# Endev: Declarative Prototyping with Data

Filip Kis, Cristian Bogdan



# Endev: Declarative Prototyping with Data

Filip Kis and Cristian Bogdan

KTH Royal Institute of Technology, Stockholm, Sweden
fkis@kth.se, cristi@kth.se

**Abstract.** The trend of Open Data and Internet-of-Things initiatives contribute to the ever growing amount of data available through web APIs. While building web applications has become easier with recent advancement in web development technologies and proliferation of JavaScript frameworks, the access to data from various APIs and data stores still poses certain challenges. It often requires complex setup and advanced programming skills that hinder the rapid prototyping efforts. Therefore, we propose Endev, a declarative framework for prototyping applications that is built on modern web technologies and supports building modern web applications, that utilize the vast amount of available data, without the need for setup or write complex JavaScript code.

**Keywords:** UI modeling, GUI generation, interactive prototypes, discourse model, query annotations

## 1 Introduction

The Open Data and the Internet-of-Things trends are producing more and more data that users are expected to have access to via GUIs of interactive applications. At the same time building such data-centric applications requires less effort than ever, for skilled developers, thanks to cloud services, popularization of scripting development technologies (e.g. JavaScript, Python) and the omnipresence of web[1]. However, the setup and advanced programming skills required to build interactive prototypes hinders the rapid prototyping efforts.

Prototyping is the key activity in human-centered software engineering process as it allows designers to explore design alternatives and include end-users early on in the process. Various prototyping methods, from sketching to using dedicated prototyping tools, are used in practice, however, the research shows that designers desire better tools that would allow them to prototype the flow of data and advanced interaction [6,9]. Furthermore, a survey of 4000 designers [3] shows that HTML is preferred prototyping tool over tools specifically designed for prototyping (e.g. InVisio) or even general design tools (e.g. Photoshop, InDesign).

Building on these research results, and with the aim to support rapid prototyping of data-centric applications, we demonstrate Endev [7] – a declarative

[1] Web is becoming the platform of choice for many applications as it is easier to maintain (no installs or updates) compared to desktop or mobile applications.

prototyping solutions. The declarative HTML annotations allow users to prototype interactions with, and connection to, data. Endev is developed as JavaScript library and utilizes cloud services to support data storage and API access without the need for server setup or dedicate development environments.

## 2 Related Work

Declarative languages have played an important role in UI development and especially in web design where HTML and CSS are dominating as markup technologies for defining UI layout. Modern client-side web development frameworks (e.g. Angular [2], Ember [1]) use declarative data-binding constraints that provide more dynamic features, such as keeping HTML elements automatically in synchronization with the application data values, though they still require significant amount of non-declarative code to access the backend or the API data.

Quilt [4] is a recent solution that provides HTML annotations to connect the interface to a spreadsheet that serves as the datastore. Quilt allows both data read and write and keeps the interface in synchronization with the spreadsheet data. Another solution based on spreadsheets as the datastore is Gneiss [5]. Unlike Quilt, Gneiss is a live programming mashup environment where, instead of using HTML annotations, the users can drag and drop widgets to the page and connect them with spreadsheet values. A key feature of Gneiss is the support for any REST[2] web service returning JSON[3] data which can be interactively combined in the spreadsheet before their data is used in the interface. The main drawback for Gneiss compared to Endev is that the users are limited to working with UI widgets existing in the system, which significantly reduces the design possibilities.

XFormsDB [10] is a declarative data binding solution that binds to server-side data. It is based on the XForms[4], a W3C Recommendation, that was designed to be the next generation of HTML forms. XFormsDB depends on having a complex server setup and supports only XML based databases, thus it is not ideal for quick prototyping. Furthermore, even though XForms are relatively old-standard (first version published in 2007), none of the major browsers currently natively supports it.

## 3 Setup-free prototyping with Endev

During prototyping, it is often important to be able to share the prototype with other stakeholders to solicit feedback. When the prototype is interactive, includes data, or requires a complex setup, it becomes hard to share it beyond screenshots or video recordings that capture only fixed path interaction. However,

[2] Representational State Transfer protocol - most widely used protocol for web service APIs

[3] JavaScript Object Notation - data format often used by exchanging data through web services

[4] https://www.w3.org/MarkUp/Forms/

Endev addresses these challenge by utilizing modern web technologies and cloud services.

