XForms

An introduction to XForms as a basis for multi-modal forms

Thomas Uhrig

Hochschule der Medien, Stuttgart

Abstract. Forms are a common element of today's web-sites. They can be found in search-engines, online-shops or social-communities - mostly implemented in HTML and supported by JavaScript. But the increasing amount of diverse devices connected to the internet brings up a conceptual problem of HTML: the violation of separation between data and view.
A solution to solve this problem, called XForms, is presented in this paper. After a general introduction to the technology, an approach how XForms can be used to build multi-modal user interfaces is described. This approach called XFormsMM is based on a paper of Mikko Honkala and Mikko Pohja and on the idea of extending XForms with modality-dependent style-sheets. It shows how the separation between data and view can lead to very flexible user interfaces. Henceforth, an example user interface including code snippets and pictures is given. Afterwards, the adaptation capabilities of XForms and XFormsMM is outlined. A closing discussion and conclusion will complete the paper.

Keywords: XForms, HTML, Multimodality, Adaptive User Interfaces

1 Introduction

HTML forms together with JavaScript are the de facto standards to enable user interaction in modern web-sites. Nevertheless, they are suffering from different problems. These problems appear especially by devising web-sites for various devices or modalities. Besides problems around scripting and others, the main issue is the violation of separation between data and view in HTML forms.

This violation of a well known principal of computer science has many impacts. The view of a form strongly depends on its data, because the data is mixed with the form markup. Therefore, it's not possible to provide different views on the same data. Also, there is no real data *structure*, which means that an important abstraction layer is missing. Hence, working in teams of form designers and data designers (e.g. a team working at the back-end of a web-application) is hardly possible. HTML forms don't provide a logical form structure either, as they only focus on a visual structure (e.g. with tables or `div`-elements).

These conceptual lacks HTML forms have, will lead to problems in combination with diverse devices and multimodality. They make it impossible to present single-authoring forms[1] on different devices (e.g. PCs and cellphones) and in different modalities (e.g. in a visual view and an aural output) in an acceptable way.

XFORMS is an approach of the W3C to improve form authoring and processing in HTML documents (see [7]). Besides an integration of existing XML technologies like XPATH[11] and XML SCHEMA[10] as well as validation/calculation mechanisms, it provides a clean separation of data and view. Hence, XFORMS provides a solution for the problems mentioned beforehand and serves as a good basis to build multi-modal applications on top of it.

2 Related Work

A general introduction to XFORMS can be found in "XFORMS Essentials" by Micah Dubinko [1]. Dubinko gives a well-founded introduction into XFORMS as well as related topics like XML SCHEMA or XPATH. However, the book doesn't cover options for creating multi-modal interfaces.

XFORMS was devised to solve problems related to form-authoring in HTML 4 [8]. Therefore, the new HTML 5 standard is a direct competitor to XFORMS. A comparison of XFORMS, HTML 5 as well as three other XML-based languages for web user interfaces can be found in [6]. Beside good approaches like several new input types, HTML 5 provides still no separation between data and view.

The information given here about XFORMSMM is based on a paper by Mikko Honkala and Mikko Pohja from the HELSINKI UNIVERSITY OF TECHNOLOGY [4]. The authors follow a generic approach of how multi-modal applications can be created using XForms and modality dependent style-sheets. Their paper also provides a list of related work to their approach.

3 XForms

XFORMS is a W3C standard to improve form authoring and processing in HTML documents. Unlike HTML, XFORMS is *not* a standalone mark-up language. Therefore, it must be used within a host language, e.g. XHTML.

XFORMS consists of four major parts:

1. The **model** is the abstract definition of the data behind the form. It is typically declared in the `head`-tag of an HTML document and is referenced by so called form controls.
2. **Bindings** are constraints or calculations linked to a specific node in the model.

[1] *Single-authoring* means that a document is written only once for different devices/ modalities, no specialized versions are needed.

3. Form **controls** and **containers** are the view on the data. They are very similar to HTML form elements, but they don't contain any data, rather then referring to it in the data model.
4. **Events** are used in XFORMS for communication, e.g. between a constraint and an error-handler.

The following describes these major parts of XFORMS in detail. In the end, the multimodality approach of Mikko Honkala and Mikko Pohja is introduced.

3.1 The XForms data model

XFORMS provides a *complete* separation between data and view. Therefore, an abstract data model must be defined at the beginning of every XFORMS document. All controls of the document will refer to that model later on.

Listing 1 shows the declaration of a data model with internal XML data. Listing 2 shows an equivalent declaration with XML data provided by an external file called `questionnaire.xml`.

Listing 1 Declaration of an inline data model

```
1 <xf:model id="model">
2     <questionnaire xmlns="">
3         <location/>
4         <age/>
5         ...
6     </questionnaire>
7 </xf:model>
```

Listing 2 Declaration of an external data model

```
1 <xf:model id="model">
2     <xf:instance id="default" src="questionnaire.xml" />
3 </xf:model>
```

Both model declarations are defining a model with the id `model` that can be referenced by any form control later in the document. At runtime (e.g. when the HTML document is rendered in a browser), a concrete in-cache instance of the model is made. If a form control referring to a specific node in the model is used, it will edit the node value of the in-cache model instance. Therefore, multiple controls can refer to a single node in the model, e.g. controls from different modalities.

