
Taneli Juga

# Ajanhallintaselainlaajennuksen kehitys

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

10.5.2022

# Tiivistelmä


| | |
|---|---|
| Tekijä: | Taneli Juga |
| Otsikko: | Ajanhallintaselainlaajennuksen kehitys |
| Sivumäärä: | 28 sivua |
| Aika: | 10.5.2022 |
| | |
| Tutkinto: | Insinööri (AMK) |
| Tutkinto-ohjelma: | Tieto- ja viestintätekniikka |
| Ammatillinen pääaine: | Ohjelmistotuotanto |
| Ohjaajat: | Lehtori Matti Oosi |
| | Lehtori Simo Silander |


---


Insinöörityön tavoitteena on perehtyä internetselainten lisäosien toimintaan sekä käydä läpi selainlaajennuksen (browser extension) kehittämisprosessi alusta loppuun. Samalla tutkitaan ja vertaillaan kehittämiseen liittyviä valintoja muun muassa selaimen valinnan, kehitysympäristön ja laajennusten toiminnallisuuden suhteen.

Työn lopputuloksena syntyi selainlaajennos, jonka tarkoituksena on helpottaa opiskelua ja lisätä tuottavuutta. Laajennus pyrkii antamaan käyttäjälle mahdollisuuksia hallita ja estää opiskelulle haitallista internetin käyttöä. Viittaan laajennukseen termillä ajanhallintalaajennus, koska se on yleisnimitys tuottavuutta lisääville ohjelmille.



Avainsanat: Selain, selainlaajennus, lisäosa

# Abstract


| | |
|---|---|
| Author: | Taneli Juga |
| Title: | Time Management Browser Extension Development |
| Number of Pages: | 28 pages |
| Date: | 10 May 2022 |
| Degree: | Bachelor of Engineering |
| Degree Programme: | Information and Communications Technology |
| Professional Major: | Software Engineering |
| Supervisors: | Matti Oosi, Senior Lecturer<br>Simo Silander, Senior Lecturer |


---


This thesis introduces the working and development of browser add-ons and goes through the development process of a browser extension from beginning to end, while discussing the various choices made during the development process regarding the development and functionality of the extension.

The end result is a time management browser extension that aims to help studying and increase productivity. The extension gives users the opportunity to govern and stop internet use that is harmful to studying.



Keywords: Browser, add-on, extension

# Sisällys

## Lyhenteet

DOM: *Document Object Model.* Web-dokumenttien ohjelmointirajapinta, jonka avulla voi dynaamisesti muuttaa dokumentin ulkonäköä ja rakennetta.

NPM: *Node Package Manager.* JavaScript-koodille tarkoitettu pakettienhallintajärjestelmä, jonka avulla voi ladata ja päivittää erinäisiä paketteja.

JSON *JavaScript Object Notation.* Yleisesti web-kehityksessä käytetty tiedostoformaatti, joka koostuu avain-arvopareista.

UML *Unified Modeling Language.* Mallintamiskieli, jota käytetään visualisoimaan ohjelmistojen toimintaa ja rakennetta.

# 1 Johdanto

Internetin suosion ja kasvun myötä on web-sovelluksista tullut yhä suurempi osa ihmisten arkipäivää ja yritysten toimintaa. Internetselaimet ovat olennainen osa, miten ihmiset pääsevät käsiksi internetin sisältöä. Näin selainten uudet ominaisuudet voivat tuoda mukanaan täysin uusia tapoja käyttää internetiä.

Tärkeänä osana uusien ominaisuuksien testaamisen ja nopean kehittämistä ovat lisäosat. Monista, alun perin lisäosien tarjoamista ominaisuuksista, on myöhemmin tullut osa internetin standardeja, tai niiden toiminnallisuus on sulautettu itse selaimeen. Selainten valmistajien lisäksi lisäosat hyödyttävät kehittäjiä, koska ne tarjoavat valmiin alustan sovelluksen kehittämiseen ja käyttäjäkunnan sitä käyttämään.

Internetselainten lisäosat (add-on) ovat ohjelmistokomponentteja, joiden avulla voi muokata internetselaimen toimintaa monella tavoin. Monet erilaiset ohjelmistot mahdollistavat toiminnallisuuden muokkaamisen erilaisten lisäosien ja laajennusten avulla, mutta internetselaimissa lisäosista on tullut merkittävä ja suosittu osa selaimien toimintaa. Eräs suosituimpia lisäosia nykyään ovat erilaiset mainonnanestäjät (add blocker), jotka suodattavat mainoksia HTML-sivuilta ja videoilta. Pelkästään USA:ssa arviolta 27 % netin selaajista käytti vuonna 2021 mainonnanestoohjelmia [1].

Kiinnostuin itse oman lisäosani kehittämisestä ja niiden toiminnan tutkimisesta, koska niistä on ajan mittaan tullut yhä tärkeämpi osa omaa internetkäyttöäni. Käytän viihteeseen, tuotteliaisuuteen sekä tietoturvaan suunniteltuja lisäosia. Lisäosien kehitys ei myöskään ole niin hyvin tunnettu asia kuin web- tai työpöytäsovellusten kehitys, joten uskon siihen tarkemmin perehtymisen olevan hyödyllinen asiasta kiinnostuneille ohjelmoijille.

Uskon myös, että lisäosan kehittäminen on hyvä aloituspaikka täysin itse tuotettujen ohjelmistojen aloittamiselle. Se ei vaadi erillisen palvelimen asentamista ja konfiguraatiota, kuten web-sovellukset, mutta se ei myöskään ole yhtä suuri

urakka kuin perinteisen työpöytäsovelluksen kehitys. Myös ohjelman julkaisu ja käyttäjien löytäminen on helppoa, koska käyttäjän ei tarvitse asentaa kokonaan erillistä ohjelmistoa.

Haluan tuoda selkoa lisäosien toimintaan, historiaan ja paikkaan web-kehityksessä sekä kuvata niiden kehitysprosessia ja toimintaa. Pyrin työlläni vastaamaan moniin kysymyksiin, kuten miten eri selainten lisäosat eroavat toisistaan, mitä lisäosilla voi tehdä, minkälaista teknologioita ja ohjelmointikieliä niiden kehittämiseen käytetään.

Aloitan kertomalla lisäosien toiminnasta ja terminologista luvussa 2. Jatkan sitten kuvaamalla lisäosien toiminnan ja kehittämisen eroja selainten välillä. Seuraavaksi kuvaan oman lisäosani suunnittelu- ja kehitysprosessia. Lopuksi käyn läpi työn aikana ilmenneet tärkeät asiat ja jatkokehityksen paikat.

# 2 Internetselaimet ja niiden lisäosat

## 2.1 Internetselaimet

Käyttäjämäärältään merkittävimmät internetselaimet ovat Chrome, Safari, Edge, Firefox, Samsung Internet ja Opera, jotka vastaavat noin 95 % käytetyistä selaimista (taulukko 1). Suurin selaimista on Google Chrome 64 %:n markkinaosuudella, joka on yli kolme kertaa suurempi kuin seuraavaksi suurimman kilpailijan, Safarin, osuus.

Taulukko 1. Selainten käyttöjakauma maaliskuu 2021 [2].

| Selain | Käyttöosuus |
|---|---|
| Google Chrome | 64,75 % |
| Safari | 18,43 % |
| Microsoft Edge | 3,37 % |
| Firefox | 3,36 % |
| Samsung Internet | 3,23 % |
| Opera | 2,31 % |
| IE | 0,62 % |
| Muut | 4,55 % |

Googlen merkittävyys selaimien saralla on vieläkin suurempi ottaen huomioon, että Chrome, Edge, Opera (ja monet muut pienemmät selaimet) käyttävät Chromium-projektin koodia pohjanaan. Chromium on Googlen kehittämä avoimen lähdekoodin kirjasto, joka sisältää kaikki tarvittavat ohjelmistokomponentit selaimen kehittämiseen, kuten JavaScript- ja grafiikkamoottorit. Chromium on Googlen aloittama hanke, mutta monet muut selainvalmistajat ovat tehneet siihen osia.

Chromium-pohjaisten selainten samanlaisuuden vuoksi kehittäjät voivat tuoda Chromelle kehitettyjä lisäosia varsin helposti Chromium-pohjaisille selaimille. Esimerkiksi Chromium-pohjaisen selaimen, Microsoft Edgen, kehittäjädokumentaatio sisältää ohjeita Chrome-lisäosien tuomiseen Edge-selaimeen [3].

## 2.2 Lisäosat ja laajennukset

### 2.2.1 Terminologia

Pohjimmiltaan lisäosilla tarkoitetaan erilaisia tapoja, joilla kolmannen osapuolen kehittäjät voivat muokata ohjelmiston toiminnallisuutta. Tällainen toiminnallisuus mahdollistetaan yleensä erinäisten ohjelmistokomponenttien kautta, jotka viestivät selaimen kanssa jonkin rajapinnan välityksellä. Ajan saatossa eri internetselaimilla on ollut monenlaisia tapoja ja teknologioita, joilla tämä on saatu aikaan. Näille teknologioille on ollut erinäisiä nimiä, kuten liitännäinen (plug-in).

Käytän tässä työssä Mozillan määritelmää, jonka mukaan lisäosa (add-on) on mikä tahansa selainta jollain tavalla, kuten ulkonäöllisesti tai toiminnallisesti, muokkaava ohjelmanosa. Termit laajennus (extension) ja liitännäinen viittaavat taas tiettyihin teknologioihin, jotka toteuttavat selaimen toiminnallisuuden muokkaamisen eri tavoin. [4.]

