

CSCI4140 - Tutorial 11  
Assignment 3 Overview  
Simplified iReserve Bot

Calvin, Kam Ho Chuen (hckam@cse)

2 Apr 2015



# Outline

- Demonstration of Assignment 3 Part I
- Chrome Storage
- OCR
- CheckList



# Demonstration

- Preliminary version only, more details to come!  
Stay tuned!



# "iReserve" Emulation Page

## Reserve and Pick Up

Email

Password

Captcha

e p G<sup>4</sup> 3 w

Continue





# "iReserve" Emulation Page

## Reserve and Pick Up

Email	<input type="text" value="Email"/>
Password	<input type="password" value="Password"/>
Captcha	<div>e p G<sup>4</sup> 3 w</div> <div></div>
	<div>Continue</div>

Email field and Password field





# "iReserve" Emulation Page

## Reserve and Pick Up

Email

Password

Captcha

e p G<sup>4</sup> 3 w

Continue

Captcha: Generated on-the-fly









# "iReserve" Emulation Page

- It contains 3 text fields: Email, Password, Captcha input.
- Your extension should be able to fill in them with stored data.
- No need to be implemented by yourself :P  
We will provide the code, link will be given later.



# Chrome Extension



## Login Details

**Email**

email

hckam@cse.cuhk.edu.hk

**Password**

password

.....

