

Tööjuhend 1: HTML ja CSS, adaptiivne veebidisain

Õpikeskkond: [Tartu Ülikooli Moodle'i
õpikeskkond](#)

Printija: Tauri Tivik

Kuupäev: 29.08.2021 22:31:53

Kursus: Veebilehtede loomine
edasijõudnutele (LTAT.03.015)

Raamat: Tööjuhend 1: HTML ja CSS,
adaptiivne veebidisain

Sisukord

- 1. Sissejuhatus
- 2. HTML
- 3. CSS
- 4. Ülesanne 1
- 5. GitHub Pages
- 6. Ülesanne 2

1. Sissejuhatus

Veebilehtede põhjaks on HTML ja CSS, kuid üks tavaline veebileht võiks alata prototüübist. Kui silmade ees on oma visualiseeritud idee, siis töö realiseerimisel on lihtsam püstitatud piiride sees püsida. Heaks tavaks adaptiivse (ekraani suurusega kohanduva) disaini puhul on mitme prototüübi loomine: mobiil- ja arvutiversioon, vahepeal ka tahvelarvutiversioon.

NB! Selle teema jaoks on meil prototüübid juba valmis, leiate need teema materjalide alt kaustast [Teema failid](#).

Millest siis tuleb alustada? Mobiilsest versioonist. Mobiilne versioon arvuti veebilehitsejas võib näha üsna vaene, kuid ikkagi kasutatav. Desktop-versioon väikesel ekraanil on aga tavaliselt pigem halb idee. Mobiilversioonist alustamise eeliseks on ka see, et kaotades umbes 80% ekraanist ([allikas: Mobile-first responsive web design](#)) saab arendaja keskenduda põhilisele sisule ja funktsionaalsusele. Kui on soov uurida, millal oleks sobivam alustada desktop-versioonist, lugege seda [Brainleaf artiklit](#).

Kui teil pole veel lemmiktestitoimetajat HTML ja CSS koodi kirjutamiseks, julgen soovitada [Visual Studio Code](#). See IDE sobib hästi veebiarenduse jaoks. Tulemuse vaatamiseks on Visual Studio Code-is mitmeid lahendusi, näiteks laiendus *HTML Preview*. Laiendusi saab alla laadida laienduste menüüst *View > Extensions*. Samuti saab valmivat lehte vaadata, avades *.html* faili otse kaustast, kuhu projekt on salvestatud. Mitmes veebilehitsejas (Google Chrome, Firefox) on olemas ka *Developer Tools* funktsionaalsus. Seda võib kasutada, et vaadata valminud lehte erineva suurusega ekraanidel.

Alustame HTML osaga. Selleks avage käesoleva tööjuhendi järgmine leht (vajutage paremas alumises nurgas oleval noolel →).

2. HTML

Käivitame toimetaja ja kohe alguses deklareerime dokumenditüübi:

```
<!DOCTYPE html>
```

Kõik ülejäänud sisu asub `<html></html>` siltide vahel ja jaguneb kaheks: osa elementidest kuulub dokumendi peasse (`<head>`) ja osa elemente – dokumendi kehasse (`<body>`).

```
<!DOCTYPE html>
<html>

  <head></head>
  <body></body>

</html>
```

Pea sisaldab metaandmeid dokumendi kohta, nende hulgas on

- `<meta>` sildid, millele lisatakse atribuudid, et anda veebilehele kontekst. Atribuutideks võivad olla kodeering (`charset`) või mitmeks otstarbeks kasutatav atribuut `name`, millele saab väärtusena lisada autori nime (`author`), lehe kirjelduse (`description`) või vaateakna suuruse (`viewport`). Atribuudi `content` väärtuseks võib olla tekst, aga saab ka järgmiselt teha: koos `name="viewport"` võiks üks meta-silt sisaldada ka `content` atribuuti väärtusega `"width=device-width, initial-scale=1.0"`;
- `<title>`, pea tähtsaim osa, veebilehe nimi, mis on nähtav veebilehitseja sakireal;
- `<link>` sildid, mille atribuutideks on, arvasite õigesti ära, lingid – stiililehtedele ja skriptidele, mida veebileht kasutab.