Eräs tunnetuimmista lisäosista on vuonna 1996 julkaistu Adobe Flash Player, joka toimi web-sivujen ”de facto”-standardina yli 10 vuoden ajan. HTML5-tandardin, joka mahdollistaa videoiden toistamisen suoraan selaimella, myötä sen suosio kuitenkin hiipui. Tästä huolimatta Flash Player toimii silti testamenttina lisäosien mahdollisuuksista ja suosiosta. [5.]

Flash Player oli liitännäinen, joka on vanhentunut teknologia, jota harvat selaimet käyttävät. Sen sijaan laajennukset ovat moderni tapa lisäosien toteuttamiseen.

### 2.2.2 Laajennukset

Chrome oli ensimmäinen selain, jolle voi kehittää selainlaajennuksia ja sen suosion myötä kaikki suurimmat selaimet ovat alkaneet käyttämään laajennuksia.

Laajennukset toteutetaan samoilla teknologioilla kuin web-sivut (HTML, CSS ja JavaScript), mikä pienentää laajennuksen kehittämiseen vaadittavaa opettelukynnystä, koska internetin suosion myötä todella monet taitavat jo näiden teknologioiden käytön. Google käyttää tästä toteutustavasta nimeä ”webbisyys” (webbiness) [6].

Toinen merkittävä ero laajennuksissa liitännäisiin verrattuna, on niiden turvallisuus. Koska selaimilla on jo valmiiksi tapoja eristää JavaScript-koodin pääsy tietokoneen muihin resursseihin, kuten tiedostojärjestelmään, voidaan näitä samoja menetelmiä käyttää eristämään laajennusten sisältämä JavaScript-koodi.

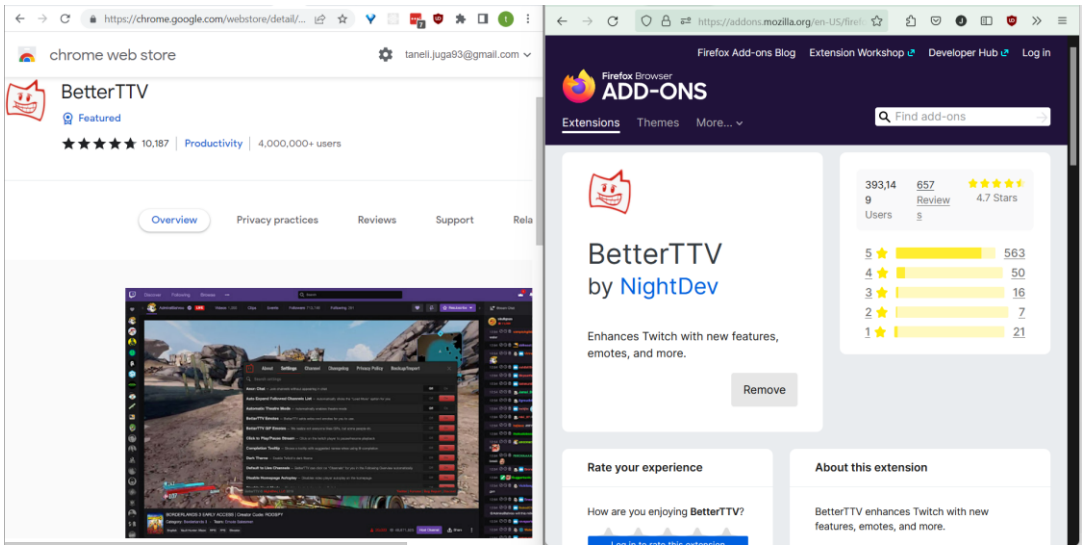
Siirrettävyyden helppouden ansiosta monet valmistajat julkaisevat laajennukset kaikilla suurimmilla selaimilla (Chrome, Safari, Firefox). Eri valmistajille on kuitenkin erilaisia kriteerejä sille, minkälaisia laajennuksia ne sallivat. Esimerkiksi

Applen Safari-selain on alkanut rajoittamaan mainonnanestolisäosien toiminnallisuutta [7], jonka ansiosta suosittu uBlock Origin on lopettanut kehittämisen Safarin uusimmille versioille [8].

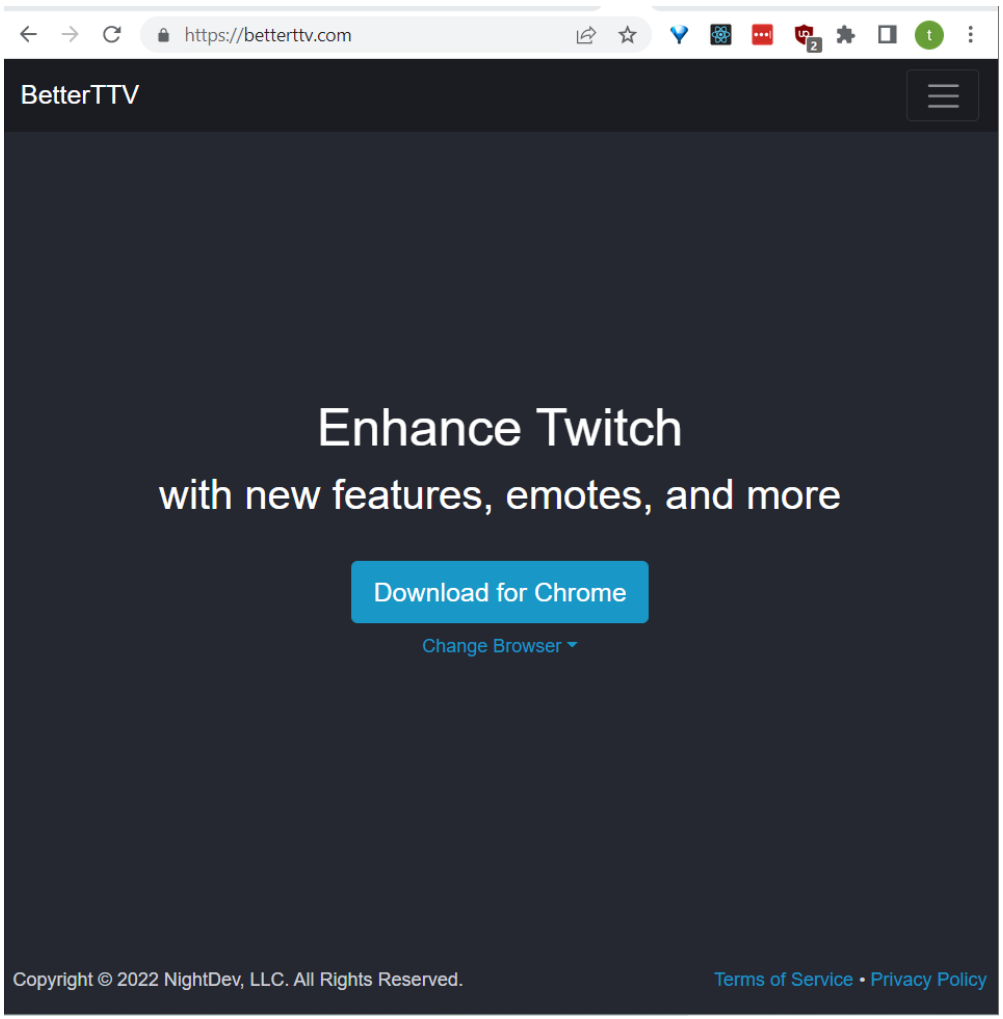
### 2.2.3 Laajennukset käyttäjän näkökulmasta

Jokaisella selainvalmistajalla on sivusto laajennusten lataamiselle (kuva 1), jotka toimivat laajennusten pääasiallisena lataamispisteenä. Monilla laajennuksilla on oma verkkosivusto, joka toimii paikkana laajennuksen mainostamiselle eri selaimille, eli suuremmalle käyttäjäkunnalle (kuva 2).

Useimmiten laajennusten verkkosivustot sisältävät linkin laajennuksen latauspisteeseen käyttäjän selaimen verkkokaupassa (kuva 2). Chrome-selaimessa tämä on ainut tapa laajennusten levittämiseen, mutta Firefox mahdollistaa laajennuksen lataamisen myös kehittäjän omasta palvelimesta. Firefox-selain mahdollistaa kuitenkin ainoastaan Mozillan digitaalisesti allekirjoittamien laajennusten asentamisen. Laajennuksen asennuksen yhteydessä selain tarkastaa allekirjoituksen ja mahdollistaa asentamisen vain allekirjoituksen täsmätessä, joten Mozillalle lähetetty versio laajennuksesta ei voi poiketa käyttäjän asentamasta versiosta.




Kuva 1. Chrome- (vas.) ja Firefox-selainten (oik.) verkkokaupat BetterTTV-laajennukselle.

