# Findings

## Initialization

Initialization seems to occur through the use of the injector.js script loading the appropriate version of the fs.js script as part of the page, instead of located in the head element as your directions for installations say. By calling <https://fruitshoppe.firebaseapp.com/?fullstory=off#/> I was able to disable recording, and it looks like I could set staging and localhost also to run other configs, which is nice.

It looks like you also set up integrations or load 3rd party libraries from Intercom, mixpanel, Bugsnag, Qualtrics, Optimizely, zopim, google analytics. Much cleaner than normal pages with a lot of script imports.

Im not familiar with Intercom, but guessing from the code in here it’s a 3rd party service that tries to help people when they hit “ragehooks” or “FrustrationEvents” if I have time I’ll need to look into that more (curiousity).

It does appear you use the orgId twice, once on requesting the fs.js, I am making an assumption here without that id matching you might not get the script? Maybe I’ll test that later. You also call it on the integrations, which appears to set up integration points with fs and any of the 3rd party scripts you load in injector.js. Maybe this is code configured in the customers dashboard for integration, kind of cool if so.

## Traffic

For traffic, it looks like once the script is initialized with the correct parameters (I see some error condition checks and seems you need valid orgId of course) then it appears to call something like

<https://rs.fullstory.com/rec/bundle?OrgId=1ENq&UserId=6389274432528384&SessionId=5489709873389568&PageId=5192788835328000&Seq=2&PageStart=1638735179773&PrevBundleTime=1638735179931&LastActivity=4861> every 5 seconds or so (appears to be 5, but seems delayed if no current activity – need to explore further)

Obvious data passed is the query string parameters:

1. **OrgId:**

1ENq

1. **UserId:**

6389274432528384

1. **SessionId:**

5489709873389568

1. **PageId:**

5683035083923456

1. **Seq:**

7

1. **PageStart:**

1638735802496

1. **PrevBundleTime:**

1638735832665

1. **LastActivity:**

8258

Which has various data on the user/session timing, where in the visit this data belongs, in this case the 7th snapshot of x number, the page id (is this set by client or is there some auto assignment based on adding to the data? Can you get this page accidentally assigned to the wrong OrgId? Is there a snapshot taken of the page to overlay the activity recorded? Does that update the PageId when the page changes incrementally/significantly? Can we overlay activity on an old version of the page?)

Then there is the payload data which appears to consist of the various mouse movements clicks and other browser activities defined in the fs.js file like

e.MUT\_INSERT=2,e.MUT\_REMOVE=3,e.MUT\_ATTR=4,e.MUT\_TEXT=6,e.MOUSEMOVE=8,e.MOUSEMOVE\_CURVE=9,e.SCROLL\_LAYOUT=10,e.SCROLL\_LAYOUT\_CURVE=11,e.MOUSEDOWN=12,e.MOUSEUP=13,e.CLICK=16,e.FOCUS=17,e.VALUECHANGE=18,e.RESIZE\_LAYOUT=19,e.DOMLOADED=20,e.LOAD=21,e.PLACEHOLDER\_SIZE=22,e.UNLOAD=23,e.BLUR=24,e.SET\_FRAME\_BASE=25,e.TOUCHSTART=32,e.TOUCHEND=33,e.TOUCHCANCEL=34

etc.

for example a viewport resize looking like

1. 27: {Kind: 19, Args: [1229, 1055, 1244, 1055], When: 1009230}
   1. Args: [1229, 1055, 1244, 1055]
   2. Kind: 19
   3. When: 1009230

Where 19 is the resize\_layout the args being the size differences (in this cases the horizontal went from 1229 to 1244 pixels but the vertical didn’t change. And there are several of these in a row and I changed the size across multiple recording sections. Similarly if I add soemthign to the cart

1. 23: {Kind: 16, Args: [316, 887, 290, 842.484375, 283.1875, 82.015625, 30], When: 1199163}
   1. Args: [316, 887, 290, 842.484375, 283.1875, 82.015625, 30]
   2. Kind: 16
   3. When: 1199163
2. 24: {When: 1199169, Kind: 8197, Args: ["Product Added",…]}
   1. Args: ["Product Added",…]
   2. Kind: 8197
   3. When: 1199169
3. 25: {Kind: 48, Args: ["log", ""Carambola embiggen user cart.""], When: 1199170}
   1. Args: ["log", ""Carambola embiggen user cart.""]
   2. Kind: 48
   3. When: 1199170
4. 26: {Kind: 2, Args: [81, 83,…], When: 1199268}
   1. Args: [81, 83,…]
      1. 0: 81
      2. 1: 83
      3. 2: [[593, 3], [596, 2], [598, 5], [603, 2], [605, 2], 19, " Carambola added to your cart. ", [609, 2],…]
         1. 0: [593, 3]
         2. 1: [596, 2]
         3. 2: [598, 5]
         4. 3: [603, 2]
         5. 4: [605, 2]
         6. 5: 19
         7. 6: " Carambola added to your cart. "
         8. 7: [609, 2]
         9. 8: [611, 2]
         10. 9: [613, 2]
         11. 10: 19
         12. 11: [616, 2]
         13. 12: [504, 6]
   2. Kind: 2
   3. When: 1199268

Where you get the click (16), the message (custom 8197), the log (48) and the insert(2)

When you look into all this in detail its very clear you will basically be able to basically have a recording of the users activity as if you were on their actual computer recording them, except instead of a video file with hard to determine exact activities and data, here you have easy to use data that can be heavily analyzed and played back as if video (if desired).

## Playback

As for stitching it together, this data looks like the way to do so”

OrgId: 1ENq

UserId: 6389274432528384

SessionId: 5489709873389568

PageId: 5683035083923456

Seq: 7

PageStart: 1638735802496

PrevBundleTime: 1638735832665

LastActivity: 8258

Specifically, the UserId and SessionId should allow a visit to be found and combined, with the sequence number determining where the activity occurred in the order. Obviously the customer should only see things belonging to their Orgid. PageId I’d imagine is used to possibly pull a screenshot when viewing a session and the timing related numbers to set pacing and such of a playback session. Otherwise those are good metadata to be able to filter and say something like I want to know what happened during the 230pm dec 5 to 330pm dec 5 sessions when we ran a flash sale and there was some issue, show me those sessions. The other data can be used to look per page (PageId), how many sessions a user had (maybe they got bounced due to error?) how many unique users (also returning/new). This is actually pretty cool stuff.

## Improve

I’ve covered most of my inferences in the above sections, so I will leave them be.

I’d love to magically find an improvement that would shock and awe, but alas, I don’t really have one at the moment. I was initially concerned with how much traffic this might be causing, but its seems pretty lightweight, and of course on my multimegabit connection its no issue. It would be interesting to see the impact of low bandwidth mobile connections, that might engage the retry logic. If I see something in further testing I will strike this and replace.