

# **BROWSER BASED EXTENSION**

**FINAL REPORT**

**OF MAJOR PROJECT**

**BROWSER BASED EXTENSION TO INTERACT WITH EVERNOTE**

**BACHELOR OF TECHNOLOGY**

**INFORMATION TECHNOLOGY**

**SUBMITTED BY:**

**CHANPREET SINGH (1311369)**

**ARVINDER KAUR (1311359)**



**GURU NANAK DEV ENGINEERING COLLEGE**

**LUDHIANA**

## ABSTRACT

A browser extension is a plug-in that extends the functionality of a web browser. Some extensions are authored using web technologies such as HTML, JavaScript, and CSS. Others are developed using machine code and application programming interfaces (APIs) provided by web browsers, such as NPAPI and PPAPI. Browser extensions can change the user interface of the web browser without directly affecting viewable content of a web page; for example, by adding a browser toolbar. By Using this Evernote extension anyone can save things you see on the web into your Evernote account. Clip the web pages you want to keep. Save them in Evernote. Easily find them on any device. This can be Great for research one can clip any article or web page and to a specific notebook and assign tags. Use Evernote to find clips on any device highlight and share key text from any website or article .Use text and visual callouts to draw attention. Share and email clips or create a URL link. User can also Clip the Gmail threads and attachments or Clip as page.

## ACKNOWLEDGEMENT

We are highly grateful to the Dr. M.S. Saini , Director, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out the major project work on topic Browser Based Extension

The constant guidance and encouragement received from Dr. K.S. Mann H.O.D. Department of IT, GNDEC Ludhiana has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We would like to express a deep sense of gratitude and thanks profusely to Mr inderjeet singh bamrah , without his wise counsel and able guidance, it would have been impossible to complete the project in this manner.

We express gratitude to other faculty members of Information Technology department of GNDEC for their intellectual support throughout the course of this work.

## **List of figures**

<b>Figure</b>	<b>Topic</b>	<b>Page No.</b>
1.1	Launch Web Clipper to start clipping	9
1.2	Share with others	10
1.3	Start clipping	10
1.4	Select a clip type	11
1.5	Browser	12
1.6	Simplified article	13
1.7	Full page	14
1.8	Bookmark	15
1.9	Custom clips for popular sites	16
1.10	Organize clips	17
1.11	Annotate key info	18

2.1	enabling extension	19
2.2	Spiral model	21
3.1	showing flowchart	22
5.1	User Interface Representation	34
5.2	Evernote Clipper	35
6.1	Shape tool	42
6.2	Add visual callouts	43
6.3	Highlight text	44
6.4	Other annotation tools:	44
6.5	Share in Google Chrome, Safari, and Opera	45
6.6	Custom clips for popular sites	45
6.7	Share in Internet Explorer and Firefox	46
6.8	Add notification alerts to web clips	46
6.9	Finds notes created with Web Clipper	47
6.10	View related notes in search engine results	48

## **Table of Contents**

<b>S. No.</b>	<b>Topic</b>	<b>Page No.</b>
1.	Introduction to Project	3
2.	Requirement Analysis and System Specification	22
3.	System Design	24
4.	Implementation, Testing, and Maintenance	32
5.	Results and Discussions	38
6.	Conclusion and Future Scope	43
7.	References	51

# CHAPTER 1. INTRODUCTION TO PROJECT

## 1.1 Browser Based Extension

Extensions allow user to add functionality to Chrome without diving deeply into native code. User can create new extensions for Chrome with those core technologies that you're already familiar with from web development: HTML, CSS, and JavaScript. If you've ever built a web page, you should feel right at home with extensions pretty quickly; we'll put that to the test right now by walking through the construction of a simple extension that will fetch an image from Google using the current page's URL as a search term. We'll do so by implementing a UI element we call a browser action, which allows us to place a clickable icon right next to Chrome's Omnibox for easy access. Clicking that icon will open a popup window filled with an image derived from the current page.

## 1.2 Objectives

1. To build browser based extension.
2. To integrate extension with Evernote using API.
3. To carry the selected data and display on Evernote.
4. To generate a basic skeleton of chrome API.
5. To interact Evernote API with the basic skeleton formed.
6. To add JavaScript code for capturing the content and transferring to Evernote.

## **1.3 Problem Formulation**

- To carry the text selection directly without the URL linkage
- To avoid the copying process of the entire clipping from the scratch
- To respond quickly on the selection of data and clipping to Evernote simultaneously

## **1.4 Recognition of a Need**

### **1.4.1 The save button for the web**

When you find something good, clip it. Every research article, travel confirmation, and bit of inspiration is collected in Evernote, forever. Organize and find it on any device.

### **1.4.2 Draw attention, send it on**

Clip something worthy of sharing then Send it right from Web Clipper. Before you do, highlight important sections or add text and visual callouts so others can see exactly what you're referring to.

### **1.4.3 Your favorite sites, perfectly captured**

Web Clipper works great everywhere. But it's brilliant when clipping from Gmail, LinkedIn, YouTube, and Amazon. Select what you want, then save it as a clean and clutter-free note.

## **1.5 Proposed System**

### **1.5.1 Web clipping**

Extracting static information from a Web site in order to display the data on a Web-enabled PDA. The idea behind Web clipping is to conserve the PDA's resources by extracting once any static data, such as graphics, logos, photos or even unnecessary text and storing that data on the PDA. The PDA will then make a wireless connection to a Web server in order to retrieve any dynamic content. Web clipping is a technology pioneered by Palm for its Palm VII handheld device. An excerpt of data taken from a Web server.



## **1.6 Unique Features of the System**

### **1.6.1 Quick actions**

A robust and highly available service to store and remember everything.

### **1.6.2 Make a note of it**

Create a project to-do list. Jot down a reminder. Or snap a picture of a sketch. A note can be anything you want it to be. And once you make a note, it's accessible wherever you go, forever.

### **1.6.3 Have it everywhere**

Capture a note once, and it's instantly available on all your devices. Never worry about where you saved something because it's in Evernote, and Evernote is wherever you are.

### **1.6.4 Find anything fast**

Whether it's text, images or documents, you keep things for a reason. That's why Evernote makes sure the notes you've saved are easy to find. You can even search for handwritten words buried deep within your notes.

### **1.6.5 Share with anyone**

Share what matters with the people who matter to you. Capture life's little moments and share them from wherever you happen to be. Or share big ideas by collaborating seamlessly.

## **1.7 Overview of Evernote**

### **1.7.1 The Evernote Community**

Help your users remember everything. User information comes in all shapes and sizes. As an Evernote partner, you have the opportunity to give your users a unique view of their own memories, regardless of format. Because you're building with Evernote, you're building on a rock-solid platform designed to store all types of user data from images and audio to web pages and PDFs — and it's all available through our API.

## 1.7.2 The App Center

### Increased Exposure

The first place our users look for cool apps, services and products that work with Evernote is the App Center, our hand-curated collection of third-party integrations. For our partners, this means increased exposure and more users. If you build something great, be sure that we'll make sure our users see it.

## 1.7.3 Evernote Developer Documentation

Everything you need to know when working with the Evernote API.

The documentation is organized into three major areas:

- Our Quick-start guides will show you how to install and configure the Evernote SDK for your chosen language or platform.
- The topical Articles describe individual concepts or functions used when interacting with the Evernote API. You'll probably spend the majority of your time reading these, as they make up the majority of our documentation. They are organized into sections by general topic.
- Finally, the API Reference contains a comprehensive listing of all types, functions and enumerations exposed by the API.

**Note:** You'll need to create an account on our development server.

Evernote Web Clipper is a simple extension for your web browser that lets you capture full-page articles, images, selected text, important emails, and any web page that inspires you.

Save everything to Evernote and keep it forever.

Available for Chrome, Safari, Internet Explorer (IE) 7+, Firefox, Opera, and Microsoft Edge (for Windows 10 or higher)

## 1.7.4 Save your first web clip in 3 steps:

### 1. Launch Web Clipper to start clipping

Click the elephant button in your browser toolbar to launch Web Clipper.

Select a clip type, full-length or sections of web pages, such as news stories or research articles.

Save time and review clipped content offline, during your commute.

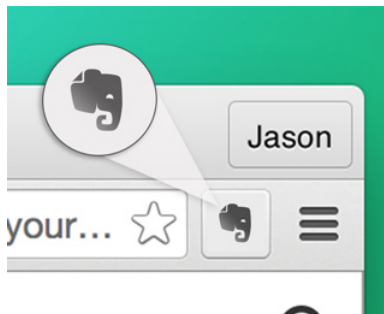


figure 1.1

## 2. Annotate key info

Currently unavailable for Microsoft Edge

Capture screenshots of web pages, then select an 'Annotate' tool to highlight and add visual callouts, such as arrows, to point out design elements you like.

