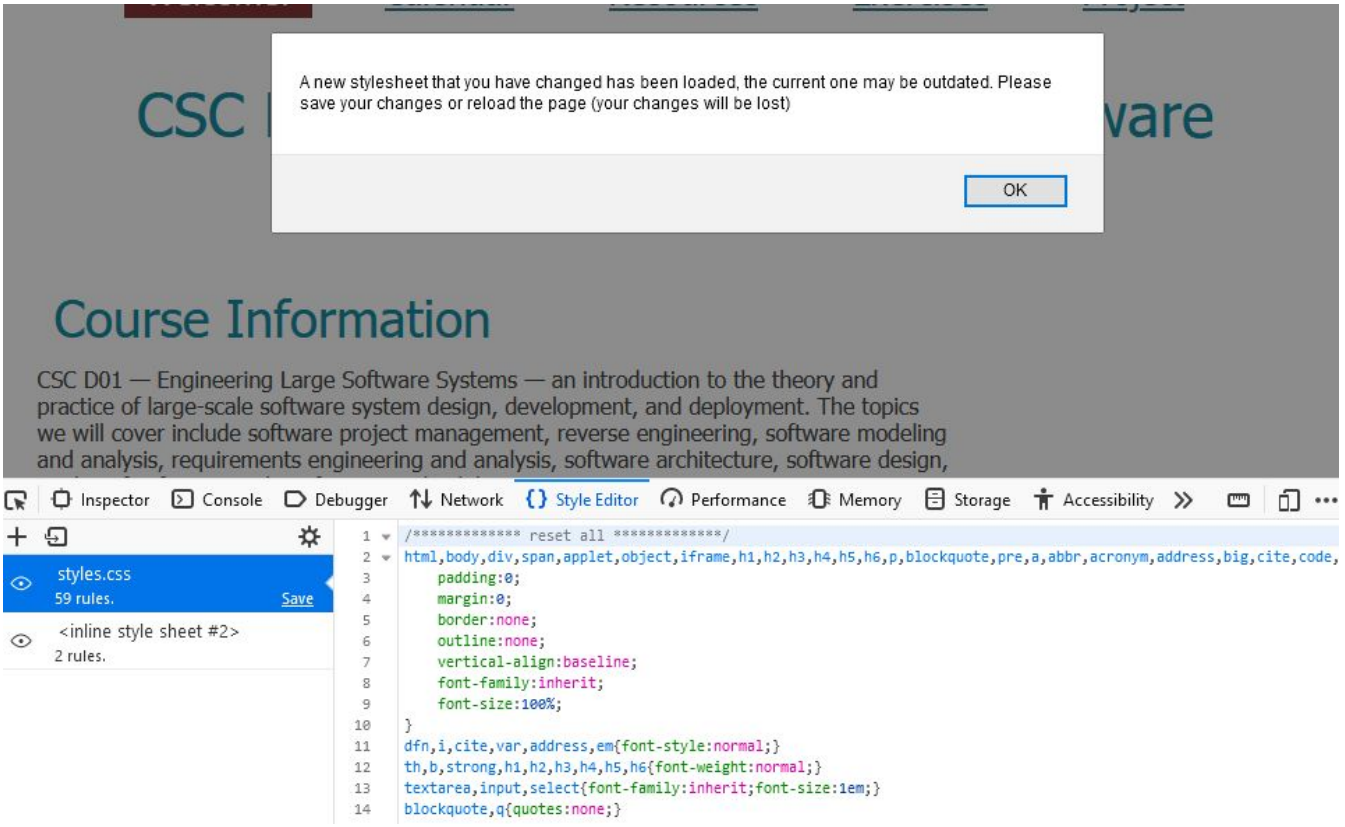
Deliverable 3

Team 29: GoodbyeWorld!

Thasitathan Sivakumaran | Ajay Rajendran | Jason Ku | Yathan Vidyananthan

Feature 1: https://bugzilla.mozilla.org/show_bug.cgi?id=1500979
Name: Track Changes - warn the user when a stylesheet is replaced with a popup

Detailed description: In the Style Editor found within Mozilla DevTools, stylesheets are shown to the user that allow them to modify the current page however they want based on the stylesheet/css that is shown in the Style Editor. The Style Editor allows for changes, as well as a “save” feature where the user can export the stylesheet so that it may be used in the future again. Sometimes, the existing stylesheet may be overridden by a “newer” one. For example, the page may have a hot reload and the stylesheet the user was working on could be outdated since there is now a more recent one. In this case, there is no problem, except that the user may have wanted to save their changes before losing them by refreshing. Of course, this is not good behaviour, as the user has now completely lost any changes they have done since the file has been overwritten. In order to fix this, it isn’t possible to disable stylesheets from being overwritten, since some pages need that feature. Instead, it would be better to prompt the user if they want to save the affected stylesheets and if they do, they must do it now. Otherwise, the sheet will be overwritten and all changes from the user are lost.