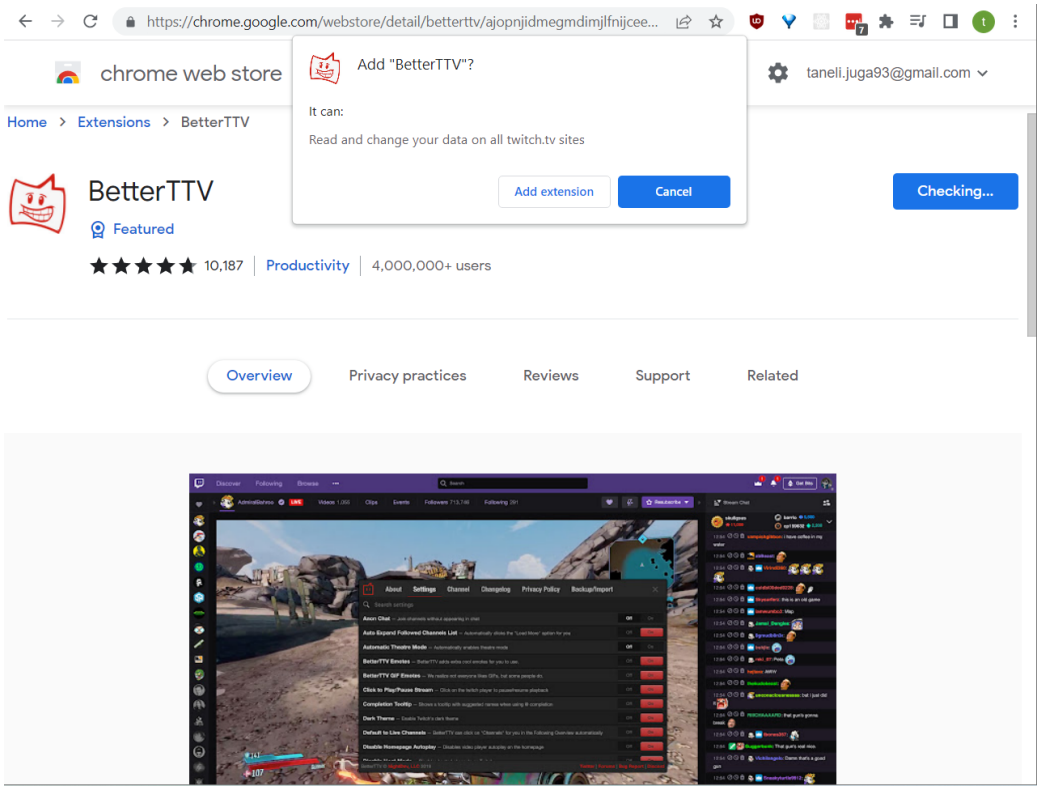



Kuva 2. BetterTTV-laajennuksen verkkosivu, jossa linkki Chromen verkkokauppaan.

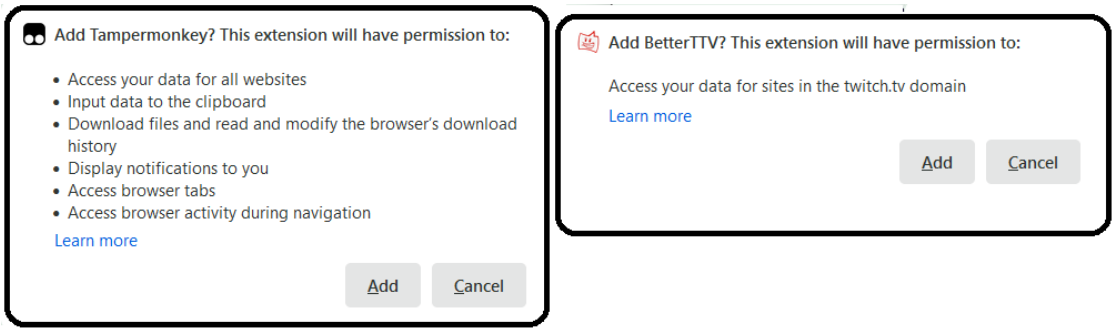
## Laajennuksen tarvitsemat luvat

Selain tarjoaa laajennukselle erilaisia toimintoja, joiden avulla voi tehdä monenlaisia asioita, kuten verkkopyyntöjen muokkaamisen tai käyttäjän selaushistorian lukemisen. Monet näistä toiminnoista kuitenkin vaativat, että selain pyytää lupaa näiden ominaisuuksien käyttämiseen. Tällä tavoin selaimella on asennuksen yhteydessä tieto laajennuksen käyttämistä toiminnoista, joten se voi antaa laajennuksen käyttäjälle tämän tiedon.

Asennuksen yhteydessä selain avaa ponnahdusikkunan, joka kertoo käyttäjälle, minkälaisia lupia laajennus vaatii selaimelta (kuva 3). Eri laajennusten vaatimat luvat voivat poiketa paljon toisistaan (kuva 4). Selainvalmistajat painottavat, että kehittäjät pyytäisivät lupia vain toimintoihin, jotka ovat välttämättömiä laajennuksen toiminnalle.

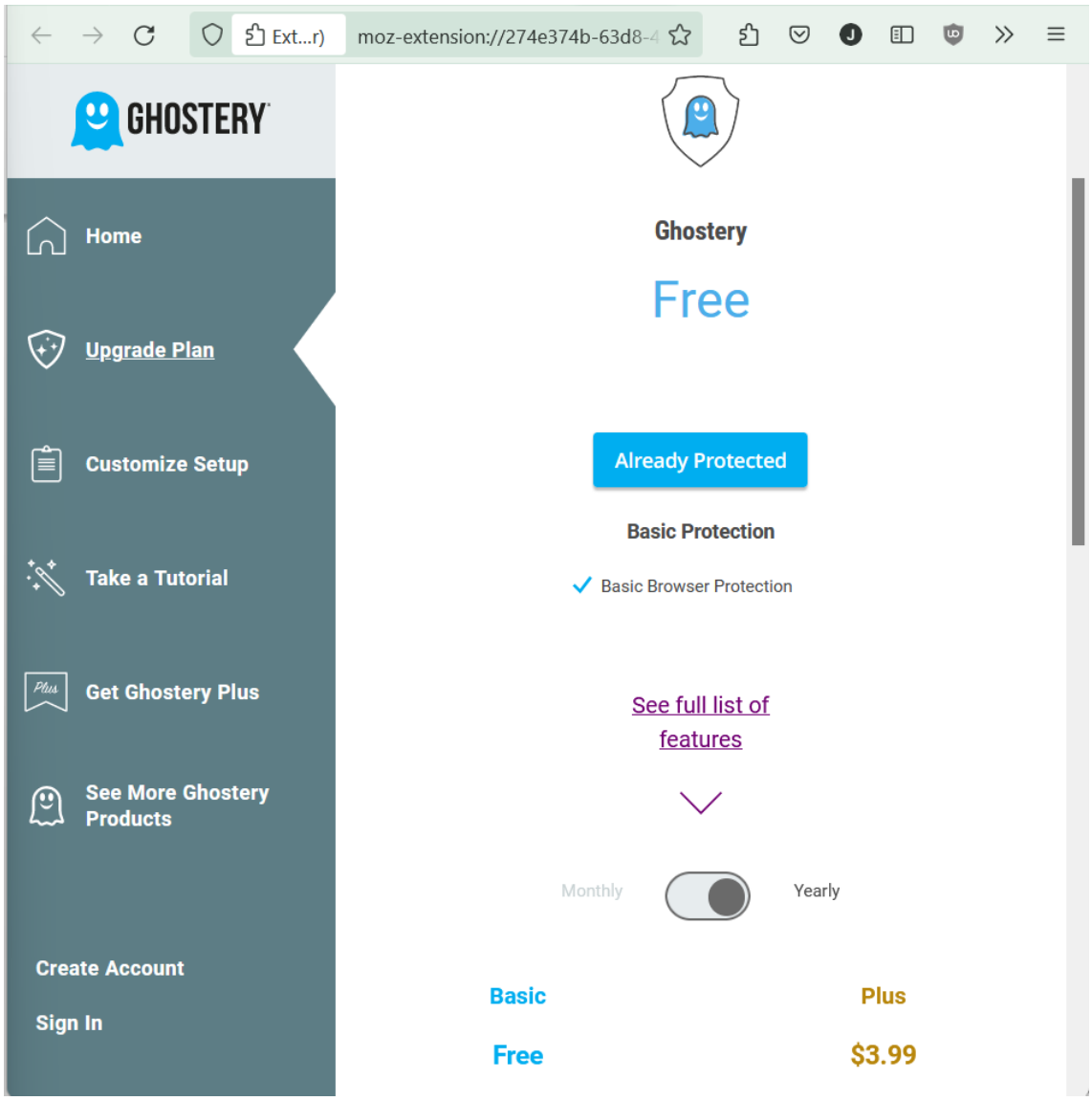



Kuva 3. BetterTTV-laajennuksen asennuksen yhteydessä avautuva ponnahdusikkuna.




Kuva 4. Tampermonkey-laajennuksen (vas.) vaatimat luvat verrattuna BetterTTV-laajennuksen (oik.) vaatimiin lupiin.

Asennuksen jälkeen laajennus on heti käytettävissä ilman selaimen uudelleenkäynnistystä. Laajennus voi tässä vaiheessa tarjota käyttäjälle opastusta laajennuksen toiminnasta sekä mahdollisuuden laajennuksen asetusten muokkaamiseen. Monien laajennusten toiminta on käyttäjälle niin itsestään selvää, etteivät ne tarjoa erillistä opastussivua. Opastussivu voi toimia myös tapana markkinoida kehittäjän muita tuotteita tai laajennuksen maksullista versiota (kuva 5).




Kuva 5. Ghostery-laajennuksen asennuksen yhteydessä avautuva HTML-dokumentti, joka tarjoaa käyttäjälle opastusta sekä markkinoi laajennuksen maksullista versiota.

Opastussivuna voi toimia linkki kehittäjän valitsemaan verkko-osoitteeseen tai se voi olla HTML-dokumentti, joka sisällytetään laajennuksen asennustiedostoihin. Mikäli opastussivun toimittaa laajennuksen mukana, ei kehittäjän tarvitse ylläpitää verkko-osoitetta oppaan ylläpitämiseen. Toinen haitta erillisessä opastussivun säilyttämisessä erillisessä verkko-osoitteessa on, että laajennuksen tulee pyytää selaimelta oikeuksia verkkopyyntöjen lähettämiselle kyseiseen verkkotunnukseen. Verkkopyynnöt laajennuksen sisäisiin tiedostoihin eivät vaadi erillisiä oikeuksia, joten niiden käyttö, aina kun se on mahdollista, on suotavaa.