## 3. Share with others

Click **Share** to share and discuss ideas with others. Clip design ideas to share with your team or post event details on a team page.

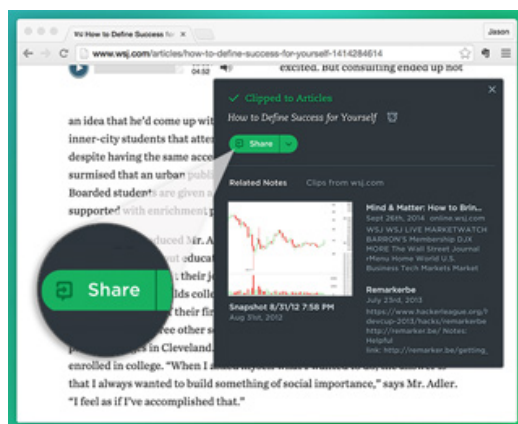


figure 1.2

## 4 . Start clipping

To start clipping, launch the Web Clipper by clicking the elephant from your browser toolbar. In Internet Explorer, you may need to add the 'Evernote' button to the toolbar, and click the greater-than signs ('>>') to expand the toolbar to display the button.

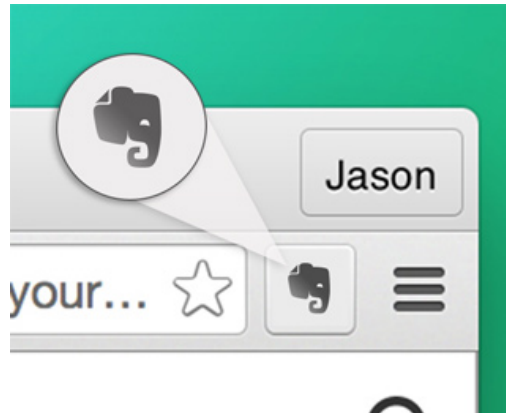


figure 1.3

## 5. Select a clip type

Select a clip type from the available options, depending on your content and layout preferences. With particular sites, such as LinkedIn, you can select only the sections of the page you'd like to save.

Note: Support for web clip types vary across browsers

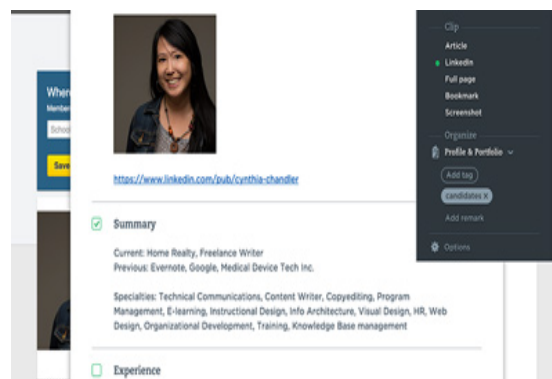


figure 1.4

## 1.8 Choose from the following options:

### 1.8.1 Article

Available for Google Chrome, Safari, Internet Explorer (IE) 7+, Microsoft Edge, and Opera. Automatically detect and clip the main section of a blog, news article, or webpage. Ignore everything else on the page. To modify the area of the webpage captured, press the + or - buttons on the screen, or use the arrow keys on your keyboard.

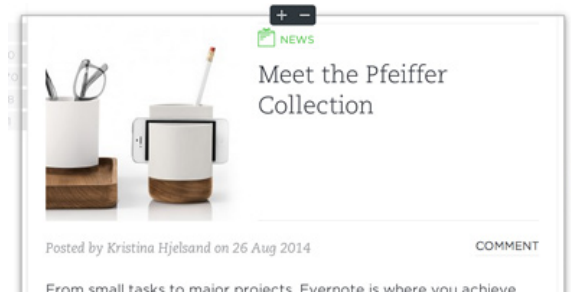


figure 1.5

### 1.8.2 Simplified article

Available for Google Chrome, Safari, Internet Explorer (IE) 7+, Microsoft Edge, and Opera. Enjoy a better reading experience on your favorite blogs and news sources. Clear away the formatting and layouts so you can focus on the content, with less distraction.

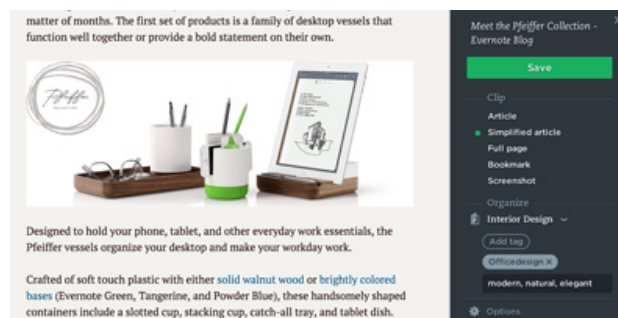


figure 1.6

## 1.8.3 Full page

Available for Google Chrome, Safari, Internet Explorer (IE) 7+, Microsoft Edge, and OperaSave a static copy of an entire page if you want to refer back to the original format and layout of the page, with all its design and navigational elements.

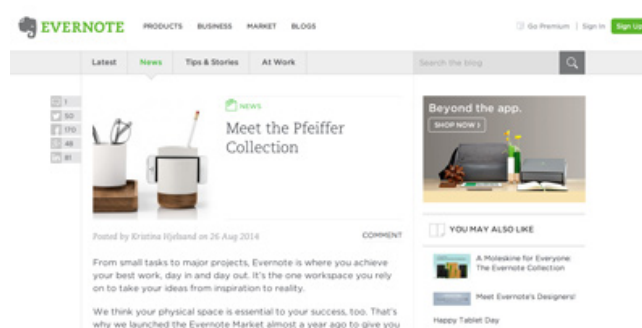


figure 1.7

## 1.8.4 Bookmark

Want to remember a site without clipping the entire page? Save the URL, an auto-generated thumbnail from the page, and a snippet of text. Add comments to the clip and recall what was inspiring or informative.

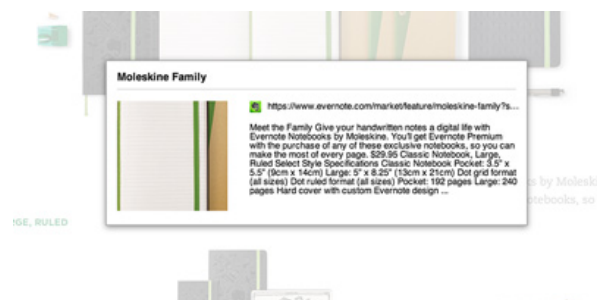


figure 1.8

### 1.8.5 Screenshot

Take a static snapshot of a page as it appears on the screen. Once you've captured the screenshot, you can crop the image and add text, shapes, stamps and other visual callouts.



figure 1.9

### 1.8.6 Selection

Select some text or pictures in a web page before opening the Web Clipper. Only the text or images you've selected are saved to Evernote.

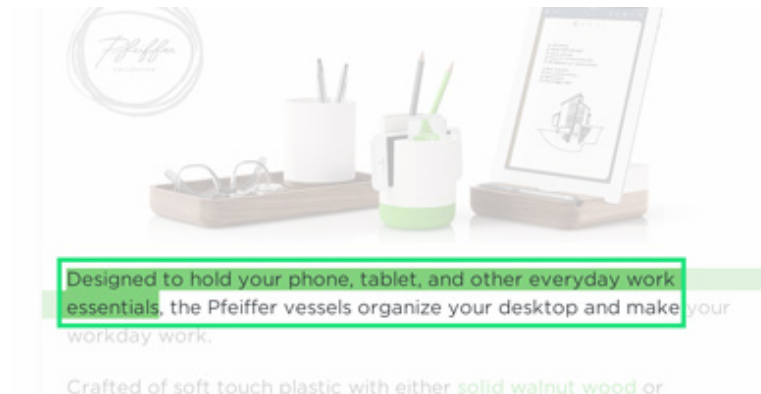


figure 1.10

## 1.9 Custom clips for popular sites

Available for Google Chrome, Safari, Internet Explorer (IE) 7+, Microsoft Edge, and OperaSave specially formatted clips of popular sites, such as Gmail, Amazon, LinkedIn, and YouTube to Evernote. Click checkboxes to select only the sections of the page you want to keep. Review a clean version of your clips as an editable note you can add comments to later.