Parts of code: The changes in the code will be as follows:

Server side

- Changes actor (devtools/server/actors/changes.js)
- Browser-Context Actor(devtools/server/actors/targets/browsing-context.js)
- StylesheetActor (devtools/server/actors/stylesheets.js)

Client side

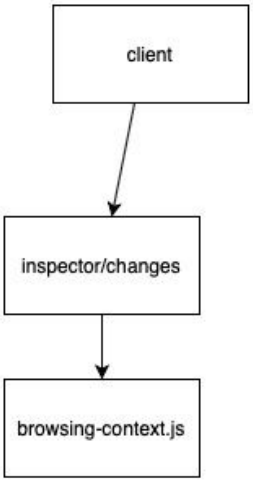
- Changeview.js (devtools/client/inspector/changes/ChangeView.js)

Designs for feature:

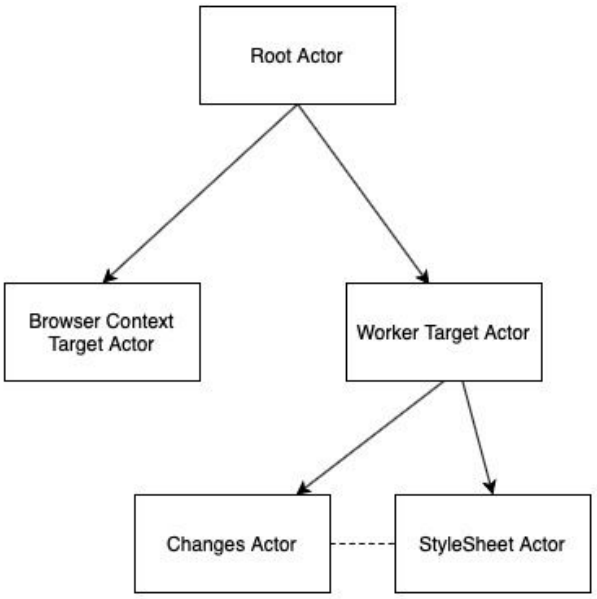
In order to implement this feature, we need to catch the “stylesheet-added” event that occurs from the browsing-context target actor in the changes actor by adding a listener. Once that listener is implemented, we have to cross reference href from the StyleSheetActor against all the hrefs in “this.changes” from the Change actor. The hrefs stored in “this.changes” are the source hrefs so when we cross reference and get a match, that means our current href has reverted back to an href that was previously saved. If this happens, we have to notify the user by using a warning in the front-end. We can implement this notification in the ChangesView.js file under client. In this file, we add a listener to listen for the event we fire from Changes actor, and when that occurs, we provide a warning stating that the changes that have happened so far may have been lost and have reverted to an old version of the href.

Uml diagrams:

Frontend UML



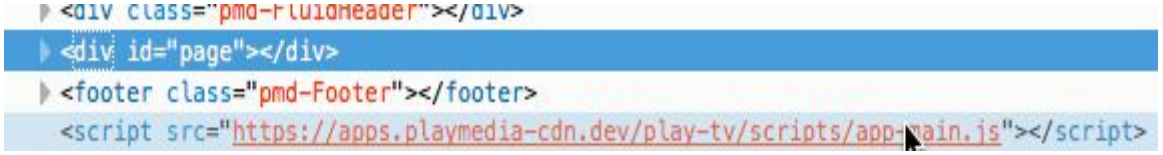
Backend UML



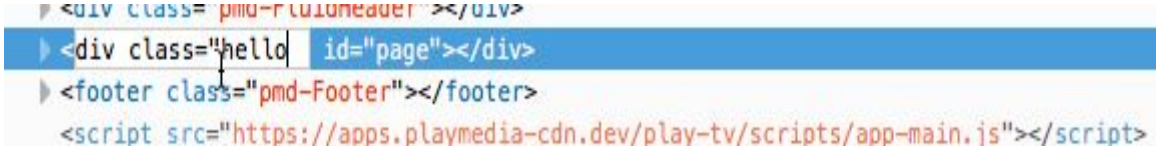
Feature 2: https://bugzilla.mozilla.org/show_bug.cgi?id=1221183
Name: Double-click on a tag shouldn't allow you to edit tag but also add whatever you want like classes or id