Kehas aga asuvad need elemendid, mis on nähtavad lehe külastajatele. Koodi struktureerimiseks ja selle puhtuse tagamiseks jagatakse keha sisu kolmeks osaks: päis (`<header>`), põhiosa (`<main>`) ja jalus (`<footer>`).

Hetkel võiks meie dokument välja näha järgmiselt:

```
<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8">
    <meta name="description" content="Terves maailmas
    pole ägedamat reisiblogi: jälgi minu seiklusi
    seitsmel kontinendil!">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0"/>

    <title>FOLLOW ME | Reisiblogi</title>

    <link rel="stylesheet" href="reset.css" />
    <link rel="stylesheet" href="style.css" />

  </head>
```

```
<body>

    <header></header>

    <main></main>

    <footer></footer>

</body>

</html>
```

Lisatud on kodeering, lühike lehe kirjeldus, mille otsingumootorid näitavad, vaateakna suurus, lehe pealkiri ja kaks linki stiililehtedele. Miks kaks? Üks nendest on `reset`-stiilileht, mis lähtestab erinevatele veebilehitsejatele omased stiilid (värvid, raamid, polstrid (ingl `padding`), veerised (ingl `margin`) jm). Seda kasutatakse põhimõtteliselt selleks, et veebilehel oleks ainult see kujundus, mida autor on sinna meelega lisanud. Detailsemalt sellest, mida ja kuidas toimib `reset.css`, saab lugeda siin: <http://html5doctor.com/html-5-reset-stylesheet/>.

NB! On tähtis, et esimesena oleks lingitud `reset.css` ja viimasena tegelik stiilileht, muidu lähtestatakse ka meie enda loodud kujundus.

Päis ja jalus võiksid olla igal veebisaidi lehel ühesugused. Paigutame päisesse meie kaks erineva taseme pealkirja ja vormistame navigatsiooni (`<nav>`) linkide nimekirjana.

```
<header>

    <h1>Follow me</h1>
    <h2>Reisiblogi</h2>

    <nav>
        <ul>

            <li><a href="#">Kodu</a></li>
            <li><a href="#">Reisilood</a></li>
            <li><a href="#">Nippe</a></li>

        </ul>

    </nav>

</header>
```

Tahaks pealkirju ja navigatsiooni kuidagi eraldada. Selleks paneme kaks pealkirja ühe sektsiooni sisse. Kuna teoreetiliselt saab sektsioone olla dokumendis mitu ja nõuded nende vormistusele võivad olla erinevad, lisame sektsiooni sildile ka klassinime.

```
<section class="header-banner">
```

```
<h1>Follow me</h1>
<h2>Reisiblogi</h2>

</section>
```

Põhiosa jagame kaheks. Paneme ühesse `div`-i (`.index-header`) horisontaaljoon (`<hr>`) ja teise taseme pealkiri *Populaarsemad*, teist osa vormistame sektsioonina (`.gallery-links`) ja lisame sinna meie pildid. Siin tuleks meeles pidada mitu asja:

- tahame, et iga pilt viiks postituse juurde, seega vormistame pilte linkidena;
- keset pilti soovime paigutada pealkirja. Üks võimalustest, kuidas seda teha, on ühendada tekst ja pilt ühe `div`-i alla sellisel kujul:

```
<div class="container">



    <p class="img-pealkiri">Pealkiri</p>

</div>
```

Tulemuseks saame

```
<main>

<div class="index-header">

    <hr />

    <h2 class="gallery-h2">Populaarsemad</h2>

</div>

<section class="gallery-links">

    <a href="#" class="gallery-image">
        <div class="container"> 
        <p class="post-header">Lijiang, Hiina</p></div>
    </a>

    <a href="#" class="gallery-image">
        <div class="container">
        <p class="post-header">Shanghai, Hiina</p></div>
    </a>
```

```

    <a href="#" class="gallery-image">
    <div class="container">
    <p class=post-header>Morocco</p></div>
    </a>

    <a href="#" class="gallery-image">
    <div class="container">
    <p class=post-header>Moskva, Venemaa</p></div>
    </a>

</section>

</main>

```

Hetkel ei pruugi töö päris õige välja näha, kuid varsti jõuame CSS-failini ja tegeleme kujundusega.