3.2 Bindings

Bindings are links between a specific node of the instance data and a calculation or validation expression that should be applied to this node. They are specified within the `model`-tag at the beginning of an HTML document. Bindings can be used to reduce or even to avoid the necessity of JavaScript and other scripting languages (this topic is discussed in detail in section 3.4).

Listing 3 shows the definition of a binding expression linked to the node `age` of the data model given in listing 1. The binding expression specifies a constraint on the node `age` that its value must always be a number. If it is not a number, an `xforms-invalid` exception will be thrown, which can be caught somewhere else in the document[2]. The expression itself is written in XPath, a W3C (query) language to select nodes in an XML document.

Listing 3 Definition of a binding expression

```
1 <xf:bind nodeset="age" constraint="string(number(.)) != 'NaN'" />
```

3.3 Form controls and containers

Form controls and containers define the "front end" of an XForms document that is presented to the user.

Form controls are similar to HTML form elements like input-fields or buttons. They are rendered on the screen and enable the user to edit data. But, unlike HTML form elements, they don't contain any data but refer to a specific node in the data model. This principle is known as the MVC pattern, introduced and described in [2].

Containers are logical components that can contain other controls or containers. They can be used to structure the document not only in a visual way, but also in a logical way. This has two benefits. On the one hand, the programmer can use containers to simplify his work. Using containers, nested elements can be treated together (e.g. to apply common attributes to all of them). On the other hand, a logical structure is important if the rendering is not only visual. Typical HTML documents rely on a visual structure using (e.g.) tables and `div`-tags, whereas an XForms document can rely on a more abstract structure based on containers. Whereas the visual structure is useless if the device or target changes dramatically, the more abstract structure of XForms can be adapted to the new rendering.

[2] Section 3.4 describes this process.

3.4 Events

HTML forms are static mark-ups for a user-interface riddled with data (see section 3.1). To give life to them, programmers have to use a scripting language like JavaScript. This brings up a couple of inconveniences:

- ▷ The use of a scripting language brings a new technology into the project that must be learned and handled.
- ▷ The maintenance of scripts is difficult, especially if the project is big and multiple scripts are used.
- ▷ Scripts are focused on the visual rendering and can hardly ever be used in another modality.

Nevertheless, the ability to execute simple "scripts" is necessary for modern websites. That's way XForms provides an event-mechanism with handlers and observers based on DOM Level 2 events and binding expression (see section 3.2) to deal with this requirement.

Events are a kind of message, thrown by a source element (e.g. a button). They are propagated along the DOM tree from the root down (the so called *capturing phase*) and from the target up again to the root (the so called *bubbling phase*). During this propagation, they can be caught by event-handlers. Whilst an event source can be any form control or binding expression, an event handler is a special tag (a so called *action*) linked to a specific event.

Listing 4 shows an event-handler that listens on an `xforms-invalid` event. Such an event is thrown e.g. if a binding expression as we see in section 3.2 is violated. If the event-handler caches such an event, a message is shown to the user.

The XForms specification defines a number of different events and actions that can be used to implement complex behaviors. Examples for important events are `xforms-invalid` (if a binding expression is violated), `xforms-submit` (if a submit-button is pressed) or `DOMFocusIn` (if a new form control is focused). Important actions are `message` (prompts a message to the user), `setvalue` (set a value to a node) or `setfocus` (set focus to a control).

Listing 4 An event-handler for a `xforms-invalid` event

```
1 <xf:message level="modal" ev:event="xforms-invalid">
2     Please enter a valid age!
3 </xf:message>
```

3.5 XFormsMM

Providing a complete separation of data and model as well as logical controls and containers, XForms serves as a suitable basis to build multi-modal applications

on top of it. An approach to this called XFormsMM was presented by Mikko Honkala and Mikko Pohja in their paper "Multimodal Interaction with XForms" [4].

The idea behind this approach is to extend XForms (including its data model, controls, etc.) with modality-dependent style-sheets (compare to figure 1). These style-sheets define the modality specific rendering for controls and grouping containers. For example a "visual" style-sheet describes how the controls should be organized on a screen whereas an "aural" style-sheet determines their sequential aural output.

