

Open Source Software Project Development

Dr. T.Y. Wong

Week 13 - 14

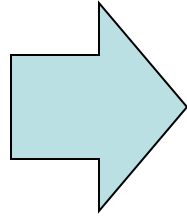
Experiencing Chrome Extensions

- Nothing is impossible for extensions.

Chrome Extension VS Firefox Extension

Firefox

Although we don't
teach it, you should
know about it.

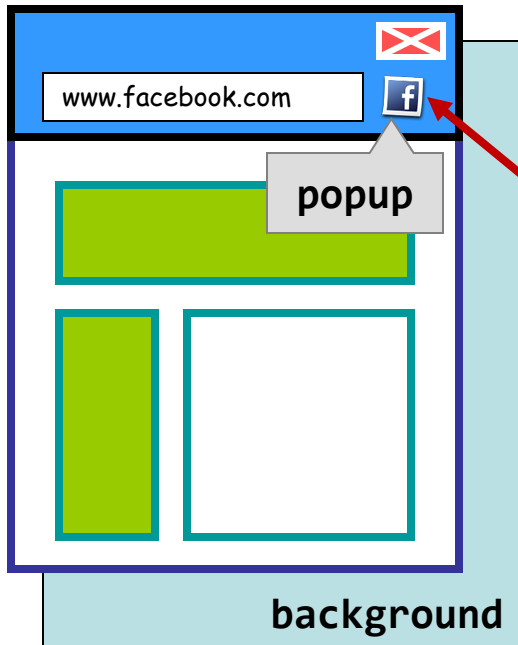


UI Design	Unlimited.	Restrictive.
Bypassing SOP: DOM & XHR	Yes.	Yes, but requiring permission
Storage	Full local file I/O, cookies.	Cookies, HTML5 storage, Partial local file I/O.
JavaScript Injection	Difficult.	Easy

Execution Environment



- The concepts:



Component #1: Background

The background is a HTML which is loaded when the extension is loaded. This is an optional component.

Component #2: BrowserAction

This is the icon you can commonly find next to the location bar. This is an optional component.

Component #3: Popup

When the BrowserAction is clicked, the extension may show a HTML called the popup. This is again optional.

Chrome Extension

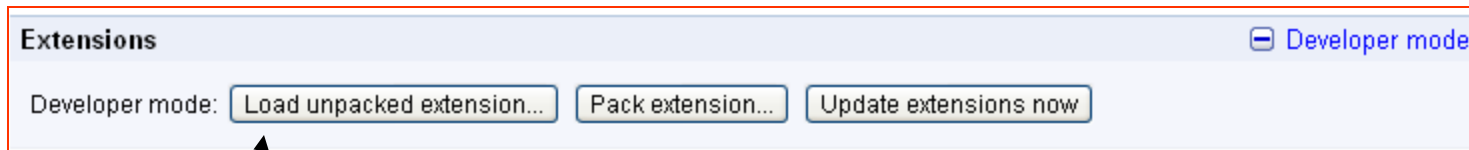
- Let's start with “*Hello World*”**

Development Environment

- Very developer-friendly.



Extensions



This allows you to load the entire development directory.

Boo... No extensions installed :-)

Want to [browse the gallery](#) instead?

Try “Menu button” -> “Tools” -> “Extensions”; Or “chrome://extensions”

Source open?