Detailed description: When using the inspector tool within the DevTools, the user can manipulate the existing HTML code on the page. Currently, if the user adds an attribute such as "class" or "id" to an existing "div" tag, the newly added class/id does not save. This is also true for tags other than "div". This is not good behaviour because some users may want to add attributes to existing tags within the browser. Additionally, the inspector tool is meant to be flexible and allow for changing any parts of the existing code, regardless of its functionality. If this bug is not fixed, it causes confusion among users who expect the inspector to add the attribute to the tag when in fact, it does not. This could lead to unexpected behaviour and issues that are hard to debug because the behaviour is not expected. Below is an example of the browser's inspector element not adding the attribute to the tag.

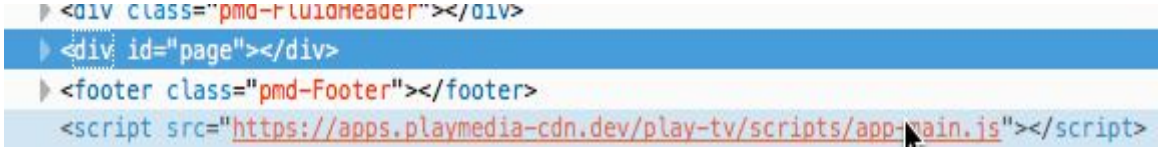
Before adding a "class" attribute:



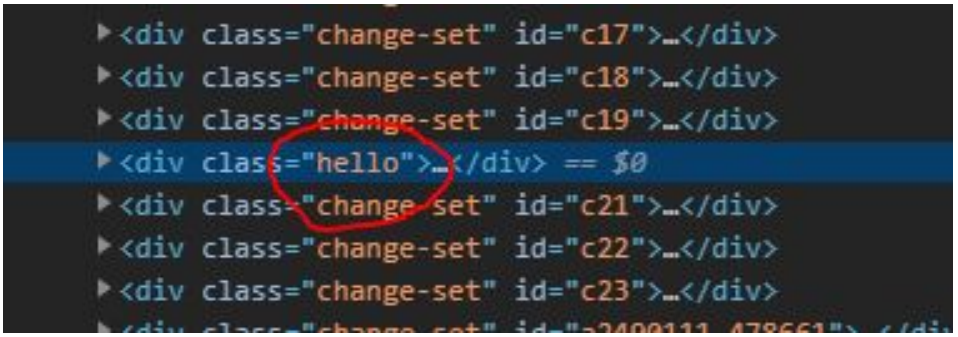
Adding "class" attribute to "div" tag:



After pressing enter (changes not saved):



Expected Result:



Parts of code:

Client side

- element-editor.js (devtools/client/inspector/markup/views/element-editor.js)

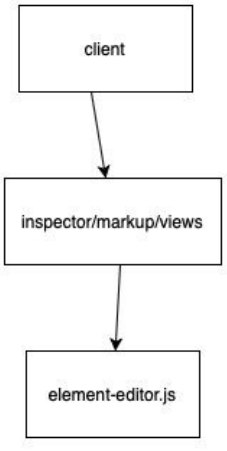
```
Origin: resource://gre/modules/devtools/server/protocol.js:907

Error: Could not change node's tagName to div id="felafel"
Stack trace:
WalkerActor<.editTagName<@resource://gre/modules/commonjs/toolkit/loader.js ->
resource://gre/modules/devtools/server/actors/inspector.js:2693:29
actorProto/</handler@resource://gre/modules/commonjs/toolkit/loader.js ->
resource://gre/modules/devtools/server/protocol.js:1013:19
DSC_onPacket@resource://gre/modules/commonjs/toolkit/loader.js ->
resource://gre/modules/devtools/server/main.js:1596:15
LocalDebuggerTransport.prototype.send/<@resource://gre/modules/commonjs/toolkit/loader.js ->
resource://gre/modules/devtools/transport/transport.js:569:11
makeInfallible/<@resource://gre/modules/commonjs/toolkit/loader.js ->
resource://gre/modules/devtools/DevToolsUtils.js:86:14
makeInfallible/<@resource://gre/modules/commonjs/toolkit/loader.js ->
resource://gre/modules/devtools/DevToolsUtils.js:86:14
```