Nägime desktop-prototüübist, et jalus on kolmeks veeruks jagatud, kuid mobiilses versioonis keskmine osa on peidetud. Vormistame *Viimased postitused* nimekirja linkide nimekirjana (`.footer-links`) ja sotsiaalmeedia ikoonide veergu (`.footer-sm`) paneme piltidele hüperlingi sildid ümber. Keskmisesse veergu lisame tavaline tekstilõik. Tuletame endale meelde, milleks kasutatakse `
` ja `` silte ja kuidas tähistatakse HTML-is täpitähti ja muid erisümboleid:

```

<footer>

    <ul class="footer-links">

        <li><p>Viimased postitused:</p></li>
        <li><a href="#">Shanghai, Hiina</a></li>
        <li><a href="#">Lijiang, Hiina</a></li>
        <li><a href="#">Moskva, Venemaa</a></li>
        <li><a href="#">Morocco</a></li>

    </ul>

    <div class="disclaimer">
        <p>K&auml;esolev veebileht on loodud<br>
<span>Veebilehtede loomine edasijõudnutele</span>
<br>aine jaoks. Tsau!</p>

    </div>

    <div class="footer-sm">

```

```

<a href="#"></a>
<a href="#"></a>
<a href="#"></a>
<a href="#"></a>

</div>

</footer>

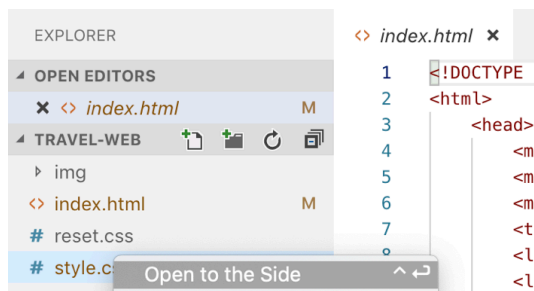
```

Nüüd liigume CSS osa juurde. Selleks avage tööjuhendi järgmine leht.

3. CSS

Nüüd liigume üle CSS-dokumendile.

Et vaadata kahte faili korraga, tuleb VSC IDE-s klikkida parema hiirenupuga teise faili peale ja valida menüüst *Open to the Side* käsu.



Esimese asjana määrame piltide laiuseks 100%, et nad ei segaks teiste elementide paigutust.

```

.gallery-image img {
    width: 100%;
}

```

Alustame päise kujundamisest.

Pöördume esimese sektsiooni poole läbi selle asukoha ja klassinime – header .header-banner. Lisame taustapildi ja määrame selle laiuseks 100% ehk kogu võimaliku laiuse. Pikkuse määramisel tutvume ühe kasuliku funktsiooniga, mida tõi kaasa CSS3: `calc()`. See väärtustab avaldise, kus vähendatavaks on 100vh ehk veebilehitseja terve akna suurus ja vähendajaks 500px – umbes nii suure osa ekraanist tahame teiste elementide jaoks jätta. Paneme paika ka muud tausta parameetrid ja nende väärtused. Soovitan erinevad väärtused iseseisvalt läbi proovida ja vaadata, mis muutub.

```
header .header-banner {  
  
    background-image: url("img/header.png");  
    width: 100%;  
    height: calc(100vh - 500px);  
    background-repeat: no-repeat;  
    background-position: center;  
    background-size: cover;  
  
}
```

Et pealkirjad oleksid samad, mis prototüübis, meil on vaja samasuguseid fonte. Kõik veebilehitsejad aga ei pruugi neid fonte teadma, seega peame neid linkima meie stiililehele. Selleks kasutame ühe nn @-reeglitest – nimelt, @import, ja Google Fonts kataloogi <https://fonts.google.com/>.