# 3 Laajennusten tekninen toteuttaminen

## 3.1 Selainten väliset erot

Laajennus kommunikoi selaimen kanssa JavaScript-koodissa rajapinnan välityksellä. Selainten tarjoamat rajapinnat ovat paljolti samanlaisia, mutta eroavat silti toisistaan, joten usein koodiin tulee tehdä muutoksia, jos laajennuksen haluaa tuoda toiselle selaimelle. Esimerkiksi Chrome-laajennuksessa rajapinnan JavaScript-objekti tuodaan nimellä Chrome, kun taas Firefoxissa nimellä Browser. Tästä huolimatta useimmat laajennukset toimivat kaikilla suurimmilla selaimilla.

Myös selainten laajennuksille tarjoamat käyttöliittymät poikkeavat hieman toisistaan. Firefox tarjoaa laajennukselle kaksi erillistä nappulaa, johon laajennus voi sitoa toimintoja käyttäjälle. Näitä nappuloita nimitetään nimillä: page action ja browser action. Kuten nimistä voi päätellä nappuloiden on tarkoitus tarjota kontekstista riippuvia toimintoja: sivukohtaisen toiminnon (page action) on tarkoitus sitoa toimintoja, jotka riippuvat sivuston osoitteesta tai sen sisällöstä. Selainkohtaiseen toimintoon taas on tarkoitus sitoa toimintoja, joiden tulisi toimia aina samalla tavoin.

## 3.2 Laajennuksen koostumus

### 3.2.1 Manifest-tiedosto

Manifesti on monella tapaa laajennuksen tärkein tiedosto. Se sisältää kaiken metatiedon laajennuksesta, kuten nimen, version, käytetyt rajapinnat, tiedostot ja niiden sijainnit. Jokainen laajennuksessa käytettävän tiedoston sijainti tulee sisällyttää manifestiin. Käytännössä manifesti on avain -arvopareista koostuva JSON-muotoinen tiedosto (esimerkkikoodi 1).

Eräs uuden manifestin suurimmista ominaisuuksista on valinnaisten oikeuksien myöntäminen. Valinnaiset oikeudet ovat oikeuksia, joita ilman laajennus toimii, mutta joita tarvitaan jonkin toiminnon toteuttamiseksi. Tämän avulla käyttäjät saavat paremman kuvan siitä, mihin laajennus oikeastaan tarvitsee sen pyytämiä oikeauksia ja antaa käyttäjälle mahdollisuuksia päättää antamiensa oikeuksien määrästä.

```
{
  "name": "Getting Started Example",
  "description": "Build an Extension!",
  "version": "1.0",
  "manifest_version": 3,
  "background": {
    "service_worker": "background.js"
  },
  "permissions": ["storage"],
  "action": {
    "default_popup": "popup.html",
    "default_icon": {
      "16": "/images/get_started16.png",
      "32": "/images/get_started32.png",
      "48": "/images/get_started48.png",
      "128": "/images/get_started128.png"
    }
  }
}
```

Esimerkkikoodi 1. Pienimuotoisen Chrome-laajennuksen manifesti.

### 3.2.2 Sisältöskripti

Sisältöskripti (content script) on JavaScript-tiedosto, joka suoritetaan jonkin HTML-sivun kontekstissa. Tämä tarkoittaa, että sillä on käytössään DOM-objekti,

joten se voi muokata sivun ulkonäköä ja toimintaa. Sisältöskripti suoritetaan aina käyttäjän navigoidessa sivulle, jolle laajennuksella on oikeudet. Sisältöskriptejä voi myös injektoida ajonaikaisesti mille tahansa välilehdelle, kunhan laajennus on pyytänyt host-oikeuksia välilehden osoitteeseen.

Sisältöskriptejä käytetään tehtävissä, joiden tulee muokata sivuston toimintaa tai ulkonäköä. Esimerkkejä tästä voi olla esimerkiksi mainonnanestolaajennukset, jotka poistavat käyttäjän määrittelemältä joukolta sivuja kaikki löytämänsä mainokset, tai laajennus, jonka tehtävänä on tarjota lisäominaisuuksia jollekin websivustolle, kuten BTTV (Better Twitch Tv) -laajennus, joka muokkaa Twitch.tv-sivuston ulkonäköä ja toimintaa.

Kehittäjä voi määritellä sisältöskriptejä laajennuksen manifestissa. Skripteille voi määritellä tietyn URL-osoitteen, jolloin selain lataa skriptit automaattisesti käyttäjän avatessa sivuston. Sisältöskripti ei pääse käsiksi kaikkiin selaimen tarjoamiin JavaScript-rajapintoihin, mutta se voi käyttää niitä epäsuorasti lähettämällä viestejä muille laajennuksen komponenteille.

### 3.2.3 Taustaskripti

Taustaskripti on laajennuksen toiminnan kannalta sen tärkein komponentti. Se ajetaan automaattisesti selaimen käynnistyksen yhteydessä ja se pääsee käsiksi kaikkiin laajennuksille tarjottuihin rajapintoihin.

Taustaskripti on nimensä mukaisesti skripti, jota suoritetaan kaiken aikaa selaimen taustalla. Taustaskripti voi kommunikoida muiden skriptien kanssa viestien välityksellä. Viestit toimivat samalla tavoin kuin tapahtumat, jossa kuuntelija rekisteröi callback-funktion, joka suoritetaan skriptin saadessa viestin. Mikä tahansa skripti voi kuunnella viestejä. Viestin sisältö on JavaScript-olio, joten se voi sisältää mitä tahansa dataa, jota JavaScript-objektiin voi laittaa.

Laajennusten taustaskriptit ovat verkkotyöskentelijöitä. Verkkotyöskentelijä (Web Worker) on teknologia, joka mahdollistaa JavaScript-tiedostojen suoritta-

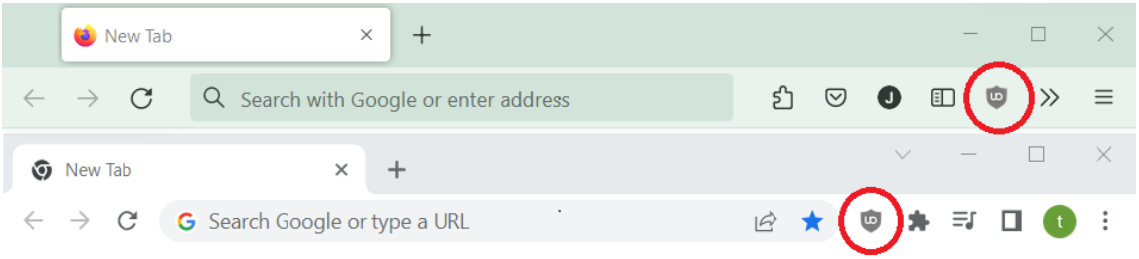
misen samanaikaisesti HTML-koodin kanssa. Tämä toiminnallisuus saavutetaan suorittamalla worker-komponentin sisältämä koodi eri säikeessä kuin HTML-dokumentin koodi. Tämän ansiosta JavaScript-koodi, joka suoritetaan verkkotyöskentelijässä, ei hidasta web-dokumentin latausta.

Verkkotyöskentelijän ympäristö ja toiminta poikkeavat jollain tavoin normaalista JavaScript-koodista. Verkkotyöskentelijästä ei pääse käsiksi DOM-olioon, joten dokumentin sisältöä ei voi muokata. Sen sijaan verkkotyöskentelijän kommunikaatio dokumentin kanssa tapahtuu epäsuorasti viestien välityksellä. Viestiin sisällytetty data voi olla yksinkertaisia, kuten numeerinen arvo, merkkijono tai JavaScript-olio.

## 3.3 Laajennusten käyttöliittymät

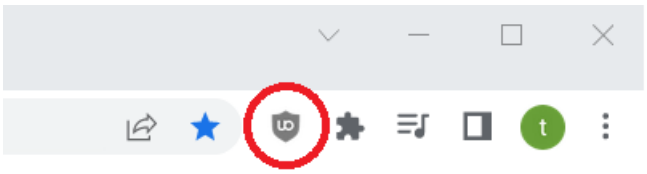
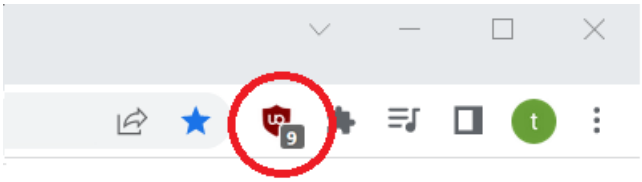
Jokainen moderni selain tarjoaa käyttäjälle vähintään kaksi käyttöliittymäkomponenttia: selaintoiminnon (browser action) ja ponnahdusikkunan (pop-up). Käyttäjälle selaintoiminto näkyy ikonina selaimen työkalupalkissa (kuva 6). Selaintoiminolla on käyttäjälle kaksi erilaista merkitystä: se viestii laajennuksen tilasta, ja sen avulla käyttäjä voi käyttää laajennuksen eri toimintoja.

