YouDanMu Sprint 2
Defects Log

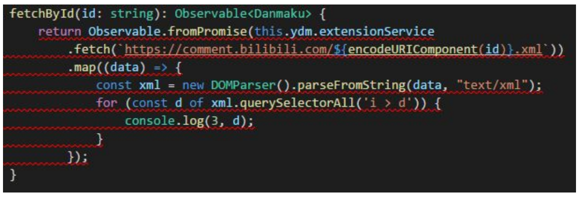
Team Member: Naiwei Zheng, Yucong Ma,
Yibo Gou, Ge Yan,
Jiaqi Zhu

A. Design Inspection Defects Log

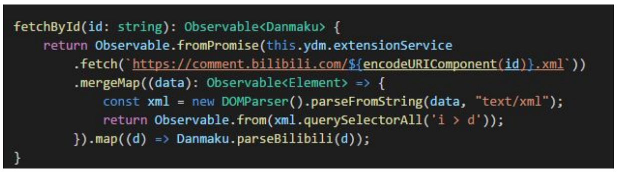
Product	YouDanMu Design Inspection		
Date	03/01/17		
Author	Ge Yan		
Moderator	Naiwei Zheng		
Inspectors	Jiaqi Zhu, Yibo Gou, Naiwei Zheng		
Recorders	Yucong Ma		
Defect#	Description	Severity	How corrected.
1	It was originally planned that our render module is isolated completely from the timeline module and use an event emitting module to communicate with each other. However, js is weak with event emission system and it is work heavy and time consuming to implement our own system.	3	We fix the issue by using rejx which is small and easy to implement to our existing code. It also works well when hooking up different events to our functions.
2.	Danmaku Source module and Video Source module were originally planned to be sharing with one of the same support interface however the video source module is far more complicated than we planned, by sharing the same interface would be a waste for Danmaku source or the interface is not enough for Video Source module.	3	Separated two modules completely and use two interfaces for inheritance. Used another Danmaku module to connect timeline between Danmaku and Video Source.
3.	The Video Source module is heavily based on the actual video website API which made it really hard to implement test cases and test stubs when come to testing	3	Written a stub as a video site that has the API to be accepted by our video source module. Fake our own video site that emit events by our manipulations. Implemented to be used in automatic testing

B. Code Inspection Defects Log

Product	Damaku Fetch module inspection		
Date	03/01/17		
Author	Yibo Gou		
Moderator	Naiwei Zheng		
Inspectors	Yucong Ma, Yibo Gou, Ge Yan		
Recorders	Yucong Ma		
Defect#	Description	Severity	How corrected.
1	When fetching Danmaku with Bilibili Web API from the injected script running on the YouTube web page, Chrome browser complains about the cross-domain reference policy violation. Thus failed to load any external file.	3	We have to redirect the HTTP call from the injected script to our extension's background page where it has no cross-domain reference policy. Then redirect the response back to the content page where data is processed.
2.	We want to iterate on the parsed XML object element query result. However, the returned value is of type NodeList, where no Array-like functions is applicable.	1	By looking up the implementation of NodeList, we found that it implemented Iterable interface, where we can directly construct array like stream with RxJS' `Observable.from` constructor. Then we can apply RxJS operators on the resulting stream.
3.	We expect the module to return a stream of Danmaku as an Observable RxJS object, but the `map` function produced a stream of array of Danmaku. It may result in type mismatch.	2	We need to use the `mergeMap` RxJS operator to flatten the produced array of Danmaku into a single stream of Danmaku.