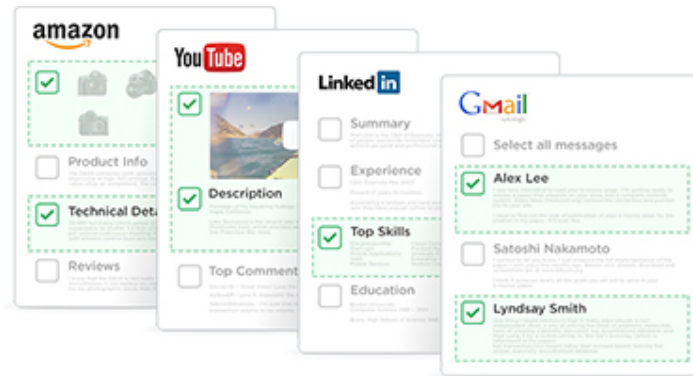


figure 1.11

### 1.9.1 Organize clips

Once you click **Save**, Web Clipper smart filing automatically stores both the contents and the URL of a web page into Evernote, as a note in your default notebook.

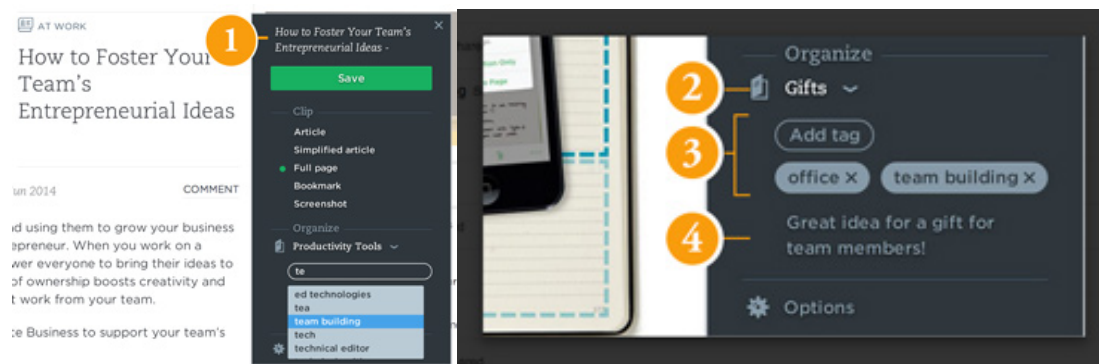


figure 1.12



If you'd prefer not to save to your default notebook, you can specify how you'd like to organize your clips before saving them to Evernote:

- **Title:** Rename the default name given to the web clip.
- **Notebook:** Select a notebook where you want to save your web clip. Evernote attempts to predict the most likely notebook based on how you've organized notes in the past.
- **Tags:** Assign one or more keyword (tags) to your web clip. To remove a tag, click the 'X' next to the keyword/phrase.
- **Remark:** Add additional comments or notes that will help you remember what you've clipped.

Once saved to Evernote anything you've clipped is fully searchable, including text in images. Enter any terms, phrases, or tags from your clips to find your clipped notes.

### 1.9.2 Annotate key info

Two ways to add visual callouts to a page:

1. **Highlight text:** Highlight text on a web page, then clip and save or share it.
2. **Add visual callouts:** Capture a screenshot of a page, then use the annotation tools to mark it up before you save and share it.



figure 1.13

# CHAPTER 2. REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION

## 2.1 Feasibility study

- Economical – API are free to get and integrate with full access and no restricted features.
- Technical – because of every denotation for programming is available it is independent to work on any hardware and software platform.
- Behavioural – easy to use as user will be acquainted easily.

An extension is a zipped bundle of files—HTML, CSS, JavaScript, images, and anything else you need—that adds functionality to the Google Chrome browser. Extensions are essentially web pages, and they can use all the APIs that the browser provides to web pages, from XMLHttpRequest to JSON to HTML5.

Extensions can interact with web pages or servers using content scripts or cross-origin XMLHttpRequests. Extensions can also interact programmatically with browser features such as bookmarks and tabs.

## 2.2 Extension UIs

Many extensions—but not Chrome Apps—add UI to Google Chrome in the form of browser actions or page actions. Each extension can have at most one browser action or page action. Choose a browser action when the extension is relevant to most pages. Choose a page action when the extension's icon should be active or inactive (and grayed out), depending on the page.

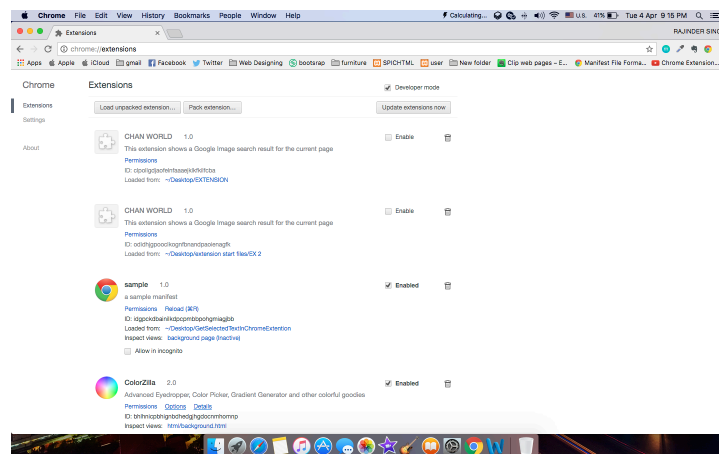


figure 2.1 enabling extension

This Google Mail Checker extension uses a browser action. This Mappy extension uses a page action and content script (code injected into a web page). This Set Page Color extension features a browser action that, when clicked, shows a popup.

Extensions (and Chrome Apps) can also present a UI in other ways, such as adding to the Chrome context menu, providing an options page, or using a content script that changes how pages look. See the Developer's Guide for a complete list of extension features, with links to implementation details for each one.

## **2.3 Software Requirements Specification Document**

- Data Requirement – API through data get for user's respective account and requires login request.
- Functional Requirement – Requires JavaScript and some API's of Evernote.
- Performance Requirement – Normal configuration of hardware with updated chrome browser and updated JavaScript
- Dependency Requirement – Google Chrome API , Evernote API , Google Chrome .
- Security Requirement – As access is through API , no issues will be present because of no loop wholes in the interface through Chrome Browser.
- Look and Feel Requirement – Icons and Graphics can be modified or animated through the use of css files.

## **2.4 Expected Hurdles**

- To check the adaptability of current activity of existing API with JavaScript based coding of the basic Chrome Extension skeleton .
- To make the cppy of the selected text only and not the URL.