Prototüübis on kasutatud *Gentium Basic* ja *Open Sans* fonte.

Leiame esimese üles ja vajutame sellel. Avaneb uus aken, kus saame valida, mis paksusega või kallakuga fonti soovime importida. Märgime kõik variandid – klõpsake kirjetel *Select this style*. Klõpsamisel paremale poole ilmub lisaaken, mis näitab valikuid.

Nüüd leiame *Open Sans* fondi ning kordame sama tegevuse, kuid siin võime paksud variandid välja jätta – prototüübis neid pole kasutusel. Näeme, et lisaaknasse on ilmunud andmed ka selle teise fondi kohta.

*Almost before we knew it, we*

Remove this style

Regular 400

Almost before we knew it,

Remove this style

Regular 400 italic

Almost before we knew it, we

Remove this style

Semi-bold 600

Almost before we knew it,

+ Select this style

Semi-bold 600 italic

Almost before we knew it, we

Remove this style

Bold 700

Almost before we knew it

Variandi valik

+ Select this style

Bold 700 italic

Almost before we knew it,

Remove this style

Gentium Basic



Regular 400



Regular 400 italic



Add more styles

Remove all

Open Sans



Header or your name

☒ <link>☐ @import

```
<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Gentium+Basic:ital@0;1&family=Open+Sans:ital,wght@0,300;0,400;1,300;1,400;1,600;1,700;1,800&display=swap" rel="stylesheet">
```

CSS rules to specify families

```
font-family: 'Gentium Basic', serif;
font-family: 'Open Sans', sans-serif;
```

Seejärel kopeerime @import käsu ja asetame selle meie stiililehe esimesele reale. <style> silte on vaja ainult siis, kui kopeerite koodi HTML-dokumendi.

Nüüd võime pealkirju kujundada.

```
header h1 {

    font-family: "Gentium Basic", serif;
    font-weight: 700;
    font-style: italic;
    font-size: 60px;
    color: #ffffff;
    line-height: 70px;
    text-transform: uppercase;
    text-shadow: 2px 2px 8px #000000;

}

header h2 {

    font-family: "Open Sans", sans-serif;
    font-weight: 300;
    font-size: 15px;
```

```
letter-spacing: 5px;
color: #ffffff;
text-transform: uppercase;
}
```

Parameetrite nimetused ja väärtused on heaks vihjeks selle kohta, mida nad teevad, kuid julgustan ikkagi ise proovida, mida täpselt muudab mõni väärtus.

CSS-tabelid võivad olla kasutatud selleks, et joondada teksti keskele nii horisontaalselt kui ka vertikaalselt. *Parent*-element saab tabeliks ja *child*-element tabeli lahtriks. Kuna meil on kaks pealkirja, oleks mugav ühendada neid ühe `div`-i alla.

```
<div class="logo">

  <h1>Follow me</h1>
  <h2>Reisiblogi</h2>

</div>
```

Seejärel lisame `.header-banner` deklaratsioonile järgmist rida:

```
display: table;
```

ja kujundame uut `.logo` klassi:

```
header .logo {

  display: table-cell;
  vertical-align: middle;
  text-align: center;
  padding: 10px 0 20px 0;

}
```

Lisasime ka polstri, et teksti ja taustapildi äärte vahel oleks natuke tühja ruumi. Polstri parameetri väärtuseks võib olla üks, kaks või neli arvu. Üks arv tähendab, et selline polster rakendub kõikidele külgedele, kaks märgistavad *top-bottom* ja *right-left* paare, neli arvu annavad väärtust kõikidele külgedele eraldi (päripäeva). Kui polstri või veerise väärtuseks on 0, siis ühikut ei kasutata.

