

[DRAFT, please don't distribute] Wildcard: a tool for modifying websites by directly manipulating data tables

Geoffrey Litt^a and Daniel Jackson^a

^a Massachusetts Institute of Technology

Abstract Browser extensions and user scripts can modify websites in useful ways—ranging from blocking ads to adding entire new features to Gmail—but many people have unique needs that aren't met by existing extensions. Today, most of those people are stuck. They can't build their own browser extensions without learning how to program, so they have no choice but to accept the way the software was built. What if things were different?

Wildcard is a platform that empowers anyone to build browser extensions and modify websites to meet their own specific needs. Wildcard shows a simplified view of the data in a web page as a familiar table view. People can directly manipulate the table to sort/filter content, add annotations, and even use spreadsheet-style formulas to pull in data from other websites. The key idea is that a table is a powerful, simple, and familiar paradigm for modifying a website.

ACM CCS 2012

- *General and reference* → *Computing standards, RFCs and guidelines*;
- **Applied computing** → **Publishing**;

Keywords programming journal, paper formatting, submission preparation

The Art, Science, and Engineering of Programming

Perspective The Art of Programming

Area of Submission Social Coding, General-purpose programming



© Geoffrey Litt and Daniel Jackson
This work is licensed under a "CC BY 4.0" license.
Submitted to *The Art, Science, and Engineering of Programming*.

1 Introduction

Todos in intro:

Do some stuff

- hook with Airbnb sort
- add more reasons
- make this shorter

People have complaints about web apps they use, but they rarely modify those apps to meet their needs. Why not? Some guesses:

- Technical skill: Most people don't know how to use Javascript and manipulate the DOM.
- Low ROI: It takes a long time to reverse engineer a site, and it's usually not worth it

These seem reasonable but they might not tell the whole story.

There have been many research projects that try to address these hurdles. For example, Chickenfoot [1] allows for people to more quickly modify sites without using Javascript or dealing with the DOM, directly addressing both of these barriers. But, a decade later, not many people have ended up using systems like this.

1.0.1 Why don't people tweak websites?

I know how to do web programming, and yet I rarely modify my apps. Sometimes it actually turns out to be pretty easy to hack on a site once I start doing it so the ROI is actually pretty high. These reasons don't seem to fully explain my behavior.

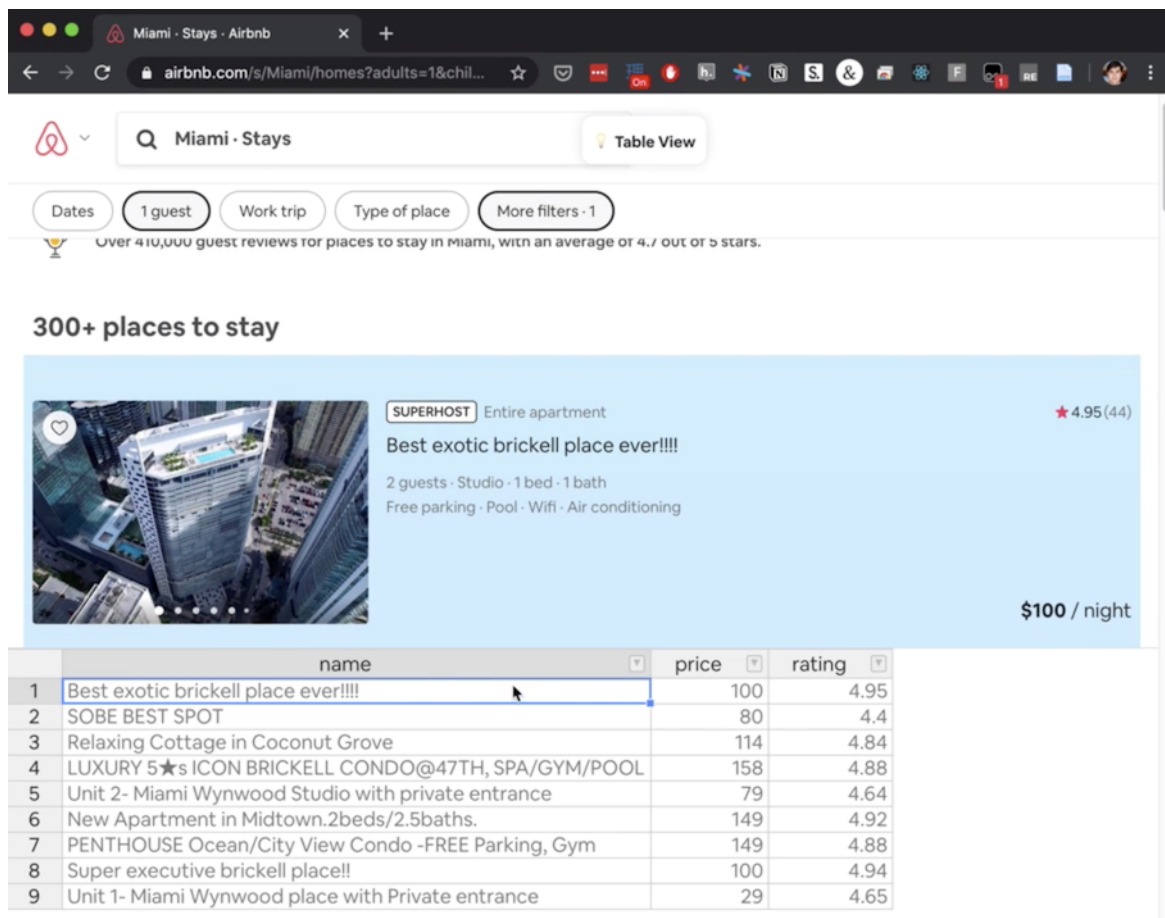
A disheartening explanation might be that most people just don't care enough to make changes. This claim might be true in the current context, but it's also important to remember that motivation is connected to culture and to the space of possibilities provided by our tools. Most people probably didn't want to write letters before mass literacy.