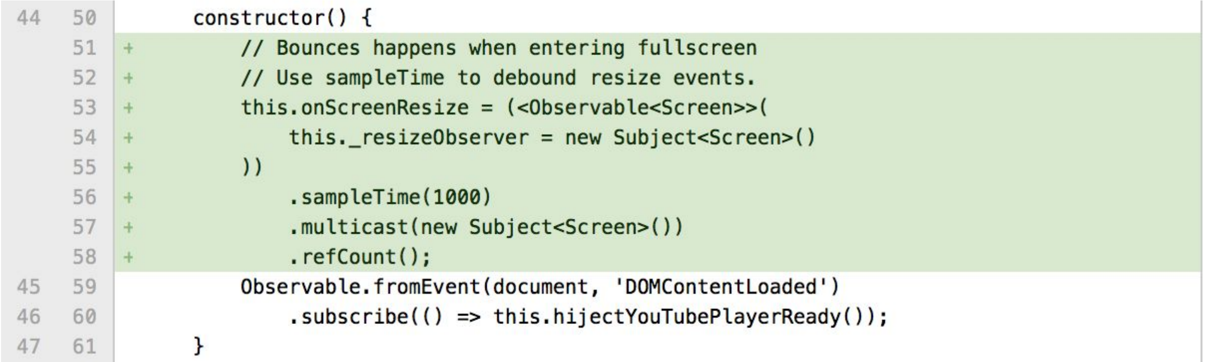
```
fetchById(id: string): Observable<Danmaku> {
    return Observable.fromPromise(this.ydm.extensionService
        .fetch(`https://comment.bilibili.com/${encodeURIComponent(id)}.xml`))
        .map((data) => {
            const xml = new DOMParser().parseFromString(data, "text/xml");
            for (const d of xml.querySelectorAll('i > d')) {
                console.log(3, d);
            }
        });
}
```



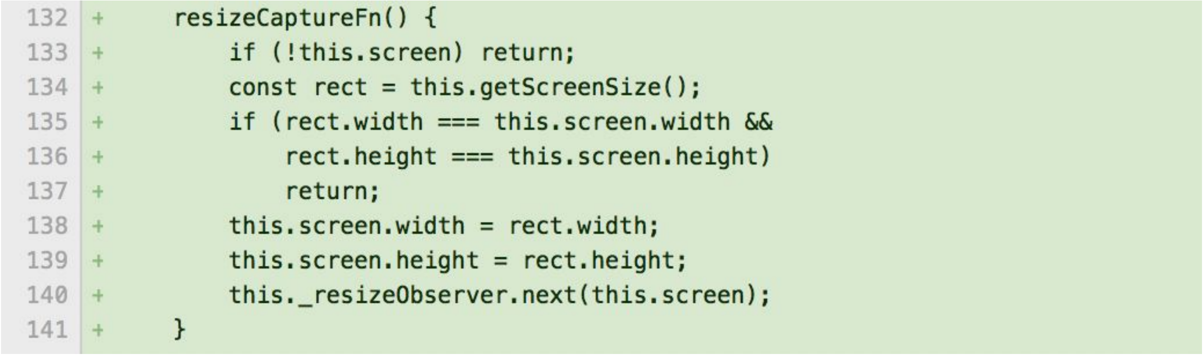
```
fetchById(id: string): Observable<Danmaku> {
    return Observable.fromPromise(this.ydm.extensionService
        .fetch(`https://comment.bilibili.com/${encodeURIComponent(id)}.xml`))
        .mergeMap((data): Observable<Element> => {
            const xml = new DOMParser().parseFromString(data, "text/xml");
            return Observable.from(xml.querySelectorAll('i > d'));
        }).map((d) => Danmaku.parseBilibili(d));
}
```

