

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

**KHOA CÔNG NGHỆ PHẦN MỀM**

**--------------------**

****

**BÁO CÁO MÔN HỌC**

**ĐỒ ÁN CHUYÊN NGÀNH**

**MÃ LỚP: SE112.I11**

**ĐỀ TÀI: CÔNG CỤ COPY MÃ HTML   
CHO LẬP TRÌNH WEB**

**CODESTEAL**

**Giảng viên hướng dẫn:**

**THS. NGUYỄN CÔNG HOAN**

**Sinh viên thực hiện:**

**Trịnh Ngọc Lợi 13520465**

***Năm học 2007 - 2008***

***Năm học 2007 - 2008***

***Năm học 2007 - 2008***

***TPHCM, 01/02/2018***

**NHẬN XÉT CỦA GIẢNG VIÊN**

**MỤC LỤC**

[**I. GIỚI THIỆU** 5](#_Toc505626923)

[**1. Giới thiệu đề tài** 5](#_Toc505626924)

[**2. Mục tiêu đề tài** 5](#_Toc505626925)

[**II. NỀN TẢNG VÀ CÔNG NGHỆ** 5](#_Toc505626926)

[**1. HTML** 5](#_Toc505626927)

[**2. CSS** 5](#_Toc505626928)

[**3. JavaScript** 6](#_Toc505626929)

[**4. DOM** 6](#_Toc505626930)

[***4.1. Cấu trúc DOM*** 6](#_Toc505626931)

[***4.2. Cấu trúc cây DOM*** 6](#_Toc505626932)

[***4.3. Thao tác với DOM*** 7](#_Toc505626933)

[***4.4. Truy xuất DOM*** 8](#_Toc505626934)

[***4.5. DOM Event*** 8](#_Toc505626935)

[**5. Chrome Extension** 10](#_Toc505626936)

[***5.1. Tổng quan*** 10](#_Toc505626937)

[***5.2. Các thành phần chính của Chrome Extension*** 10](#_Toc505626938)

[***5.3. Kiến trúc của Chrome Extension*** 11](#_Toc505626939)

[***5.4. API & Message*** 14](#_Toc505626940)

[**III. XÂY DỰNG CHROME EXTENSION** 16](#_Toc505626941)

[**1. Mô tả đề tài** 16](#_Toc505626942)

[**2. Phân tích luồng xử lý** 18](#_Toc505626943)

[**3. Các chức năng chính của đề tài** 20](#_Toc505626944)

[***3.1. Lựa chọn Element trực tiếp*** 20](#_Toc505626945)

[***3.2. Truy xuất và Beautify mã Html*** 26](#_Toc505626946)

[***3.3. Truy xuất và Beautify mã CSS*** 27](#_Toc505626947)

[***3.4. Hiển thị kết quả trực tiếp trên webpage*** 37](#_Toc505626948)

[***3.5. Copy to Cliboard*** 39](#_Toc505626949)

[***3.6. View On JsFiddle*** 41](#_Toc505626950)

[**4. Hướng dẫn sử dụng** 42](#_Toc505626951)

[**IV. ĐÁNH GIÁ & HƯỚNG PHÁT TRIỂN** 46](#_Toc505626952)

[**1. Đánh giá sản phẩm** 46](#_Toc505626953)

[**2. Hướng phát triển** 47](#_Toc505626954)

[**3. Lời kết** 47](#_Toc505626955)

[**V. TÀI LIỆU THAM KHẢO** 47](#_Toc505626956)

**I. GIỚI THIỆU**

**1. Giới thiệu đề tài**

Ngày nay với sự phát triển không ngừng của mạng Internet, các website theo đó mà ra đời ngày một nhiều. Công việc thiết kế website đang dần thu hút được đông đảo người tham gia. Với những người mới bắt đầu bước chân vào công việc này thì mọi thứ có vẻ tương đối khó và đôi khi họ cần những sự trợ giúp từ các website đã có sẳn.

Việc copy mã Html và Css từ trước tới giờ đơn giản là việc mở devtool và xem mã Html, Css của đối tượng, tuy nhiên công việc ấy lại khá tốn thời gian và khó theo dõi với các người mới. Sẽ rất hữu ích nếu có một công cụ cho phép người dùng lựa chọn một đối tượng trên website và xem mã Html và Css của nó một cách trực tiếp ngay tại website mà không cần phải vào một giao diện khó nhìn nào khác. Đây chính là lý do để em lựa chọn phát triển công cụ copy mã Html và Css cho lập trình web: **CodeSteal**. CodeSteal là một chrome extenstion hỗ trợ người dùng thực hiện các công việc trên một cách dễ dàng

**2. Mục tiêu đề tài**

Như đã trình bày ở trên, công cụ CodeSteal ra đời với mục đích giúp người dùng có thể copy mã Html và Css một cách nhanh chóng và thuận tiện nhất

Ngoài ra CodeSteal còn đặt ra mục tiêu giúp người dùng có thể kiểm tra các mã code mà mình vừa lấy được để tăng sự tin tưởng khi sử dụng sản phẩm. Và hơn hết, giao diện và cách sử dụng công cụ mới này phải thật dễ dàng và trực quan.

**II. NỀN TẢNG VÀ CÔNG NGHỆ**

**1. HTML**

- HTML (Hyper Text Markup Language) là ngôn ngữ siêu văn bản được sử dụng để tạo ra các website.

- Một tài liệu HTML được hình thành bởi các phần tử HTML (HTML Elements) được quy định bằng các cặp thẻ (tag), các cặp thẻ này được bao bởi bởi một dấu ngoặc nhọn và gồm 1 thẻ mở, 1 thẻ đóng (ví dụ: <html></html>)

- Một tập tin HTML sẽ bao gồm nhiều phần tử HTML và được lưu lại với đuôi .html

- HTML đóng vai trò giúp xây dựng cấu trúc siêu văn bản trên một website, hoặc khai báo các tập tin kỹ thuật số như video, hình ảnh, âm thanh.

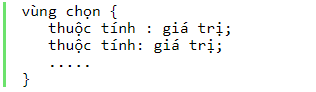
- Nói chung, HTML như là một bộ xương cấu tạo nên cơ thể con người.

**2. CSS**

**-** CSS (Cascading Style Sheets) là ngôn ngữ được sử dụng để tìm và định dạng các phần tử được tạo ra bởi ngôn ngữ HTML. Nếu HTML đóng vai trò tạo ra cấu trúc website thì CSS sẽ giúp website có thêm một chút “phong cách”.

**-** Phương thức hoạt động của CSS là nó sẽ tìm dựa vào các vùng chọn, vùng chọn có thể là tên một thẻ HTML, tên một ID, class hay nhiều kiểu khác. Sau đó là nó sẽ áp dụng các thuộc tính cần thay đổi lên vùng chọn đó.

**-** Cấu trúc của CSS:



- Nghĩa là nó sẽ được khai báo bằng vùng chọn, sau đó các thuộc tính và giá trị sẽ nằm bên trong cặp dấu ngoặc nhọn {}. Mỗi thuộc tính sẽ luôn có một giá trị riêng, giá trị có thể là dạng số, hoặc các tên giá trị trong danh sách có sẵn của CSS. Phần giá trị và thuộc tính phải được cách nhau bằng dấu hai chấm, và mỗi một dòng khai báo thuộc tính sẽ luôn có dấu chấm phẩy ở cuối. Một vùng chọn có thể sử dụng không giới hạn thuộc tính.

**3. JavaScript**

- JavaScript là một ngôn ngữ lập trình của HTML và Web

- JavaScript tạo ra các sự kiện tương tác với hành vi của người dùng (ví dụ như nhấp vào ảnh sẽ có hiệu ứng phóng to)

- Ngoài ra cũng có thể sử dụng JavaScript để thêm các hành vi động, lưu trữ thông tin, xử lý các yêu cầu và phản hồi trên một trang web.

- Nếu chỉ sử dụng HTML và CSS thì website của bạn sẽ thật đơn giản và nhàm chán, JavaScript giúp bạn tạo ra một website sinh động và phức tạp hơn.

**4. DOM**

***4.1. Cấu trúc DOM***

**-** DOM (Document Object Model) là một chuẩn được định nghĩa bởi W3C dùng để truy xuất và thao tác trên các tài liệu có cấu trúc dạng HTML hay XML bằng các ngôn ngữ lập trình thông dịch (scripting language) như Javascript, PHP, Python.

**-** Có thể nói Webpage = DOM + JavaScript, mặc dù hiện nay có thể tương tác với DOM bằng PHP, Python,…; tuy nhiên, JavaScript đã theo suốt lịch sử phát triển của DOM nên sử dụng DOM JavaScript để phát triển web là một lựa chọn tốt nhất

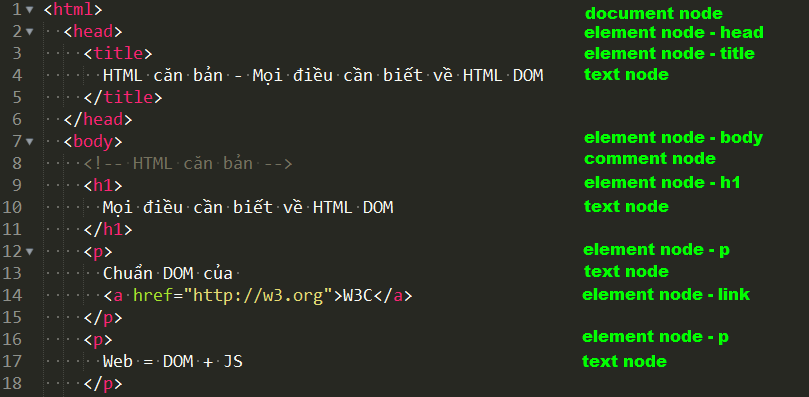
***4.2. Cấu trúc cây DOM***

- DOM được chia thành các Node, được biể diễn trên một cấu trúc dạng cây gọi là DOM Tree. Có 3 loại Node quan trọng là Document Node, Element Node và Text Node:

+ Document Node: là tài liệu HTML, biểu diễn bởi cặp thẻ *<html> </html>*

+ Element Node: biểu diễn 1 phần tử HTML

+ Text Node: Một đoạn ký tự trong tài liệu HTML



- Các Node trong DOM có quan hệ với nhau:

+ Document Node là node đầu tiên

+ Các Node không phải Node gốc đều có một Node cha (parent)

+ Một Node có thể có nhiều lớp con (children) hoặc không có node con nào

+ Những Node có cùng Node cha gọi là các Node anh em (siblings)

+ Trong các Node anh em, Node đầu tiên là firstChild và Node cuối cùng là lastChild

***4.3. Thao tác với DOM***

- Thao tác với DOM thông qua các phương thức và các thuộc tính. Dưới đây là một số phương thức và thuộc tính quan trọng sử dụng trong công cụ CodeSteal, các bạn có thể tham khảo đầy đủ tại <https://www.w3schools.com/jsref/dom_obj_all.asp> .