One interesting explanation is that **people can't estimate the difficulty of a change**. It's not motivating to think about what changes I might want when I don't know whether they would take a few minutes or are completely impossible. For me, an engineer, this gets at the heart of the issue. I know I could probably implement the change, but it's so hard to estimate how long it would take (minutes? months?) that I don't even bother trying. In general, when something seems expensive to do (or even possibly expensive), it can discourage casual lightweight experimentation.

So, perhaps to encourage people to casually modify software, **apps need to provide more consistent affordances indicating what changes are possible and easy**. Maybe if it were more obvious that certain types of changes could be achieved in mere minutes, programmers and non-programmers would end up modifying our software more.

1.0.2 The Wildcard platform

Wildcard adds a panel to the bottom of a web page that shows a structured table view of some of the core data in the page. When the user manipulates the table, the



Miami - Stays - Airbnb

airbnb.com/s/Miami/homes?adults=1&chil...

Miami - Stays

Table View

Dates 1 guest Work trip Type of place More filters - 1

Over 410,000 guest reviews for places to stay in Miami, with an average of 4.7 out of 5 stars.

300+ places to stay

SUPERHOST Entire apartment ★ 4.95 (44)

Best exotic brickell place ever!!!!

2 guests · Studio · 1 bed · 1 bath

Free parking · Pool · Wifi · Air conditioning

\$100 / night

	name	price	rating
1	Best exotic brickell place ever!!!!	100	4.95
2	SOBE BEST SPOT	80	4.4
3	Relaxing Cottage in Coconut Grove	114	4.84
4	LUXURY 5★s ICON BRICKELL CONDO@47TH, SPA/GYM/POOL	158	4.88
5	Unit 2- Miami Wynwood Studio with private entrance	79	4.64
6	New Apartment in Midtown.2beds/2.5baths.	149	4.92
7	PENTHOUSE Ocean/City View Condo -FREE Parking, Gym	149	4.88
8	Super executive brickell place!!	100	4.94
9	Unit 1- Miami Wynwood place with Private entrance	29	4.65

■ **Figure 1** Opening a table corresponding to search results on Airbnb

original page also gets modified. We aim to make the mapping between the table and the page as direct and intuitive as possible.

Todos here:

- a better demo is actually sorting too at this point too, to show the bidirectional nature
- hint at future possibilities
- introduce an image fallback for the PDF version

For example, in Fig. 1 we open up a table view that corresponds to search results on the Airbnb travel site.