Designs for feature: The purpose of this feature is so that users are able to add attributes to tags which have no attributes. Currently, if we have a div tag which has no attributes (ex attributes: classes, id) then we are not able to add any extra attributes. We are able to edit the tag by double clicking it but if we attempt to append an attribute beside the tagname, it will just undo back to just the tag name once you finish your edit. Also right now in the backend when you attempt to add an attribute while editing the tag name, there is an error that is thrown(picture above). This error is because the whole user input is treated as a tag name which is an invalid tag name. This error is suppressed so that it is not displayed to the user. So when we make our modifications this error should not trigger if the user provides a valid tag name and a set of valid attributes.

In order to add the functionality to add attributes to tags with no attributes, we need to make a modification to the element-editor.js file. In this file there is a method called onTagEdit which is run when there is a tag being edited. We need to work from this method because when there is a tag with no attributes the user can only edit the tag. So we must add functionality to this method to take the user input and split it into a list with tagname and the extra attributes if given. We would first make the change to the tag name using the existing code but we will add extra functionality to update the extra attributes. In order to update the attributes we will reuse the existing method, applyAttributes. The method, applyAttributes, will take the new attributes and add it to the html element.

Uml diagrams:



Feature we selected and explain your decision (including detailed implementation plans):

The feature we selected was https://bugzilla.mozilla.org/show_bug.cgi?id=1500979 (the first one). This title of the feature is: *Track Changes - warn the user when a stylesheet is replaced*. The reason we selected this feature to work on is because our group believes that it is important to keep the user up to date with any system changes. It is important to be transparent about the system status to the user in order to minimize any confusion, as this will save a lot of debugging time for both the user and developers. Additionally, our group believes that this feature would give us in-depth knowledge of the existing code, exposing us to a lot of code that will allow us to gain experience with open-source coding.

(*the following is referenced from the explanation at the top) The detailed implementation details is as follows, we need to catch the “stylesheet-added” event that occurs from the browsing-context target actor in the changes actor by adding a listener. Once that listener is implemented, we have to cross reference href from the StyleSheetActor against all the hrefs in “this.changes” from the Change actor. The hrefs stored in “this.changes” are the source hrefs so when we cross reference and get a match, that means our current href has reverted back to an href that was previously saved. If this happens, we have to notify the user by using a warning in the front-end. We can implement this notification in the ChangesView.js file under client. In this file, we add a listener to listen for the event we fire from Changes actor, and when that occurs, we provide a warning stating that the changes that have happened so far may have been lost and have reverted to an old version of the href.

Acceptance tests:

Pre-requisites:

- Github
- Mozilla DevTools

Setup:

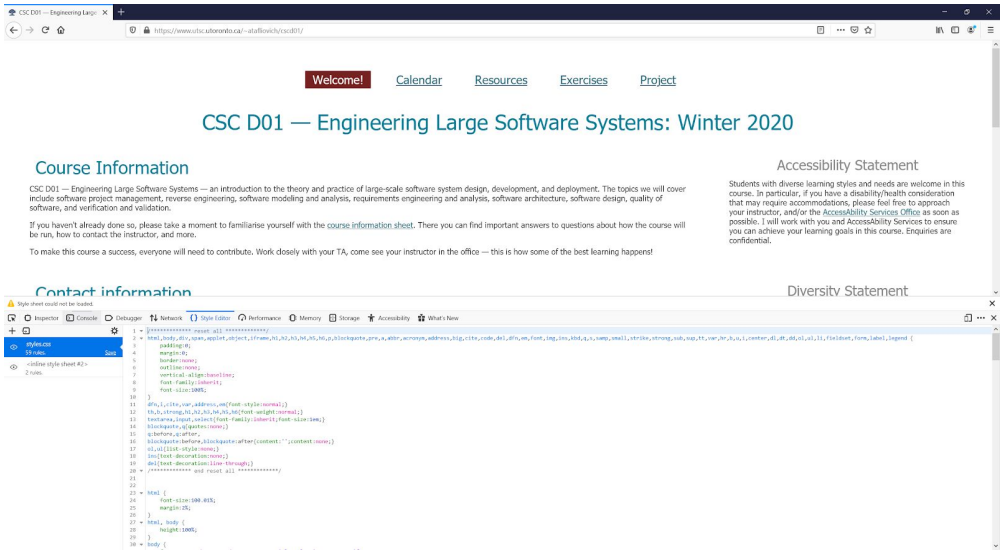
1. git clone https://github.com/thasitathan/gecko-dev.git
2. cd mozilla-central
3. ./mach build (takes up to 1.5 hours if never done before, otherwise run ./mach build faster)
4. ./mach run
5. Download the add-on for hot-reloading: https://addons.mozilla.org/en-CA/firefox/addon/css-reloader/