## 2.5 SDLC model used

- Spiral Model

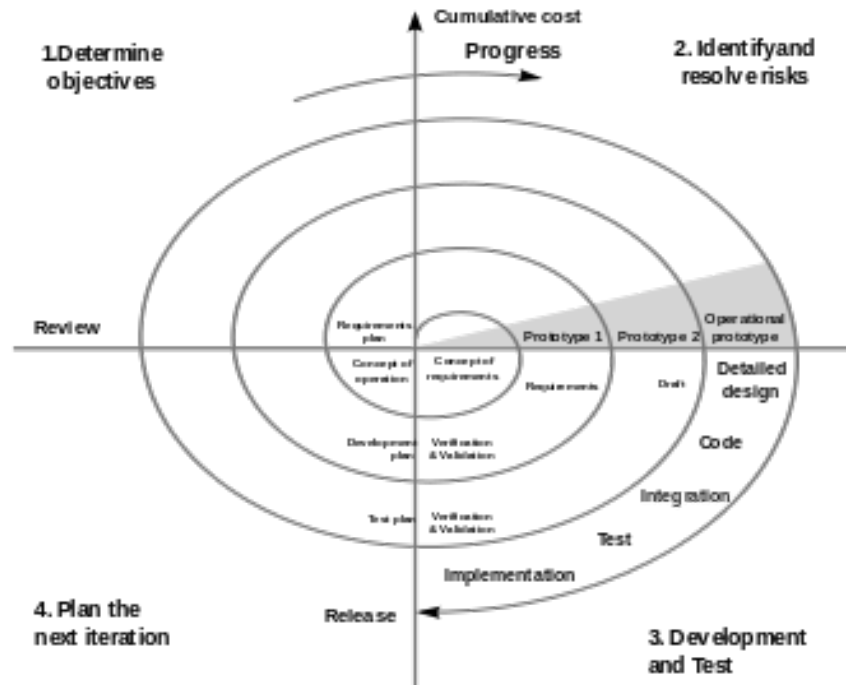


figure 2.2 spiral model

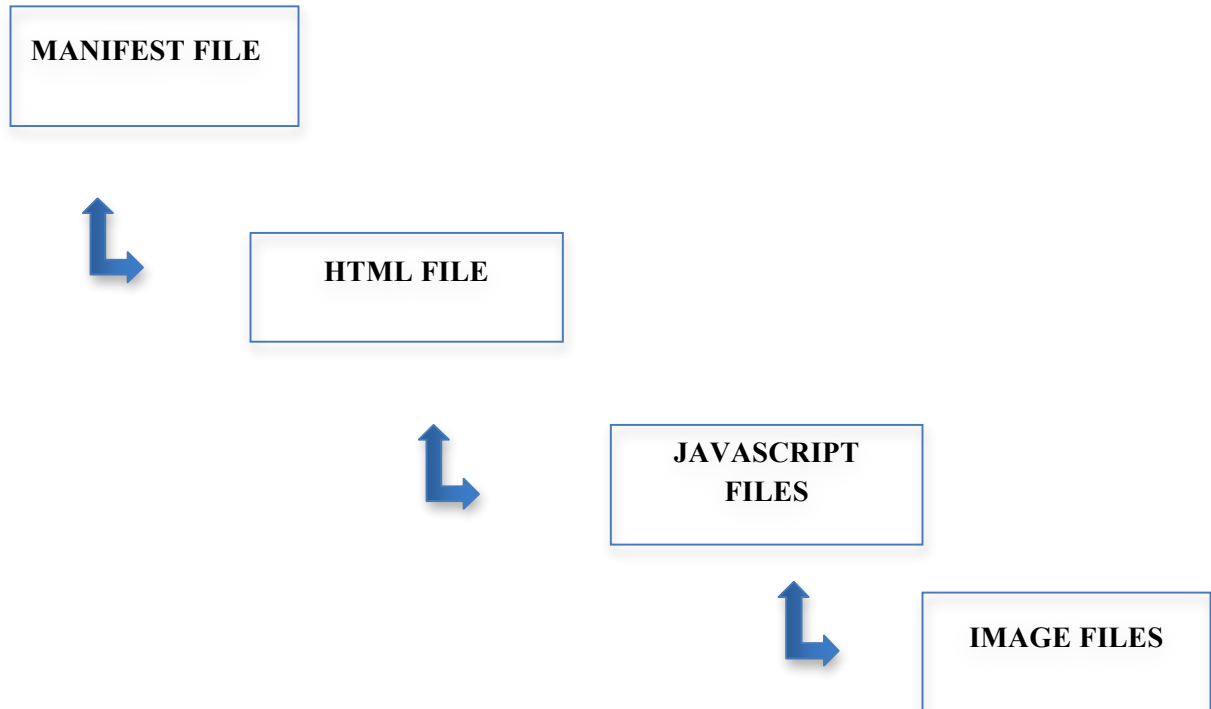
The spiral model is a risk-driven process model generator for software projects. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping.

## CHAPTER 3. SYSTEM DESIGN

### 3.1 Detail Design

Each extension has the following files:

1. A manifest file
2. One or more HTML files (unless the extension is a theme)
3. Optional: One or more JavaScript files
4. Optional: Any other files your extension needs—for example, image files.



(Figure 3.1 showing flowchart)

While you're working on your extension, you put all these files into a single folder. When you distribute your extension, the contents of the folder are packaged into a special ZIP file that has a .crx suffix. If you upload your extension using the Chrome Developer Dashboard, the .crx file is created for you. For details on distributing extensions, see [Hosting](#).

## **3.2 The manifest file**

The manifest file, called manifest, Json gives information about the extension, such as the most important files and the capabilities that the extension might use. Here's a typical manifest file for a browser action that uses information from google.com:

## **3.3 JavaScript**

"JS" for short is a full-fledged dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on websites JavaScript is incredibly versatile. You can start small, with carousels, image galleries, fluctuating layouts, and responses to button clicks. With more experience you'll be able to create games, animated 2D & 3D graphics, comprehensive database-driven apps, and much more JavaScript itself is fairly compact yet very flexible. Developers have written a large variety of tools complementing the core JavaScript language, unlocking a vast amount of extra functionality with minimum effort.

## **3.4 Front End**

### **3.4.1 Architecture of files included**

Application Programming Interfaces (APIs) built into web browsers, providing functionality like dynamically creating HTML and setting CSS styles, collecting and manipulating a video stream from the user's webcam, or generating 3D graphics and audio samples.

Third-party APIs to allow developers to incorporate functionality in their sites from other content providers, such as Twitter or Facebook.

Third-party frameworks and libraries you can apply to your HTML to allow you to rapidly build up sites and applications.

### **3.4.2 Architecture**

Many extensions have a background page, an invisible page that holds the main logic of the extension. An extension can also contain other pages that present the extension's UI. If an extension needs to interact with web pages that the user loads (as opposed to pages that are included in the extension), then the extension must use a content script.

## 3.5 Methodology

### 3.5.1 DOM and JavaScriptEDIT

The short example above, like nearly all of the examples in this reference, is JavaScript. That is to say, it's written in JavaScript, but it uses the DOM to access the document and its elements. The DOM is not a programming language, but without it, the JavaScript language wouldn't have any model or notion of web pages, HTML documents, XML documents, and their component parts (e.g. elements). Every element in a document—the document as a whole, the head, tables within the document, table headers, text within the table cells—is part of the document object model for that document, so they can all be accessed and manipulated using the DOM and a scripting language like JavaScript.

In the beginning, JavaScript and the DOM were tightly intertwined, but eventually they evolved into separate entities. The page content is stored in the DOM and may be accessed and manipulated via JavaScript, so that we may write this approximative equation:

$$\text{API (HTML or XML page)} = \text{DOM} + \text{JS (scripting language)}$$

The DOM was designed to be independent of any particular programming language, making the structural representation of the document available from a single, consistent API. Though we focus exclusively on JavaScript in this reference documentation, implementations of the DOM can be built for any language, as this Python example demonstrates:

### 3.5.2 Overview of DOM

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It provides a structured representation of the document and it defines a way that the structure can be accessed from programs so that they can change the document structure, style and content. The DOM provides a representation of the document as a structured group of nodes and objects that have properties and methods. Essentially, it connects web pages to scripts or programming languages.

A Web page is a document. This document can be either displayed in the browser window, or as the HTML source. But it is the same document in both cases. The Document Object Model (DOM) provides another way to represent, store and manipulate that same document. The DOM is a fully object-oriented representation of the web page, and it can be modified with a scripting language such as JavaScript.

The W3C DOM and WHATWG DOM standards form the basis of the DOM implemented in most modern browsers. Many browsers offer extensions beyond the standard, so care must be

exercised when using them on the web where documents may be accessed by various browsers with different DOMs.

All of the properties, methods, and events available for manipulating and creating web pages are organized into objects (e.g., the document object that represents the document itself, the table object that implements the special `HTMLTableElement` DOM interface for accessing HTML tables, and so forth). This documentation provides an object-by-object reference to the DOM implemented in Gecko-based browsers.

### 3.5.3 XML Http Request

Use XML Http Request to request data from one or more servers. The permissions field of the manifest specifies which hosts the extension can send requests to.

HTML5 and other emerging APIs

Google Chrome supports HTML5 features, along with other emerging APIs. Here are some of the APIs you can use:

- audio
- application cache
- canvas
- geolocation
- local storage
- notifications
- video
- web database

### 3.5.4 WebKit APIs

Because Google Chrome is built upon WebKit, your extensions can use WebKit APIs. Especially useful are the experimental CSS features such as filters, animations, and transformations. Here's an example of using WebKit styles to make the UI spin. Because JSON is in V8, you don't need to include a JSON library to use JSON functions.



## **1. APIs in bundled libraries**

If you want to use a library that the browser doesn't provide (for example, jQuery), you can bundle that library's JavaScript files with your extension. Bundled libraries work in extensions just as they do in other web pages.

## **2. Cross-Origin XMLHttpRequest**

Regular web pages can use the XMLHttpRequest object to send and receive data from remote servers, but they're limited by the same origin policy. Extensions aren't so limited. An extension can talk to remote servers outside of its origin, as long as it first requests cross-origin permissions. Each running extension exists within its own separate security origin. Without requesting additional privileges, the extension can use XMLHttpRequest to get resources within its installation.

For example, if an extension contains a JSON configuration file called config.json, in a config\_resources folder, the extension can retrieve the file's contents. If the extension attempts to use a security origin other than itself, the browser disallows it unless the extension has requested the appropriate cross-origin permissions.

## **3. Description**

Use the chrome.permissions API to request declared optional permissions at run time rather than install time, so users understand why the permissions are needed and grant only those that are necessary.