**Captcha**

captcha

captchaText

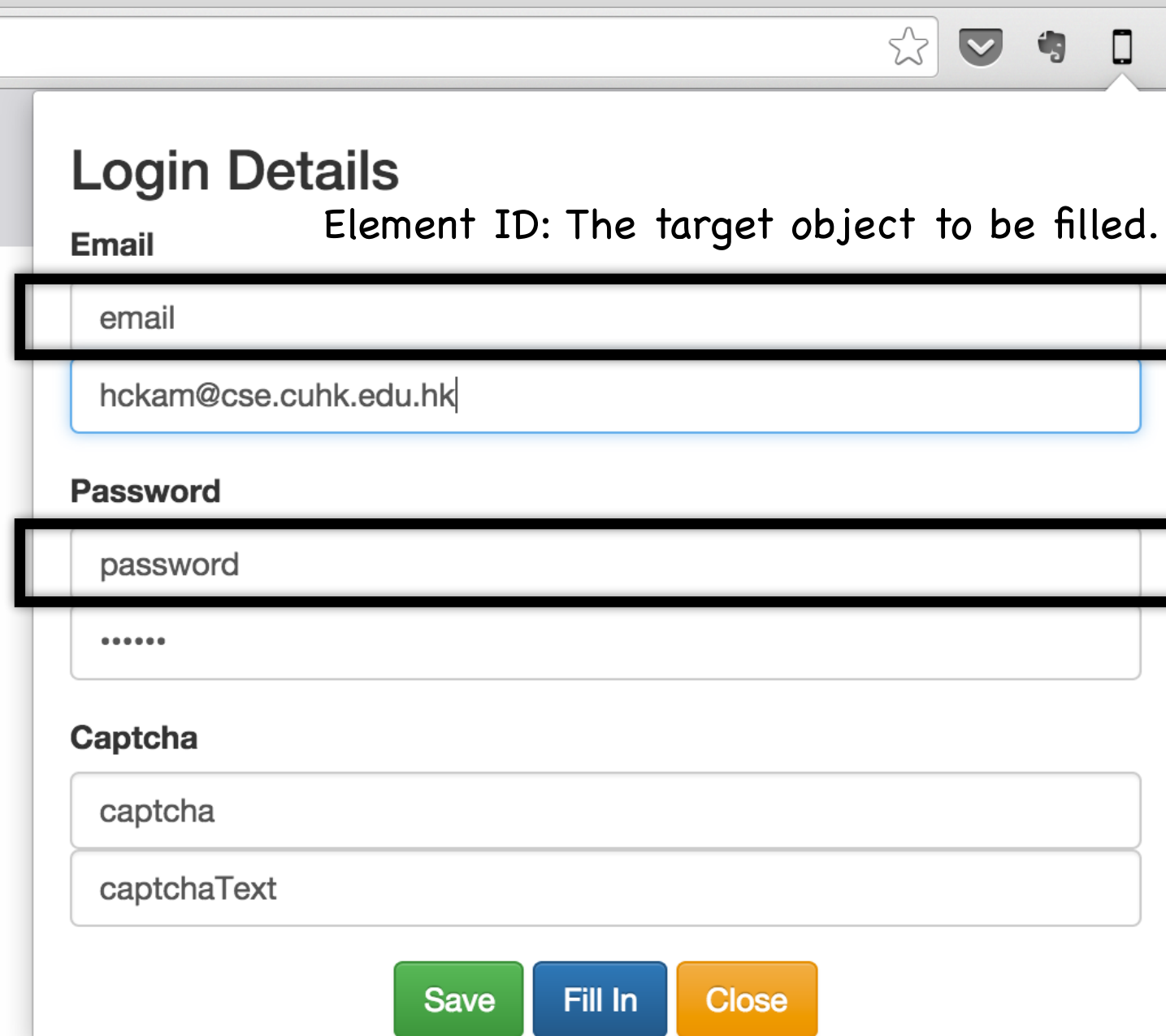
Save

Fill In

Close



# Chrome Extension



**Login Details**

Email Element ID: The target object to be filled.

email

hckam@cse.cuhk.edu.hk

**Password**

password

.....

**Captcha**

captcha

captchaText

Save Fill In Close



# Chrome Extension

☆

☑

🔍

📱

## Login Details

**Email**      Element ID: The target object to be filled.

email

hckam@cse.cuhk.edu.hk

**Password**

password

.....

**Captcha**

captcha	Captcha Picture ID
captchaText	Captcha Input TextField ID

Save

Fill In

Close



# Chrome Extension

**Login Details**

Element ID: The target object to be filled.

**Email**

email

hckam@cse.cuhk.edu.hk

**Password**

password

... Save & Fill in the form immediately

**captcha**

captchaText

Save Fill In Close

Saving to LocalStorage

Close the popup



# Chrome Extension

- The extension should have a popup, allowing user to input the pre-filled information.
- There are two textfields for email and password, namely #ID and value.
- For captcha, two textfields are also required: one is for captcha picture #ID, another is for Captcha input #ID.



# Program Flow

- Load the Extension
- An icon appears next to the address bar, a popup page appears when it is clicked.
- When the iReserve page is loaded (reloaded), the content script will be injected automatically, i.e. form filling is done when the page finishes loading.
- Form filling can also be done without reloading by clicking the "Fill" button in the popup page.
- User details can be saved to local storage for later retrieval.



# Saving Data Locally in Chrome

- Chrome provides a handy tool to store user data, namely `storage.sync` and `chrome.local`.
- `storage.sync` will allow Chrome to sync across each Chrome browser with user logged in.
- `storage.local` will store the data in local machine only. (In this case we will use it).



# storage.local

- Remember to set "Storage" permission!

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



# storage.local store values

```
chrome.storage.local.set({'key': "value", 'key2': "value2"},  
function(e){});
```

- It stores the data in a key-value pair manner.
- callback on success.



# storage.local get values

```
chrome.storage.local.get(null,function(e){  
  console.log(e["key"]);  
});
```

- The first parameter is to define which keys to retrieve (in String or array of string). If it is null, then all keys are retrieved.
- If on success, the value will be stored in parameter of callback function (e).



# Optical Character Recognition (OCR)

- To bypass the captcha, OCR is needed to recognise the characters 😈.
- In our chrome extension case, "OCRAD.js" is recommended.

## Ocrad.js Optical Character Recognition in JS

Ocrad.js is a **pure-javascript** version of the **Ocrad** project, automatically converted using **Emscripten**. It is a simple **OCR (Optical Character Recognition)** program that can convert scanned images of text back into text. Clocking in at **about a megabyte** of Javascript with no hefty training data dependencies (looking at you, **Tesseract**), it's on the lighter end of the spectrum.

This was made by [antimatter15](#) (please follow me on [Twitter](#) or [G+](#))



# Ocrad.js

- Include it in content script section at manifest  
`"js": ["ocrad.js", "action.js"],`
- Easy to use. Require only one sentence of code!!!  
(YEAH) `var string = OCRAD(image);`
- However, it only accepts a canvas element and a Context2D instance. That means it **does not accept img object!**
- We need to preprocess the captcha image..😞