- Một số thuộc tính quan trọng:

+ id: Định danh, là duy nhất cho mỗi phần tử DOM, dùng để truy xuất trực tiếp và nhanh chóng

+ className: Tên class, sử dụng để truy xuất phần tử DOM nhưng 1 phần tử có thể có nhiều className

+ tagName: tên của thẻ HTML

+ innerHTML: Trả về mã HTML bên trong phần tử hiện tại

+ outerHTML: Trả về mã HTML của phần tử hiện tại: outerHTML = innerHTML + tagName

+ style: Tập các thiết lập CSS của phần tử hiện tại

- Một số phương thức quan trọng:

+ getElementById(id): tham chiếu đến Node duy nhất có id giống với id truyền vào

+ getElementsByClassName(className): tham chiếu đến các Node có className giống với className truyền vào

+ appendChild: thêm một Node con vào Node hiện tại

+ removeChild: Xóa một Node con khỏi Node hiện tại

- Các thuộc tính quan hệ:

+ parentNode: Node cha

+ childNodes: Các Node con

+ firstChild: Node con đầu tiên

+ lastChild: Node con cuối cùng

***4.4. Truy xuất DOM***

- Truy xuất gián tiếp: Truy xuất qua các thuộc tính quan hệ, ví dụ muốn truy xuất đến Node cha của Node hiện tại: *Node.parentNode*

- Truy xuất trực tiếp: truy xuất qua các phương thức như getElementById(id) hoặc getElementsByClassName(className). Với cách này sẽ nhanh hơn và đơn giản hơn truy xuất gián tiếp vì không cần phải biết nhiều quan hệ và vị trí của Node

***4.5. DOM Event***

- Event và các hàm xử lý event của DOM là một phần quan trọng của JavaScript. Event sẽ xảy ra khi có sự tương tác giữa người dùng hoặc từ chính trang web. Có rất nhiều event DOM ví dụ như: onclick, mouseout, onload, ready, resize, … Tham khảo đầy đủ hơn tại <https://developer.mozilla.org/en-US/docs/Web/Events>

- Có 3 cách để sử dụng Event và các function xử lý nó:

+ Chèn trực tiếp vào thẻ HTML: Cách này nhanh chóng và dễ sử dụng, tuy nhiên lại phức tạp hóa code khi xây dựng các ứng dụng lớn.



+ Chèn vào thuộc tính event của DOM: Cách này khá phổ biến và được sử dụng nhiều vì sự đơn giản và hiệu quả của nó:



+ Dùng phương thức addEventListener: Sử dụng phương thức này khi bạn quan tâm tới chất lượng code hoặc muốn xây dựng them mô hình hướng đối tượng



**5. Chrome Extension**

***5.1. Tổng quan***

- Chrome Extension là một tập các file như: html, css, javascript, images, … được nén lại trong 1 thư mục. Nó nằm mục đích cung cấp thêm các chức năng mở rộng cho trình duyệt Google Chrome. Chrome Extension về cơ bản là những web pages như những website bình thường khác và có thể sử dụng tất cả các API mà trình duyệt chrome cung cấp

- Nhìn chung, Chrome Extension thực sự đơn giản và không hề phức tạp như chúng ta vẫn nghĩ về nó.

***5.2. Các thành phần chính của Chrome Extension***

**-** Extension Uis: Mỗi extension có thể là một browers action hoặc một page action. Bạn chọn nó là một browers action khi nó có ảnh hưởng đến tất cả các trang, và chọn nó là một page action khi nó chỉ ảnh hưởng đến một số trang nhất định: Những extension có thể biểu diễn theo những cách khác nhau như: thêm vào Context Menu, cung cấp một option page, hoặc sử dụng content script để thay đổi hiển thị lên webpage

**-** Files: Mỗi Extension cơ bản gồm các files sau:

**+** Một file ***manifest***

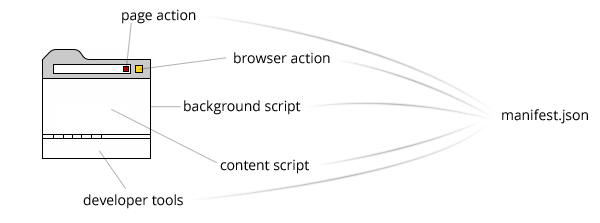
**+** Các file HTML để thiết lập hiển thị cho extension

**+** Optional: Các file javascript để xử lý dữ liệu, thông điệp, tương tác với người dùng

**+** Optional: images để đặt làm icon hiển thị hoặc bất cứ thứ gì mà bạn cần thêm

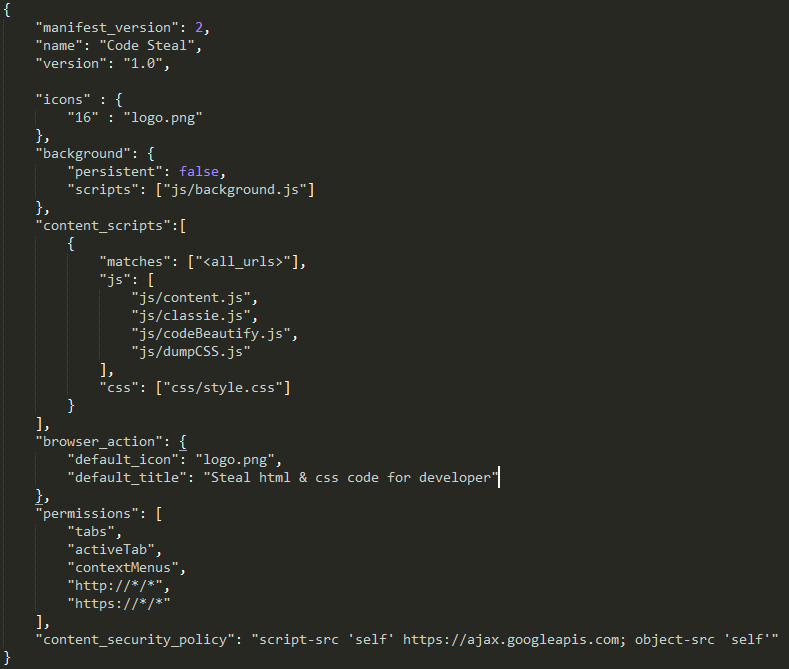
***5.3. Kiến trúc của Chrome Extension***

- Dưới đây là sơ đồ kiến trúc của một chrome extension cơ bản:



- ***File manifest.json:***

+ Là một file json quan trọng nhất của Chrome Extension, file này có nhiệm vụ khai báo các thông tin cơ bản về extension như: Name, version, description, background, brower\_action, permissions, …



+ Trên đây là một minh họa cho file manifest.json. Cách trình bày của file bắt buộc ở dạng json. Các bạn có thể xem chi tiết từng thuộc tính của file manifest.json tại <https://developer.chrome.com/extensions/manifest> .

- ***Background page:***

+ Mỗi extension có một background page không thể nhìn thấy và được chạy bởi trình duyệt.

+ Có 2 loại background page: Trang cố định và trang sự kiện. Trang cố định được hoạt động mọi lúc, trang sự kiện chỉ hoạt động khi cần thiết. Tuy nhiên, google khuyến khích các nhà phát triển nên sử dụng các trang sự kiện vì nó giúp tiết kiệm bộ nhớ và cải thiện hiệu suất tổng thể của trình duyệt.

+ Background page chính là nơi đặt các xử lý logic và những khởi tạo cho extension của bạn, và nó đóng vai trò như là cầu nối giữa các phần khác nhau của extension.

+ Để khai báo sử dụng background page chúng ta khai báo thuộc tính:

*"background": {*

*"persistent": false,*

*"scripts": ["js/background.js"]*

*},*

+ *persistent* đặt *fasle* nếu bạn sử dụng trang sự kiện và *true* nếu bạn sử dụng trang cố định

+ *script:* nơi bạn khai báo tập tin script để thực hiện các xử lý cho background

- ***Content Script:***

+ Được sử dụng nếu bạn cần truy cập vào DOM của trang hiện tại. Code trong file content.js sẽ được chạy trong ngữ cảnh của trang web hiện tại, nghĩa là nó như một thành phần script của trang web và sẽ được thực thi mỗi lần trang web làm mới.

+ Để khai báo sử dụng content script, chúng ta cần khai báo thuộc tính:

*"content\_scripts":[*

*{*

*"matches": ["<all\_urls>"],*

*"js": [*

*"js/content.js",*

*"js/classie.js",*

*"js/codeBeautify.js",*

*"js/dumpCSS.js"*

*],*

*"css": ["css/style.css"]*

*}*

*],*

+ *matches:* xác định các trang web mà script được sử dụng, đặt giá trị all\_urls như trên để sử dụng cho tất cả các trang web.

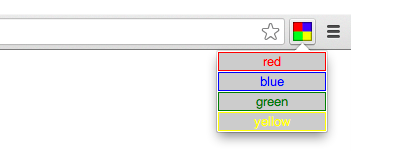
+ *js*: Nơi bạn khai báo các file js sẽ được sử dụng để đưa vào ngữ cảnh cửa trang web

+ *css*: tương tự như *js*

- ***Brower Action:***

+ Là một thành phần UI, và được sử dụng bởi hầu hết các nhà phát triển khi xây dựng chrome extension.

+ Khi sử dụng nó, một biểu tượng sẽ được hiển thị ở bên phải thanh địa chỉ, người dùng có thể nhấp vào biểu tượng và mở một cửa sổ để tương tác.



+ Để khai báo sử dụng browser\_action, chúng ta cần khai báo thuộc tính:

*"browser\_action": {*

*"default\_icon": "logo.png",*

*"default\_popup": "popup.html",*

*"default\_title": "Steal html & css code for developer"*

*},*

+ *default\_icon*: khai báo tập tinh hình ảnh làm biểu tượng cho Browser action

+ *default\_popup*: khai báo tập tin HTML, chính là thể hiện khi người dùng click vào biểu tượng

+ *default\_title*: Tiêu đề hiển thị khi người dùng hover vào biểu tượng

***5.4. API & Message***

- ***API*** là một phần không thể thiếu của Chrome Extension, có rất nhiều thứ khác nhau để thực hiện trong chrome extentsion nhờ vào sự trợ giúp của API, như truy cập vào bookmark, lịch sử người dùng, di chuyển, tạo các tab chrome, …

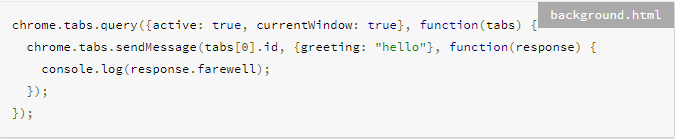
- Có thể tham khảo thêm về API và cách sử dụng của từng loại ở đây: <https://developer.chrome.com/extensions/api_index>

- Tuy nhiên, không phải tất cả các API đều có sẵn trong mọi phần của Extension và không phải lúc nào cũng có thể truy cập vào API mà chúng ta muốn sử dụng. Lúc này, chúng ta nên nghĩ đến việc sử sử dụng ***Message***

- Có 2 kiểu Message là: Long-lived conections và One-time request

+ One-time request: Kiểu giao tiếp này chỉ sảy ra khi có yêu cầu, ví dụ bạn gửi tin nhắn và chờ một câu trả lời:

• Để đơn giản hóa, nên sử dụng runtime.sendMessage hoặc tabs.sendMessage để gửi thông điệp giữa những phần khác nhau trong Extension:



• Khi đã gửi thông điệp, ở content.js, chúng ta cần set up một trình lắng nghe sự kiện:



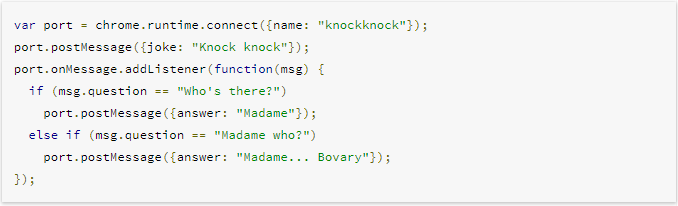
• Như vậy, việc gửi yêu cầu một lần của Chrome Extension sảy ra tương đối đơn giản và dễ hiểu. Tuy nhiên có một lưu ý nhỏ: Nếu như có nhiều page cùng lắng nghe một Message gửi đi thì chỉ page đầu tiên gửi reponse được chấp nhận, tất cả các reponse trả về sau đó sẽ bị bỏ qua.