Nüüd kujundame navigatsiooni. Kuna linkisime *reset*-stiililehe, ei ole meie nimekirjana vormistatud navigatsiooni mingeid sümboleid ees. Piirdume nelja parameetriga nimekirja jaoks (`margin` ja `text-align` on siin ühe ja sama eesmärgiga, kuna Internet Exploreris joondamine läbi `margin: auto;` ei tööta) ja kahe parameetriga nimekirja elemendi jaoks:

```
header nav ul {  
  
    display: block;  
    margin: 0 auto;  
    text-align: center; /*IE*/  
    width: fit-content;  
  
}  
  
header nav ul li {  
  
    display: inline-block;  
    padding: 16px 16px;  
  
}
```

Lõpuks vormistame navigatsiooni linke:

```
nav ul li a {  
  
    font-family: "Open Sans", sans-serif;  
    font-weight: 400;  
    font-size: 18px;  
    letter-spacing: 3px;  
    color: #172029;  
    text-transform: uppercase;  
    text-decoration: none;  
  
}
```

Jätame hetkeks pildigalerii kujundamise vahele ja tegeleme jalusega. Määrame pikkuseks `calc(100% - 80px)`, kasutame vabad 80px polstri jaoks, lisame jalusele taustavärvi ja üles veerise, et galeriipiltide ja jaluse vahel oleks ruumi.

```
footer {  
  
    width: calc(100% - 80px);  
    padding: 40px 40px;  
    background-color: #172029;  
    margin-top: 20px;  
  
}
```

Peidame ära keskmise veeru:

```
.disclaimer {  
  
    display: none;
```

```
}
```

Vormistame *Viimased postitused* nimekirja (tehke kindlaks, et mäletate, mis kõik need parameetrid teevad). Et joondada see korralikult vasakule, kasutame `float` atribuuti:

```
footer ul {  
    width: fit-content;  
    float: left;  
    padding-left: 0px;  
}  
  
footer ul li {  
    display: block;  
    padding: 0 0 10px 0;  
}  
  
footer ul li p {  
    font-family: "Open Sans", sans-serif;  
    font-weight: 300;  
    font-size: 18px;  
    letter-spacing: 2px;  
    color: #ffffff;  
    text-transform: uppercase;  
}  
  
footer ul li a {  
    font-family: "Open Sans", sans-serif;  
    font-weight: 400;  
    font-size: 18px;  
    letter-spacing: 3px;  
    color: #ffffff;  
    text-transform: uppercase;  
    text-decoration: none;  
}
```

Mis juhtus jalusega? `float` atribuudil põhineval paigutamismeetodil on üks puudus: kuna elemendid nõ ujuvad konteineris, ei tea konteiner, kui suur ta peab olema. Olukorra parandamiseks lisame `footer` deklaratsioonile sellise rea:

```
overflow: hidden;
```

Sotsiaaltööde ikoonid paigutame samal moel, aga paremale:

```
.footer-sm {  
    width: 32px;  
    float: right;  
}  
  
.footer-sm img {  
    width: 100%;  
    margin-bottom: 10px;  
}
```

Jalus valmis, tegeleme nüüd meie lehe põhiosaga – pildigaleriiga.

Kujundame üleval asuva horisontaaljoone ja pealkirja:

```
hr {  
    color: #4f5663;  
    border-width: 2px;  
    width: 30%;  
    margin: 0 auto;  
    margin-bottom: 5px;  
}  
  
.gallery-h2 {  
    font-family: "Gentium Basic", sans-serif;  
    font-weight: 600;  
    font-size: 28px;  
    letter-spacing: 2px;  
    padding: 10px;  
    color: #3a383d;  
    text-transform: uppercase;  
    text-align: center;  
}
```