## CHAPTER 4. IMPLEMENTATION, TESTING, AND MAINTENANCE

### 4.1 Tools and Techniques used for Implementation

#### 4.1.0 Hardware Requirements:

Hardware requirements include that hardware which is required for its working. It includes:

1. Pentium 4 Computer
2. 512 MB RAM
3. High Speed Internet Connection(DSL/Cable)

#### 4.1.2 Software Requirements

1. The technical specifications of requirements for the software are as follows:
2. Any Operating System (Windows, Linux, MAC)
3. Java run time environment
4. Netbeans (Java IDE)
5. Java SDK (Software Development Kit)
6. Any web browser(Chrome, Firefox, etc)

### 4.2 Coding Standards of Language used :

#### 4.2.1. Requesting cross-origin permissions :

By adding hosts or host match patterns (or both) to the permissions section of the manifest file, the extension can request access to remote servers outside of its origin.

```
{ "name": "My extension", ... "permissions": [ "http://www.google.com/" ], ... }
```

Cross-origin permission values can be fully qualified host names, like these:

- "http://www.google.com/"

- "http://www.gmail.com/"

Or they can be match patterns, like these:

- "http://\*.google.com/"
- "http://\*/"

A match pattern of "http://\*/" allows HTTP access to all reachable domains. Note that here, match patterns are similar to content script match patterns, but any path information following the host is ignored.

Also note that access is granted both by host and by scheme. If an extension wants both secure and non-secure HTTP access to a given host or set of hosts, it must declare the permissions separately:

```
"permissions": [ "http://www.google.com/", "https://www.google.com/" ]
```

## 4.2.2 Security considerations

When using resources retrieved via XMLHttpRequest, your background page should be careful not to fall victim to cross-site scripting. Specifically, avoid using dangerous APIs such as the below:

```
var xhr = new XMLHttpRequest();

xhr.open("GET", "http://api.example.com/data.json", true);
xhr.onreadystatechange = function() { if (xhr.readyState == 4)
{ // WARNING! Might be evaluating an evil script! var
  resp = eval("(" + xhr.responseText + ")"); ... } }
xhr.send();

var xhr = new XMLHttpRequest(); xhr.open("GET",
"http://api.example.com/data.json", true);
xhr.onreadystatechange = function() { if (xhr.readyState == 4)
{ // WARNING! Might be injecting a malicious script!
  document.getElementById("resp").innerHTML = xhr.responseText;
... } } xhr.send();
```

Instead, prefer safer APIs that do not run scripts:

Additionally, be especially careful of resources retrieved via HTTP. If your extension is used on a hostile network, an network attacker (aka a "man-in-the-middle") could modify the response and, potentially, attack your extension. Instead, prefer HTTPS whenever possible.

### 4.2.3 Interaction with Content Security Policy

If you modify the default Content Security Policy for apps or extensions by adding `content_security_policy` attribute to your manifest, you'll need to ensure that any hosts to which you'd like to connect are allowed. While the default policy doesn't restrict connections to hosts, be careful when explicitly adding either the `connect-src` or `default-src` directives.

### 4.2.4 Permission Warnings

To use most `chrome.*` APIs and extension capabilities, your extension must declare its intent in the manifest, often in the "permissions" field. Some of these declarations result in a warning when a user installs your extension.

When you autoupdate your extension, the user might see another warning if the extension requests new permissions. These new permissions might be new APIs that your extension uses, or they might be new websites that your extension needs access to `chrome.permissions`.

### 4.2.5 Content scripts

If your extension needs to interact with web pages, then it needs a content script. A content script is some JavaScript that executes in the context of a page that's been loaded into the browser. Think of a content script as part of that loaded page, not as part of the extension it was packaged with (its parent extension). Content scripts can read details of the web pages the browser visits, and they can make changes to the pages. In the following figure, the content script can read and modify the DOM for the displayed web page. It cannot, however, modify the DOM of its parent extension's background page.

A browser window with a browser action (controlled by `background.html`) and a content script (controlled by `contentscript.js`).

Content scripts aren't completely cut off from their parent extensions. A content script can exchange messages with its parent extension, as the arrows in the following figure show. For example, a content script might send a message whenever it finds an RSS feed in a browser page. Or a background page might send a message asking a content script to change the appearance of its browser page.

## 4.2 Testing techniques and Test Plans

### 4.3.0 Automated testing tools

#### 4.3.1 Sikuli Testing

[Sikuli](#) is a tool to automate graphical user interfaces ([GUI](#)) using “Visual Image Match” method. In Sikuli, all the web elements should be taken as an image and stored inside the project. Sikuli will trigger GUI interactions based on the image visual match, the image which we have passed as the parameter along with all methods.

Sikuli can be very much useful to automate flash objects (which do not have ID or name). It can be useful in the situation, where we have a stable GUI (i.e. GUI components not changing).

Even Window based applications can also be automated using Sikuli. Sikuli provides very friendly Sikuli-script.jar, which can be easily used together with Selenium WebDriver. We can even automate Adobe Video/Audio player, Flash Games on website using Sikuli. With simple API, it makes coding easier.

#### 4.3.2 Practical Uses

1. Sikuli can be used to automate Flash Objects / Flash Websites.
2. It can be useful to automate Window based application. We can automate, what we are seeing on the screen.
3. It provides, simple API. i.e. all methods can be accessed using screen class object.
4. It can be easily integrated with Selenium and all other tools.
5. Using Sikuli we can automate desktop applications.
6. Most of the automation testing tools will not support flash object automation (E.g. Selenium). Sikuli provides extensive support to automate flash objects.
7. It uses powerful “Visual Match” mechanism to automate desktop & flash objects.

#### 4.3.3 Benefits

- Open source Tool.
- One of the biggest advantage of Sikuli is that, it can easily automate Flash objects.
- It makes easy to automate windows application.

- When you're testing an application under development and you don't know the ID/name of the elements, then you can go with Sikuli. It will check the appearance of the image and if match found, it will interact with the image accordingly.

Prerequisites:

Before getting started , we need to download and install the following software:

- Any screenshot capturing tool (E.g. Duck Capture, or q Snap)
- JDK
- Eclipse

#### **4.3.4 Selenium**

Selenium is a set of different software tools each with a different approach to supporting test automation. Most Selenium QA Engineers focus on the one or two tools that most meet the needs of their project, however learning all the tools will give you many different options for approaching different test automation problems. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One of Selenium's key features is the support for executing one's tests on multiple browser platforms.

#### **4.4 Testing permission warnings**

If you'd like to see exactly which warnings your users will get, package your extension into a .crx file, and install it. To see the warnings users will get when your extension is autoupdated, you can go to a little more trouble and set up an autoupdate server. To do this, first create an update manifest and point to it from your extension, using the "update\_url" key (see Autoupdating). Next, package the extension into a new .crx file, and install the app from this .crx file. Now, change the extension's manifest to contain the new permissions, and repackage the extension. Finally, update the extension (and all other extensions that have outstanding updates) by clicking the chrome://extensions page's Update extensions now button.

## 4.5 Match Patterns

Host permissions and content script matching are based on a set of URLs defined by match patterns. A match pattern is essentially a URL that begins with a permitted scheme (http, https, file, or ftp, and that can contain '\*' characters. The special pattern <all\_urls> matches any URL that starts with a permitted scheme. Each match pattern has 3 parts:

- scheme — for example, http or file or \*

Note: Access to file URLs isn't automatic. The user must visit the extensions management page and opt in to file access for each extension that requests it.

- host — for example, www.google.com or \*.google.com or \*; if the scheme is file, there is no host part

- path — for example, /\*, /foo\*, or /foo/bar. The path must be present in a host permission, but is always treated as /\*.

Here's the basic syntax:

```
<url-pattern> := <scheme>://<host><path><scheme> := '*' | 'http' | 'https' | 'file' | 'ftp' <host> := '*' | '*'.' <any char except '/' and '*'>+ <path> := '/' <any chars>
```

The meaning of '\*' depends on whether it's in the scheme, host, or path part. If the scheme is \*, then it matches either http or https, and not file, or ftp. If the host is just \*, then it matches any host. If the host is \*.hostname, then it matches the specified host or any of its subdomains. In the path section, each '\*' matches 0 or more characters.

## 4.6 Hosting

Warning: As of Chrome 33, Windows users can only download extensions hosted in the Chrome Web store, except for installs via enterprise policy or developer mode (see Protecting Windows users from malicious extensions). As of Chrome 44, no external installs are allowed from a path to a local .crx on Mac (see Continuing to protect Chrome users from malicious extensions).

This page tells you how to host .crx files on your own server. If you distribute your extension, app, or theme solely through the Chrome Web Store, you don't need this page. Instead, consult the store developer documentation.

By convention, extensions, installable web apps, and themes are served—whether by the Chrome Web Store or by a custom server—as .crx files. When you upload a ZIP file with the Chrome Developer Dashboard, the dashboard creates the .crx and file for you. If you aren't publishing the

dashboard, you need to create the .crx file yourself, as described in Packaging. You can also specify autoupdate information to ensure that your users will have the latest copy of the .crx file.

A server that hosts .crx files must use appropriate HTTP headers, so that users can install the file by clicking a link to it.