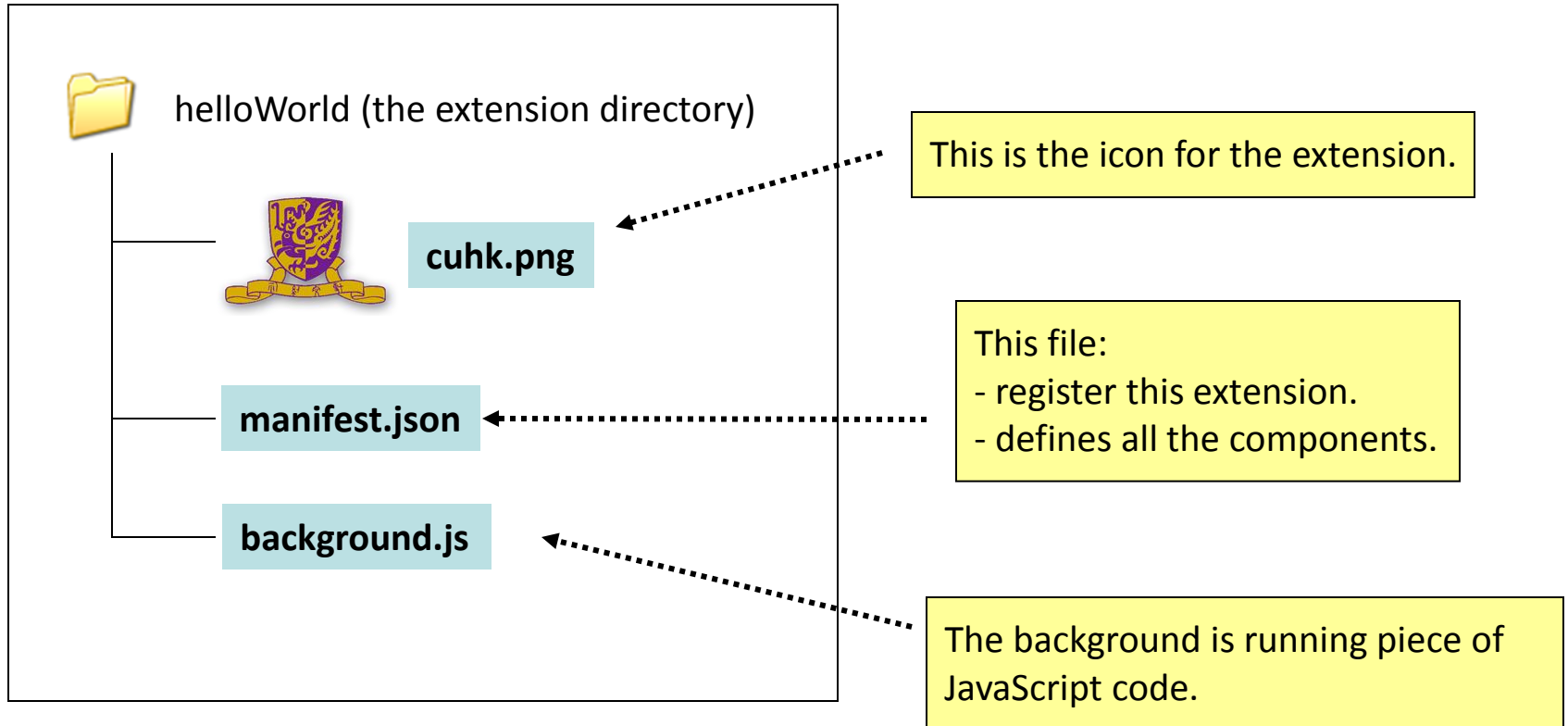
- You can dig out the source easily!

Mac OS X: ~/Library/Application Support/Google/Chrome/Default/Extensions

Linux: ~/.config/google-chrome/Default/Extensions

**Windows: C:\Users\tywong\AppData\Local\Google\Chrome\
User Data\Default\Extensions**

Example #1: Development directory



See "helloWorld.zip"

Example #1: Manifest JSON

- So, the configuration is a JSON object...

```
{  
  "name": "Hello World",  
  "manifest_version": 2,  
  "version": "1.0",  
  "description": "For CSCI4140",  
  "icons": {  
    "48": "cuhk.png"  
  },  
  "background": {  
    "scripts": ["background.js"]  
  },  
  "browser_action": {  
    "default_icon": "cuhk.png"  
  }  
}
```

Basic things. Not so important.

Defining the background script.

Defining the browserAction.

We first omit the popup HTML in this example.

See “manifest.json” in “helloWorld/”

Example #1: background

```
var count = 0;  
chrome.browserAction.onClicked.addListener(  
  function(tab) {  
    alert("Hello!");  
    count++;  
    console.log("Count = " + count);  
  }  
);
```

The console helps a lot while debugging!
But, where is it?

Fires with the icon is clicked!

However, it won't fire when
the browserAction has its
popup HTML registered.



Reference: <http://code.google.com/chrome/extensions/browserAction.html#event-onClicked>

See “background.html” in “helloWorld/”

Example #2: Adding popup in Manifest JSON

```
{
  "name": "Hello World with popup",
  "manifest_version": 2,
  "version": "1.0",
  "description": "For CSCI4140",
  "icons": {
    "48": "cuhk.png"
  },

  "background": {
    "scripts": ["background.js"]
  },

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

Old stuffs...

New stuff!

This defines the HTML when the browserAction's icon is clicked.

See “manifest.json” in “helloWorld_popup/”

Example #2: popup