With Endev the users can prototype web applications that provide data storage and other features that contemporary users expect (e.g. drag-and-drop interaction, real-time data synchronization) by writing HTML and annotating it with declarative expressions. Such prototypes can be executed in any browser, which makes them easily sharable. In other words, the prototype can be sent by email, shared over Dropbox or put on-line in one of the code playgrounds (e.g. CodePen[5], JsFiddle[6]) for easy access and modification.

We will use an example Wish List app (shown in Figure 1) to demonstrate how a prototype can be built with Endev. The goal of the Wish List app is to give the users possibility to setup their wish list (e.g. for a birthday or Christmas). Each item on the list can optionally include a picture that the users can retrieve by searching through public pictures on Flickr.

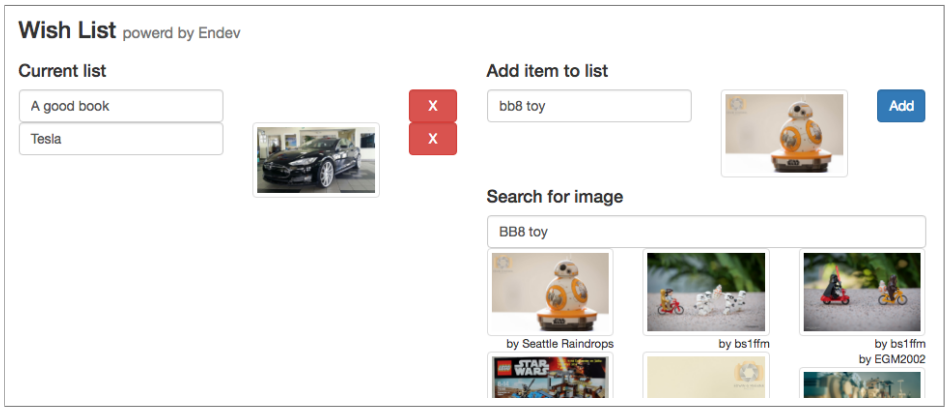


**Fig. 1.** The Wish List prototype showing the items on the list (with name and optionally an image) on the left (Current list) with possibility to add a new item on the right (Add item to the list). At the top right the users can type the name of the new item, while in the bottom right they can search and select a picture that should be associated to the item.

The following sections will describe how the two main challenges (storing the wish list data and getting the pictures from Flickr) are achieved through prototyping with Endev. The more comprehensive and interactive tutorial covering the main features of Endev is available on-line[7].

[5] http://codepen.io/
[6] https://jsfiddle.net/
[7] http://www.endevjs.org/tutorial

## 4 On-the-fly creation of own data

Storing the data is typically not a concern that is addressed during UI prototyping. The designers normally use dummy data that is either not possible to edit or, at a more advanced stage of the prototyping, the edits are temporary (e.g. refresh of the page or reload of the app reset the data). However, there is a value in working with the data that is stored. For instance, when testing with users they get to experience the flow of the application by experiencing the interaction with the data. Furthermore, if there is a need to quickly modify the design, the user-entered data is still there and thus makes the comparison of alternatives easier to compare.

Listing 1.1 shows the HTML code with Endev annotations needed to list the items from the wish list and add a new item to the list. The code is the same code used for the application shown in Figure 1 without the additional layout-only HTML code.

```
Current list
<div data-from="firebase:WishList item" data-auto-update="true">
  <input data-value="item.name"/>
  <img src="{{item.image}}"/>
</div>

Add item to list
<input data-value="newItem.name"/>
<img src="{{newItem.image}}"/>
<button data-insert-into="firebase:WishList"
        data-click="insert({name:newItem.name,image:newItem.image})">
  Add
</button>
```

**Listing 1.1.** Endev code of Wish List app (see Figure 1) for listing the items in the wish list and adding a new item to the list.

In this example `data-from` and `data-insert-into` annotations are used to connect to the data storage for which we use Firebase[8] cloud storage. Firebase is a document-based storage, therefore, there is no need to define the data structure before hand. Instead, data is stored as-is and on-the-fly.