## 4.7 file installing configuration

Google Chrome considers a file to be installable if either of the following is true :

- The file has the content type application/x-chrome-extension
- The file suffix is .crx and both of the following are true:
  - o The file is not served with the HTTP header X-Content-Type-Options: nosniff
  - o The file is served with one of the following content types:
    - empty string
    - "text/plain"
    - "application/octet-stream"
    - "unknown/unknown"
    - "application/unknown"
    - "\*\*/\*"

The most common reason for failing to recognize an installable file is that the server sends the header X-Content-Type-Options: nosniff. The second most common reason is that the server sends an unknown content type—one that isn't in the previous list. To fix an HTTP header issue, either change the configuration of the server or try hosting the .crx file at another server.



## CHAPTER 5. RESULTS AND DISCUSSIONS

### 5.1 User Interface Representation

#### 5.1.1 Clip something worthy of sharing

Send it right from Clipper. Before you do, highlight important sections or add text and visual callouts so others can see exactly what you're referring to. How to Make the Most Out of Evernote Clipper Evernote Clipper is one of the best tools around if your job involves browsing the web all the time – for example, you're a news or article writer (or you just like to keep well-informed). Here's how to make the most of this great tool.

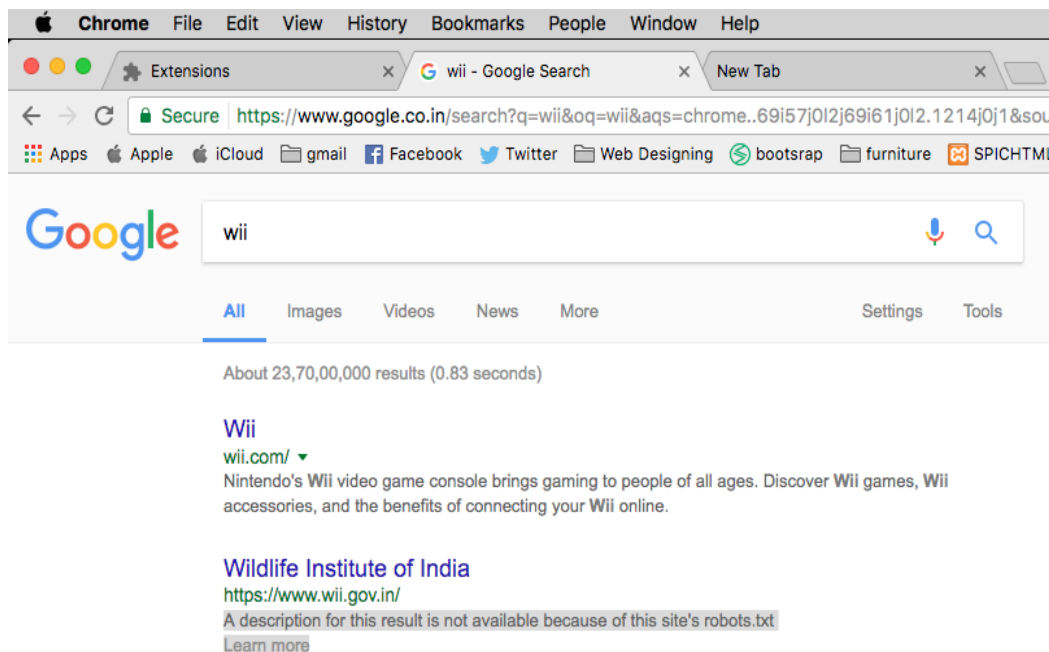


figure 5.1 User Interface Representation

### 5.2 Evernote Clipper

First of all, why is it important to use a tool like Evernote Web Clipper? Well, because it helps you save content you'll want to review later. Also, said content is stored in your Evernote account, where you can easily find and organize it. These things are done via its Chrome extension, which has received a plethora of new and useful features lately, features we'll be taking a look at. Start by installing the extension from the Chrome Web Store if you haven't already done so. After installation, you'll notice that you have a new button on your Chrome toolbar, a button that looks like an elephant. This little button will be your friend from now on. When you've browsed onto a page that you want to keep for later viewing, click on your tiny elephant friend and he will offer a few very useful options. Let's take a look at the coolest ones.

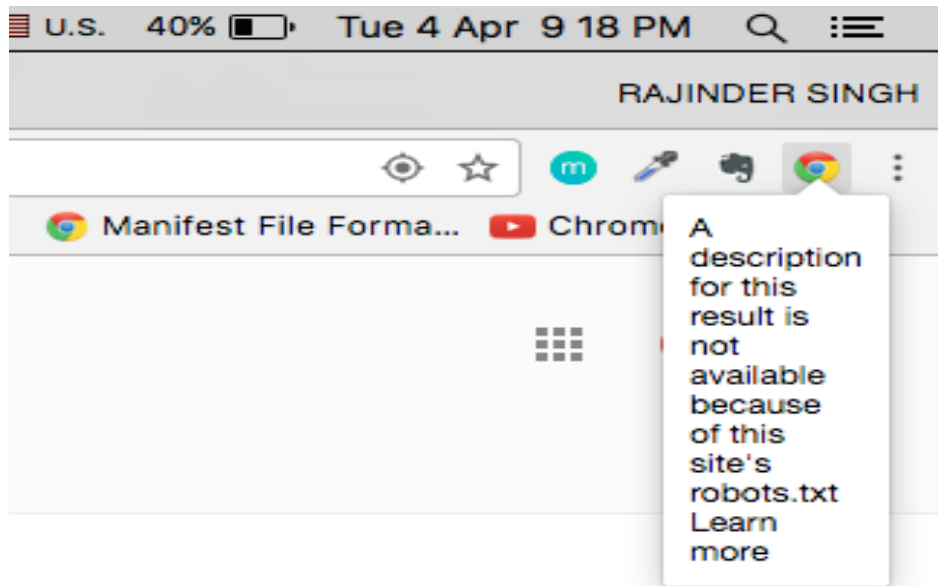


figure 5.2 Evernote Clipper

### 5.3 Save As Needed

This is one of the best parts. Maybe you just want the text part of an article, for a simple read, without all the images in it. You can save just the article itself, in simplified form without all the graphics, or even the full page if you so desire. However, if you just need a bookmark, that's perfectly doable, too.

### 5.4 Keep Organized

If you're the kind that likes to keep your information compartmentalized, this extension is excellent for facilitating organization. The Web Clipper allows you to save your clippings to any notebook you want. I've created a special notebook just for my web clippings. You can even create specific notebooks for work, personal clippings, and much more. That way it's much easier to find your content. You can also save directly to shared notebooks. This allows you to easily share your browsing discoveries with everyone who has access. You can easily change the notebook you're saving to; just click the arrow next to the active notebook's title. You can also add tags and remarks to what you're clipping in order to make searching easier. Save a PDF Straight to Evernote This is a feature I really like. When you're viewing a PDF online, you can easily save it (or bookmark it) in your Evernote account for later viewing. If you click the button in the toolbar in the PDF viewer, you'll notice that the menus are changed. You can now clip the PDF itself or bookmark it.

## **5.5 Easy Access**

You don't need to go through the toolbar menu if you just want to quickly clip something. It's possible to just right-click the page you want to clip; the menu will include the necessary options. If you have a specific feature of Evernote Web Clipper that you're constantly using, know that there's most certainly a shortcut for it. To see what it is, click the Web Clipper button in your Google Chrome toolbar, then click Options. Now go to the Keyboard shortcuts tab. I'm sure the feature you use most has its own shortcut and that's all you'll need to remember. You can also edit the shortcuts for a more personalized, convenient experience.

## **5.6 Screenshots with Your Own Notes**

I've saved one of the best parts for last. If you choose screenshot in the menu, you can take a screenshot of a section of the page you're looking at, which you can annotate with everything you want. You can even blur parts of the image, on top of being able to insert text or mark up what you need to keep in mind.

## **5.7 Bottom Line**

Together with Evernote itself, Evernote Web Clipper creates a package that's quite hard to beat when looking for a tool to save and catalog the articles, images, and other information you run into while browsing online.

## 5.8 Brief Description

### 5.8.1 Chrome Platform API's

Google APIs is a set of application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services. Examples of these include Search, Gmail, Translate or Google Maps. Third-party apps can use these APIs to take advantage of or extend the functionality of the existing services.

The APIs provide functionality like analytics, machine learning as a service (the Prediction API) or access to user data (when permission to read the data is given). Another important example is an embedded Google map on a website, which can be achieved using the Static maps API, Places API or Google Earth API.

### 5.8.2 Authentication and authorization

Usage of some of the APIs requires authentication and authorization using the OAuth 2.0 protocol. OAuth 2.0 is a simple protocol. To start, it is necessary to obtain credentials from the Developers Console. Then the client app can request an access token from the Google Authorization Server, and uses that token for authorization when accessing a Google API service.