• Với việc sử dụng thông điệp như vậy, chúng ta có thể dễ dàng lấy thông tin DOM từ webpage và xử lý chung một cách đơn giản.

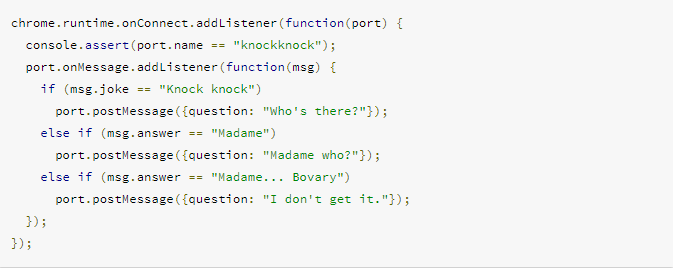
+ Long-lived connections: Sử dụng kiểu truyền thông điệp này nếu chúng ta cần một kênh giao tiếp ổn định.

• Có thể mở kênh giao tiếp từ content script tới extension page hoặc ngược lại bằng cách sử dụng runtime.connect hoặc tabs.connect. Mỗi kênh giao tiếp sẽ có một port riêng để phân biệt, thường sử dụng runtime.Port để khai báo cổng port cho kênh giao tiếp

• Ví dụ đơn giản, từ content script, sẽ mở một kênh giao tiếp để gửi và lắng nghe thông điệp



• Để lắng nghe các sự kiện được gửi trong kênh giao tiếp, ở extension page chúng ta cần set up một sự kiện runtime.onConnect để lắng nghe thông điệp, và tất nhiên, chỉ khi cổng port nhận được đúng với port của kênh giao tiếp thì thông điệp mới có thể nhận được.



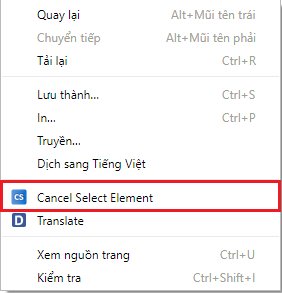
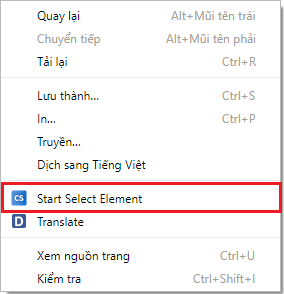
- Trên đây là một số ví dụ về Message trong Chrome Extension, tùy vào mục đích sử dụng mà chúng ta sẽ sử dụng loại truyền thông điệp nào.

**III. XÂY DỰNG CHROME EXTENSION**

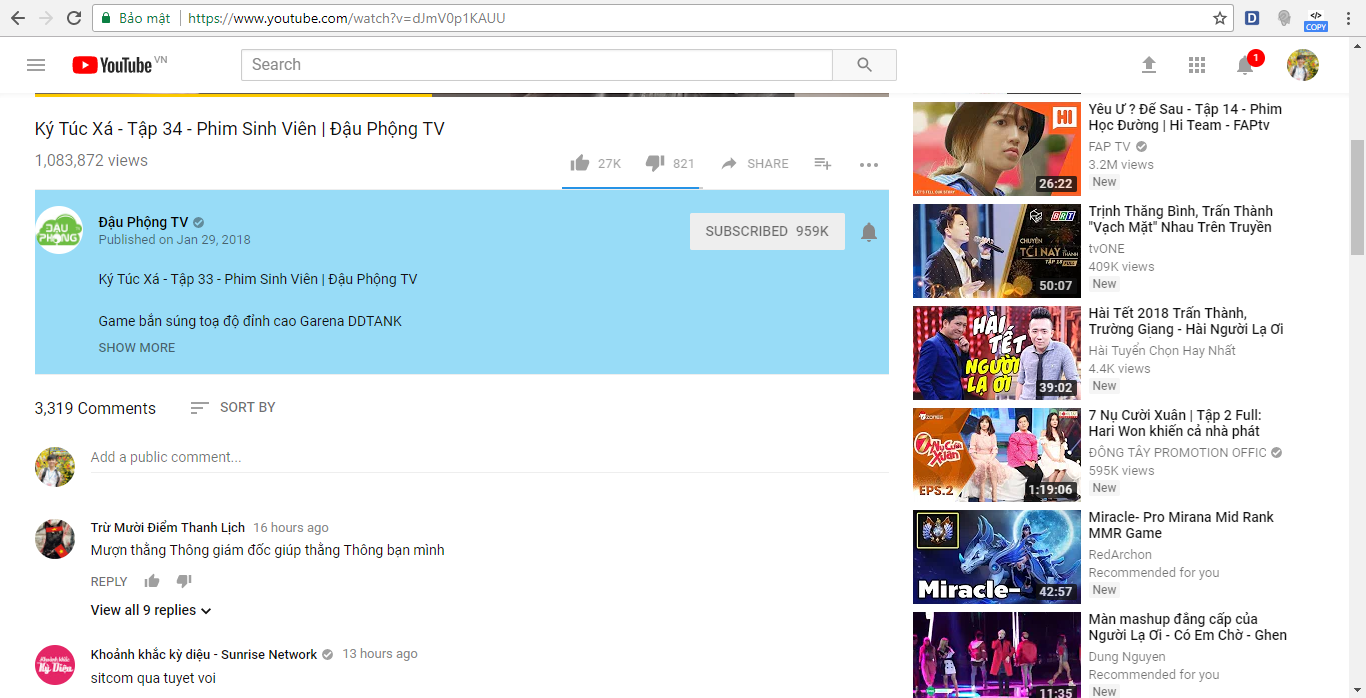
**1. Mô tả đề tài**

- CodeSteal là một chrome extension hỗ trợ lập trình viên có thể truy xuất được mã html và css của một đối tượng DOM (một element hay một node). Bên cạnh đó sản phẩm cho phép người dùng có thể xem code trước trên JSFIDDLE.COM (một trang web cho phép bạn chạy các mã html, css và javascript).

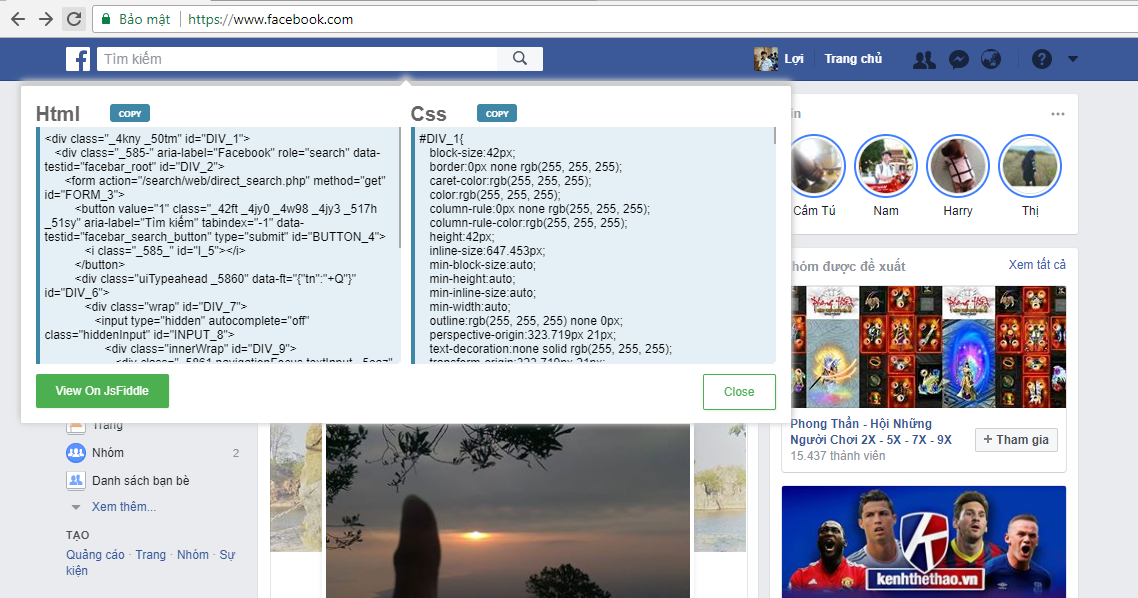
- Với giao diện và cách sử dụng đơn giản, CodeSteal dễ dàng để người dùng mới tiếp cận và sử dụng. Các thao tác với công cụ này cũng tương đối trực quan, không yêu cầu người dùng phải thực hiện các hành động phức tạp và khó khăn.



***ContextMenu: Thêm một đối tượng vào context menu***



***Lựa chọn element: Các element sẽ được highlight khi mouseover***



***Kết quả mã HTML và CSS sau khi click vào element***

**2. Phân tích luồng xử lý**

**- *Trạng thái lựa chọn element:*** là trạng thái mà người dùng có thể biết mình đang rê chuột qua element nào thông qua nền highlight màu xanh và có thể click vào element để xem mã code HTML và CSS của element đó. Trong trạng thái này, các sự kiện onclick (như click vào các link) sẽ bị vô hiệu hóa, vì thế người dùng có thấy một bảng kết quả khi click vào bất cứ element nào họ muốn. Các sự kiện onclick sẽ được khôi phục lại khi trạng thái lựa chọn element được tắt.