We have seen, in the related work, some solutions use spreadsheets to allow building applications quickly without the need for complex database setup. However, document-based data storage provides more complex data objects compared to spreadsheets thus allowing for more complex use-cases. Furthermore, Firebase is just one of several data providers offered by Endev and others (e.g. spreadsheet storage) can be added easily.

Other annotations like `data-value` and `data-click` serve to bind the values to some UI elements or user actions respectively. Finally, the `data-auto-update` annotation enables automatic updates of the values in the *Current list*, in other words, as soon as the users modify any of the items in the list the changes are saved automatically.

[8] https://firebase.google.com/

## 5 Seamless integration of API data

The second challenge of the Wish List app is to integrate with Flickr search API so that an image can be associated with the wish list item. Traditionally such feature would be prototyped with dummy data instead of having a real API integration. However, with more and more web services generating data and applications that work with them it is important that prototyping includes working with real API data. Web service data, compared to proprietary domain-data, comes with certain challenges (e.g. quality, reliability, latency) that are often out of control of application developers. For instance, the end-users or designers might expect the results from API to be the same as when they use the actual service where the data comes from. While, in reality, the service provider might have different algorithms for these two cases. Experiencing the differences early on in the prototyping allows for better management of expectations and thus better design.

Accessing APIs requires certain amount of complex and error-prone code [8] that needlessly increases the prototyping effort. Endev addresses this by providing seamless mechanism for reading web service data that is seen by the users as just another data provider. In other words, the difference between reading their own data (as seen in Listing 1.1) and the data coming from an API is in the string that defines the data source (e.g. `data-from` annotation).

Listing that follows shows how the users can read data from Flickr API based on the inputed search term.

```
Search for an image
<input data-value="searchTerm"/>
<div data-from="yql:flickr.photos.search result"
     data-where="result.text= searchTerm AND
                 result.api_key = '_API_KEY_OMITTED_'">
  <div data-from="result.photo photo">
    <img src="http://farm{{photo.farm}}.staticflickr.com/{{photo.server}}/{{
        photo.id}}_{{photo.secret}}.jpg"
         data-insert-into="firebase:PhotoCollection"
         data-click="newItem.image = '_LONG_URL_OMITTED_'">
    <small data-from="yql:flickr.people.info2 people"
           data-where="people.user_id = photo.owner AND
             people.api_key = '_API_KEY_OMITTED_'">
      by {{people.person.username}}
    </small>
  </div>
</div>
```

**Listing 1.2.** Endev code of Wish List app (see Figure 1) for searching public images on Flickr and, when one image is clicked, adding it to the new item.

The first `data-from` has similar meaning as in Listing 1.1. However, instead of getting data from Firebase, Endev now uses Yahoo Query Language platform[9] to directly access the Flickr search API. Since the data returned by the API is hierarchical (i.e. contains an item called `photo` that contains an array of actual photo results) the second `data-from` is used to access each item in the array. The final `data-from` is used to access another Flickr API which returns the

[9] https://developer.yahoo.com/yql/

information about the owners of the photos based on their ids. Finally, the `data-click` sets the value of the image of the `newItem` which was used in Listing 1.1 when creating the new item for the list.

## 6 Behind the scenes

Endev is implemented as JavaScript framework with architecture shown in Figure 2. Endev Core is responsible for caching (on the client-side) and querying the data from one of the Endev Providers which, in turn, access the data and keep it in sync with the data storage. Endev Annotations are built on top of Anuglar, therefore, Endev supports evolutionary prototyping as the prototype can evolve beyond the capabilities provided by Endev. Thus, Endev can be used to either quickly prototype a completely new application or just a new feature in already existing applications. In both cases the prototype can evolve into a more stable system without the need of re-implementing the whole interface from scratch.