The Mozilla browser should now be opened and ready for testing

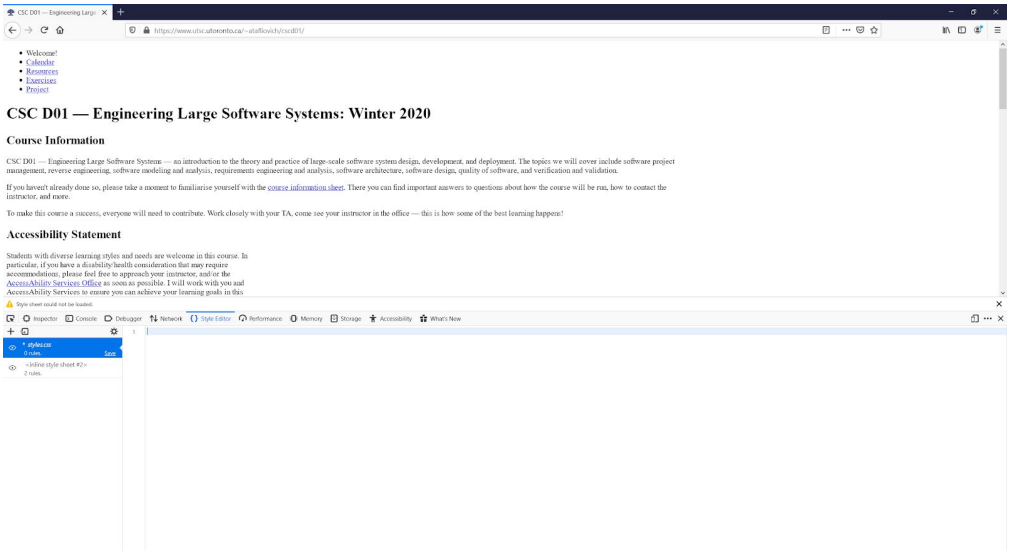
Test Cases:

Warning appears after new stylesheet is loaded

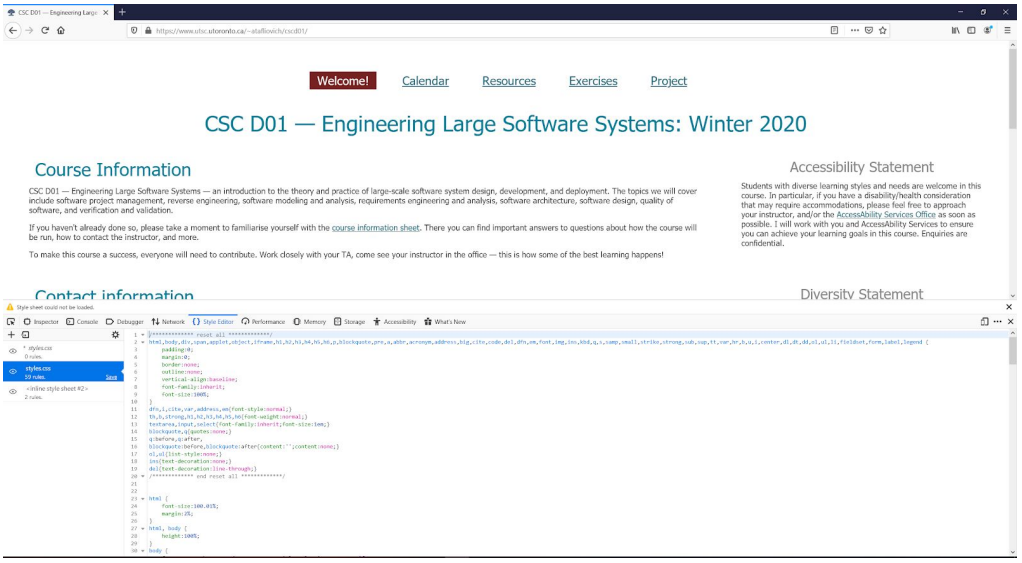
1. Go to https://www.utsc.utoronto.ca/~atafliovich/cscd01/index.html
2. Right click -> inspect element -> Style Editor