```
function init() {  
    var colorArray = ["red", "green", "blue", ... ];  
    var index = Math.floor( Math.random() * colorArray.length );  
    document.body.style.color = colorArray[index];  
    console.log(colorArray[index]);  
}  
  
addEventListener("load", init, false);
```

```
<html>  
  
<script type="text/javascript" src="popup.js"></script>  
  
<body>  
<p><noBr><b>Hello World!</b></noBr></p>  
</body>  
</html>
```

Well, whether this HTML (1) acts like “background”, i.e., always running, or (2) runs on demand?

Let's try it out!

See “popup.html” and “popup.js” in “helloWorld_popup/”


Chrome Extension

- Let's interact with the tabs!

Learn with a purpose!


- Let's write an extension to detect whether you're playing Facebook or not!

Background's Task




Set this icon when:

- your selected tab is on Facebook;



Set this icon when:

- your selected tab is not on Facebook;



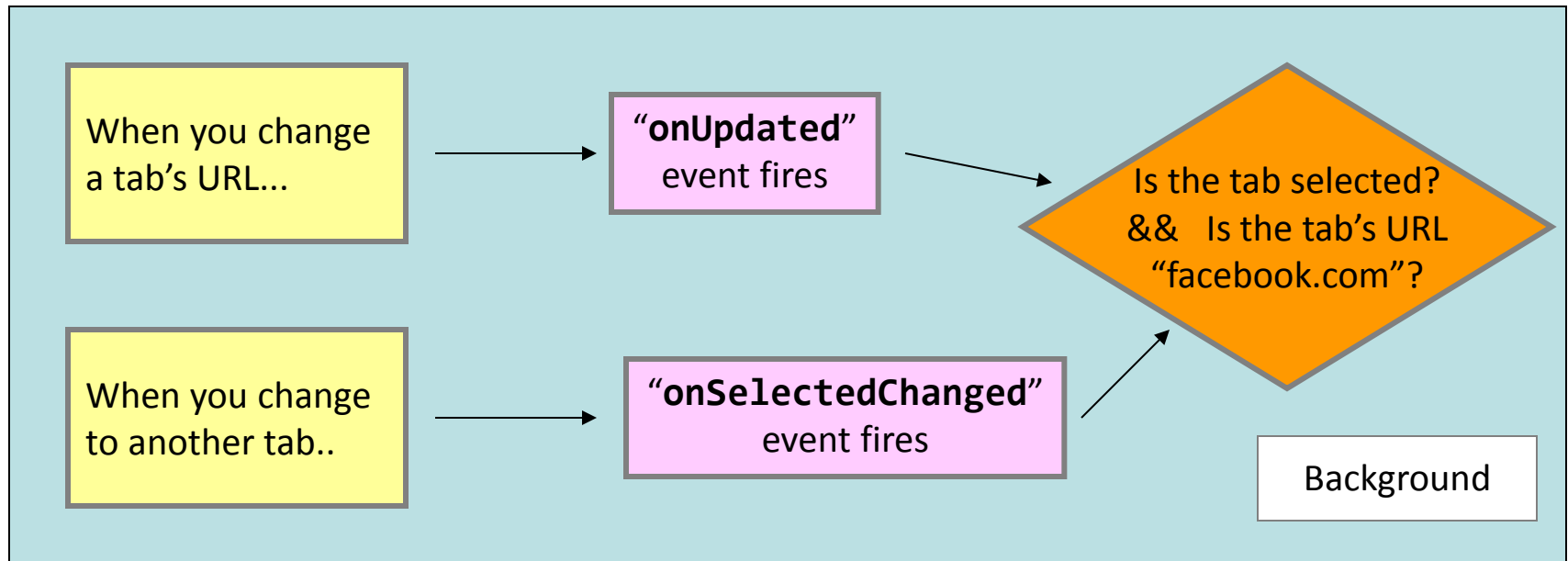
Increase this count when:

- you change the location of the selected tab to Facebook;
- you change to another tab which is on Facebook;

See “**chrome.browserAction.setBadge*()**”:
<https://developer.chrome.com/extensions/browserAction.html>