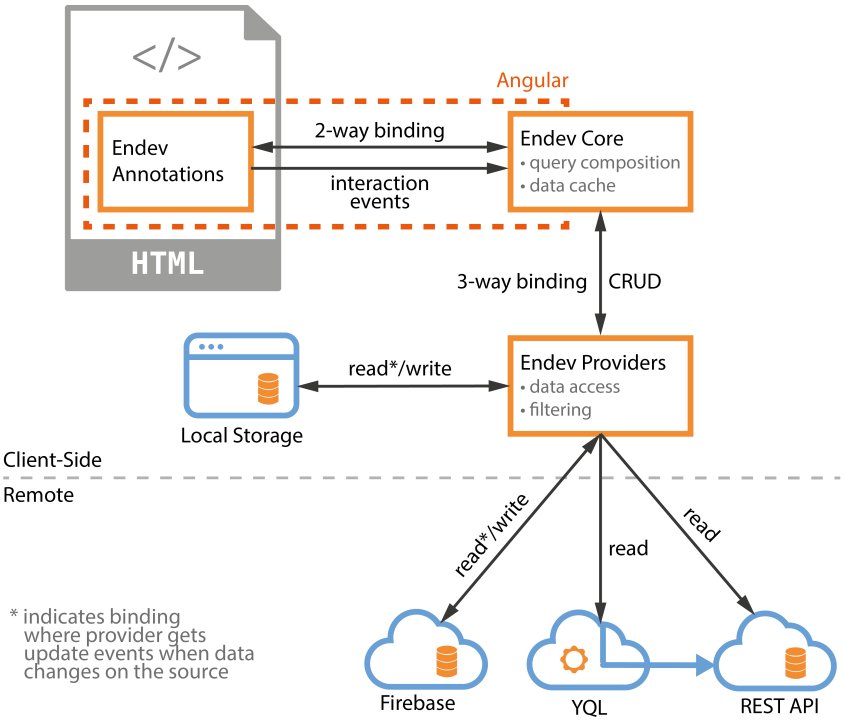


**Fig. 2.** Endev architecture

## 7 Conclusion and future work

In this paper we demonstrated Endev, the tool for prototyping interactive applications with data. Endev enables prototyping with real data, created by the users or coming from a web service, through declarative annotations without the need for complex setup or server orchestration. The prototypes are simple HTML files that can easily be shared among all the stakeholders and require only a browser to be executed.

While the declarative annotations employed by Endev provide a uniform way of accessing data, there is a challenge in finding and understanding the growing amount of web service data. Endev currently supports a basic way of exploring the data returned from an API, however, in the future we would like to explore how to better support the discovery and understanding of APIs during prototyping.

## References


1. Ember.js - A framework for creating ambitious web ... (2011), http://emberjs.com/
2. AngularJS - Superheroic JavaScript MVW Framework (2014), https://angularjs.org/
3. The Tools Designers Are Using Today (2015), http://tools.subtraction.com/
4. Benson, E., Zhang, A.X., Karger, D.R.: Spreadsheet driven web applications. In: Proceedings of the 27th annual ACM symposium on User interface software and technology (UIST). pp. 97–106. ACM Press, New York, New York, USA (2014)
5. Chang, K.S.P., Myers, B.A.: Creating Interactive Web Data Applications with Spreadsheets. In: ACM symposium on User interface software and technology (UIST). pp. 87–96. ACM Press, New York, New York, USA (2014)
6. Grigoreanu, V., Fernandez, R., Inkpen, K., Robertson, G.: What designers want: Needs of interactive application designers. In: Visual Languages and Human-Centric Computing, 2009. VL/HCC 2009. IEEE Symposium on. pp. 139–146 (sep 2009)
7. Kis, F., Bogdan, C.: Declarative Setup-free Web Application Prototyping Combining Local and Cloud Datastores. In: 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE Computer Society (2016)
8. Myers, B.A.: Separating application code from toolkits. In: Proceedings of the 4th annual ACM symposium on User interface software and technology (UIST). pp. 211–220. ACM Press, New York, New York, USA (1991)
9. Myers, B.A., Park, S.Y., Nakano, Y., Mueller, G., Ko, A.J.: How designers design and program interactive behaviors. In: 2008 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). pp. 177–184. IEEE Computer Society, Washington, DC, USA (2008)
10. Vuorimaa, P., Laine, M., Litvinova, E., Shestakov, D.: Leveraging declarative languages in web application development. World Wide Web 19(4), 519—-543 (2016)