STARWATCH

Natalie Podrazik, 29th Street Publishing

Wh ?

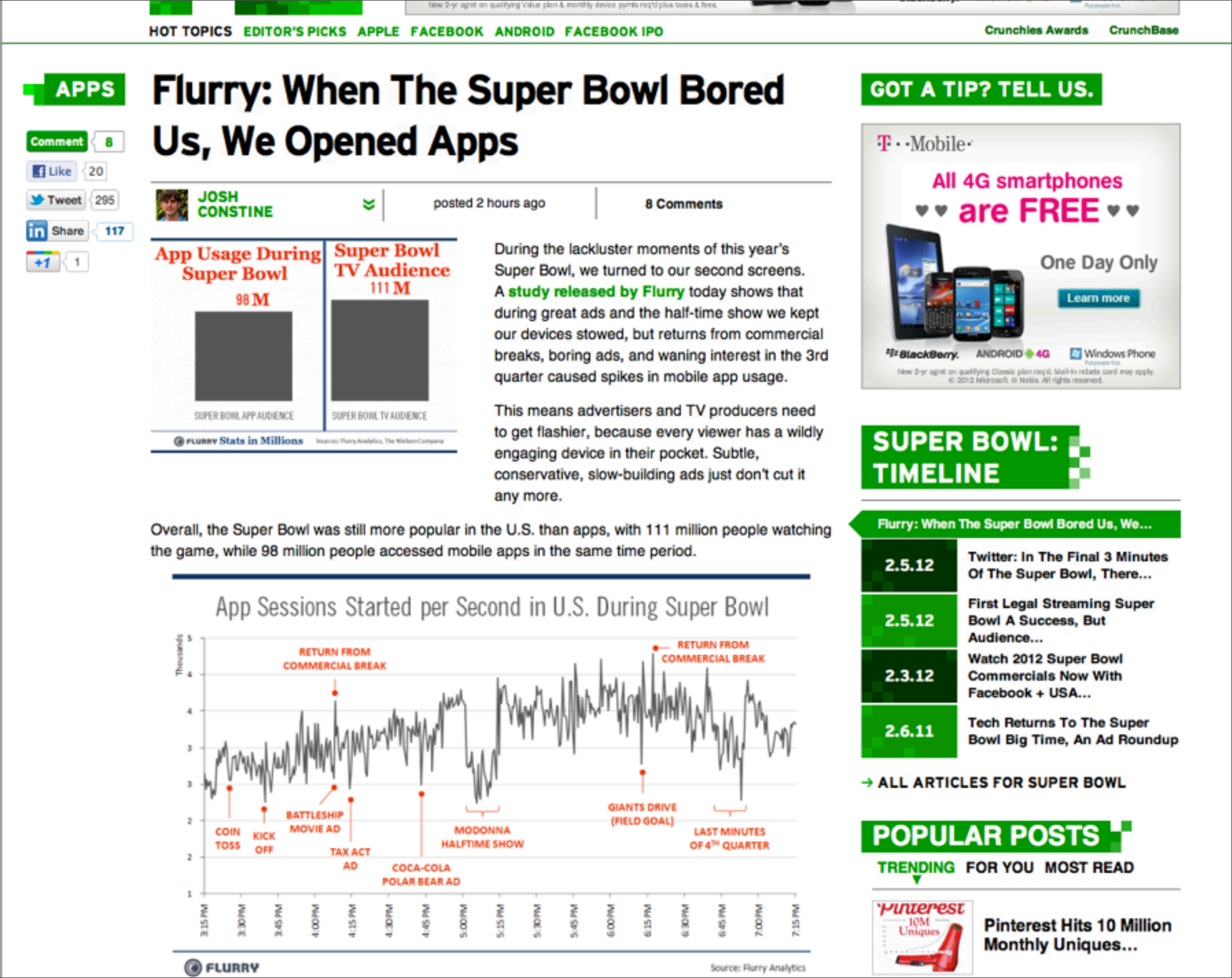
29th Street Publishing

@bdeskin
@lettertojane
@djacobs
@nataliepo

@29pco

V as in Victor
V as in Victor
Library
Menu
Read

Why ?



Flurry: When The Super Bowl Bored Us, We Opened Apps

JOSH CONSTINE posted 2 hours ago **8 Comments**

During the lackluster moments of this year's Super Bowl, we turned to our second screens. A **study released by Flurry** today shows that during great ads and the half-time show we kept our devices stowed, but returns from commercial breaks, boring ads, and waning interest in the 3rd quarter caused spikes in mobile app usage.