Veel üks elementide paigutamise viis CSS-is on positsioneerimine (ingl. k. *positioning*). Kasutame seda, et asetada postituste pealkirju piltide peale. `position: relative` tähendab, et elemendi asukoht muutub selle elemendi algse asukoha suhtes (nt `position: relative; left: 30px;` tähendab, et element on nihutatud oma algsest asukohast 30 piksli võrra

paremale) ja `position: absolute` tähendab, et elemendi asukoht sõltub lähedama positsioneeritud *parent*-elemendi asukohast (meie olukorras on *parent*-elemendiks pilt ja *child*-elemendiks tekst).

```
.gallery-image {  
    margin: 10px;  
    position: relative;  
    text-align: center;  
}  
  
.post-header {  
    /*kujundus*/  
  
    font-family: "Open Sans", sans-serif;  
    font-weight: 700;  
    font-size: 22px;  
    letter-spacing: 3px;  
    line-height: 24px;  
    color: #ffffff;  
    text-transform: uppercase;  
    text-shadow: 3px 3px 9px #111111;  
  
    /*positsioneerimine*/  
  
    position: absolute;  
    top: 50%;  
    left: 50%;  
    transform: translate(-50%, -50%);  
}
```

Piltide paigutamiseks aga kasutame `float` atribuudi ja positsioneerimise uuemat ja adaptiivsemat alternatiivi: *CSS Flexbox* meetodit.

Flexbox on mugav võimalus dünaamiliselt skaleerida lehe elemente. See on võimas ja kasulik meetod, millel põhineb näiteks Bootstrap 4, millega tutvume järgmises tunnis. Et paremini aru saada selle töö põhimõtetest, on olemas mäng *Flexbox Froggy*: <http://flexboxfroggy.com/#eng>. Proovige järele: lahendage vähemalt 12 ülesannet ja esitage oma tulemuste ekraanitõmmis Moodle'sse [Ülesande 1](#) kaudu.

Kuigi uuemad veebilehitsejate versioonid peaksid *Flexboxi* toetama, igaks juhuks võib ikka prefikseid kasutada. *Parent*-elemendiks ehk konteineriks on meil `.gallery-links` klass ja *child*-elementideks `.gallery-image` objektid.

```
.gallery-links {
```

```

display: flex;
flex-flow: row wrap;
display: -webkit-flex;
-webkit-flex-flow: row wrap;
display: -moz-flex;
-moz-flex-flow: row wrap;
justify-content: center;
-o-flex-flow: row wrap;
justify-content: center;
}

```

Viimaseks lisame `.gallery-image` deklaratsioonile `flex-basis` parameetri:

```

-webkit-flex-basis: 300px;
-moz-flex-basis: 300px;
flex-basis: 300px;

```

Mobiilversioon tundubki valmis – tegeleme nüüd arvutiversiooniga. Muutusi hakkame lisama teise `@-reegli` abil: `@media` ehk meediapäring (ingl.k. *media query*).

`@media` käitub nagu `if-lause`: parameetrid väärtustatakse ainult siis, kui on täidetud mingi tingimus, tavaliselt kas minimaalne või maksimaalne ekraani suurus. Kuna alustasime mobiilversiooniga, võime minna ainult suuremaks. Olgu siis meie katkepunktiks 950px:

```

@media only screen and (min-width: 950px) {}

```

Kui ekraan on laiem kui 950 pikslit, mis peaks olema teisiti? Meediapäringuga saab parameetritele väärtusi lisada või olemasolevaid täielikult ümber kirjutada. Näiteks:

```

@media only screen and (min-width: 950px) {

  header .header-banner {

    height: calc(100vh - 400px);

  }

  header h1 a {

    font-size: 80px;

  }

  header h2 {

    font-size: 24px;
    margin-top: 20px;
  }
}

```

```
}

header nav ul li {

    padding: 16px 42px;

}

}
```

On aeg kujundada ka meie jaluse keskmist veergu, mille oleme mobiilses versioonis ära peitnud. Meediapäringuid võib CSS dokumendis olla ka mitu: kui kood on jagatud päiseks, põhiosaks ja jaluseks, võib teha igale osale eraldi meediapäringuid.

```
@media only screen and (min-width: 950px) {

    .disclaimer {

        display: block;
        font-family: "Open Sans", sans-serif;
        font-weight: 300;
        font-size: 18px;
        letter-spacing: 2px;
        line-height: 24px;
        color: #ffffff;
        text-transform: uppercase;
        float: left;
        margin: 0 80px;

    }

    .disclaimer span {

        font-style: italic;

    }

}
```

Tundubki valmis! Proovige muuta ekraanisuurust: kas töötab õigesti?