*-* ***JsFiddle:*** là một website cho phép review code html và css trực tiếp. Tương tự CodePen

- ***Start Select Element:*** Khởi động trạng thái lựa chọn Element

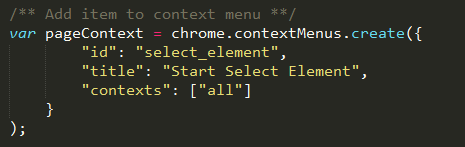
- ***Cancel Select Element:*** Tắt trạng thái lựa chọn Element

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Người dùng** | **Background Script** | **Content Script** |
| 1 | Click vào ContextMenu: Chọn “*Start Select Element”* |  |  |
| 2 |  | Nhận sự kiện và gửi thông điệp cho biết người dùng yêu cầu chọn element |  |
| 3 |  |  | Bật trạng thái cho phép người dùng chọn element đồng thời gửi thông điệp phản hồi đã thay đổi trạng thái |
| 4 |  | Nhận thông điệp đã thay đổi trạng thái và thay đổi hiển thị của ContextMenu: *Start Select Element* => *Cancel Select Element* |  |
| 5 | Click vào một element để xem kết quả HTML và CSS của element đó |  |  |
| 6 |  |  | Tắt trạng thái chọn element đồng thời gửi thông điệp phản hồi đã thay đổi trạng thái |
| 7 |  | Nhận thông điệp đã thay đổi trạng thái và thay đổi hiển thị của ContextMenu: *Cancel Select Element* => *Start Select Element* | Truy xuất mã HTML và CSS của element vừa được chọn từ người dùng |
| 8 |  |  | Chèn một element mới (thẻ <div>), chứa nội dung là mã HTML và CSS vừa truy xuất được vào trang web hiện tại để hiển thị cho người dùng xem và tương tác. |
| 9 | Click button copy để copy mã code |  |  |
| 10 |  |  | Nhận sự kiện click vào button copy và gửi thông điệp cho Background với nội dung là mã code người dùng muốn copy |
| 11 |  | Nhận thông điệp và copy mã code vào clipboard để người dùng có thể sử dụng. |  |
| 12 | Click button “View On JsFiddle” |  |  |
| 13 |  |  | Gửi code HTML và CSS đến JsFiddle để hiển thị. |
| 14 | Click button “Close” hoặc click ngoài vùng thẻ <div> đang hiển thị nội dung mã code HTM và CSS |  |  |
| 15 |  |  | Xóa bỏ element (thẻ <div>) khỏi trang web hiện tại. Đặt trang web trở lại trạng thái bình thường |

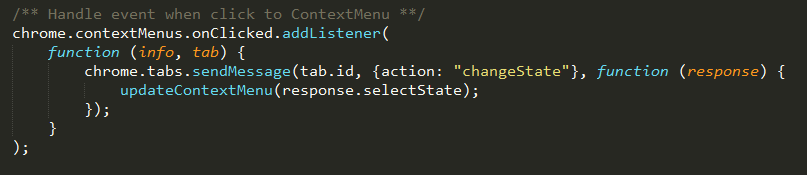
**3. Các chức năng chính của đề tài**

***3.1. Lựa chọn Element trực tiếp***

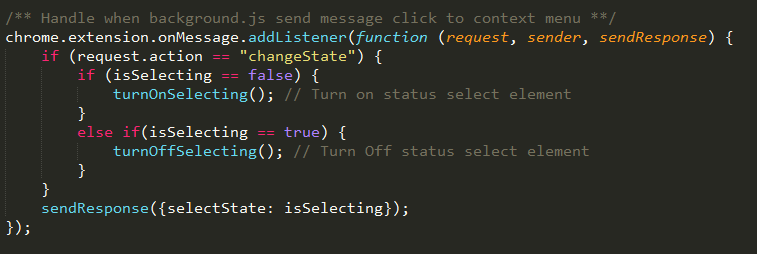
- Thông thường, để có thể lựa chọn một element trong một web page, chúng ta phải vào devtool (ấn f12) và lựa chọn element. Tuy nhiên để người dùng có thể tương tác một cách trực tiếp và thuận tiện nhất. CodeSteal hỗ trợ người dùng chức năng lựa chọn element một cách trực tiếp (không cần mở devtool).



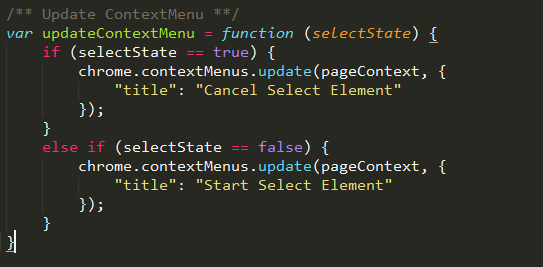
- Hỗ trợ chèn thêm một đối tượng vào ContextMenu với tên hiển thị là “Start Select Element”.



- Xử lý sự kiện khi người dùng click vào ContextMenu, background gửi Message tới content.js để cho biết người dùng yêu cầu “changeState”.

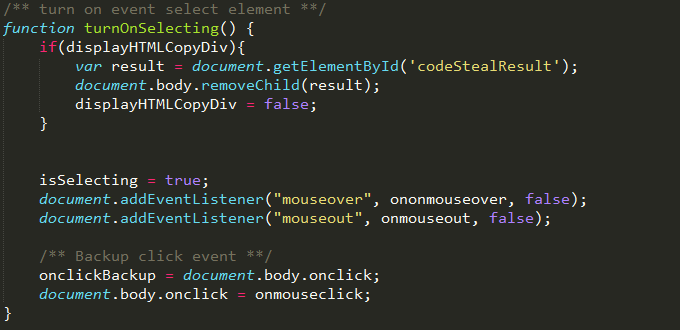


- content.js bắt sự kiện, kiểm tra xem thông điệp nhận được có phải là “changeState” hay không, nếu đúng sẽ kiểm tra trạng thái hiện tại của webpage (đã ở trong trạng thái select element hay chưa), và gửi một reponse về cho background là trạng thái của hiện tại của webpage. Đồng thời bật tắt trạng thái select elemnt thông qua 2 function *“turnOnSelecting”* và *“turnOffSelecting”*



- Sau khi có phản hồi từ phía content.js, sẽ tiến hành update lại ContextMenu thông qua function “updateContextMenu”.

* Trạng thái lựa chọn element trực tiếp:



- function trên được sử dụng để khởi động trạng thái lựa chọn element cho webpage:

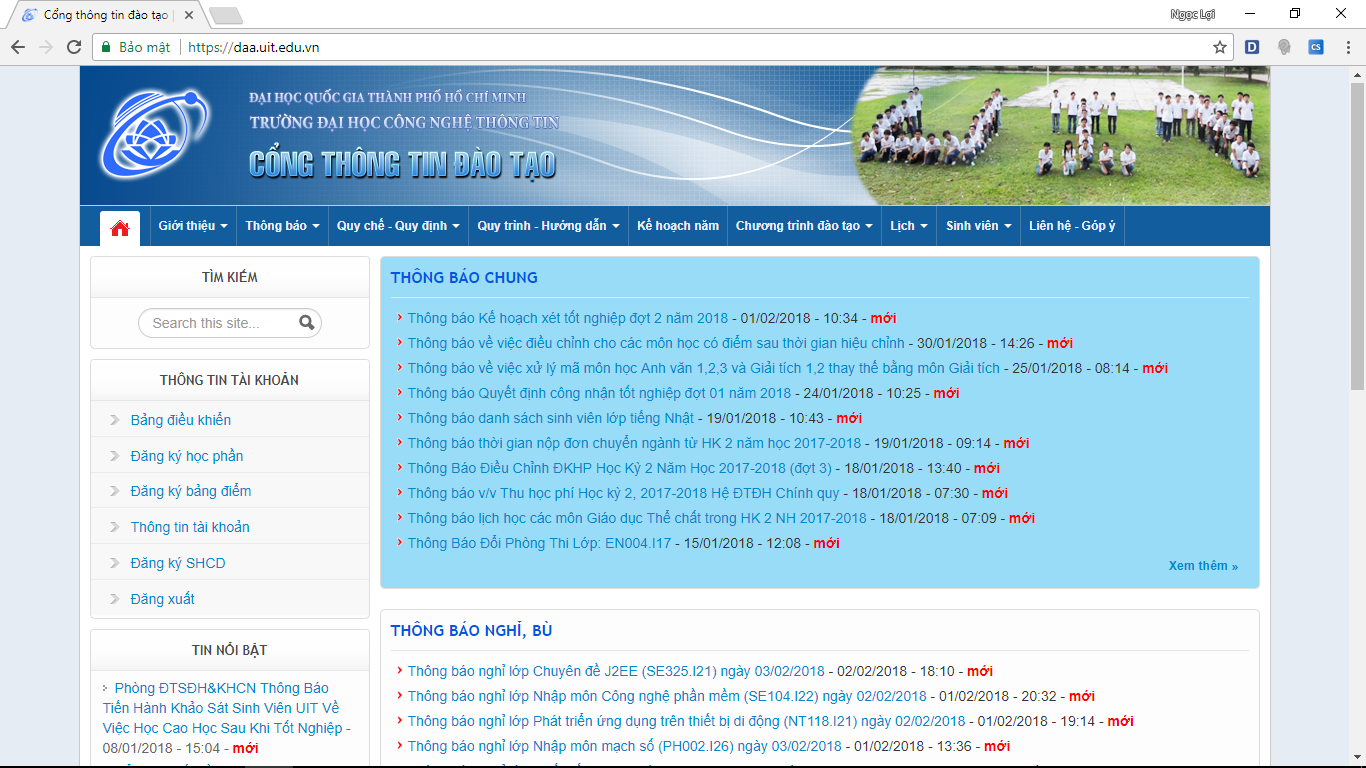
+ Đầu tiên sẽ kiểm tra xe có bảng kết quả nào đang được hiển thị hay không? Nếu có thì remove khỏi webpage. Phòng trường hợp người dùng chưa tắt bảng kết quả nhưng lại muốn tiếp tục xem element khác.

+ Sau đó sẽ gán một biến “isSelecting = true” để đánh dấu là đang trong trạng thái lựa chọn element

+ Thay thế 2 sự kiện mouseover và mouseout của document webpage. Backup lại trạng tất cả các event onclick của document (như click vào các link, ajax, ...) và thay thế bằng event onclick của **CodeSteal:**



• onmouseover: function hỗ trợ thêm mới một className vào element. CodeSteal phát hiện các element dựa vào target event. Khi hover qua một đối tượng DOM (element) bất kỳ, function trên sẽ bắt đối tượng đó và gán bằng biến toàn cục *“clickedEl”.* Sau đó sẽ chèn thêm *class highlight* (với thuộc tính background-color là màu xanh nước biển) vào element đó để người dùng có thể biết mình đang rê chuột qua đối tượng nào, vị trí ở đâu.



*Phần “THÔNG BÁO CHUNG” đang được rê chuột qua nên đổi được đổi màu nền*

• onmouseout: function hỗ trợ xóa class highlight khi rê chuột ra ngoài phạm vi đối tượng để trả đối tượng về trạng thái hiển thị bình thường.



• onmouseout: function xử lý khi người dùng click trong trạng thái lựa chọn element. Nếu người dùng click chuột trái (*event.button == 0*) thì tiến hành xử lý. Đầu tiên sẽ tắt trạng thái lựa chọn element, đưa webpage về trạng thái bình thường.