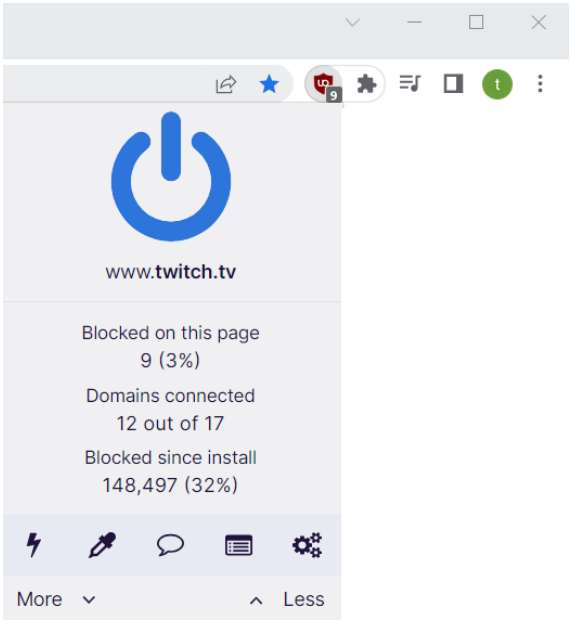
Vaikka selaintoiminto on varsin yksinkertainen komponentti, käyttävät monet laajennukset sen toiminnallisuuksia varsin monipuolisesti. Tästä hyvänä esimerkkinä toimii sisällön suodattamiseen (content filtering) suunniteltu laajennus nimeltä uBlock Origin -laajennus. Laajennusta käytetään esimerkiksi estämään nettisivuston mainoksia sekä käyttäjän seurausta tiedonkeruupalvelujen avulla. Laajennus toimii estämällä selainta lähettämästä tai vastaanottamasta pyyntöjä palvelimiin, jotka käyttäjä on määrittänyt estettäviksi. Estetyt pyynnöt voivat joskus olla sivuston toiminnan kannalta haitallisia, joten käyttäjän kannalta on tärkeää pystyä näkemään, mitä pyyntöjä laajennus milläkin hetkellä estää, ja muokkaamaan, mitä pyyntöjä sallitaan tai estetään. Laajennus suorittaa kaiken tämän selaintoiminnon ja ponnahdusikkunan avulla (taulukko 2)




Kuva 6. uBlock Origin -laajennuksen selaintoiminto (ympyröity punaisella) Firefox (ylh.) ja Chrome (alh.) -selainten työkalupalkissa.

Taulukko 2. uBlock Origin -laajennuksen selaintoiminnon toimintaa havainnollistava taulukko.

| Tilanne | Seuraus | Kuva |
|---|---|---|
| Käyttäjä on sivustolla, joka ei sisällä estettäviä pyyntöjä. | Selaintoiminto poistettu käytöstä, joten sen ikoni muuttuu harmaan väriseksi. |  |
| Käyttäjä on sivustolla, jossa on yhdeksän estettyä pyyntöä. | Selaintoiminto on käytössä, joten sen ikoni muuttuu värilliseksi ja ilmaisee estettyjen pyyntöjen määrän numerolla. |  |

| | | |
|---|---|---|
| Käyttäjä painaa selaintoiminnon ikonia. | Laajennus avaa ponnahdusikkunan selaintoiminnon alapuolelle. Ponnahdusikkunan kautta käyttäjä voi käyttää laajennuksen eri toimintoja (esim. valita mitä elementtejä sallia tai estää). |  |

# 4 Ajanhallintalaajennuksen kehittäminen

## 4.1 Laajennuksen tarkoitus

Miettiessä omia tapojani käyttää internetselainta eräs tärkeimmistä havainnoistani oli sen rooli työnteon välttelyssä sekä tehokkaan työnteon estämisessä. Välttely ilmenee usein työnteon aloittamisen lykkäämisellä erilaisten huomiota vievien sivustojen, kuten uutisten tai sosiaalisen median, selaamisella. Työnteon tehokkuuteen vaikuttaa taas samanlaisen käyttäytymisen ilmeneminen kesken työnteon.

Ohjelmointialalla työskenteleville ja opiskeleville henkilöille selaimen käytön välttäminen kesken työnteon on myös usein vaikeaa, koska internet toimii usein tärkeänä resurssina erilaisen tiedon etsimiseen, joten selaimen toiminnan muokkaaminen on luonteva ratkaisu. Päätin kehittää laajennuksen, jonka pää-

asiallisena tehtävänä olisi välttelykäyttäytymisen vähentäminen ja häiriötekijöiden poistaminen. Tällaisia ohjelmia kutsutaan usein nimikkeellä ajanhallintaohjelmistot, joten viittaan tällä nimikkeellä kehittämääni laajennukseen.

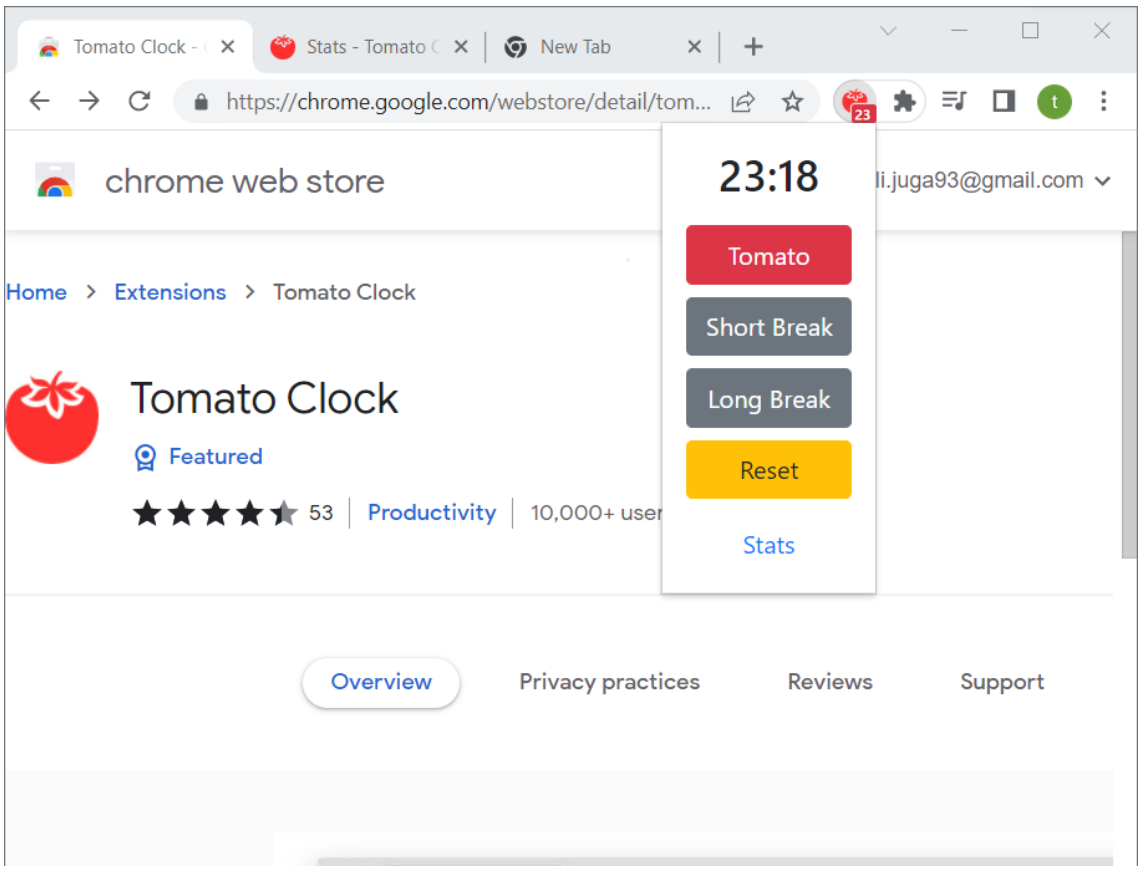
## 4.2 Kartoitus

Aloitin suunnittelun kartoittamalla laajennuksia, jotka vastaavat samoihin käyttötarkoituksiin. Eräs tavoitteitani vastaavista laajennuksista oli Tomato Clock [9]. Laajennus toimi antamalla käyttäjälle mahdollisuuden luoda erilaisia aikajaksoja, joiden aikana on tarkoitus työskennellä (kuva 7). Laajennus pohjautuu tunnettuun Pomodoro-ajanhallintamenetelmään.

Pomodoro on ajanhallintamenetelmä, joka perustuu työnteon jakamisen tietyn pituisiin aikajaksoihin ja niiden välissä pidettäviin taukoihin. Yleinen sääntö on tehdä neljä 25 minuutin mittaista jaksoa työskentelylle, joiden välissä pidetään viiden minuutin mittainen tauko. Kun nämä on suoritettu, pidetään pidempi 20-30 minuutin tauko. Tekniikan tarkoituksena on auttaa keskittymistä ja tehostaa työskentelyä. [10.]

Tomato Clock -lisäosa luokitellaan Chromen ja Firefoxin verkkokaupoissa kuuluvaksi tuottavuuskategoriaan. Tuottavuuslisäosien tarkoitus on, nimensä mukaisesti, lisätä käyttäjän tuottavuutta. Tämän kategorian lisäosat tarjoavat käyttäjälle tuottavuutta lisääviä toimintoja, kuten kirjanpitoa ja kalenterointia.

Testatessani laajennusta huomasin, että vaikka laajennus auttoi tehtävien ajastamisessa, se ei auttanut keskittymään itse työntekoon. Päätin kehittää laajennuksen, joka pystyisi parantamaan Tomato Clock -laajennuksen toimintaa tällä saralla.




Kuva 7. Esimerkki Tomato Clock -laajennuksen toiminnasta.

## 4.3 Laajennuksen toiminnalliset vaatimukset

Halusin aloittaa lisäosan suunnittelun helpoilla ja konkreettisilla tavoitteilla. Tärkeimpiä toiminnallisuuksia lisäosalle ovat aikajaksoissa työskentely ja häiritsevien sivustojen estäminen.

Käyttäjä suunnittelisi ennalta aikajakson, jolloin haluaisi työskennellä ja tämän aikajakson aikana käyttäjän määrittelemien haitallisten sivustojen selaaminen olisi kielletty.

