet

Tässä luvussa

- opit ymmärtämään ohjelmoinnin perusperiaatteet.
- opit ohjelmoimaan koodia kirjoittamalla.
- opit JavaScript-ohjelmointikieltä.

Teksti

Johdantoteksti

JavaScript on **ohjelmointikieli**, jota käytetään HTML:n ja CSS:n kanssa yhdessä verkkosivuja luotaessa. JavaScriptin avulla voidaan verkkosivulle lisätä toiminnallisuutta. JavaScriptiä voidaan käyttää myös pelien ohjelmoinnissa ja mobiilisovellusten luomisessa. JavaScript-ohjelmointikieltä ei tule sekoittaa Java-ohjelmointikieleen.

JavaScript-koodia varten kannattaa luoda oma tiedostoonsa. JavaScript-tiedoston tunnistaa **.js-tiedostopäätteestä**.

<huomiolaatikko 1>

JavaScript-koodia voidaan kirjoittaa suoraan HTML-tiedostoon joko sijoittamalla JavaScript-koodi <head>- ja </head>-tagin väliin tai varsinaisen HTML-tiedoston sisällön sekaan <body>- ja </body> -tagin väliin merkitsemällä skriptiä <script>- ja </script> -tagilla.

Ohjelmoinnin peruspalikat

Muuttujat

Muuttuja on tietokoneen muistista varattu alue, johon voidaan tallentaa tietoa. Muuttuja **alustetaan** (eli kerrotaan tietokoneelle, että muistista varataan alue) tunnisteella **var**. Alustamisen lisäksi muuttajaan tallennetaan jokin **arvo**. Esimerkiksi muuttujaan **ika** voidaan heti muuttujan julistamisen yhteydessä tallentaa luku 25:

```
var ika = 25;
```

<huomiolaatikko 2>

Kirjoitettua koodia kannattaa ikään kuin lukea oikealta vasemmalle: “sijoitetaan arvo 25 muuttujaan ika”.

Muuttuja voidaan myös alustaa aiemmin alustettujen muuttujien avulla:

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

Muuttujan **vanhempi** arvo on 25 + 20 eli 45.

<online-kokeilu>

<huomiolaatikko 3>

Joissain muissa ohjelmointikielissä **muuttujan tyyppi** tulee julistaa muuttujan määrittelyssä, esim. C#-ohjelmointikielessä:

```
int ika = 25;
```

Muuttujalle ika määritetään tyyppi int eli kokonaisluku. Muita muuttujatyyppejä on kirjaimet (char), kirjainjonot (string), totuusarvo (boolean, jolloin *muuttuja saa arvon true tai false*) ja long (liukuluku/desimaaliluku). JavaScriptillä ohjelmoitaessa tyyppimäärityksiä ei tarvitse tehdä.

Ehtolauseet

Suorien komentojen lisäksi ohjelmoinnissa voidaan määrittää toimintoja erilaisilla **ehdoilla**. Ehtojen käyttäminen tapahtuu if-, else ja if-else-**ehtolauseilla**.

If-ehtolause:

If-ehtolauseen avulla voidaan verkkosivulle lisätä esimerkiksi toiminta, jossa verkkosivun kävijä toivotetaan tervetulleeksi tietyn kellonajan mukaan. Tällöin tietokoneen pitää tehdä toiminta:

Tarkista kellonaika. Jos kello on vähemmän kuin 18.00, toivota kävijälle "Hyvää päivää!".

Koodina vastaava toiminta näyttää tältä:

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

Ehtona toiminnassa on siis tietty kellonaika ja toiminnan toteutumiseksi tämän ehdon täytyy toteutua eli olla **tosi**.

If-else-ehtolause:

Tarkistettava ehto voi myös jäädä toteutumatta, jolloin ehto on **epätosi**. Myös toteutumattomalle ehdolle voidaan määritellä toiminta. Tämä voidaan toteuttaa **If-else-ehtolauseen** avulla:

Tarkista kellonaika. Jos kello on vähemmän kuin 18.00, toivota "Hyvää päivää!"

Muuten toivota "Hyvää iltaa!".

Koodina vastaava toiminta näyttää tältä:

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

If-else-ehtolauseella voidaan määritellä myös useampi vaihtoehtoinen toiminta:

Tarkista kellonaika. Jos kello on vähemmän kuin 11.00, toivota "Hyvää huomenta!".

Muuten jos kello on vähemmän kuin 20.00, toivota "Hyvää päivää!".

Muuten toivota "Hyvää iltaa!".

Koodina vastaava toiminta näyttää tältä:

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

<online-kokeilu>

<huomiolaatikko 4>
eri silmukoista ja niiden käyttökohteista

Silmukka

Silmukoita käyttämällä voidaan toistaa sama koodi useamman kerran. Näin haluttu toiminta saadaan ohjelmoitua lyhyemmällä koodilla ja tehokkaammin. Silmukoita voidaan tehdä for- ja while-silmukoilla.

