

Dnevnik rada

1. tjedan (09.10.2017.-15.10.2017.)

Dosadašnji rad

Na sastanku s mentorom i asistentima kolega Štoković i ja smo dobili zadatak zajedno obraditi temu plugina za browsere. Kao primjer je bio zadan plugin koji ispisuje opis i značenje riječi pri prelasku miša preko riječi.

Daljnji koraci

Proučiti kako se rade pluginovi te što se sve može s njima postići i ostvariti.

2. tjedan (16.10.2017.-22.10.2017.)

Dosadašnji rad

Proučio kako se rade ekstenzije za browsere, tj. specifično za Google Chrome. Budući da je potrebno znanje Javascript-a i HTML-a za razvoj pluginova sam poradio i na tome jer se nisam prije susretao s tim programskim jezicima. Napravio dva jednostavna plugina za Google Chrome od kojih je jedan za izbor boje pozadine trenutne web stranice, a drugi za zamjenu svih pojava neke riječi ili izraza koju korisnik sam odabere nekom također proizvoljnom riječi ili izrazom na trenutno otvorenoj web stranici.

Daljnji koraci

Proučiti kako napraviti kompleksnije ekstenzije kao npr. riječnik koji čita iz neke datoteke ili iz nekog drugog izvora te traži riječ nad kojom se nalazi miš.

3. tjedan (23.10.2017.-29.10.2017.)

Dosadašnji rad

Proučio razlike u razvoju ekstenzija Firefox-a i Google Chrome-a.

Glavna datoteka:

- manifest.json (Chrome)
- package.json (Firefox)

Dodavanje korisničkom sučelju browsera:

- Ako želimo dodati izbornike, gumbе na alatnoj traci ili slično u Google Chrome-u, moramo ih statički uključiti u manifest.json datoteku
- Firefox je fleksibilniji što se tiče toga jer su elementi za interakciju specificirani u glavnom modulu te se ekstenzija prilagođava stranici

Sadržajne skripte (Content Scripts):

- Chrome je najjednostavniji jer se Javascript i CSS mogu ubaciti u bilo kojem trenutku pomoću API poziva
- Za Firefox se koriste razni modovi za uključivanje skripti budući da je dosta kompliciraniji proces od Chrome-ovog

Pozadinski procesi i slanje poruka (Message Passing):

- Chrome ima jednostavan model za slanje poruka korištenjem prilično laganog API-ja, poruke se mogu slati međusobno iz iskočnih prozora browsera, pozadinskih skripti ili procesa te sadržajnih skripti korištenjem takozvanih "message listeners"
- Firefox također ima prilično kvalitetan način za slanje poruka tako da se jednostavno doda "event listener" na određenu poruku koju želimo obraditi

Kodiranje:

- Može se koristiti bilo koji standardni HTML i Javascript editor
- Za ubacivanje ekstenzije u Chrome jednostavno iz browsera odaberemo lokaciju direktorija ekstenzije s računala te ako izmijenimo kod ekstenzije sve što trebamo napraviti da bi browser pokretao najnoviju verziju ekstenzije jest pritisnuti reload gumb unutar Chrome-a
- Firefox je dosta kompliciraniji od Chrome-a po tome što svaka promjena u kodu dovodi do obaveze za ponovnim kompajliranjem i reloadanjem ekstenzije

Debugiranje:

- Chrome je ponovno bolji budući da sadrži "web developer toolbar" te se unutar tog jednog sučelja mogu debuggati baš svi dijelovi ekstenzije uključujući pozadinske procese, sadržajne skripte te korisnička sučelja browsera, jednostavan dio Chrome-a console.log ispisuje sve poruke u konzoli
- Firefox nije toliko ugodan kao Chrome što se tiče debugiranja te je poprilično kompliciraniji, ali s određenim dodacima se može taj proces olakšati

Na temelju kratkog istraživanja zaključio sam da je Chrome jednostavniji za korištenje te sam i ja sam korisnik Chrome-a zbog čega sam se opredijelio za razvoj ekstenzije za Google Chrome.

Daljnji koraci

Učitati Javascript i HTML detaljnije kako bih bio spreman na sve potrebe razvoja ekstenzije.

4. tjedan (30.10.2017.-05.11.2017.)

Dosadašnji rad

Kod razvoja ekstenzije u pregledniku Google Chrome imamo opciju dodati iskočni prozor koji se aktivira klikom na ikonu desno od adresne trake kao na [slici](#).



Za početak moramo napraviti datoteku s nastavkom .html i spremiti ju unutar direktorija u kojem su sve ostale datoteke koje definiraju našu ekstenziju. Zatim trebamo navesti novu datoteku koju smo nazvali npr. popup.html unutar manifest.json datoteke u kojoj se nalaze glavne odrednice naše ekstenzije.

```
{
  "manifest_version": 2,

  "browser_action": {
    "default_icon": "icon.png",
    "default_popup": "popup.html"
  },

  "permissions": [
    "activeTab",
    "storage"
  ]
}
```

U navedenom isječku gledamo dio koda koji spada pod browser_action. Pod default_icon navodimo puno ime slike za koju želimo da bude zadana slika naše ekstenzije, a pod default_popup navodimo puno ime datoteke koju smo napravili za naš iskočni prozor. Slika koju smo izabrali će biti vidljiva pored adresne trake te će se na nju moći kliknuti što će izazvati otvaranje iskočnog prozora koji izgleda onako kako smo definirali u datoteci koju smo naveli pod default_popup. Budući da je izgled i funkcionalnost iskočnog prozora definirana html datotekom, uređujemo ga kao bilo koju drugu stranicu koju bi uređivali pomoću HTML-a.