Khi người dùng click vào element thì trước đó nó đang được chuột rê qua nên biến clickedEl lúc này đã được gán, tức là biến clickedEl lúc này chính là element vừa được click.

Đầu tiên, sẽ tiến hành truy xuất toàn bộ mã CSS của element. Sau đó sẽ lấy toàn bộ mã HTML và tiến hành format, beautify. Sau khi đã có đầy đủ element, mã HMTL và CSS đã được beautify thì sẽ chèn thêm một đối tượng Div để hiển thị kết quả cho người dùng.

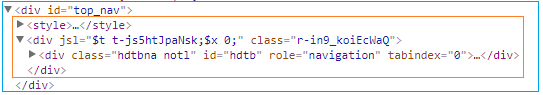
Đồng thời, gửi một Message cho background cho biết rằng element đã được click để backgound.js update lại ContextMenu: *“Cancel Select Element” => “Start Select Element”.*

***3.2. Truy xuất và Beautify mã Html***

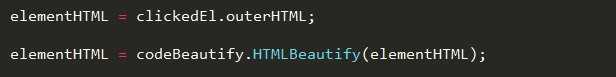
**-** Với DOM, để lấy được mã html của một đối tượng, thường sử dụng 2 phương thức: innerHTML và outerHTML:

+ innerHTML: truy xuất mã HTML bên trong thẻ tag của đối tượng. (Phần code bên trong khung màu đỏ)

+ outerHTML: truy xuất toàn bộ HTML của đối tượng, bao gồm cả thẻ tag (Phần code bên trong khung màu xanh)



- Khi copy mã HTML, thường hay bị lỗi format và sẽ rất khó nhìn với các mã HTML chưa được format như vậy. Đễ hỗ trợ người dùng thuận tiện hơn khi sử dụng, CodeSteal đã tích hợp việc format và beautify mã HTML về đúng định dạng.



**-** Gán outerHTML cửa element cho một biến string. Và sau đó format lại code HTML qua *function HTMLBeautify.*



- function trên hỗ trợ format và beautify mã HTML. Đầu tiên sẽ chia nhỏ mã HTML thành các dòng tag đơn lẻ thông qua format của HTML, các tag có dạng <tag-name>. Sau đó sử dụng regex (một biểu thức chính quy xử lý chuỗi nâng cao) để format lại các dòng tag đơn lẻ và cuối cùng là nối các dòng đơn lẻ trở lại thành một đoạn code HTML đã được beautify.

***3.3. Truy xuất và Beautify mã CSS***

**-** Theo tìm hiểu có nhóm, có 2 cách để truy xuất mã CSS của một đối tượng:

**+** Sử dụng styleSheets và cssRules

**+** Sử dụng getComputedStyle

***3.3.1. StyleSheets***

- styleSheets là một đối tượng DOM chứa tất cả các style liên quan đến trang web hiện tại, được định nghĩa bằng thẻ <style> hoặc <link rel=“stylesheet”>. Hiểu đơn giản, mỗi styles sheet giống như một file .css:

+ Truy xuất styleSheets trong javascript: ***var sheets = document.styleSheets;***

+ styleSheets là một mảng chứa nhiều sheet nên truy xuất tới từng sheet theo số thứ tự của nó: ***var sheet = sheets[0];***

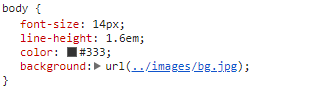
- cssRules[] là một mảng chứa tất cả các CSS rules của một style sheet:

+ Truy xuất styleSheets trong javascript: ***var rules = sheet .cssRules;***

- rule là một đối tượng trong mảng cssRules[], quy định CSS cho một element HTML, hay chính là các mã CSS của element.

+ Truy xuất rule trong javascript: ***var rule = rules[0];***

- Mỗi rule bao gồm các properties:



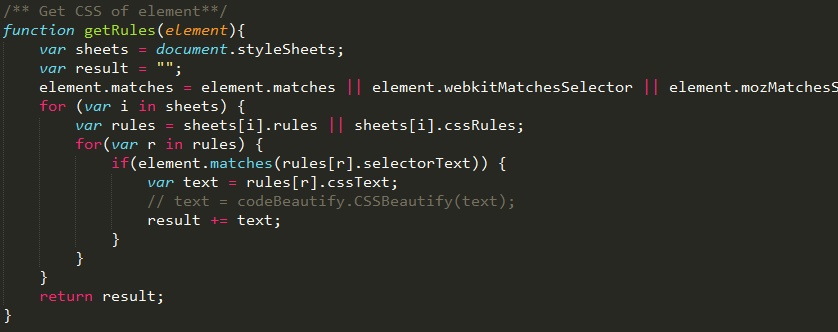
+ cssText: Trả về nội dung của một CSS rule. Ví dụ phía trên là 1 rule

+ selectorText: Trả về tên đại diện của rule, chính là tên của các class, tag, id. Trong ví dụ trên *selectorText là body*

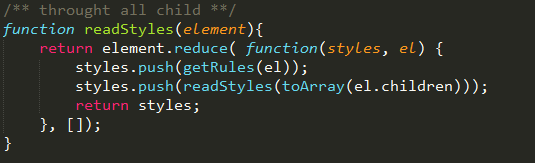
+ style: Mảng đối tượng các rule của element cho phép người dùng read/write thông qua phương thức này:

*if (rule* ***.****style.color != "red")*

*rule.style.color = " red";*



- Function getRules: lọc qua tất cả các rule, kiểm tra nếu như selectorText của rule bằng với element thì lấy cssText của rule và gán vào kết quả.

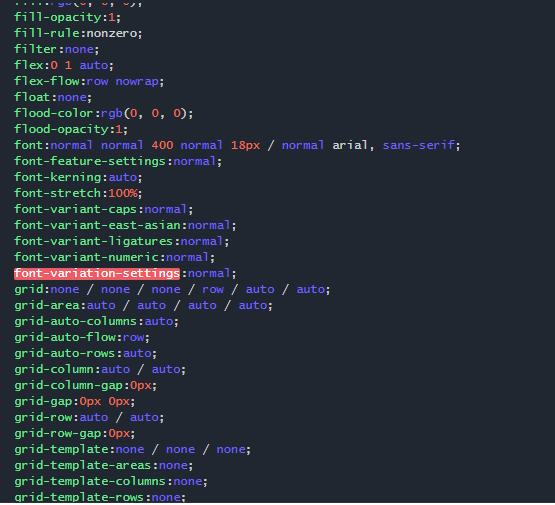


- function readStyles: sử dụng hàm đệ quy để lọc qua tất cả các node con của element truy xuất toàn bộ mã CSS.

***3.3.2. getComputedStyle***

**-** getComputedStyle là một function cho phép lấy ra tất cả các style cho element. Và các style này là các style hiển thị thực tế (actual showing). Ví dụ như style background, một element có thể có nhiều rule quy định style background cho nó, nhưng getComputedStyle chỉ lấy ra một style duy nhất mà element đang sử dụng để hiển thị, các style đã bị ghi đè sẽ không được lấy ra.

**-** Tuy nhiên, getComputedStyle lại trả về một bộ tất cả các style mà CSS có, nghĩa là với những style mà element không sử dụng cũng được trả về với giá trị default.



*Kết quả mà getComputedStyle trả về*

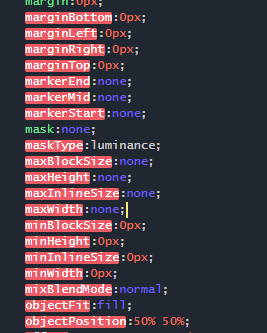
- Để có một bộ CSS ngắn gọn và đúng yêu cầu, chúng ta sẽ loại bỏ các giá trị default thông qua ***function getStylesWithoutDefaults:***



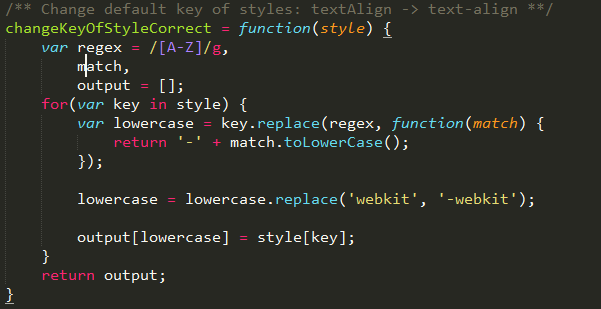
+ Thêm vào document một element và tiến hành getComputedStyle của element mới. Bộ style lấy được chính là bộ style mặc định vì element mới không được mô tả gì cả, chỉ bao gồm thẻ đóng và thẻ mở.

+ Lấy bộ CSS của element chính, lọc qua từng đối tượng trong bộ CSS mặc định và tiến hành so sánh với đối tượng trong bộ CSS của element. Nếu style không có trong bộ CSS của element hoặc 2 style là giống nhau, thì đó là style mặc định. Chúng ta chỉ lưu lại các style không phải mặc định vào mảng output.

- Vấn đề tiếp theo khi sử dụng getComputedStyle: Nếu tên của các style từ 2 từ trở lên thì nó không phải ở dạng chuẩn:



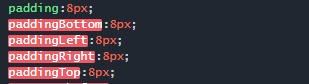
- Để định dạng lại tên đúng chuẩn, sử dụng ***function changeKeyOfStyleCorrect:***



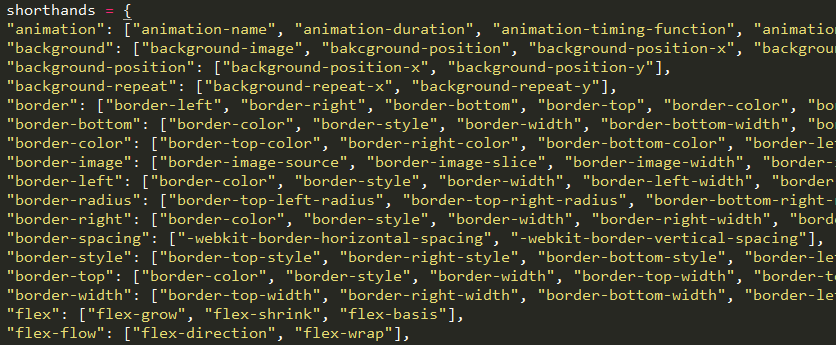
+ Function này tìm các ký tự chữ hoa trong tên của style và thay vào đó là ký tự thường cùng dấu “-” phía trước.

+ Đồng thời thay thế trực tiếp với các style webkit.

- Vấn đề tiếp theo là các style có chứa các style con: Ví dụ như “*margin gồm margin-top, margin-bottom, margin-left, margin-right”*, nếu người dùng sử dụng ít hơn 5 style trên thì getComputedStyle vẫn trả về cả 5 style và *function getStylesWithoutDefaults*không thể remove các trường hợp này:



- Để loại bỏ bớt các style con, sử dụng **function keepOnlyShorthandProperties:**





**+** Function trên tìm kiếm các shorthand trong danh sách style của element, nếu như tồn tại shorthand thì sẽ đưa các con của shorthand vào removeList.