For-silmukka:

For-silmukassa tiettyä toimintaa toistetaan niin kauan kuin tietty ehto on tosi.

<kommentti>

For-silmukan syntaksi on seuraavanlainen:

```
      for (alustuslauseke; ehtolauseke; kasvatus-/vähennyslauseke, joka tehdään toiminnan jälkeen) {
          toiminta
      }
```

Alustuslausekkeena on muuttujan arvon asettaminen. **Ehtolausekkeena** on ehto, jonka mukaan silmukka jatkaa toimintaansa. Ehtolausekkeen tulee olla sellainen, että silmukan suoritus päättyy jossain vaiheessa tai muuten tuloksena on ikuinen silmukka.

Viimeisenä on **kasvatus-/vähennyslauseke**, joka määrittää, mitä suoritetun toiminnan jälkeen tehdään. Toiminnan jälkeen yleensä kasvatetaan tai vähennetään alustuslausekkeessa määritettyä muuttujaa.

<kommentti>

Tehdään laskuri, joka tulostaa parilliset luvut 0-10 väliltä:

```
var luku = 0;
for (var i = 0; i<11; i+=2){
        luku = i;
        console.log(luku);}
```

HTML:n puolella silmukkaa voitaisiin hyödyntää seuraavasti: 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;
      }
```

While-silmukka:

Myös while-silmukassa tiettyä toimintaa toistetaan niin kauan, kun tietty ehto on tosi.

<kommentti>

While-silmukan syntaksi on kuitenkin erilainen kuin for-silmukan:

```
while (ehto) {
toiminta, joka suoritetaan ehdon ollessa tosi
}
```

<online-kokeilu>

<VIDEO ehtolauseista ja silmukasta>

<tehtävä>

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

</tehtävä>

Funktiot

Funktioita käyttämällä voidaan useampi samalla kertaa suoritettava komentojen sarja nimetä yhdeksi kokonaisuudeksi. Funktiota voidaan käyttää silloin, kun sitä tarvitaan. Kun funktiota halutaan käyttää, täytyy sitä **kutsua.** Funktio käyttää **syötteitä eli parametrejä,** joille asetetaan arvo kutsumisen yhteydessä. Parametrejä voi olla yksi tai useampia. Funktion syntaksi on JavaScriptissä seuraavanlainen:

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

Esimerkiksi suorakulmion pinta-alan laskemiseen voidaan tehdä funktio, joka saa parametreinä suorakulmion pituuden ja leveyden, ja niitä hyödyntäen funktio tulostaa suorakulmion pinta-alan:

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

Tätä funktiota kutsutaan esimerkiksi `arvoilla pituus=2 ja leveys=5`, se palauttaa suorakulmion pinta-alan eli $2 \cdot 5$ eli 10.<video?>

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.nimi[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-sulkutagia, jotta HTML-sivu on luonut kaikki tarvittavat elementit sivulle, ennen kuin komento suoritetaan.

Edellisessä esimerkissä skripti hakee dokumentista (dokumentti on siis tässä objekti!) elementin id:n 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, näytetään päivämäärä:

```
<button onclick="naytaPvmJaKlo()">Näytä päivämäärä ja kellonaika</button>
```

JavaScript-tiedostossa olisi funktio:

```
function naytaPvmJaKlo(){
  document.getElementById('demo').innerHTML=Date()
}
```

<kommentti>

<huomilaatikko >Yleisiä tapahtumankäsittelytapoja</huomiolaatikko>

Huomiolaatikot

Metodeista

Metodi on funktio, jota olio voi tehdä tai suorittaa. Metodi voi siis esimerkiksi tulostaa tiedon olion sisällöstä. Esimerkissä meillä on ollut document -niminen olio, jolle kutsutaan getElementById -metodia (`document.getElementById('demo')`). Metodi muodostetaan samalla tavoin kuin oliokin, eli käyttämällä function() -komentoa. Kun metodi halutaan ottaa käyttöön, kutsutaan sitä syntaksilla: `Olio.metodi()`;

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

Lopullinen tehtävä - tarkistimen tekeminen

HTML-pohja <tästä> ...

<Javascript-tiedosto, jossa valmiina window.onload-funktio>

→ Hae (HTML-)dokumentista elementti id:n perusteella (HTML-dokumentin tarkistus-buttonin id), ja lisää sille addEventListener() -metodi (parametreina click, buttonin id).

Kirjoita funktio tarkista, jossa alustat oikeiden vastausten lukumäärä-muuttujan. Jos dokumentissa olevan elementin nimen perusteella kohta 0 tai 1 on valittu, niin kasvata oikeinden vastausten lukumäärä -muuttujaa. Lisää niin monta riviä, kuin on väitteitä.

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.

Lisää html-sivulle kaksi buttonia, joille annat omat id:t. Määritä javascriptissä näitä buttoneita painamalla uudet css-tiedostot.

Videot

Silmukat

Ehtolauseet