## 4.4 Käyttötapaukset

Aloitin toiminnallisten vaatimusten kartoittamisen käyttötapausten suunnittelulla. Käyttötapauksilla kuvataan käyttäjän vuorovaikutusta ohjelmiston kanssa käyt-

täjän näkökulmasta. Jokaiseen käyttötapaukseen tulee liittyä tavoite, joka kertoo, mitä käyttäjä pyrkii saavuttamaan kyseisellä vuorovaikutuksella, sekä ehdot, joita vaaditaan käyttötapauksen aloittamiseen ja suorittamiseen.

Käyttötapausten avulla pystytään havainnollistamaan, millä eri tavoin ja miksi ohjelmistoa käytetään, joten ne toimivat hyvänä aloituskohtana suunnittelussa. Käyttötapauksiin liittyvät ehdot toimivat hyvänä tapana pohtia, miten ohjelmiston oikeaoppinen toiminta varmistetaan, joten ne toimivat hyvänä pohjana testien kehittämiselle.

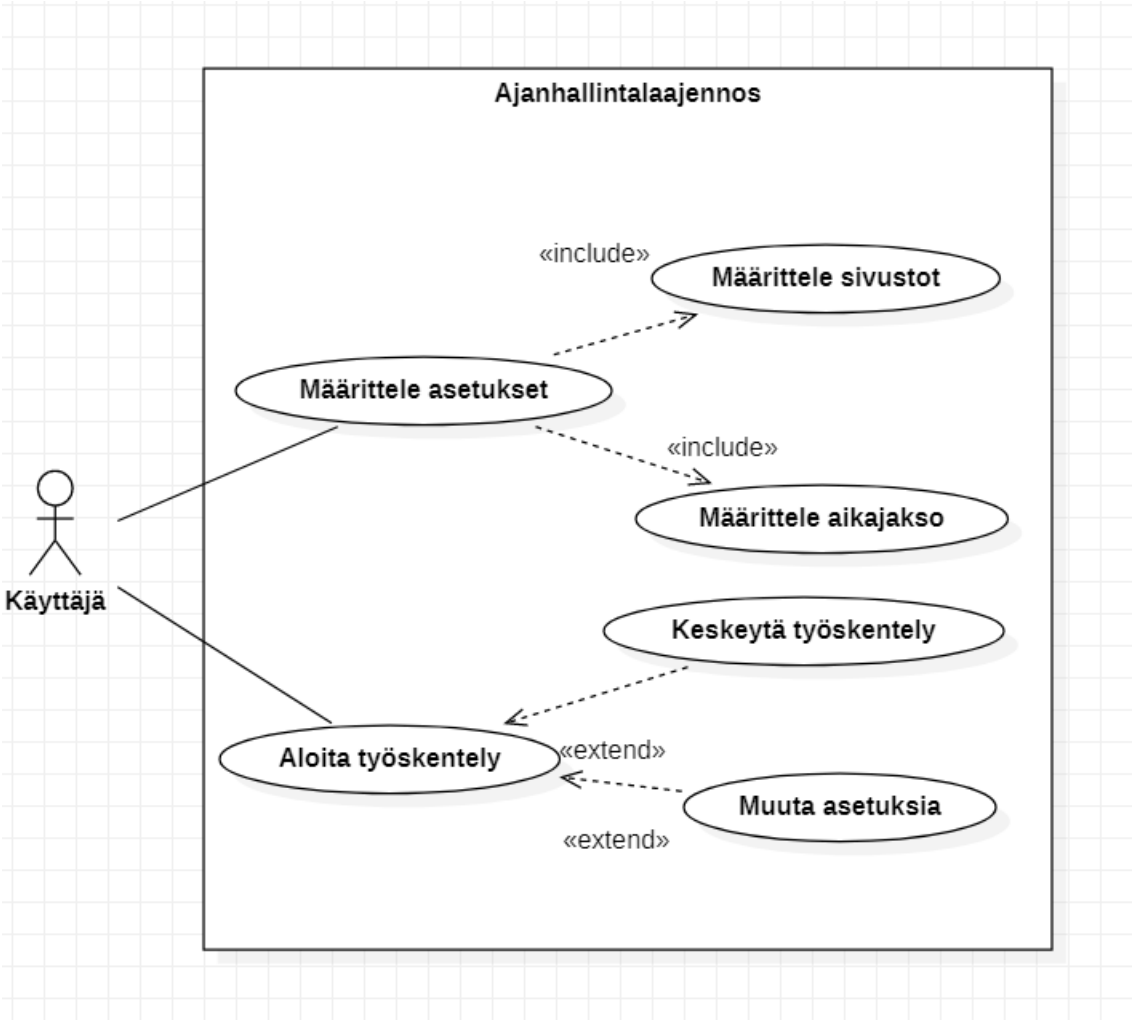
Yksittäinen käyttötapaus voi koostua siihen kuuluvista (include) tai sitä laajentavista (extend) alikäyttötapauksista, joiden avulla voi tarkemmin kuvata käyttötapauksen suorittamiseen liittyviä askelia (include) tai kuvata vaihtoehtoisia polkuja käyttötapauksen suorittamiselle (extend).

Aloitin käyttötapausten luomisen pohtimalla, miten voisin toteuttaa tärkeimmät tavoitteeni. Häiriöiden estämisen osalta tärkeä asia on selaamisen estäminen häiritseville sivustoille. Esteen luominen häiritsevän toiminnan välille tekee impulsiivisen toiminnan vaikeammaksi ja tarjoaa mahdollisuuden harkita tekoja tarkemmin.

Koska tarkoituksena on, että häiritsevä käyttäytyminen vaikeutuisi vain työskentelyn ajaksi, tulee käyttäjällä olla jokin tapa, millä kertoa ohjelmalle, milloin hän haluaa työskennellä. Käyttäjän tulee pystyä myös määrittelemään, mitkä sivustot hän luokittelee häiritseviksi.

Näiden havaintojen avulla loin kaksi erilaista käyttötapausta: määrittele asetukset ja aloita työskentelyjakso. Loin käyttötapausten pohjalta UML-kaavion, joka havainnollistaa tapauksia käyttäjän näkökulmasta (kuva 7). Käyttötapaus määrittele asetukset koostuu kahdesta alikäyttötapauksesta, jotka tarkentavat sen toimintaa. Käyttötapausta aloita työskentely taas laajennetaan kahdella alikäyttötapauksella: keskeytä työskentely ja muuta asetuksia. Laajentavat käyttöta-

paukset merkitsevät poikkeustilanteita, jotka eivät ole välttämättömiä käyttötapauksen suorittamiselle, mutta jotka voivat esiintyä käyttäjän toimien seurauksena.




Kuva 8. Laajennuksen käyttötapaukset

# 5 Laajennuksen toteutus

## 5.1 Kehitysympäristö

Koska laajennuksen kehitys vastaa melko paljon web-kehitystä, onNode.js-ympäristö luonteva valinta. Se mahdollistaa myös ohjelman jakamiseen komponentteihin, mikä helpottaa laajennuksen ylläpitoa ja testien luomista. Node.js-ympäristölle on olemassa myös suuri määrä erilaisia kehitystyötä helpottavia kirjastoja.

Käytin pakettien lataamiseen NPM-ohjelmaa. Se vaatii toimiakseen package.json-tiedoston, joka määrittelee projektin riippuvuudet sekä sen testaamiseen ja rakentamiseen käytettävät skriptit (esimerkkikoodi 2).

Node.js ympäristön käyttämät CommonJS-moduulit poikkeavat selainten käyttämistä moduuleista, joten moduulien käyttämiseen laajennuksessa vaaditaan WebPack-moduulia, joka ohjelman rakentamisen yhteydessä yhdistää moduulien koodin yhdeksi JavaScript-tiedostoksi.

Käytin Karma-kirjastoa yksikkötestien luomiseen, koska se mahdollistaa testien ajamisen selaimella.  Karmalle kerrotaan vain testien ja selaimen sijainti, ja se osaa automaattisesti ajaa testit selaimessa.

Web-ext on Mozillan tuottama moduuli, joka nopeuttaa kehitystä monella tapaa. Se muun muassa vahtii muutoksia laajennuksen sisältämissä tiedostoissa ja lataa ne välittömästi selaimen muistiin, nopeuttaen kehitysprosessia huomattavasti.

Sinon-chrome on moduuli, joka mahdollistaa yksikkötestien luomisen. Se sisältää korvikeoliot (mock-objects) Firefox- ja Chrome-selainten JavaScript-rajapinnoille. Korvikeolioiden avulla voidaan varmistaa, että testien epäonnistuminen johtuu varmasti virheestä koodissa, eikä selaimessa olevasta virheestä. Ne mahdollistavat myös testaamisen, ilman itse selaimen avaamista, joten testien ajaminen voidaan automatisoida integraatiotestausta varten.

```
{
  "name": "Ajanhallinta työkalu",
  "version": "1.0.0",
  "description": "Extension for helping with studying.",
  "main": "index.js",
  "scripts": {
    "test": "karma start",
    "test:debug": "karma start --no-single-run --auto-watch",
    "web-ext": "web-ext run -s ./addon",
    "build": "webpack"
  },
  "keywords": [],
  "author": "Taneli Juga",
  "license": "ISC",
  "devDependencies": {
    "webpack-cli": "^4.9.2",
    "chai": "^4.3.6",
    "expect.js": "^0.3.1",
    "karma": "^6.3.19",
    "karma-firefox-launcher": "^2.1.2",
    "karma-mocha": "^2.0.1",
    "mocha": "^10.0.0",
    "sinon-chrome": "^3.0.1",
    "web-ext": "^6.8.0",
    "webpack": "^5.72.0"
  }
}
```

Esimerkkikoodi 2. Projektin package.json-tiedosto

## 5.2 Laajennuksen rakenne

Laajennus koostuu neljästä tiedostosta (taulukko 3), joilla kullakin on oma vastuunsa laajennuksen toiminnassa. Laajennus koostuu kolmesta HTML-dokumentista, joiden kautta käyttäjä hallinnoi laajennuksen toimintaa, sekä yhdestä taustaskriptistä, joka sisältää laajennuksen bisneslogiikan.