**+** Sau đó sẽ so sánh removeList và danh sách style, nếu như style nào không nằm trong removeList thì mới giữ lại, còn lại sẽ xóa bỏ khỏi danh sách style.

- Sau khi đã loại bỏ các default style và các shorthand, chúng ta sẽ tạo ra một đối tượng JSON chứa style cho element, đối tượng này gồm **id** là **id của element** và **property** chính là danh sách các **style của element:**

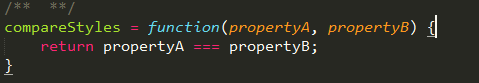


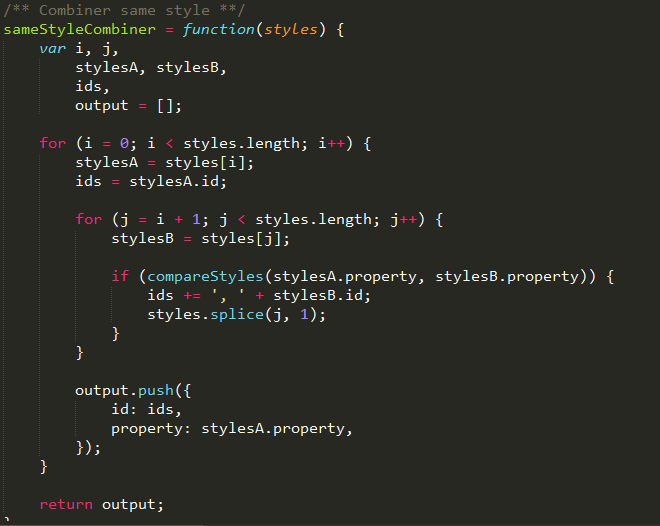
- Và để có thể lọc hết CSS của tất cả element con, chúng ta cần một function đệ quy lặp qua tất cả các element con:



- Sau khi lấy được style của các element, chúng được lưu vào một mảng, với mỗi phần tử là một đối tượng JSON chứ style của từng element con.

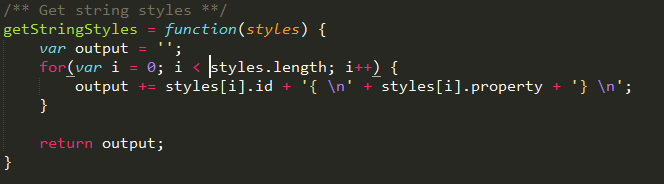
- Thêm một vấn đề nữa xuất hiện tại đây, đó là các element con có thể sử dụng các bộ style giống nhau, như vậy chúng ta cần gom chúng lại để phần code CSS có thể gọn gàng nhất, ***function sameStyleCombiner*** sẽ giải quyết vấn đề này:

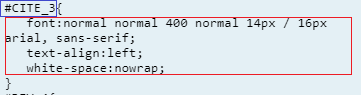




+ Function trên sử dụng 2 vòng lặp lồng nhau để kiểm tra xem có property của 2 element nào trùng nhau hay không. Nếu có thì sẽ loại bỏ 1 trong 2, và phần id sẽ được gom lại thành “*id\_1, id\_2”.*

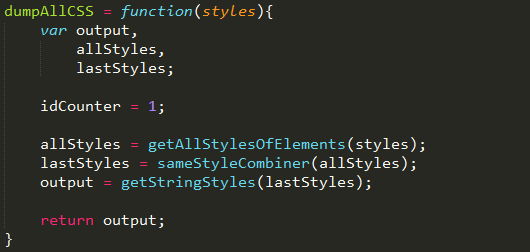
- Công việc cuối cùng để có một bộ style hoàn chỉnh cho element, chúng ta sẽ convert mảng các style sang dạng string, ***function getStringStyles***:





+ Kết quả của function trên là hình ảnh bên dưới với phần **màu xanh** là **id** và phần **màu đỏ** là **property** của đối tượng

- Tất cả những phần nội dung phía trên sẽ được đưa vào ***function dumpAllCSS*** để sử dụng tại content.js:



***✦ So sánh 2 phương pháp***

|  |  |
| --- | --- |
| **styleSheets** | **getComputedStyle** |
| - Lấy đúng định dạng được trình bày trong các file CSS nên không cần chỉnh sửa hay convert nhiều | - Chỉ lấy các property nên cần phải định dạng lại đúng chuẩn và phải thêm id cho element để người dùng hiểu style nào sử dụng cho element nào |
| - Mã CSS lấy ra bị lỗi, cần phải format và beautify mới có thể sử dụng tốt | - Beautify trong quá trình xây dựng chuỗi string cho CSS |
| - Chỉ lấy được các sheet external, nên sẽ không sử dụng được ở nhiều trang | - Có thể sử dụng ở tất cả các trang |
| - Lấy tất cả các CSS liên quan đến đối tượng nên không biết style nào đang được thể hiện lên (nhiều style trùng nhau) | - Chỉ lấy các style đang áp dụng để hiển thị ra nên sẽ tối ưu hơn cho người sử dụng |
| - Thực hiện đơn giản | - Thực hiện phức tạp hơn |
| **Kết luận:** Tuy phức tạp hơn nhưng hiệu quả sử dụng cao hơn nên CodeSteal hiện đang sử dụng phương pháp getComputedStyle. | |

***3.4. Hiển thị kết quả trực tiếp trên webpage***

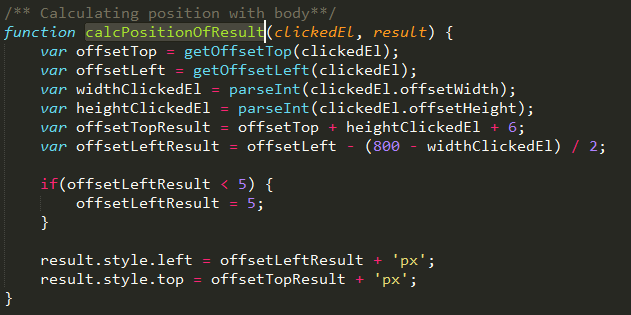
**-** Để người dùng có thể sử dụng thuận tiện nhất, CodeSteal hỗ trợ hiển thị kết quả ngay trên webpage tại vị trí element mà người dùng muốn xem:

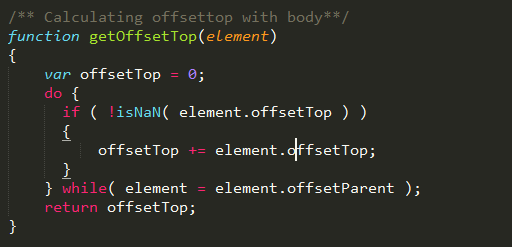


- Đầu tiên sẽ tạo ra một element **result** bằng function createElement, sau đó sẽ thiết lập innerHTML cho **result**.

- Chúng ta gán **id** cho **result** để sau này cần gọi tới khi xóa bỏ nó khỏi document của webpage

- Tính toán vị trí sẽ hiển thị của **result** dựa vào vị trí hiện tại của element bằng **function calcPositionOfResult:**





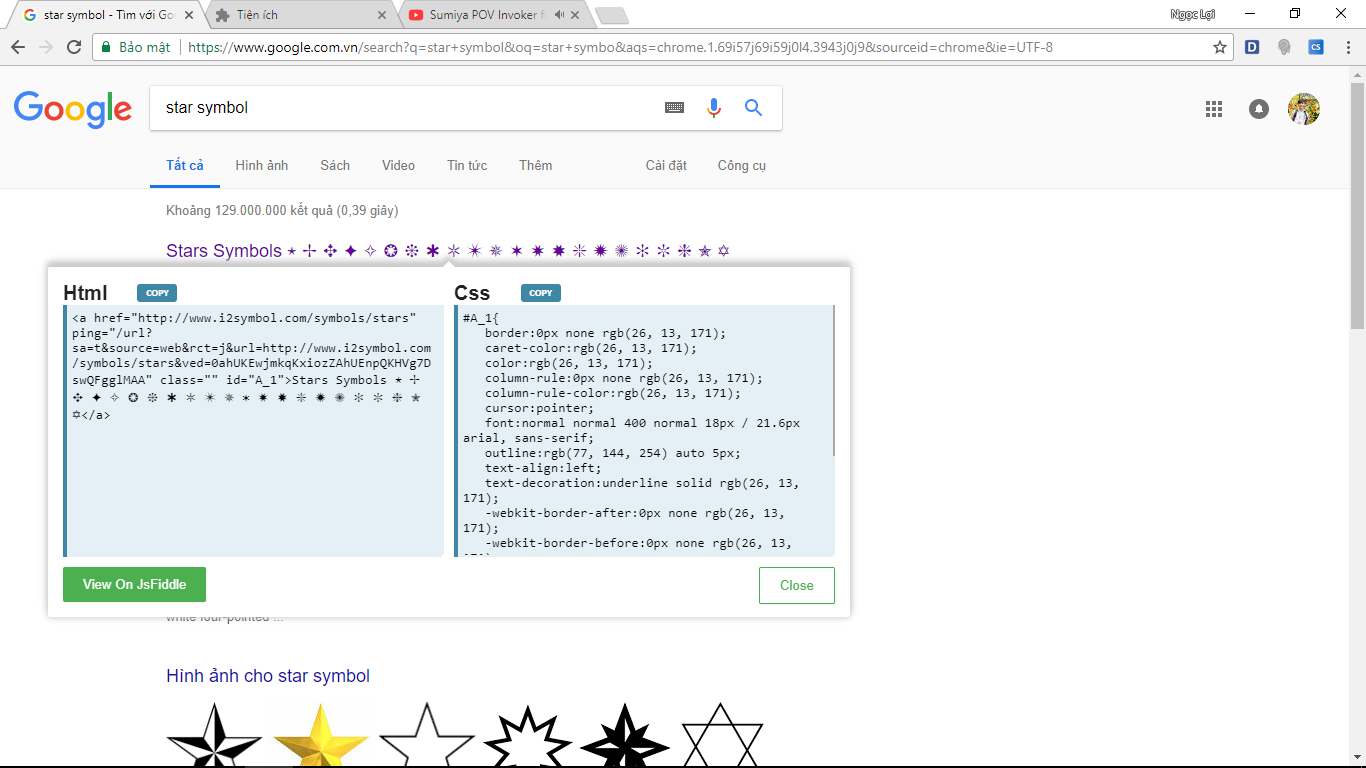
**+** Sử dụng phương thức offsetTop và offsetLeft của một element trong DOM để tìm vị trí hiện tại của nó so với parent. Và tiếp tục tìm vị trí của parent với parent lớn hơn cho tới khi không còn parent nữa. Chúng ta sẽ lấy ra được vị trí của element so với body của webpage.

**+** Tính toán lại vị trí cần đặt result và gán offsetTop và offsetLeft cho **result.**

- Sau khi có được innerHTML, tọa độ và id, chúng ta sẽ thêm **result** vào body của webpage để nó hiển thị cho người dùng có thể xem.