5. tjedan (06.11.2017.-12.11.2017.)

Dosadašnji rad



Daljnje istraživanje JavaScripta i načina na koji možemo implementirati označavanje teksta. Zaključili da će se trebati koristiti Chrome-ov `runtime.sendMessage` API i funkcija `addListener`. Funkcija `addListener` osluškuje događaj koji smo definirali, tj. čeka da se taj isti događaj dogodi te potom poduzima neku akciju koju također definiramo. Mi želimo dodati Listener na kraj selekcije budući da nam to signalizira kraj selekcije teksta. Za to nam koristi događaj “`mouseup`” što označava otpuštanje klika miša. Osim `mouseup` eventa, tj. događaja postoje još i:

- `onclick`,
- `ondblclick`,
- `onmouseover`,
- `onmouseout`,
- `oncontextmenu`,

i mnogi drugi. Unutar funkcije koja se izvede kada se naš `mouseup` događaj dogodio smo koristili jedan poziv funkcije `runtime.sendMessage` te smo kao parametar iskoristili tekst koji smo dobili ranije. Korištenjem `sendMessage` funkcije smo tako označeni tekst poslali u pozadinsku, odnosno background, skriptu kojoj mogu pristupati svi ostali elementi, odnosno skripte, ekstenzije. U pozadinskoj skripti smo koristili funkciju `onMessage` pomoću koje smo primili i spremili dobiveni tekst u varijablu.

6. tjedan (13.11.2017.-19.11.2017.)

Dosadašnji rad

Osmislili i započeli dizajn iskočnog prozora uređivanjem `popup.html` datoteke. Cilj nam je da ekstenzija bude jednostavna za koristiti te samim time i da ima i jednostavan izgled. Na vrhu će biti ime naše ekstenzije (Sequoia), postojat će izbor jezika u `select` elementu te gumb na čiji klik će se prevesti označeni tekst. Ispod gumba će biti 2 `paragraph` elementa pravokutnog oblika unutar kojih se pojavljuju označeni tekst, odnosno prijevod.

7. tjedan (20.11.2017.-26.11.2017.) MEĐUISPITI

8. tjedan (27.11.2017.-03.12.2017.) MEĐUISPITI

9. tjedan (04.12.2017.-10.12.2017.)

Dosadašnji rad

Napisali kod za popup.html datoteku, tj. napravili dizajn iskočnog prozora te je sve ostalo kako je bilo planirano. Za manevriranje podacima i obavljanje prijevoda smo napravili skriptu popup.js u kojoj iz select elementa iskočnog prozora preuzmemo odabrani jezik te pomoću chrome.runtime.getBackgroundPage() funkcije povlačimo označeni tekst, koji smo prethodno pohranili u background skriptu, u novonapravljenju skriptu popup.js.

10. tjedan (11.12.2017.-17.12.2017.)

Dosadašnji rad

Kako bi mogli ostvariti željeni prijevod, počeli smo razmatrati opcije za ostvarivanje istog. Kao opcije smo imali korištenje API-ja nekog online prevoditelja, stvaranje tekstualne datoteke s bazom riječi i definicijama istih ili preuzimanje tekstualne datoteke s riječnikom određenog jezika s nekog web poslužitelja i čitanje definicija odabranih riječi iz te tekstualne datoteke. Odlučili smo se za korištenje API-ja online prevoditelja budući da se tako može lakše dinamički prevesti željeni tekst na bilo koji jezik s puno manje napora.

Prva opcija nam je bio Google Translate, no budući da se korištenje njegovog API-ja plaća (postoji free trial opcija koja traje mjesec dana) trebali smo naći neku drugu opciju te smo se odlučili za Yandex i njegov Translate API. Na njihovoj službenoj stranici su bile sve upute koje su potrebne za slanje API zahtjeva.

[Yandex Translate API dokumentacija](#)

11. tjedan (18.12.2017.-24.12.2017.)

Dosadašnji rad

Nakon biranja načina prevođenja, jedino nam je preostalo doraditi kod kako bi se ostvario prijevod. Za to koristimo klasu XMLHttpRequest. Ta klasa sadrži metode "open" i "send" kojima definiramo vrstu zahtjeva koji šaljemo te URL koji je napravljen po uputama na Yandex Translate API dokumentaciji. Unutar još jedne metode XMLHttpRequest klase, "onreadystatechange", smo definirali ponašanje u trenutku kada readyState varijabla bude 4 (operacija zahtjeva je završena) te status varijabla bude 200 (dobiven je odgovor te pohranjen unutar tijela naše instance XMLHttpRequest klase). Yandex Translate API vraća odgovor u formatu XML te smo iz njega izvukli prijevod našeg teksta razdvajanjem odgovora po tagovima (slično kao tagovi u HTML-u).

XMLHttpRequest readyState

XMLHttpRequest status

12. tjedan (25.12.2017.-31.12.2017.) PRAZNICI

From:
<http://studentski-izvjestaji.zesoi.fer.hr/> - Studentski izvještaji

Permanent link:
http://studentski-izvjestaji.zesoi.fer.hr/doku.php?id=studenti:shadi_zahran:sz_dnevnik&rev=1517184687

Last update: **2023/06/19 18:20**