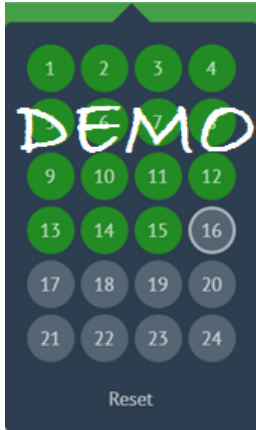
► Harjutus

Jaluses kasutasime elementide paigutamiseks `float` atribuuti. Proovige nüüd saavutada sedasama tulemust, kasutades Flexbox tehnikat.

Varsti uurime, kuidas saab oma veebilehe üles laadida *GitHub Pages*'i, aga enne seda sooritage järgmisel lehel toodud Ülesanne 1.

4. Ülesanne 1

Kui ei ole veel läbinud, siis nüüd on aeg läbida mängu Flexbox Froggy (<http://flexboxfroggy.com/#eng>) vähemalt 12 etappi ja Moodle [Ülesande 1](#) kaudu esitada ekraanitõmmis sooritatud ülesannetest (*näidake lahendatud ülesannete "sisukord", mis asub paremas ülemises nurgas. Lahendatud ülesanded on rohelised, lahendamata aga hallid*).



5. GitHub Pages

Viimase sammuna võiks oma veebilehe veebi üles laadida. Kui teil on olemas GitHub konto, võib selleks kasutada GitHub Pages hosting-teenust – ja kui konto puudub, praegu on õige aeg see luua. Soojalt soovitan paigaldada arvutisse ka GitHub Desktop, kui te just käsurea kõige suurem fänn ei ole.

GitHub Pages lubab luua ühe kasutaja kohta ühe personaalse veebilehe ja lisada sellele lõpmatult palju projektilehti. All on toodud kirjeldus, **kuidas saab luua** personaalse veebilehe.

1. Looge uus GitHub repo.

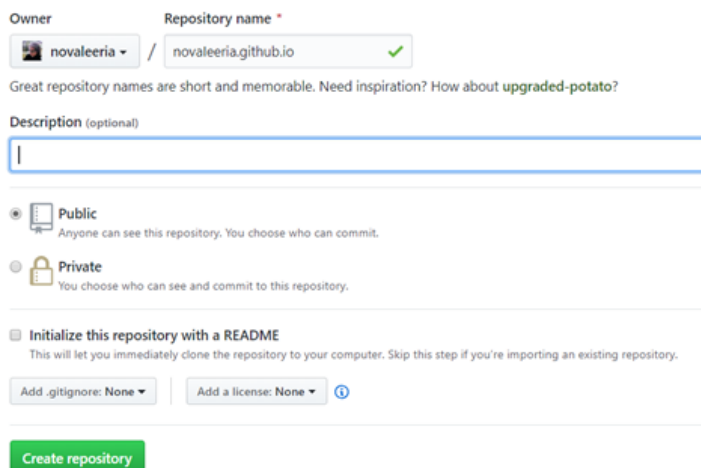
Repositories



2. Pange talle nimeks *kasutajanimi.github.io* (**väga tähtis!**).

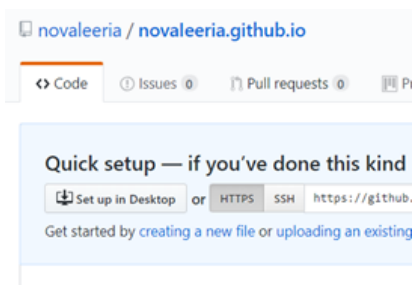
Create a new repository

A repository contains all project files, including the revision history.