### 5.8.3 Client libraries

There are client libraries in various languages which allow developers to use Google APIs from within their code, including Java, JavaScript, .NET, Objective-C, PHP and Python.

The Google Loader is a JavaScript library which allows web developers to easily load other JavaScript APIs provided by Google and other developers of popular libraries. Google Loader provides a JavaScript method for loading a specific API (also called module), in which additional settings can be specified such as API version, language, location, selected packages, load callback and other parameters specific to a particular API. Dynamic loading or auto-loading is also supported to enhance the performance of the application using the loaded APIs.

### 5.8.4 Google Apps Script

Google Apps Script is a cloud-based JavaScript platform which allows developers to write scripts that can manipulate APIs of services such as Calendar, Docs, Drive, Gmail, and Sheets and easily create Add-Ons for these services with chromium based applications.

### 5.8.5 Common use cases

- User registration is commonly done via Google+ sign in, which allows users to securely log in to 3rd party services with their Google+ account using the Google+ API. This is currently available from within Android, iOS or JavaScript. It is popular to include a “Sign in with Google” button in Android apps, as typing login credentials manually is time-consuming due to limited screen size. As the user is usually signed into their Google account on their mobile device, signing-in/signing-up for a new service with a Google is usually a matter of a few button clicks.
- Drive apps are various web applications (often third party) which work within Google Drive using the Drive API. Users can integrate these apps into their Drive from the Chrome Web Store which allows them to do work entirely in the cloud. There are many apps available for collaborative document editing (Google Docs, Sheets), picture/video editing, work management or for sketching diagrams and workflows.
- Custom Search allows web developers to provide a search of their own website by embedding a custom search box and using the Custom Search API. They can customize the search results and make money off the ads shown using AdSense for Search.
- App Engine apps are web apps that run on the Google App Engine, a platform-as-a-service (PaaS) cloud computing platform which allows web developers to run their websites in Google datacenters. These web apps often take advantage of APIs to manipulate services such as TaskQueue (a distributed queue), BigQuery (a scalable database based on Dremel) or DataStore.
- Gadgets are mini-applications built in HTML, JavaScript, Flash and Silverlight that can be embedded in webpages and other apps. They can run on multiple sites and products (even writing them once allow users to run them in multiple places).

## 5.9 Web APIs

In addition to the chrome.\* APIs, extensions can use all the APIs that the browser provides to web pages and apps. If the browser doesn't support an API you want to use, you can bundle additional API libraries into your extension.

Here's a sampling of the APIs that extensions can use:

## 5.9.1 Standard JavaScript APIs

These are the same core JavaScript and Document Object Model (DOM) APIs that you can use in ordinary web apps.

## 5.10 Back Ends Representation

### 5.10.1 The background page

Background pages defined by `background.html` can include JavaScript code that controls the behavior of the extension. There are two types of background pages: persistent background pages, and event pages. Persistent background pages, as the name suggests, are always open. Event pages are opened and closed as needed. Unless you absolutely need your background page to run all the time, prefer to use an event page.

### 5.10.2 UI pages

Extensions can contain ordinary HTML pages that display the extension's UI. For example, a browser action can have a popup, which is implemented by an HTML file. Any extension can have an options page, which lets users customize how the extension works. Another type of special page is the override page. And finally, you can use `tabs.create` or `window.open()` to display any other HTML files that are in the extension.

The HTML pages inside an extension have complete access to each other's DOMs, and they can invoke functions on each other.

The architecture of a browser action's popup's contents are a web page defined by an HTML file (`popup.html`). This extension also happens to have a background page (`background.html`). The popup doesn't need to duplicate code that's in the background page because the popup can invoke functions on the background page. A browser window containing a browser action that's displaying a popup.

The popup's HTML file (`popup.html`) can communicate with the extension's background page (`background.html`). See [Browser Actions, Options, Override Pages, and the Communication between pages](#) section for more details.

## **5.11 Declaring Permissions**

### **5.11.1 Implementing optional permissions:**

Decide which permissions are required and which are optional:

An extension can declare both required and optional permissions. In general, you should:

Use required permissions when they are needed for your extension's basic functionality.

Use optional permissions when they are needed for optional features in your extension.

### **5.11.2 Advantages of required permissions:**

Fewer prompts: An extension can prompt the user once to accept all permissions.

Simpler development: Required permissions are guaranteed to be present.

### **5.11.3 Advantages of optional permissions:**

Better security: Extensions run with fewer permissions since users only enable permissions that are needed.

Better information for users: An extension can explain why it needs a particular permission when the user enables the relevant feature.

Easier upgrades: When you upgrade your extension, Chrome will not disable it for your users if the upgrade adds optional rather than required permissions.

### **5.11.4 array of string (optional) permissions**

List of named permissions (does not include hosts or origins). Anything listed here must appear in the `optional_permissions` list in the manifest.

### **5.11.5 array of string (optional) origins**

List of origin permissions. Anything listed here must be a subset of a host that appears in the `optional_permissions` list in the manifest. For example, if `http://*.example.com/` or `http://*/`

appears in `optional_permissions`, you can request an origin of `http://help.example.com/`. Any path is ignored.

## **5.12 Clicking the Re-enable button brings up the following warning:**

### **5.12.1 Warnings and their triggers**

It can be surprising when adding a permission such as "tabs" results in the seemingly unrelated warning that the extension can access your browsing activity. The reason for the warning is that although the `chrome.tabsAPI` might be used only to open new tabs, it can also be used to see the URL that's associated with every newly opened tab (using their `tabs.Tab` objects).

### **5.12.2 Permissions that don't cause warnings**

The following permissions don't result in a warning:

- "activeTab"
- "browsingData"
- "clipboardWrite"
- "contextMenus"
- "cookies"
- "experimental"
- "idle"
- "storage"
- "unlimitedStorage"
- "webRequest"
- "webRequestBlocking"



## CHAPTER 6. CONCLUSION AND FUTURE SCOPE

### 6.1 Highlight text

To highlight text on a web page, simply click and drag the highlighter cursor across text on a page. To remove highlights, hover over the highlighted text and click the 'X'.



figure 6.1

**Highlighter tool:** When the Web Clipper detects text on a page, the cursor becomes a Highlighter tool.

When you select '**Article**', '**Simplified article**', or '**Full page**', your cursor turns into a highlighting tool, so you can point out or remind yourself of important or inspiring sections on a page.

### 6.2 Add visual callouts

Mark up your screenshot with arrow, shapes, and text as visual callouts for quicker and clearer communication.



figure 6.2

## 6.3 Shape tool

Draw arrows, circles, lines, and other shapes on a screenshot.



figure 6.3

1.8.6 Color options: Choose or edit the color of your new or selected shapes.

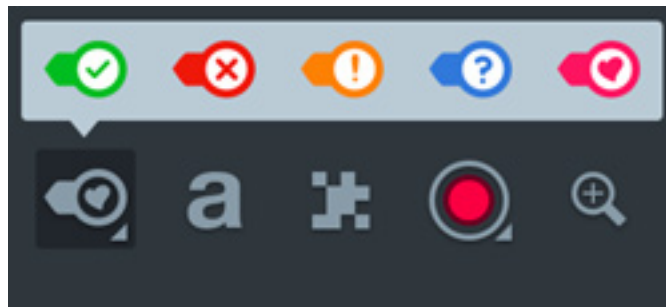








figure 6.4

## 6.4 Stamp tool

Add stamps with labels and arrows. Select and position your stamp on your screenshot. Click the '+' or the 'a' to add an arrow or text label.

## 6.5 Other annotation tools:

-  **Pen tool:** Draw with digital ink.
-  **Type tool:** Add text annotations.
-  **Pixelator tool:** The pixelator tool is perfect for blurring portions of images that need to be kept anonymous, such as faces or other personally identifiable information.
-  **Crop tool:** Drag the lines to crop a screenshot.
-  **Zoom in and Out:** Zoom in and out of your screenshot.
-  **Zoom in and Out:** Zoom in and out of your screenshot.