Product	Youtube Integration module inspection		
Date	03/01/17		
Author	Naiwei Zheng		
Moderator	Naiwei Zheng		
Inspectors	Yibo Gou		
Recorders	Jiaqi Zhu		
Defect#	Description	Severity	How corrected.
1	Some of the event name are hard coded that needed to hook to the website events, it will be hard to find out the issue if the API of YouTube has changed.	3	Define all constant variables in one place. When these names are to be used. used the variables instead of hard coding the names of the events.
2.	Ads plays and videos plays are handled at the same place. This caused much confusion when come to editing to code to add more functions as some functions should not be run when the ads played. However, some functions needs to be run only when the ads played.	3	Separate ads play and video play functions and distinguish the event emission between these two play events.
3.	To capture the screen size change event, we listened on the size button clicked event, where it fires before the screen size is actually changed.	2	Use asynchronous busy waiting to capture the moment where the screen size is changed.
4.	When entering to fullscreen, the screen size is changed several times, when using the above method, we can only capture the first change, and the ending screen size may be different.	2	We rewrite the screen size change detection mechanism. We use a slow clock to busy loop a detector to constantly check for screen size change. We produce a RxJS stream of screen size change events, then we apply the RxJS `sampleTime` operator to “debounce” a fast changing sequence of

			events and only take the last event in a short duration of time.
5.	Sometimes ads play before video starts, and thus the play event is not fired. However the video is actually buffering, and we want to wrap it with a “cue” event to indicated that video is about to play and meta info is available.	1	We listen on “buffering” event to managed the state of video info initialization, and wrap it inside our “cue” event.
6.	To be able to destroy rendering screen, we need an additional “unplay” event to indicate video meta info becomes invalid.	2	We listen on the “unstarted” event that indicate video is stopped, and wrap it inside our “unplay” event.



```
    constructor() {
+       // Bounces happens when entering fullscreen
+       // Use sampleTime to debound resize events.
+       this.onScreenResize = (<Observable<Screen>>(
+           this._resizeObserver = new Subject<Screen>()
+       ))
+           .sampleTime(1000)
+           .multicast(new Subject<Screen>())
+           .refCount();
        Observable.fromEvent(document, 'DOMContentLoaded')
            .subscribe(() => this.hijectYouTubePlayerReady());
    }
```



```
+   resizeCaptureFn() {
+       if (!this.screen) return;
+       const rect = this.getScreenSize();
+       if (rect.width === this.screen.width &&
+           rect.height === this.screen.height)
+           return;
+       this.screen.width = rect.width;
+       this.screen.height = rect.height;
+       this._resizeObserver.next(this.screen);
+   }
```

Product	Chrome Extension module Inspection.		
Date	03/01/17		
Author	Yucong Ma		
Moderator	Naiwei Zheng		
Inspectors	Jiaqi Zhu, Ge Yan		
Recorders	Jiaqi Zhu		
Defect#	Description	Severity	How corrected.
1	When implementing the injected script <-> background page function call redirection channel, we found out that no direct method is available to communicate between the two environments.	3	We found out that the content script can listen on events on the page element, and the injected script can dispatch custom events. Moreover, there’s `chrome.sendMessage` interface on content script that can be used to communicate with background script. Thus we can use content script as a proxy to listen on injected script’s event and forward it to background page with Chrome extension API, and get back response, and create custom event again and dispatch to injected script again.
2.	The user settings are not synced when they log in on another device, the setting in the new device overwrite the original settings.	2	When the current settings are not in default status, promote user to select between “Use saved settings” or “Save and apply current settings”.

Product	Render module inspection.		
Date	03/01/17		
Author	Yibo Gou		
Moderator	Naiwei Zheng		
Inspectors	Jiaqi Zhu, Naiwei Zheng		
Recorders	Naiwei Zheng		
Defect#	Description	Severity	How corrected.
1	Unicode have special character(subscript,superscript) which will exceed the screen that cause problem for the calculation of the rendering module.	3	Filter out those special characters which are trival for normal users.
2.	When user change the overall font size of Danmaku. The dynamicly changing sometimes will cause overlap of Danmaku.	2	Recalculate and reset the default denstiy of Danmaku when user adjust the font size.’’
3.	When user change the overall font family, some characters cannot be matched in new font family.	1	Make those untranslatable characters to question mark.