There's somewhat of a tension here between directness and structure. Manipulating the original page itself might seem most "direct." But the whole problem we're dealing with is that the original web page doesn't provide affordances for end-user modification, and there's no consistent structure that people can learn to work with across many sites.

1.0.3 Directness

The table view is perhaps one hop less “direct,” but in turn provides other benefits. Because many people are already familiar with spreadsheets, they can quickly intuit which changes are easy to make in this system. Also, people can learn to work with the same consistent spreadsheet view across many sites. This consistency is very important—learning a generic tool that can be applied to many different specific cases is very powerful, and connected to the idea of literacy in a medium (just ask anyone who has used vim, or spreadsheets, or a pencil and paper). Wildcard aims to balance this tension, with a workflow that involves both the table view and the original webpage.

Wildcard is fairly general and can support many useful changes to websites, which will be demoed later:

- sorting and filtering data: eg sorting shopping results
- using 3rd-party APIs and performing small computations to add new data, in the style of “web mashups”: eg adding walkability scores to hotel listings
- adding private user annotations to the page: eg taking notes on different options
- using alternate UI widgets to enter data into a page: eg using a personal datepicker widget with private calendar data, to enter the right dates for taking a flight

The overall goal is to provide generic tools that fit well with the table paradigm and enable many specific useful changes. But it's important to note that Wildcard doesn't aim to provide maximum coverage all the possible ways someone might want to modify a web page. Rather, it aims to provide a useful, simple subset of modifications, and to provide consistent affordances so that users confidently understand which modifications they can make.

1.0.4 The big picture

Eventually web apps might provide the structured data table themselves. In the meantime, we need some sort of adapter to make this system work with existing sites.

Wildcard provides a system for creating a wrapper on top of existing websites. This wrapper defines how structured data can be extracted out of the page, and also how manipulating the table should modify the page.

The most basic way of building these wrappers is for skilled programmers to manually build and maintain them for popular sites. This approach beats the status quo because many people, including end users, can benefit from the generic wrapper and use it in many ways. This is different from the current world where programmers build use-case-specific browser extensions, and each extension has to implement its own interactions with the low-level DOM of a page. There's also a greater incentive for many people to collectively maintain a wrapper if it's shared.

A more advanced way would be to make these wrappers partially or totally automated, and enable end users to create them. This future work could leverage existing research on wrapper induction but isn't the focus of the current work.

Again, ultimately we hope that first-party sites would find it beneficial and straightforward to provide a structured data view themselves. Wildcard doesn't require sites

to expose some complex Semantic Web schema; it merely asks for a simple structured data view.

2 Demo

- Sorting search results:
 - Airbnb took away search
 - once you see the tabular view, there's an obvious interaction available to anyone familiar with tables
- Injecting new data into the page
 - Take own custom notes, saved in the browser. Maybe shared in the future
- Formulas
 - Compute values to inject into the page
 - TBD: styling
- Using a custom date picker

3 Design principles

3.1 Expose generic structure

- Consistent across all sites, can gain familiarity
- Nightmare bike: users learn the structure they can exploit

3.2 No API needed

- 3p-only is enough. 1p help is optional.
- Can even do things the 1p didn't want to expose.

3.3 In-place toolchain

- meet the user where they are, in the page
- very native to the web style (dev tools is kinda like this)

3.4 Decouple UI from data

- bring your own views of the data
- bring your own widgets for data entry
- instrumental interaction

4 Analysis

Cognitive dimensions?

5 Related work

- Instrumental interaction
- Web automation
- Wrapper induction
- Personal data
- Extension helpers

6 Future work / Open Questions

- How far does functionality go?
 - workflows? triggers?
 - what can and can't be done?
- automated wrappers?
 - lean on existing tech
- usability studies

7 Conclusions

8 Todos

- remove bold text for the proceedings version

Acknowledgements

References

- [1] Robert C. Miller, Michael Bolin, Lydia B. Chilton, Greg Little, Matthew J. Webber, and Chen-Hsiang Yu. "Rewriting the Web with Chickenfoot". In: 2010. DOI: 10.1016/B978-0-12-381541-5.00003-1.

About the authors

Geoffrey Litt is bla bla bla

Daniel Jackson is bla bla bla