The screenshot shows the GitHub 'Create a new repository' form. At the top, it says 'Owner' with a dropdown menu showing 'novaleeria' and 'Repository name' with a text input containing 'novaleeria.github.io'. Below this, it says 'Description (optional)' with a text input. Then, there are two radio buttons: 'Public' (selected) and 'Private'. Below these, there is a checkbox for 'Initialize this repository with a README'. At the bottom, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None'. A green 'Create repository' button is at the bottom right.

3. Kloonige repo oma arvutisse (kui vajutada *Set up in desktop*, avaneb paigaldatud *GitHub Desktop*, seal saate vajutada *Clone* ja valida nimekirjast just loodud repo).



4. Lubage luua arvutisse *GitHub* kaust, kuhu paigutatakse teie repo kaust. Tekitage sinna HTML fail nimega *index.html* ja suvalise lihtsa sisuga, nt

```
<!DOCTYPE html>

<html>

  <body>

    <h1>Hello World</h1>
    <p>I'm hosted with GitHub Pages.</p>

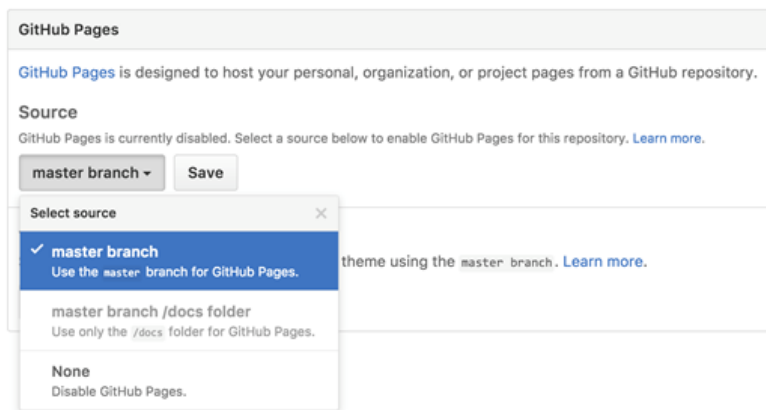
  </body>

</html>
```

5. Salvestage (*Commit*) ja laadige (*Push*) muudatused üles *GitHub Desktop*'ist.

Voilà – nüüd aadressil *kasutajanimi.github.io* asubki teie veebileht. **Laadime üles ka valminud projekti.** Selleks tuleb tegutseda järgmiselt:

1. looge uus repo – nüüd võite nimeks panna ükskõik mida;
2. leidke repo sätetest *GitHub Pages* lõik ja valige seal *None* asemel *master branch*;



3. korrake *GitHub Desktop*'is samad tegevused, mis enne: kloonige repo lokaalsesse kausta, viige sinna üle projektifailid (*index.html*, stiililehed, pildid), salvestage ja laadige üles muudatused.

Nüüd peaks aadressil *kasutajanimi.github.io/teisereponimi* asuma teie projekt. Ongi valmis!

Ingliskeelne video *GitHub Desktop*'i kasutamisest:



6. Ülesanne 2

Teema materjalide alt leiate kausta [Ülesande 2 failid](#). Vaadates prototüüpe (prototüübid pole nii ranged, kui see, mida praegu kasutasime), tehke valmis üks adaptiivse disainiga retseptiraamatu maandumisleht ja laadige see *GitHub Pages* abil üles. Iseseisvalt tuleb leida juurde 7 pilti: bänner, foto kokkadest Meist seksiooni jaoks, pildid, mis iseloomustaksid eesti, aasia ja itaalia rooge, ja 2 sotsiaalvõrgustike ikooni omal valikul. Kui on soov, teksti võib asendada proovitekstiga (<https://et.lipsum.com/>).

Lisaülesanne eriti seiklushimulistele: meenutage, kuidas toimivad CSS-animatsioonid, ja tehke nii, et roogade päritolu maad näidataks ainult siis, kui kursor on pildil.

Laadige oma projekt *GitHub Pages*'i ja Moodle [Ülesande 2](#) kaudu esitage link tööle.