- Cuối cùng là gán nội dung của HTML và CSS đã lấy được ở trên vào textarea của **result**

- Kết quả sẽ là một element với hiển thị như sau:

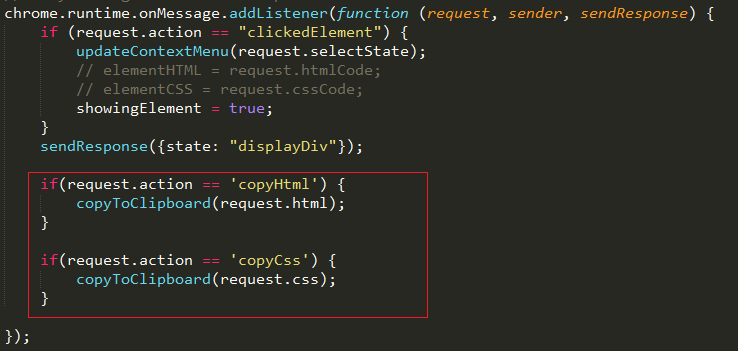


***3.5. Copy to Cliboard***

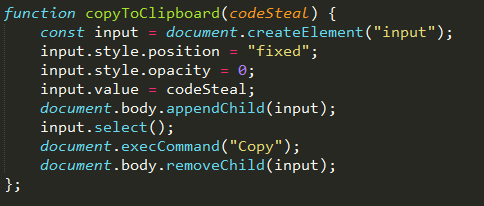
**-** Chrome Extension không hỗ trợ bạn thực hiện code copy trong file content.js, vì vậy để có thể copy được mã HTML và CSS chúng ta cần chuyển nội dung của nó tới background.js và từ đó sẽ tiến hành copy và Clipboard cho người dùng sử dụng:



- Khi có sự kiện click vào button copy, một Message sẽ được gửi tới background với nội dung là mã HTML hoặc CSS.



- Background sẽ nhận message và tiến hành **copyToClipboard** mã code mà nó nhận được:



+ Thêm mới một element vào body với value là mã code mà background vừa nhận được, sao đó sẽ copy vào Clipboard nhờ function **execCommand**

+ Người dùng có thể sử dụng code bằng cách paste hoặc Ctrl + V.

***3.6. View On JsFiddle***

- Để người dùng có thể kiểm tra đối tượng vừa được lấy dễ dàng hơn và nhanh hơn. CodeSteal hỗ trợ chức năng “View On JsFiddle”

- Như đã nói ở trên, JsFiddle là một trang web cho phép người dùng có thể xem trước các mã HTML, CSS và JavaScript của mình. Vì thế việc xem trước trên JsFiddle sẽ cho người dùng biết những mã code mà CodeSteal vừa lấy được sẽ hiển thị như thế nào khi sử dụng.

- Với sự hỗ trợ của API POST từ JsFiddle, cho phép bạng post code tới trang web và running ngay lập tức, việc xem trước code đã trở nên dễ dàng hơn bao giờ hết:



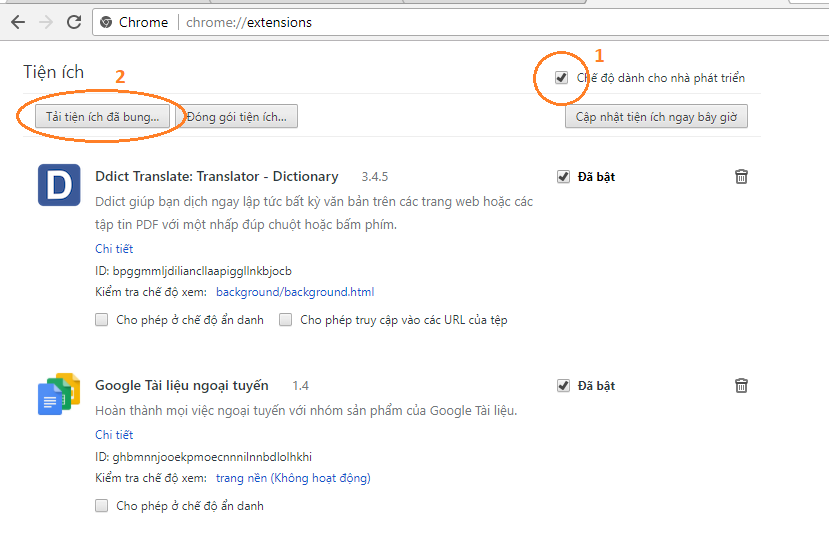
+ Cấu trúc sử dụng “View On JsFiddle” tương đối đơn giản. Bao gồm một **form** với action sẽ gửi tới trang prefill code của JsFiddle và sử dụng method **POST**, và một button **submit**

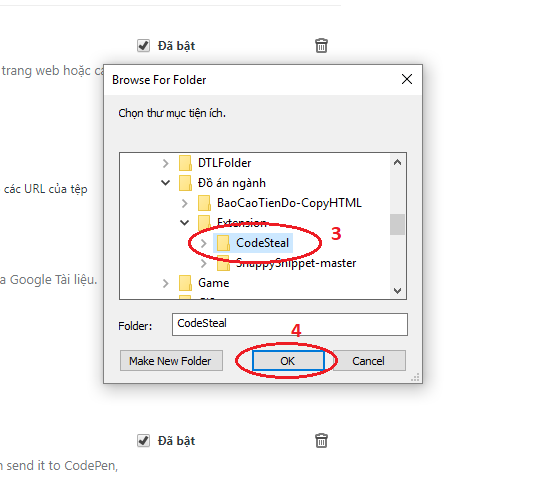
+ Khi có sự kiện click vào button submit, nó sẽ tìm kiếm các phần element có name là “*html*” và “*css*” để lấy value của nó và POST tới JsFiddle.

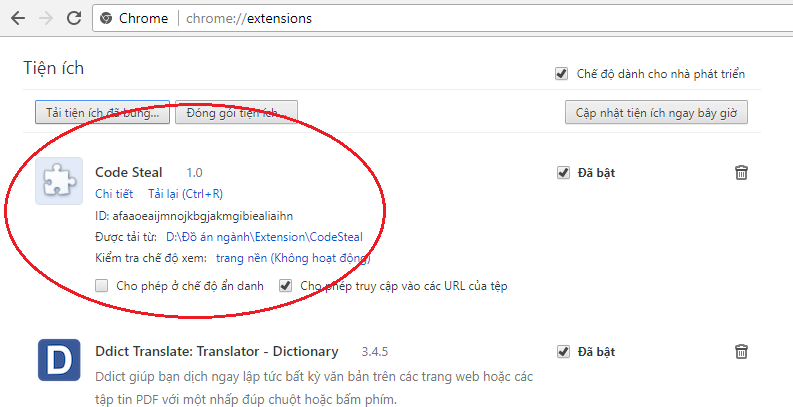
**4. Hướng dẫn sử dụng**

- Cài đặt: vì chưa được publish trên chrome store nên chúng ta sẽ tiến hành cài đặt thủ công:

+ Các bạn vào <chrome://extensions/>

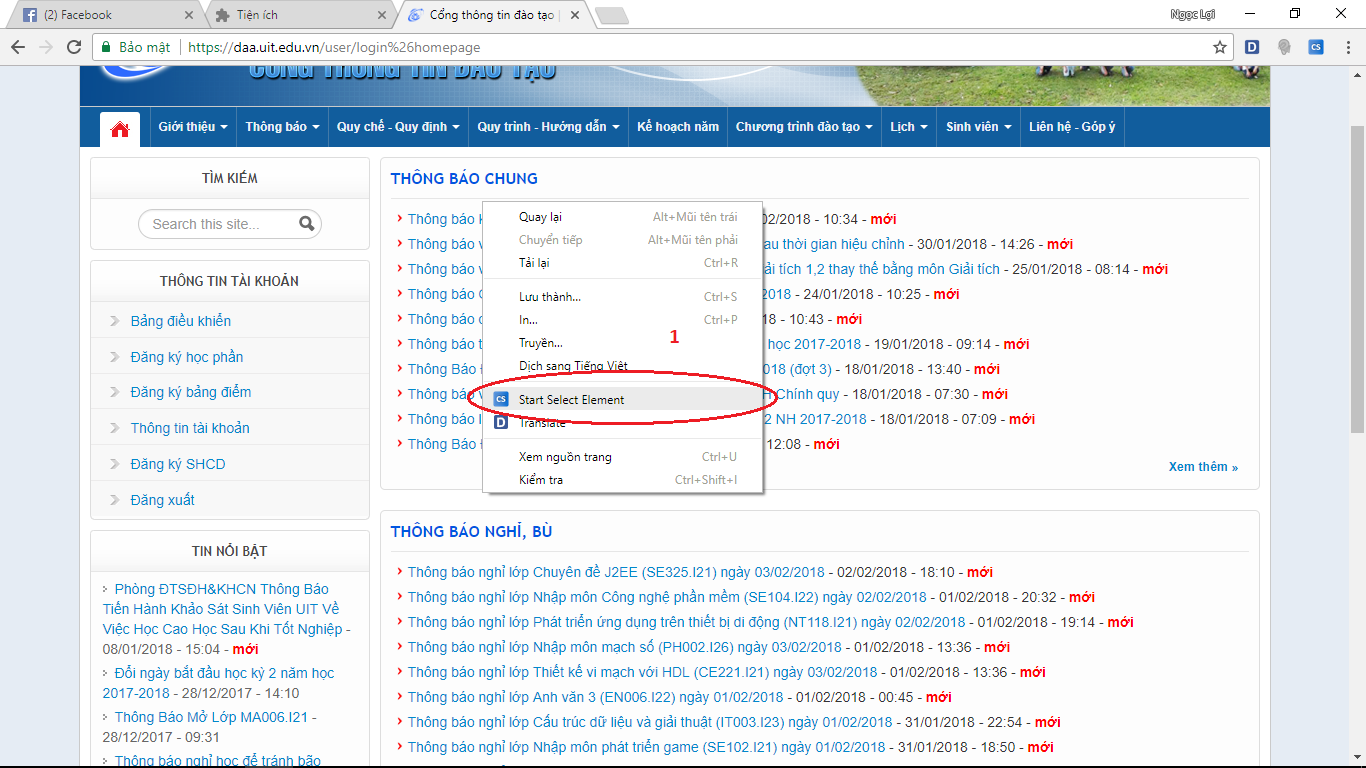




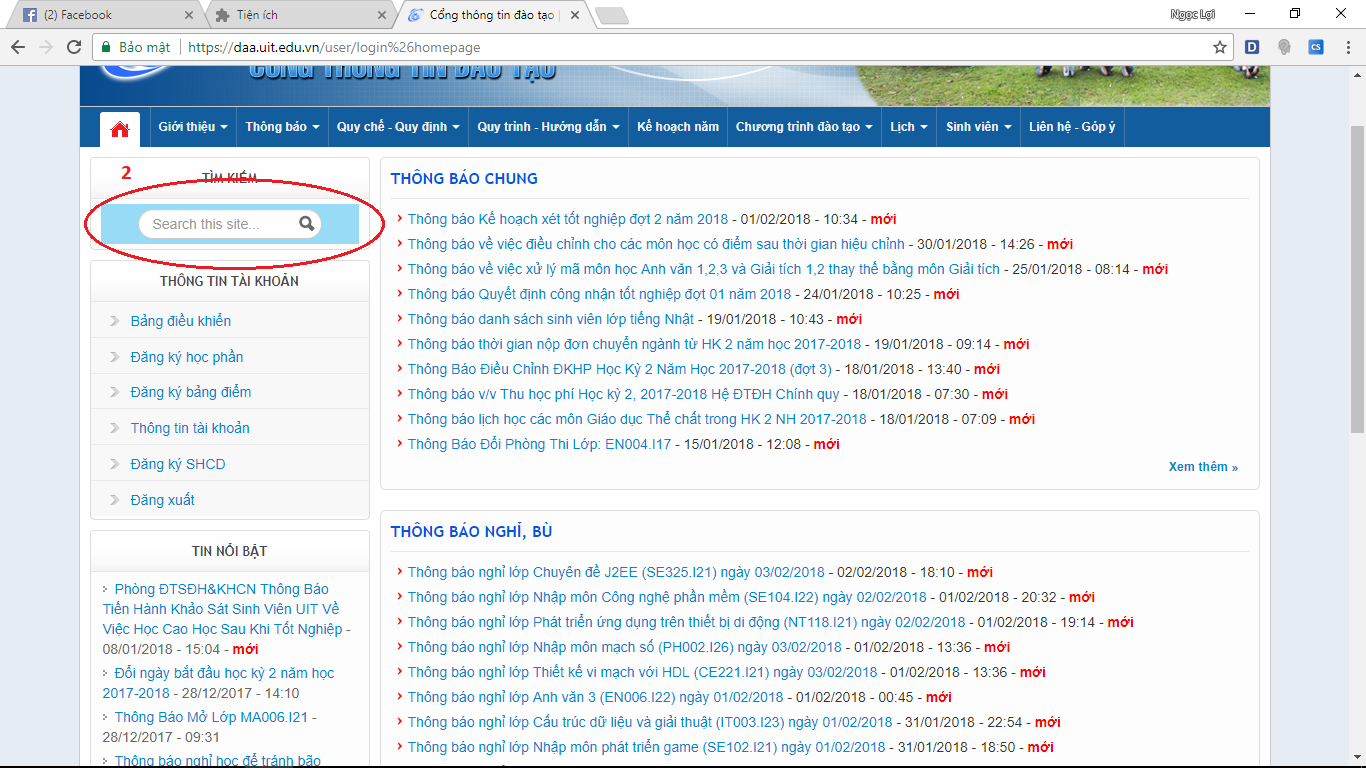


- Sử dụng Code Steal:

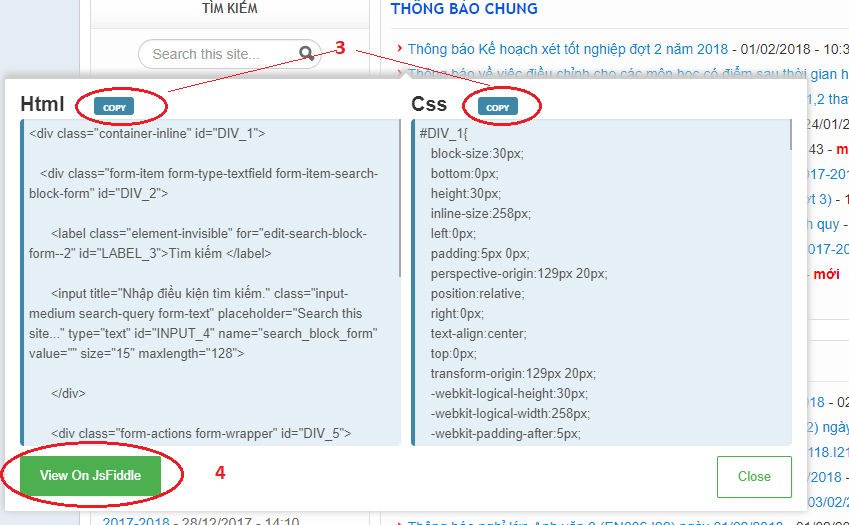
+ Đầu tiên, click chuột phải để hiện ContextMenus, sau đó click vào “Start select Element”



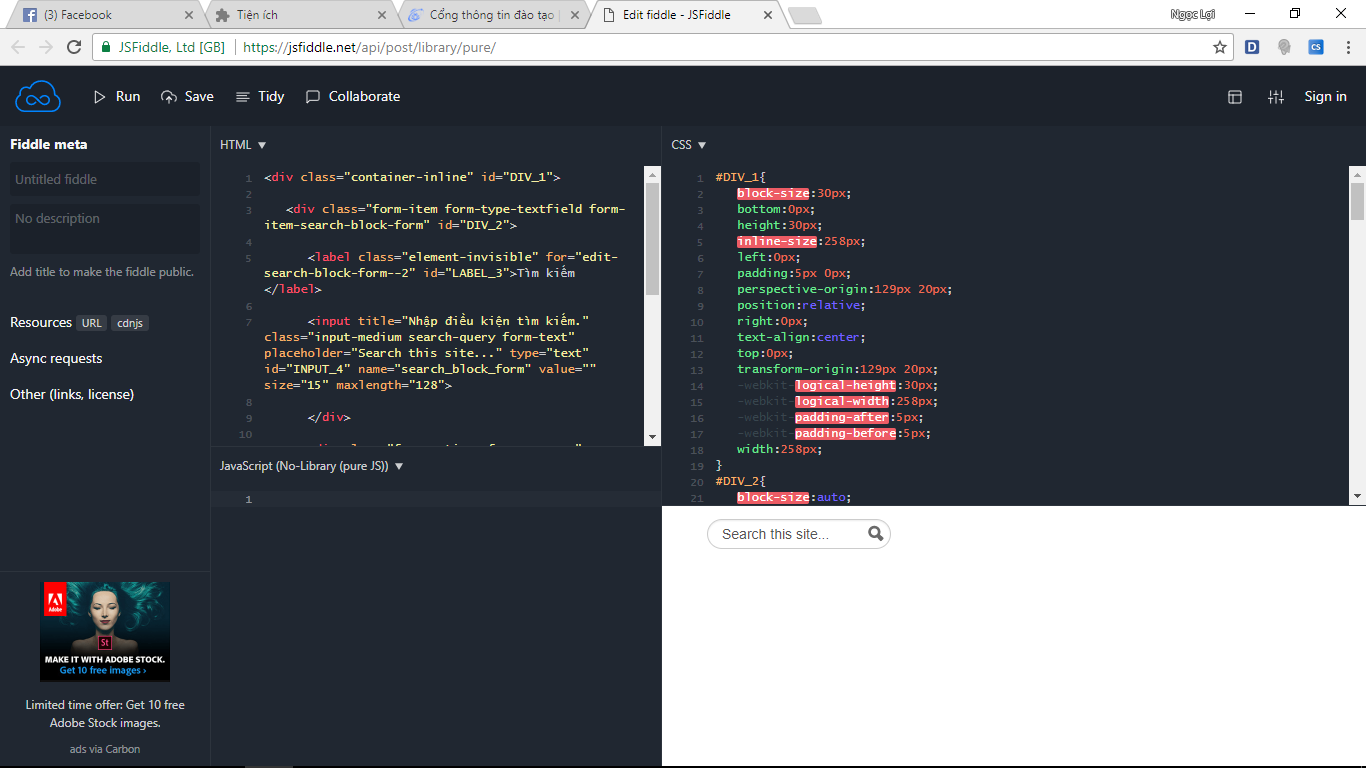
+ Sau đó bắt đầu lựa chọn element và click vào element muốn xem mã code:



+ Một bảng kết quả xuất hiện, có thể thực hiện copy từng mã code để dán vào nếu muốn sử dụng hoặc có thể xem trước trên JsFiddle bằng cách nhấn vào button “View on JsFiddle”



+ Kết quả được hiển thị bên JsFiddle:



- Trên đây là một hướng dẫn đơn giản để sử dụng Code Steal để copy code HTML và CSS của website. Chúc các bạn thành công!

**IV. ĐÁNH GIÁ & HƯỚNG PHÁT TRIỂN**

**1. Đánh giá sản phẩm**

- Công cụ hiện tại đã hỗ trợ việc truy xuất các mã HTML và CSS của các element web một cách tương đối chính xác.

- Các chức năng hiện có:

+ Lựa chọn element một cách trực tiếp

+ Truy xuất mã HTML, CSS

+ Beautify mã HTML, CSS

+ Hiển thị kết quả ngay tại element muốn lấy mã HTML và CSS

+ Copy To Clipboard

+ Xem trước trên JsFiddle

- Các nhược điểm tồn tại:

+ Nhiều Element chưa highlight được trong quá trình chọn element

+ Beautify mã HTML còn chưa xử lý với các element không có tag đóng (như thẻ image, video, …)

+ Dump CSS còn một số style rác chưa thể xử lý

+ Kết quả trả về chưa đạt độ chính xác tối đa.

**2. Hướng phát triển**

- Điều chỉnh lại class highlight để có thể highlight tất cả các element.

- Beautify hiệu quả hơn với mã HTML

- Loại bỏ các style rác và đưa ra các style chính xác nhất sử dụng cho element

- Thêm nhiều tùy chọn copy và lựa chọn element hơn cho người dùng: Chọn kiểu lấy CSS nào? (1 trong 2 phương pháp bên trên), …

- Phát triển trên nhiều brower khác: IE, Cốc cốc, Firefox, …

**3. Lời kết**

Sau hơn 3 tháng thực hiện đề tài, với sự giúp đỡ và định hướng của Thầy Nguyễn Công Hoan, em đã cơ bản xây dựng xong công cụ CodeSteal (Hỗ trợ copy mã HTML và CSS cho lập trình web). Tuy còn nhiều khiếm khuyết trong công cụ nhưng cũng đã phần nào hỗ trợ người dùng trong việc lập trình.

Em xin gửi lời cảm ơn sâu sắc tới Thầy Nguyễn Công Hoan và cũng rất mong nhận được sự hỗ trợ, góp ý của thầy trong quá trình phát triển công cụ sau này.

**V. TÀI LIỆU THAM KHẢO**

<https://developer.chrome.com/extensions>

<http://doc.jsfiddle.net/api/post.html>

<https://stackoverflow.com/questions/18675828/unable-to-get-value-of-margin-property-from-result-getcomputedstyle>

<https://www.w3schools.com/jsref/dom_obj_document.asp>

<https://www.w3schools.com/jsref/dom_obj_all.asp>

<https://www.w3schools.com/jsref/dom_obj_event.asp>

<http://www.javascriptkit.com/domref/cssrule.shtml>

<https://stackoverflow.com/questions/22907735/get-the-computed-style-and-omit-defaults>

Và một số trang web khác…