Taulukko 3. Laajennuksen tiedostot.

| Tiedostot | Tyyppi | Tarkoitus |
|---|---|---|
| • request-handler.js | Taustaskripti | Estää selaimen navigoinnin estetyille verkkotunnuksille. |
| • options.html<br>• options.js<br>• options.css | HTLM-dokumentti | Sivu, jonka avulla käyttäjä konfiguroi estettävät verkkotunnukset. |
| • popup.html<br>• popup.js<br>• popup.css | HTLM-dokumentti | Sivu, jonka avulla käyttäjä hallinnoi session kulkua. |
| • dashboard.html<br>• dashboard.js<br>• dashboard.css | HTLM-dokumentti | Sivu, jolle käyttäjä ohjataan, navigoituaan estetylle sivulle. Sisältää käyttäjälle hyödyllistä tietoa laajennuksen tilasta. |

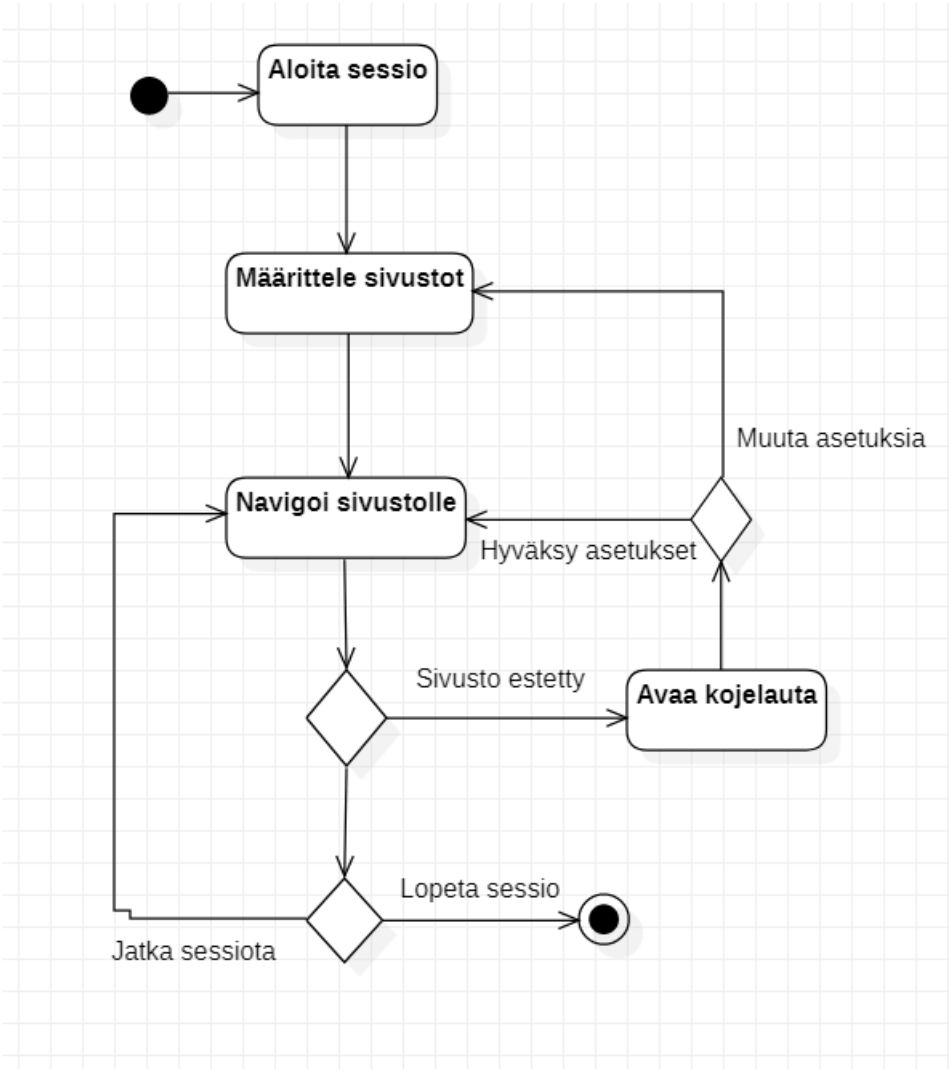
Nämä tiedostot ja niiden toiminta määritellään myös manifestissa (esimerkkikoodi 3). Manifestissa määritellään myös laajennuksen selaintoiminnon ulkonäkö ja sen yhteyteen kytketään ponnahdusikkuna. Tämä saavutetaan antamalla browser_action-avaimen sisäiselle default_icon-avaimelle arvo, joka osoittaa kuvatiedostoon, sekä antamalla default_popup-avaimelle arvo, joka osoittaa ponnahdusikkunan tiedostoon (popup.html).

```
{
  "manifest_version": 2,
  "name": "Time manager",
  "description": "A browser extension to help with time management.",
  "version": "1.0",

  "browser_action": {
    "browser_style": true,
    "default_title": "Time manager",
    "default_popup": "popup/popup.html",
    "default_icon": {
      "19": "icons/block_inactive.svg",
      "38": "icons/block_inactive.svg"
    }
  },

  "icons": {
    "48": "icons/block_inactive.svg",
    "96": "icons/block_inactive.svg"
  },

  "background": {
    "scripts": [
      "background_scripts/request-handler.js"
    ]
  },

  "options_ui": {
    "page": "options/options.html",
    "browser_style": true
  },

  "web_accessible_resources": [
    "dashboard/dashboard.html"
  ],

  "permissions": ["webRequest", "webRequestBlocking", "storage",
"<all_urls>"]
}
```

Esimerkkikoodi 3. Laajennuksen manifest.json-tiedosto

## 5.3 Laajennuksen toiminta

Aloitin toiminnan suunnittelun minimivaatimusten määrittelyllä aktiviteettikaavion muodossa (kuva 9). Sessiolla tarkoitetaan aikajaksoa, jolloin laajennus on toiminnallinen.




Kuva 9. Aktiviteettikaavio laajennuksen toiminnasta. Kun laajennus havaitsee käyttäjän menneen estetylle sivustolle, laajennus ohjaa käyttäjän kojelauta-sivulle (dashboard.html), josta käyttäjä voi muuttaa laajennuksen asetuksia tai jatkaa selausta.

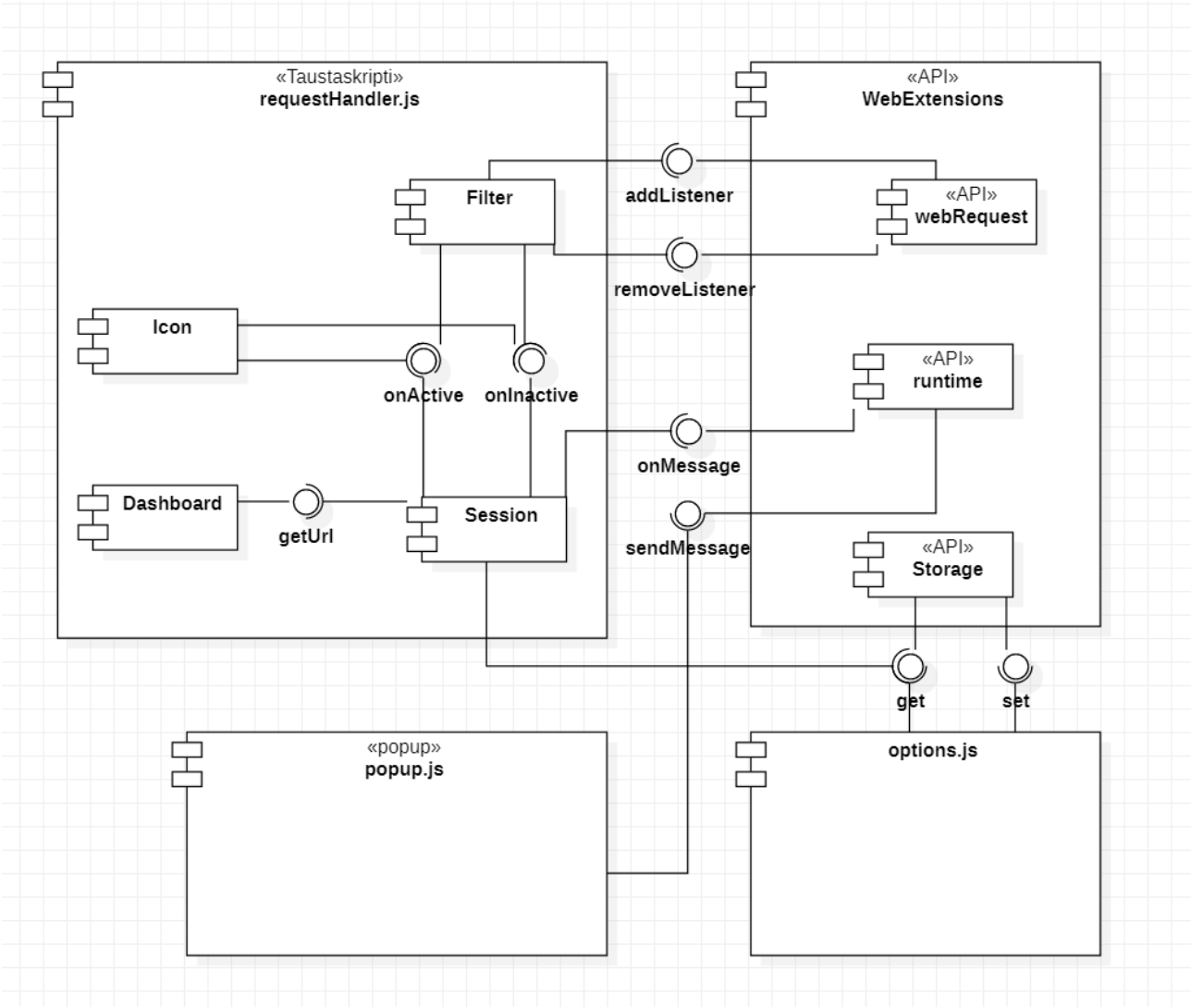
Käyttäjä hallitsee laajennusta painamalla laajennuksen selaintoimintoa, joka avaa ponnahdusikkunan. Ponnahdusikkunan kautta käyttäjä voi aloittaa tai lopettaa opiskelusession.