Tab events...

- Tab-related event handling:
 - (1) **onUpdated** and (2) **onSelectionChanged**.



Reference: <http://code.google.com/chrome/extensions/tabs.html>

See "background.html" in "detect_facebook/"

Misc...

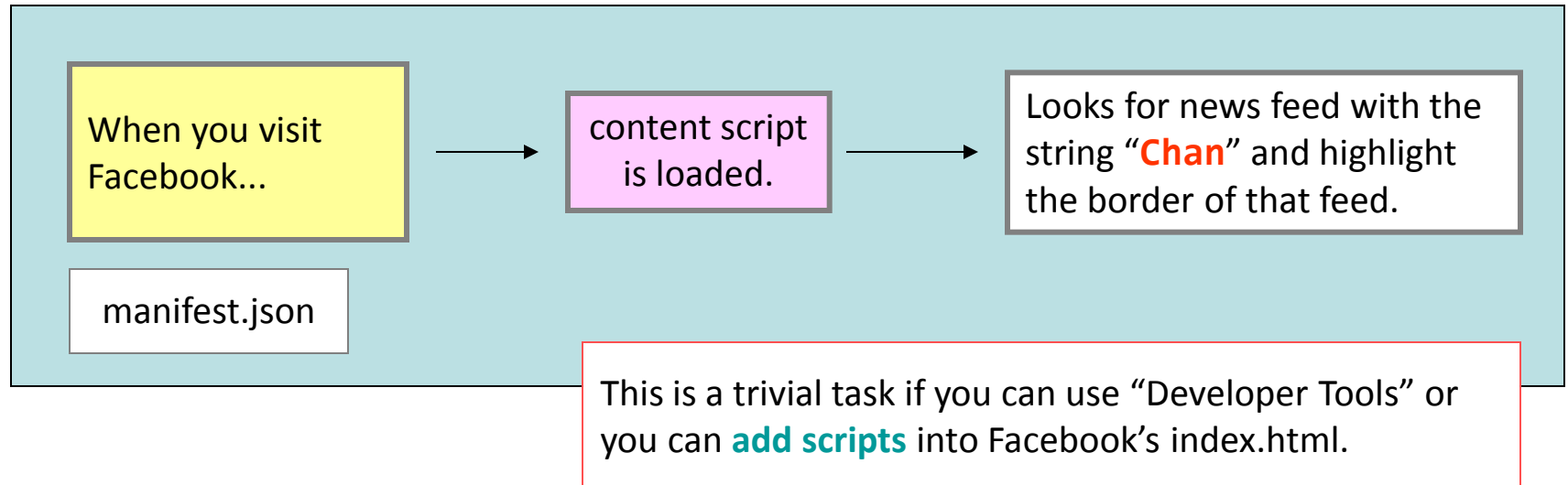
- Permission in “**manifest.json**”
 - <http://code.google.com/chrome/extensions/tabs.html#manifest>
- Display the “*Facebook-playing counter*” and changing icons?
 - <http://code.google.com/chrome/extensions/browserAction.html>

Chrome Extension

- Let's try injecting codes & message passing!

JavaScript injection?