# Preprocess the image for OCRAD.js

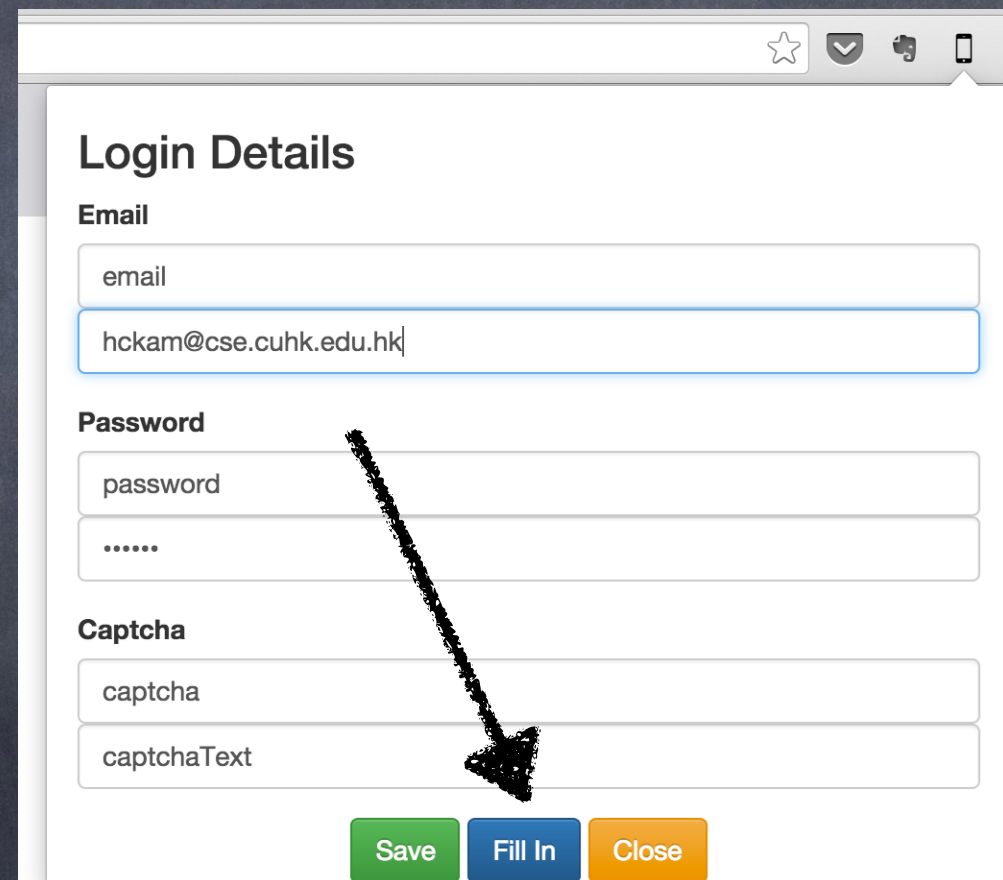
```
var image = new Image();  
image.src = document.getElementById("image").src;  
// Initialize a canvas  
var canvas = document.createElement('canvas');  
canvas.height = image.height;  
canvas.width = image.width;  
var imgDraw = canvas.getContext('2d');  
imgDraw.drawImage(image, 0, 0);  
var string = OCRAD(imgDraw);
```

- First a image object is created and make the source pointing the <img> object.
- Then the image will be drawn on canvas and it can be passed to OCRAD.js library!
- If the environment is hell-like (Open\_\_\_\_\_), how can we ensure the script runs after the image completely loaded? Use `unload` function of image.



# Message Passing from popup to content script

- If you want to send content script messages from the popup page like this:
- You need another function to do this:



The screenshot shows a web form titled "Login Details" within a browser window. The form has three main sections: "Email", "Password", and "Captcha". The "Email" section has a text input field with the placeholder "email" and the value "hckam@cse.cuhk.edu.hk". The "Password" section has two text input fields, both with the placeholder "password"; the second field contains six dots. The "Captcha" section has two text input fields, both with the placeholder "captcha"; the second field contains the text "captchaText". At the bottom of the form are three buttons: "Save" (green), "Fill In" (blue), and "Close" (orange). A thick black arrow points from the second password field to the "captchaText" field.



# Message Passing from popup to content script

```
chrome.tabs.query({active: true, currentWindow: true},  
function(tabs) {chrome.tabs.sendMessage(tabs[0].id,  
{key: "value"},function(response){});  
});
```

- This will find the current active tab and then get the id. This id is necessary to specify the recipient of the action : )
- The content script uses the normal listener to handle the message sending.
- Last Reminder: need "tabs" permission!



# CheckList

- How do I define a Chrome Extension? [Tut 1].
- How do I save data locally? [Tut 2].
- How do I access the DOM Object? [Tut 1].
- How do I manipulate the webpage object? [Tut 1].
- How do I recognize the character? [Tut 2].



# Reference

- <https://developer.chrome.com/extensions>
- <http://antimatter15.com/ocrad.js/demo.html>



# Thank You!

- Next Tutorial: Assignment 3 Part II.
- See You : )