3. Click on styles.css, select all the text and delete all of it (should be visual change on site)

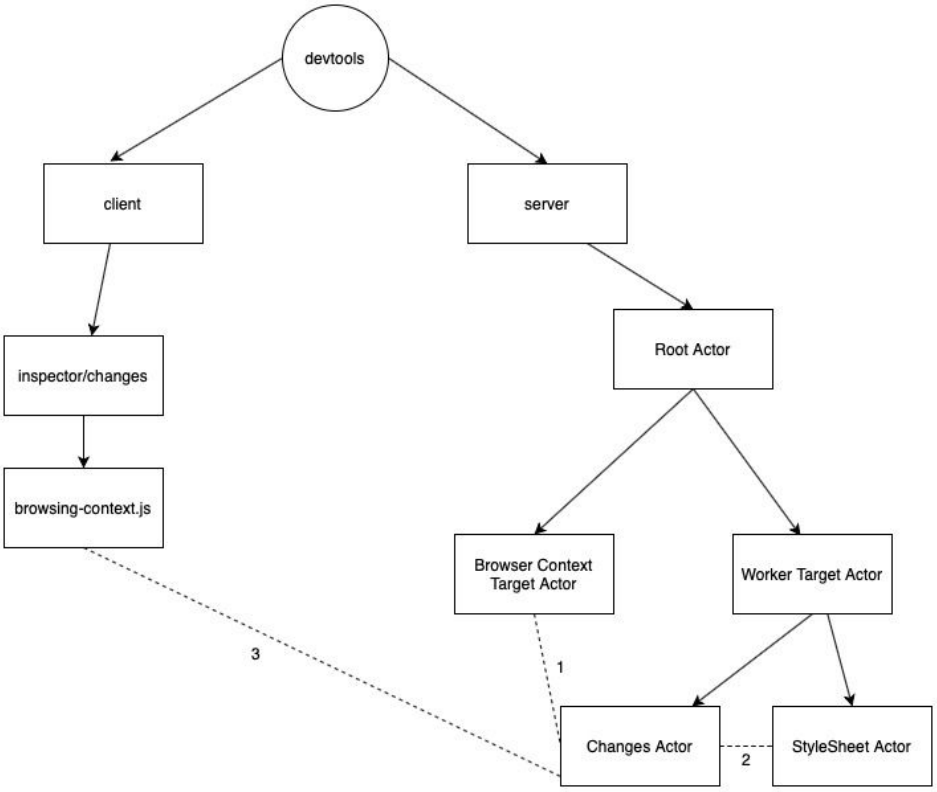


4. Right click on the page and click reload css (this was part of the extension you downloaded in step 5 in setup)



   - In the style editor there should be a new css file and an asterisk(*) beside the css file that you edited in step 3
   - The page should have gone back to how it originally looked
5. Verify that an alert was prompted stating that there was a new stylesheet that was picked up and being used because of a hot reload.

Higher-level UML Diagram Architecture Document



The above is the overall architecture for this feature in devtools. The files that will be modified on the client front end side is browsing-context.js. The server side contents being changed are the actors, which include browser context actor, changes actor, and stylesheet actor. The browser context actor is a target actor exposed directly by the root actor upon loading up devtools, and the changes actor and stylesheet actor are target sourced actors exposed by the worker target actor. The following explains the connections between files (labeled on diagrams with numbers)

(1) This is the listener between the observer (changes actor) and the observable (Browser Context Target Actor). Observer listens for the stylesheet added event
(2) This is the cross reference between this.changes href from changes actor to the current href in stylesheet actor
(3) This is the listener between the observer (browsing-context.js) and the observable (Changes Actor). Observer listens for the stylesheet changed event thrown by changes actor and create warning

The code base has Observer design patterns which is a software design pattern in which an object maintains a list of its dependents called observers, and notifies them when a state changes usually by calling one of their methods. We are implementing two listeners, changes actors and browsing context actors, they are both observers. The browsing context the observable changes is the actor it is listening for the events we will create. The listener for changes actor the observable is browser context target actor and it listens for the stylesheet added event in the target actor.