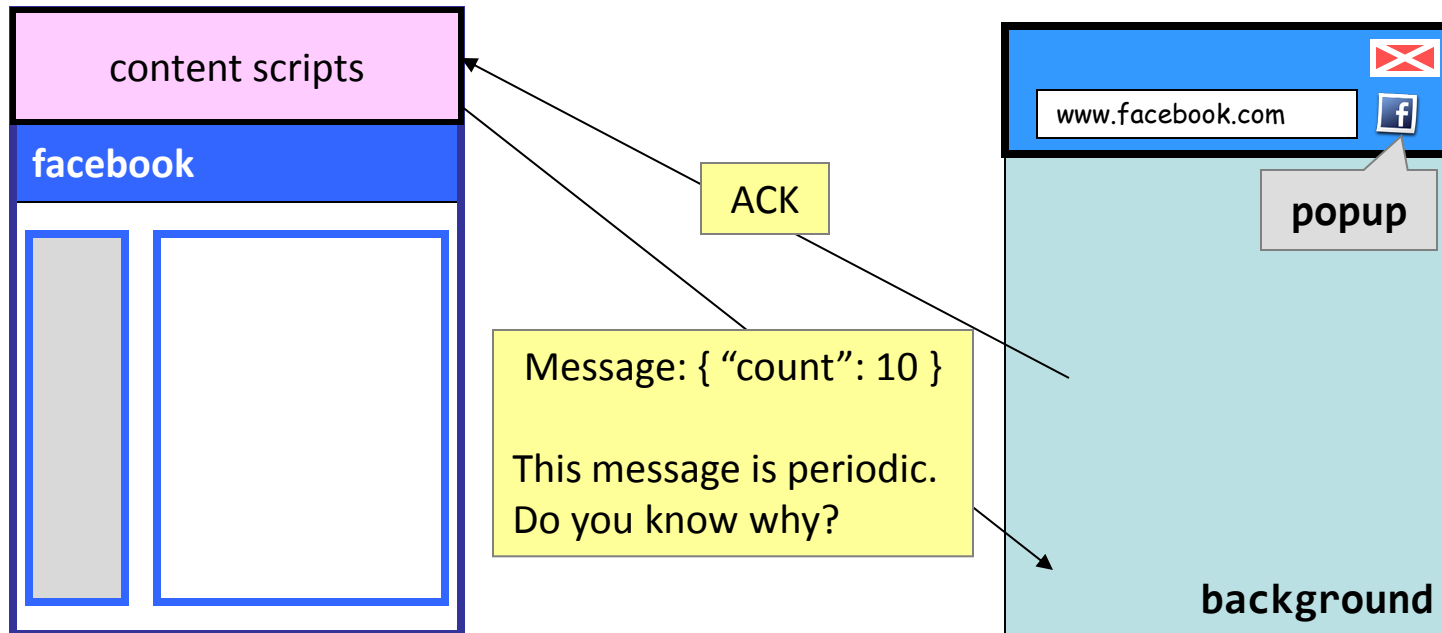
- It is called content scripts.
 - http://code.google.com/chrome/extensions/content_scripts.html
- Our purpose:



See "content.js" in "detect_people/"

What is message passing?

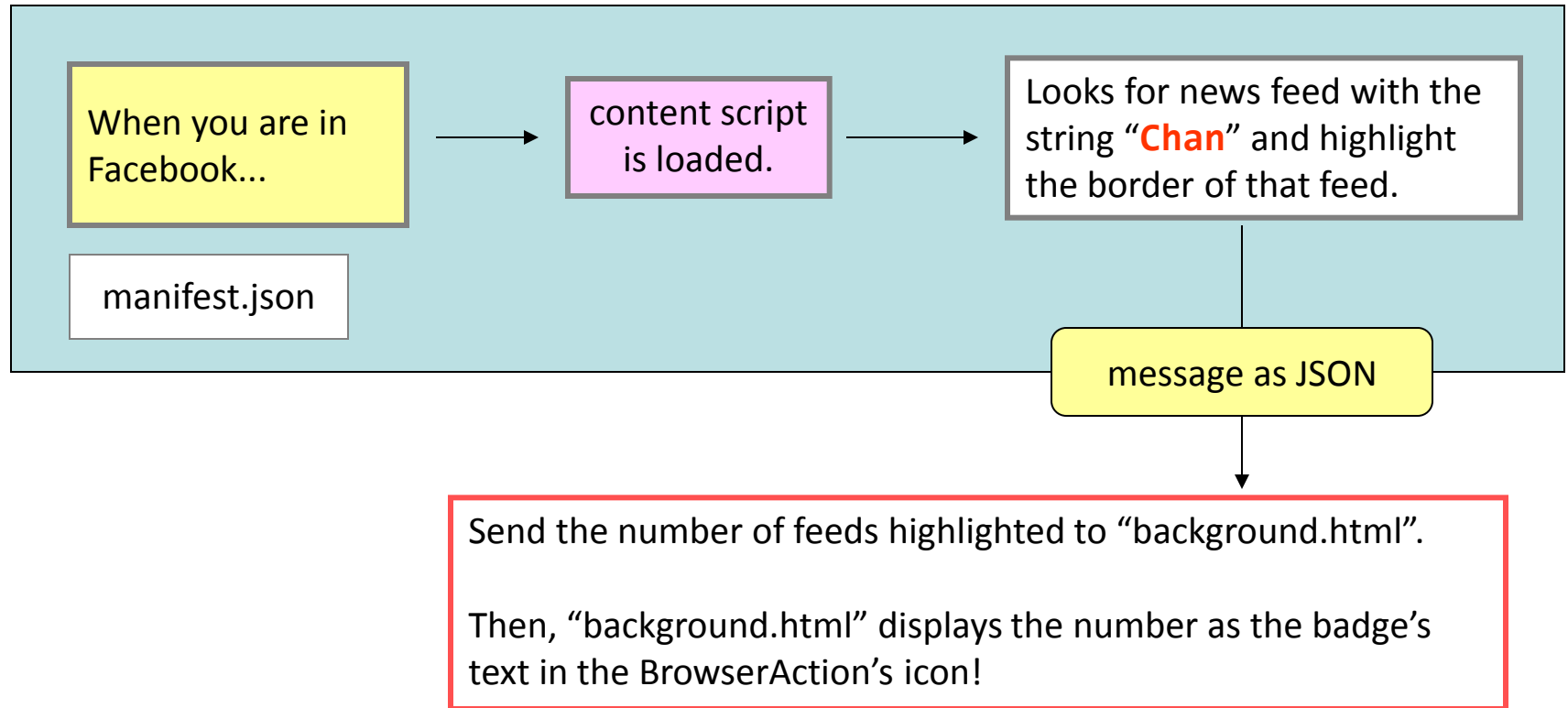
- An extension cannot directly access a page.
 - Messages are sent through the content script.



Reference: <http://code.google.com/chrome/extensions/messaging.html>

Adding things together...

- The flow:



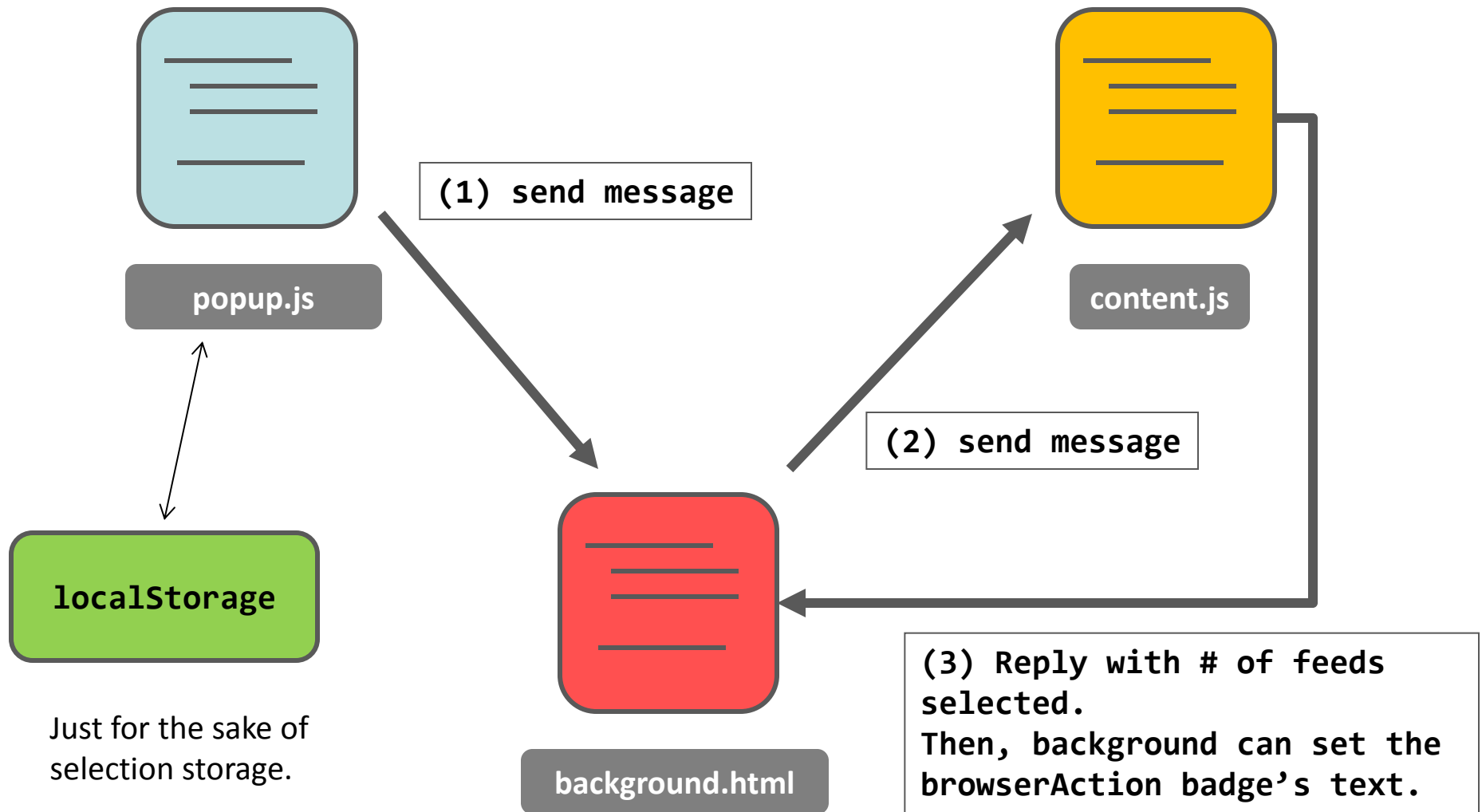
See "content.js" & "background.html" in "detect_people_with_count/"

A finishing touch...