Product	Bilibili Danmaku paser module inspection.		
Date	03/01/17		
Author	Jiaqi Zhu		
Moderator	Naiwei Zheng		
Inspectors	Naiwei Zheng, Ge Yan		
Recorders	Yucong Ma		
Defect#	Description	Severity	How corrected.
1	Danmaku sources will return GBK encoding which is not acceptable for our format will cuase error.	3	Check the encoding before pasing
2	During parsing process, Danmaku with same author name will be consider as the same author. It can have different user with same user name.		Parsing Danmaku based on user ID instead of user name.

C. Unit Test Defects Log

We do use automation to assist our testing. Since our source code is written in TypeScript, we want to benefits from the type checking and IDE static code analysis by writing our test cases also in TypeScript. Thus we need to compile our test scripts along with our source code. To do this we used Gulp.js, a stream-interfaced asynchronous JavaScript task runner; node-typescript, a package that stream the source files to the TypeScript compiler, and several other toolings.

We choose Jasmine.js as our test framework, with assistance with Chai.js as an environment independent assertion library, Mocha.js as the task runner in browser environment, and coverall.js as the code coverage reporter.

The testing strategy is to write a set of test cases for every single module. We refactored our modules into single files, and we can dynamically substitute other modules with dummy stubs with a static testing interface. This way we can just import a module, replaces other modules with stubs, then write assertion with set of test cases to conduct testing on a single module.

Product	Youtube Integration module Unit test		
Date	03/01/17		
Author	Naiwei Zheng		
Moderator	Naiwei Zheng		
Inspectors	Ge Yan, Yibo Gou		
Recorders	Jiaqi Zhu		
Defect#	Description	Severity	How corrected.
1.	Adplay event can happen during the playback of video, this could cause conflict to the render module where there is an ad inserted in the middle of the video. Input: adplay event from Youtube event Output: fucntion to handle adplay event	1	Monitor videoPause event when adPlay trigger to find out if the ads is played during the video playback and pause the danmaku if necessary.
2.	If the player is already full screen before the event is being hooked to our event system, will cause problem to the render module. Input: Screen size and fullscreen event Output: parameters to be parsed to render	2	When getting the screen size for the first time. Do not assume that the video is played using window mode

	module		but check if the fullscreen mode is activated and react to that.

Product	Chrome extension module Unit test		
Date	03/01/17		
Author	Yibo Gou		
Moderator	Naiwei Zheng		
Inspectors	Naiwei Zheng, Yucong Ma		
Recorders	Jiaqi Zhu		
Defect#	Description	Severity	How corrected.
1.	If the profile being synced to the chrome cloud is not correct and being read by the extension module. The module does not check if the profile is valid and this will cause problem if the profile is not valid even the plugin will completely crashed even with restart Input: user profile Output: plugin settings	3	Check the bounds of each value in profile, check the validity of each definitions in the profile and throw out error if the profile is not valid. Load the default profile when the profile is valid.

Product	YouDanMu render module Unit test		
Date	03/01/17		
Author	Yibo Gou		
Moderator	Naiwei Zheng		
Inspectors	Yucong Ma, Naiwei Zheng		
Recorders	Jiaqi Zhu		
Defect#	Description	Severity	How corrected.
1.	If the danmaku is being input into the render module is empty, the danmaku will	2	ignore empty danmaku with only

	still be render but the calculation to dodge other danmaku will cause division of zero Input: danmaku to render Output: render result		whitespace character.

Product	Danmaku Timeline module Unit test		
Date	03/01/17		
Author	Yibo Gou		
Moderator	Naiwei Zheng		
Inspectors	Yucong Ma, Naiwei Zheng		
Recorders	Jiaqi Zhu		
Defect#	Description	Severity	How corrected.
1.	Danmaku that are blocked by the user will be render again once the danmaku is parsed for the parser once more (refresh, reset, reload). Input: Danmaku to fill in timeline Output: Danmaku to be rendered in timeline	3	Use the pair of vid and uid of the danmaku to record the danmaku being blocked rather than using solely the uid of danmaku to record the danmaku being blocked by user.