## 6.6 Share web clips with others

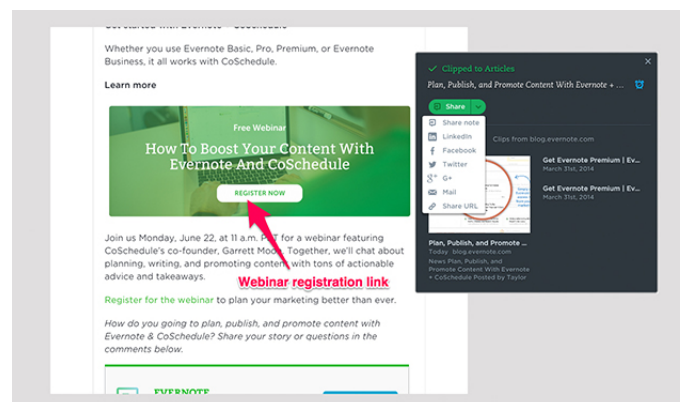


figure 6.5

Web Clipper makes it easy for you to share the knowledge and inspiration you find on the web with your colleagues and friends:

- Take a screenshot of a web site you like, mark it up, and email it to your designer
- Share an interesting web site with friends on your social networks
- Paste a link to your Web Clip or a site right into a chat session with colleagues

## 6.7 Share in Google Chrome, Safari, and Opera

Once you've successfully saved a Web Clip, click **Share** and share the Web Clip note via LinkedIn, Facebook, Twitter, Work Chat or email. You can also share a Web Clip as a public link you can copy and paste in any document or messaging application.

### Share as an email

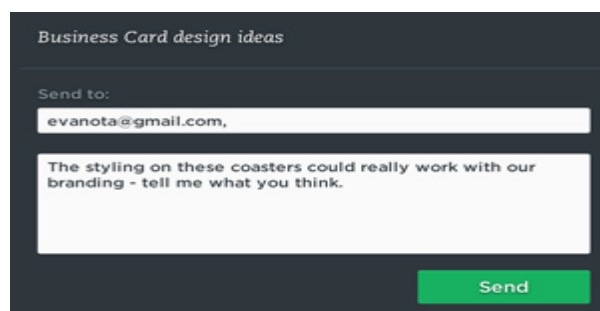


figure 6.6

### Share as a link

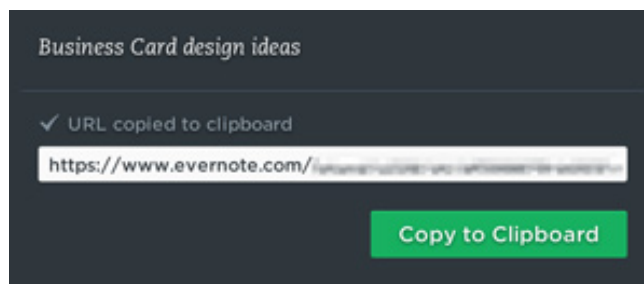


figure 6.7

## 6.8 Share in Internet Explorer and Firefox

Save your Web Clip from any browser, then share it from your Evernote account.

1. After you save your clip, open your Evernote account (log in if prompted).
2. Open the note(s) you've clipped and select the note sharing button in the toolbar near the top of your note. Choose one of the sharing options listed in the window: Work Chat, Facebook, Twitter, LinkedIn, or Email.

## 6.9 More tips

1. Clip emails from Gmail or Outlook
2. Clip from Gmail

From Gmail, launch the Web Clipper, then click **Email** to save a clean, clutter-free version of Gmail messages (and selected threads) directly into your Evernote account.

3. Clip from Microsoft Outlook

From Outlook, click **New Clip** from the toolbar to clip emails, contacts, and calendar events directly to your Evernote account.

4. Add notification alerts to web clips

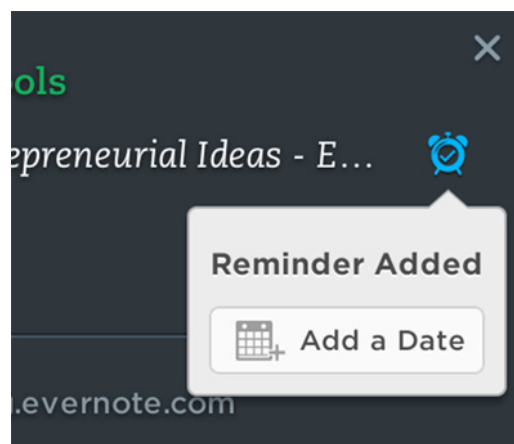


figure 6.8

Clip event details or travel confirmations with email or in-app notifications to remind you of important deadlines.

## 5. Finds notes created with Web Clipper

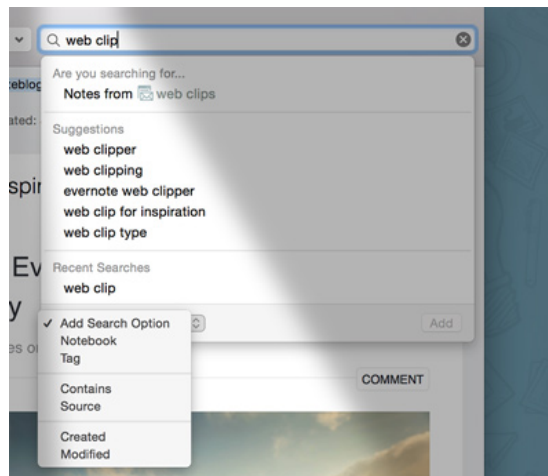


figure 6.9

Evernote lets you use a search filter to show only notes created using the Web Clipper. To bring up a list of all the notes created with the Web Clipper, enter "source:web.clip" in the search bar.

Note: To bring up a list of notes clipped from the Web Clipper on Mac, click inside the search bar, then type "web clips" or select **Add Search Option > Source > Web page**.

## 6. Explore notes related to web clips

Evernote Web Clipper is more than a tool for capturing new notes, it also helps you remember other information you have saved to your Evernote account. After you capture a Web Clip, the confirmation box displays Related Notes from your Evernote account, as well as notes that you have previously captured from the same website.

Based on the content of your new Web Clip, Evernote attempts to suggest other notes in your account that may inspire you. Open these notes in Evernote from the Web Clipper confirmation panel.

## 7. View related notes in search engine results

To view related notes from your Evernote account whenever you search the Web with some of the most common search engines, including Google, Bing, or Yahoo, enable the related results option from your Web Clipper settings.

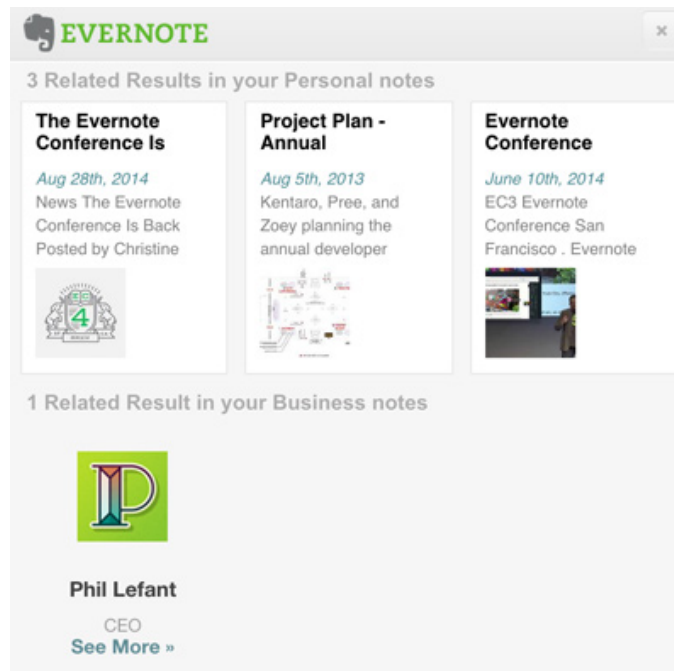


figure 6.10

To enable the related results feature, launch the Web Clipper, click '**Options**', check the 'Related results' box, then click **Done**. You can also enable or disable the option inside your browser's preferences for 'Extensions' or 'Add-Ons'.

## 7. REFERENCES

1. P.Mate and H. Chavan, —Browser Based Continuous Query Processing in API Based Services‡, International Journal of Advanced Research in Computer Science and Software Engineering, Vol.4, Issue 5, May 2014.
2. U. Thakur, V. Nagare and M.Shimpi, —GOOGLE EXTENSION developer Guide System using Augmented Reality: A Review‡, International ISSN: 2278 – 7798 All Rights Reserved © 2016 JSETR:773
3. International Journal of Science, Engineering and Technology Research (IJSETR), Volume 5, Issue 3, March 2016
4. Journal of Soft Computing and Engineering (IJSCE) Vol-5 Issue-5, November 2015.
5. S. A. Jordan and Irbid, —Building Browser based extension to interact with Applications using Different Development Mobile Platforms‡, International Journal of Advanced Science and Technology Vol. 54, May, 2013.
6. B .S.Reddy and Dr R.P Sam,‡Mobile Location-Based Extension developer tools International Journal of Computer Trends and Technology (IJCTT) - volume4 Issue5–May 2013.
7. Google Chrome URL:<https://www.google.co.in/>accessed on 20-March 2017.
8. Google Chrome URL:<https://developer.chrome.com/extensions/devguide> accessed on 23-March 2017
9. Google Chrome URL:<https://developer.chrome.com/extensions/getstarted> accessed on 27-March 2017