- Can we search patterns other than “**Chan**”?
 - (1) I mean we can change the pattern on demand...
 - (2) More nicely, can we change save the pattern, too?
- Solution:
 - (1) sending reverse messages from “popup.html” to “content scripts”.
 - (2) With the help of **HTML5 localStorage**.

See “facebook_selector/”

Facebook Selector



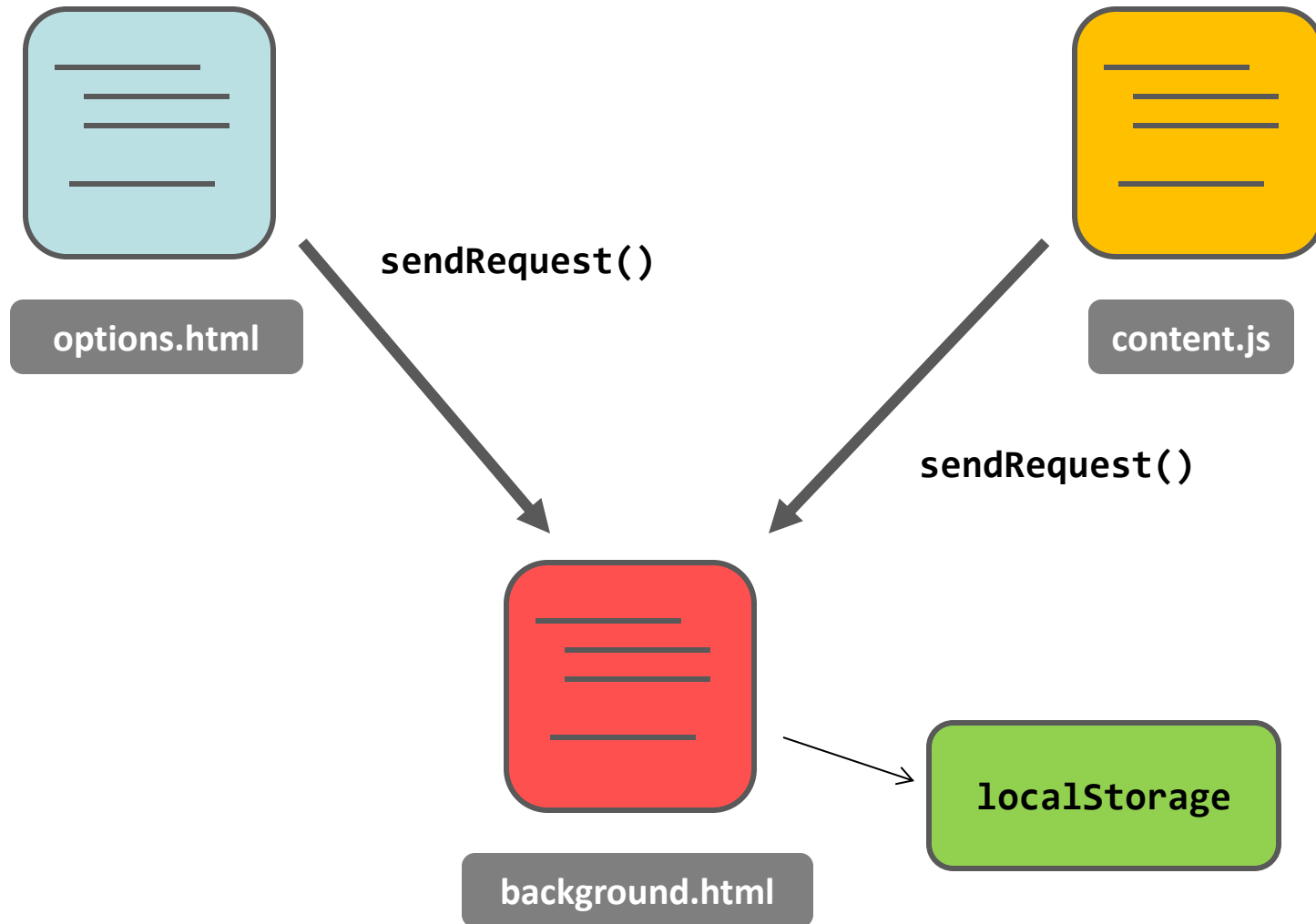
See “facebook_selector.zip”

Chrome Extension

Bonus: ERGWAVE Automatic Login Extension

**Note: my code does not work since Chrome 18.
Let me show you my design.**

ERGWAVE Automatic Login Extension