This means advertisers and TV producers need to get flashier, because every viewer has a wildly engaging device in their pocket. Subtle, conservative, slow-building ads just don't cut it any more.

Overall, the Super Bowl was still more popular in the U.S. than apps, with 111 million people watching the game, while 98 million people accessed mobile apps in the same time period.

SUPER BOWL: TIMELINE

Flurry: When The Super Bowl Bored Us, We...

2.5.12	Twitter: In The Final 3 Minutes Of The Super Bowl, There...
2.5.12	First Legal Streaming Super Bowl A Success, But Audience...
2.3.12	Watch 2012 Super Bowl Commercials Now With Facebook + USA...
2.6.11	Tech Returns To The Super Bowl Big Time, An Ad Roundup

→ ALL ARTICLES FOR SUPER BOWL

POPULAR POSTS

Pinterest Hits 10 Million Monthly Uniques...

GREAT
TIMES
Do not lean on door
FLY YOUR OWN FL
NOW.
ERT TICKETS ARE AVAILABLE
STAY ALERT
Voter Registration
To Wakefield-241 St, Bronx. Late nights via local
Late nights 4 via Lex stops here

HOMELAND

Goals

- iOS-targeted
- Transparency in collection methods and easy to extend
- User-data collected anonymously and results in near-nil UX overhead
- Real-time access to quality reports

Concept

- iOS app ensures small sqlite db is in place and we have an **installation id** generated
- Standard app actions trigger log statements, which are written to the sqlite db
- If the user is connected to the internet, the sequential log statements are posted to a cloud-based database.
- A remove service parses the log data to find summaries and critical app info

from http://blog.aliceeuphemia.com/?p=3584

Starwatch logs, sends, and parses anonymous usage data.

DEMO

V as in Victor App

```json
    "actions": [
        {
            "timestamp": "20121014 16:05:38",
            "global_id": "5050b34cc873d95b1966be6a",
            "view": "Cover"
        },
        {
            "timestamp": "20121014 16:05:38",
            "global_id": "5050b34cc873d95b1966be6a",
            "view": "Cover"
        },
        {
            "timestamp": "20121014 16:05:41",
            "global_id": "5050b34cc873d95b1966be6a",
            "view": "TitlePage"
        },
        {
            "timestamp": "20121014 16:05:43",
            "global_id": "5050a4cdc873d95b1966be5d",
            "view": "Article"
        },
        {
            "timestamp": "20121014 16:05:48",
            "global_id": "5050b634c873d929cee543a2",
            "view": "Gallery"
        },
        {
            "timestamp": "20121014 16:05:55",
            "global_id": "5050a5e5c873d95b1966be62",
            "view": "Article"
        },
        {
            "timestamp": "20121014 16:05:58",
            "global_id": "5050a674c873d95b1966be63",
            "view": "Article"
        },
        {
            "timestamp": "20121014 16:06:28",
            "global_id": "5050b34cc873d95b1966be6a",
            "view": "Flap"
        },
        {
            "timestamp": "20121014 16:06:29",
            "global_id": "5050b34cc873d95b1966be6a",
            "view": "Flap"
        },
        {
            "timestamp": "20121014 16:06:29",
            "global_id": "",
            "view": "Library"
        },
        {
            "timestamp": "20121014 16:06:35",
            "global_id": "5009b5ec362fe2046c000005",
            "view": "Cover"
        },
        {
            "timestamp": "20121014 16:06:35",
            "global_id": "5009b5ec362fe2046c000005",
            "view": "Cover"
        },
        {
            "timestamp": "20121014 16:06:37",
            "global_id": "5009b5ec362fe2046c000005",
            "view": "TitlePage"
        },
        {
            "timestamp": "20121014 16:06:38",
            "global_id": "5009b74f362fe2046c000006",
            "view": "Article"
        }
    ],
    "device": "GX9LMY2K2958KRYLZO0KNMB9",
    "start_time": "20121014 16:05:37",
    "end_time": "20121014 16:06:42",
    "usage_time": 65,
```

How

do I use this?