Ponnahdusikkunan JavaScript-tiedosto lähettää laajennuksen viesti-rajapinnan avulla (runtime.sendMessage) viestin taustaskriptille, joka käynnistää session. Tämän jälkeen sessio-objektiin yhdistetyt kuuntelijat suoritetaan.

Kun sessio on käynnissä, ikoni muuttaa väriä ilmaisemaan käyttäjälle, että laajennus on toiminnassa. Se aiheuttaa myös Filter-objektin aktivoitumisen, joka yhdistää selaimen WebRequest-rajapintaan kuuntelijan, joka vastaanottaa selaimen lähettämiä pyyntöjä. Filter-objekti tarkistaa, vastaako selaimen lähettämän pyynnön URL-osoite käyttäjän antamien osoitteiden listalla. Jos osoite on listalla, ohjataan käyttäjä dashboard-sivulle, josta hän näkee laajennuksen tilan ja voi mahdollisesti muuttaa sen asetuksia. Käyttäjä voi esimerkiksi sallia estetyn sivuston, joten estettyä sivustoa voi päästä selaamaan.

Osoitteen ei tarvitse vastata täydellisesti, koska laajennus tarkistaa vain osoitteen verkkotunnuksen, johon ei kuulu URL-osoitteen polkua tai protokollaa. Tästä seuraa, että jos käyttäjä on estänyt osoitteen ”www.example.com”, estää laajennus myös osoitteet ”https://www.example.com” ja ”www.example.com/some/resource”.

Koska laajennus on tehty Firefox-selaimelle, se käyttää WebExtension-rajapintaa, joka poikkeaa hieman Chromen rajapinnoista. Laajennus käyttää yhteensä kolmea WebExtensions-rajapintaa toiminnallisuuden saavuttamiseen (kuva 10). Ponnahdusikkuna viestii taustaskriptille käyttäen message-rajapintaa, jonka taustaskripti (requestHandler.js) vastaanottaa. Asetussivu (options.js) tallentaa käyttäjän asetukset storage-rajapinnan avulla selaimen muistiin, josta voi noutaa ne myöhemmin. Taustaskripti tarvitsee webRequest-rajapintaa selaimen lähettämien pyyntöjen vastaanottamiseen.



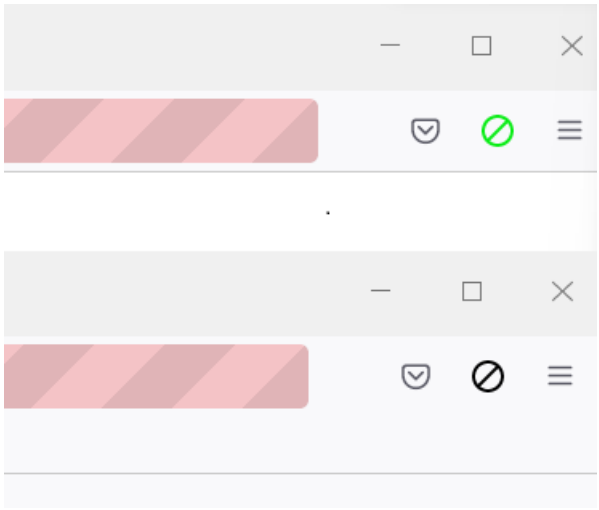

Kuva 10. Komponenttikaavio laajennuksen rakenteesta

Taustaskripti koostuu kolmesta pienemmästä komponentista: Iconista (ikoni), Filtteristä (suodatin) sekä Sessionista (sessio). Käytännössä ne ovat JavaScript-olioita. Sessio-olio sisältää rajapintanaan kaksi metodia (onActive, onInactive), joita ikoni- ja suodatinkomponentit käyttävät. Metodit vastaanottavat parametrinä callback-funktion, joka suoritetaan käyttäjän aloittaessa session. Näin oliot voivat kytkeä toiminnallisuutta session alkamiseen ilman, että itse sessio-oliota tarvitsee muokata. Vastaavasti, jos kaikki logiikka toteutettaisiin sessio-olion sisällä, jouduttaisiin sitä muokkaamaan aina, kun halutaan lisätä toiminnallisuutta sessioon, joten se olisi tiukemmin kytketty muihin komponentteihin, mikä vaikeuttaa jatkokehitystä.

Ikoniolion vastuuna on vaihtaa selaimen selaintoiminnon ulkonäköä, kun laajennus on toiminnassa. Tämä saavutetaan vaihtamalla ikonin kuvaa browserAction-rajapinnan avulla (esimerkkikoodi 3). Kun laajennus ei ole aktiivinen, ikoni on musta, mutta vaihtuu aktivoitumisen jälkeen vihreäksi (kuva 11).

```
const Icon = {
  setIcon(img_name){
    const path = `icons/${img_name}`
    browser.browserAction.setIcon({path: path});
  },
  async init(){
    Session.onActive(this.setIcon("block_active.svg"))
    Session.onInactive(this.setIcon("block_inactive.svg"))
  }
}
```

Esimerkkikoodi 4. Ikoniolion toteutus koodissa.



Kuva 11. Ikonin ulkonäkö käyttäjän aloitettua session (ylh.) ja käyttäjän lopetettua session (alh.).

# 6 Yhteenveto

Halusin tällä työllä tutkia selainlaajennusten kehitystä sekä koittaa itselleni hyödyllisen ohjelmiston luomista. Laajennuksen kehittäminen oli kaikkiaan mielenkiintoinen prosessi, jossa voi hyödyntää web-kehityksen taitoja, mutta hieman

erilaisessa kontekstissa. Koin myös hyvänä selainten yksinkertaisen käyttöliittymän, koska kehittämisen voi aloittaa matalalla kynnyksellä, joten voi suoraan keskittyä laajennuksen logiikan toteuttamiseen.

Ohjelmiston lopullinen toiminnallisuus jäi hieman vajaaksi alkuperäisestä suunnitelmasta, mutta toimii silti hyvänä alkuna jatkokehitykselle. Alkuperäisessä suunnitelmassa käyttäjä pystyisi luomaan aikajaksoja, jolloin laajennus olisi toiminnassa, mutta aikarajoitteiden takia tämä toiminnallisuus jäi puuttumaan.

Jatkokehityksen kannalta mielenkiintoista olisi laajennuksen toiminnallisuuden parantaminen vastaamaan alkuperäistä suunnitelmaa. Toiminnallisuutta voisi koittaa myös lisätä. Esimerkiksi käyttäjän kannalta tärkeää olisi pystyä seuraamaan omia saavutuksiaan sekä estettyjen sivustojen määrää.

**Lähteet**


1 Statista Research Department. 2021. Ad blocking user penetration rate in the United States from 2014 to 2021. Verkkoaineisto <https://www.statista.com/statistics/804008/ad-blocking-reach-usage-us/>. Luettu 18. Elokuu 2021.

2 Statcounter. 2021. Browser Market Share Worldwide. Verkkoaineisto <https://gs.statcounter.com/browser-market-share#monthly-202105-202105-bar>. Luettu 18. Elokuu 2021.

3 Microsoft. 2022. Port a Chrome extension to Microsoft Edge. Verkkoaineisto <https://docs.microsoft.com/en-us/microsoft-edge/extensions-chromium/developer-guide/port-chrome-extension>. Luettu 18. Elokuu 2021.

4 Mozilla. 2021. Add-ons. Verkkoaineisto <https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons>. Luettu 18. Elokuu 2021.

5 Arstechnica. 2020. The rise and fall of Adobe Flash. Verkkoaineisto <https://arstechnica.com/information-technology/2020/07/the-rise-and-fall-of-adobe-flash/>. Luettu 9.5.2022.

6 Google 2020. Extensions platform vision. Verkkoaineisto <https://developer.chrome.com/docs/extensions/mv3/intro/platform-vision/>. Luettu 9.5.2022.

7 Zdnet. 2019. Apple neutered ad blockers in Safari, but unlike Chrome, users didn't say a thing. Verkkoaineisto <https://www.zdnet.com/article/apple-neutered-ad-blockers-in-safari-but-unlike-chrome-users-didnt-say-a-thing/>. Luettu 9.5.2022.

8 uBlock. 2019. Safari platform. Verkkoaineisto <https://github.com/gorhill/uBlock/blob/master/platform/safari/README.md#safari-platform>. Luettu 9.5.2022.

9 Chrome Web Store. 2021. Tomato Clock. Verkkoaineisto <https://chrome.google.com/webstore/detail/tomato-clock/enemipdanmallpjakiehedcgjmibjihj?hl=en>. Luettu 10.5.2022.

10 Koulutus.fi 2022. Pomodoro on kokeilun arvoinen ajanhallintamenetelmä. Verkkoaineisto <https://www.koulutus.fi/artikkelit/pomodoro-tekniikka-on-toimiva-ajanhallintamenetelma-17172>. Luettu 10.5.2022.