ERGWAVE Automatic Login Extension

1. Injecting JavaScript that fills in the credential automatically. Then, trigger the **submit()** of the form. Retrieving credential through messaging.

2. When there is no credential stored in the localStorage, read user input by registering **onsubmit**.

content.js



content.js

sendRequest()

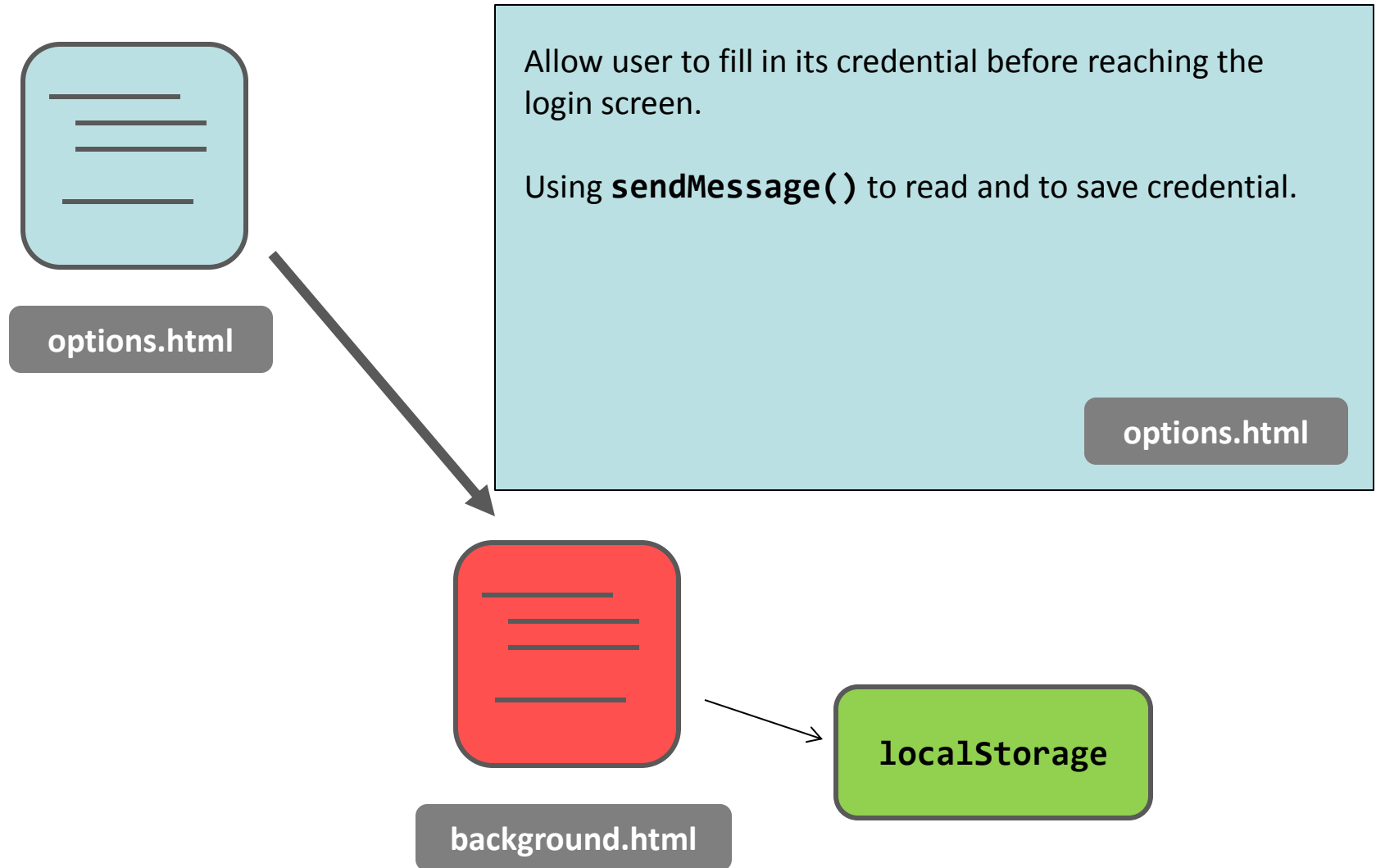


background.html



localStorage

ERGWAVE Automatic Login Extension



ERGWAVE Automatic Login Extension

Centralized agent to receive all kinds of requests.

The only one that access the **localStorage**.

background.html