0. https://github.com/29thStPublishing/Starwatch
1. Set up a DB
2. Add iOS hooks into your app
3. Verify data collection into DB
4. Parse collected logs via script

App Hooks: Basic

Step One: Set up and log info only

```
#import "SWCUtility.h"

// add to application:didFinishLaunchingWithOptions

    // This will make sure our DB's are in the correct place,
    // we've begun tracking actions against this unique device id,
    // and increments the number of times this user has opened the app.
    [SWCUtility begin];

    // This method takes in a dictionary of your custom key-value pairs
    // and, along with general information about this device and user,
    // prepares to send it to the remote db.
    [SWCUtility logInfo:
        [NSDictionary dictionaryWithObjects:[NSArray arrayWithObjects:
                                             [NSString stringWithFormat:@"%d",
                                                [SWCUtility getNumOpens]], nil]
                                    forKeys:[NSArray
arrayWithObjects:@"num_opens",nil]]];
```

Step Two: Send data on app exit

```
// in applicationDidEnterBackground to send data on exit:

    [SWCUtility send_data];
```

App Hooks: Advanced

Step One: Set up and send on exit

```
#import "SWCUtility.h"

// in application: didFinishLaunchingWithOptions:

    // This will make sure our DB's are in the correct place,
    // we've begun tracking actions against this unique device id,
    // and increments the number of times this user has opened the app.
  [SWCUtility begin];
  // then triggers the “start app “action.
    [SWCUtility logAppStart];

// in applicationDidEnterBackground:application

    // mark that the user’s session is over, and send the data.
  [SWCUtility logAppEnd];
  [SWCUtility send_data];
```

Step Two: Add any hooks you’d like.

Sample App Hooks

```
// ---- SWCUtility.h ----
// appStart and appEnd demarcates the
user's "session" for parsing
+(void)logAppStart;
+(void)logAppEnd;


/* general logAction method:
  'name': the name of the view you're
logging the action from
  'action': the verb to describe what
triggered this log (view_begin, tap,
swipe, error, etc.) Check out the many
SW_ACTION_* defined in SWCUtility.h.
  'global_id': a unique identifier of
the contents of your container view
  'metadata': a placeholder for more
information to log */
+(void)logAction:(NSString*)name
    action:(NSString*)action
    global_id:(NSString*)global_id
    metadata:(NSString*)metadata;
```

```
// ---- SWCViewController.h -----
/* parameters: view's name, view's
global_id (empty string OK),
respond_to_callbacks flag. Turn it on
to inherit the viewDidAppear/disappear
log statements. Turn it off to
manually log when your view appears.
*/
- (id)init:(NSString*)new_name
    new_global_id:(NSString*)
new_global_id
    new_respond_to_callbacks:(BOOL)
new_respond_to_callbacks;


// ---- SWCUIButton.h -----
/* parameters: the button's name.
Call this
method to "activate" a SWCUIButton
which will automatically add a target
to log every time it's pressed. */
-(void)activate:(NSString*)newName;
```

Verify & Parse

-
  ```
      # Looks for the “info” actions and fills in a device summary
      table
      python parse_logs.py info

      # Looks for “feedback” actions and puts them in one collection
  python parse_logs.py feedback

  # Parses through (start_app|became_active) and entered_background
  action and notes everything in between oiinto concise Sessions.
      python parse_logs.py session
  ```

What

is next?

- Making better reading experiences to reinforce the relationship between an artist and their audience
- Learning from our subscribers to improve our design
- Integrating answers to reader’s specific questions directly in our authoring & packaging system
- Open sourcing Starwatch to share our work and so others may use it and improve it

What's the total number of minutes spent in this issue? What's the average number of minutes spent per issue for this device? For iPads only? How many people read this issue? What counts as a "read"? How many people open the app and never come back? How many people go to the library and never come back? What's the medium minutes per reader? What's the issue open rate? How does it differ for subscribers? How often do subscribers come back compared to people who buy in-app purchases only? How many times have people ever opened the app? How many times do people who downloaded the app typically open it? What time of day do they open the app? What day of the week do they typically open the app? How many articles are read to the bottom in every issue? How many articles are opened in every issue? What's the average read time for one article? What's the breakout of the story opened in one issue? Which article results in the most close-apps? Where are users most likely to send Feedback? Are users using Feedback as notes to the author or for bug reports? How many subscribers read on the iPhone and iPad? When do users share -- after weeks of reading or after a friend refers them? **SO MANY s.**

https://github.com/29thStPublishing/Starwatch

hello@29.io