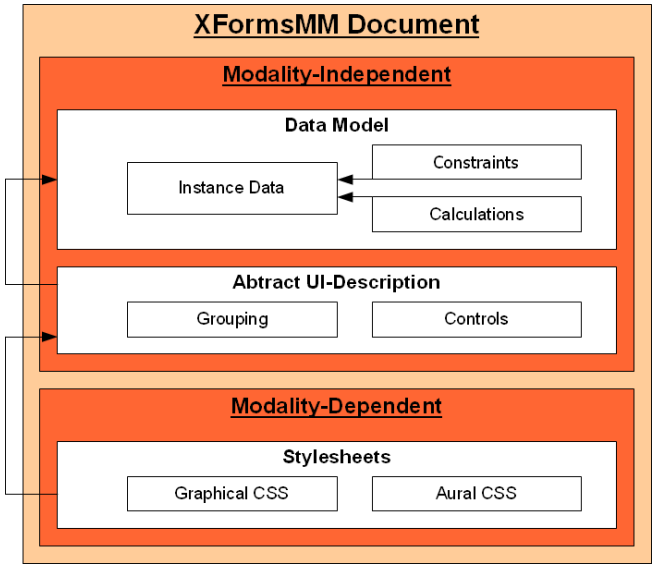

Fig. 1. Architecture of an XFormsMM document (based on [4])

In addition to their theoretical idea, Mikko Honkala and Mikko Pohja devised a reference implementation for the XSmiles web-browser[12]. Figure 2 shows the main parts of this implementation, which is described in the following.

An XForms processor provides the basis for the implementation.This processor performs the XML parsing, the XML Schema handling and other basic tasks needed for XForms. On top of this processor, there is a thin layer called *interaction manager*. It controls the complete communication and synchronization between the XForms document and the different rendering systems.

A rendering system provides a special view on the XForms document for a (single) modality. For example, the graphical rendering system provides a visual view whereas the aural rendering system provides a speech output (and respectively a visual or an aural input).

All rendering systems are operating on the same data model through the interaction manager. If a rendering system changes the model, the interaction man-

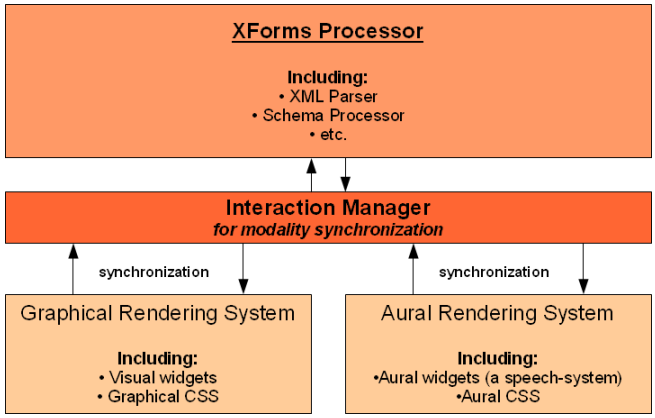

Fig. 2. Architecture of an XFormsMM implementation (based on [4])

ager will immediately inform the other rendering system(s) about that change, so that they can update themselves. This enables a simultaneous[3] and supplementary[4] use of the different modalities. A complementary[5] use is also possible, by excluding specific controls from a modality through the style-sheets.

4 An Example User Interface

The following part describes how a simple user interface for a questionnaire can be implemented using XForms. Beside the visual user interface (a web-site), an aural user interface according to XFormsMM will be given.

Figure 3 shows the example questionnaire rendered in Google Chrome using the FormFaces[5] framework. Due to the fact that all major browsers don't implement XForms currently, a framework must be used to translate the XForms code into common HTML. A certain amount of frameworks address this issue either on the server-side (e.g. betterForms or Orbeon Forms) or on the client-side (e.g. FormFaces). Though, only a short example for a user interface will be provided, the JavaScript-based framework FormFaces is used as no installation is necessary.

The questionnaire is based on the model introduced in section 3.1 including two binding constraints and a submit-method. All controls are referring to nodes of this model. The document is structured in two logical groups (marked in green and yellow in figure 3) using the `group` container described in section 3.3. This grouping is not visible in the visual rendering, as it is only a logical structure. But it's "visible" in the aural rendering as a branch, as shown in listing 5.

[3] This means that multiple modalities can be used at the same time. They will always synchronize each other through the interaction manager.
[4] This means that each modality can present any control.
[5] This means that not every control is rendered in every modality.

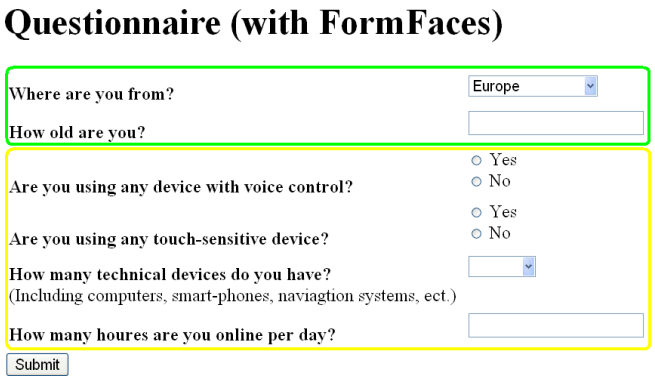

Fig. 3. Visual rendering of the example in Google Chrome

Listing 5 Aural rendering of the example according to XFormsMM

```
>> System: Edit user information, edit questionnaire, Submit.
<< User: Edit user information.
>> System:Edit location, edit age, back.
<< User:Edit location.
>> System:Select "Europe" or "USA", back.
<< User:Europe.
>> System:Selected "Europe". Select "Europe" or "USA", back.
<< User:Back.
>> System: Edit location, edit age, back.
<< User: Back.
>> System: Edit user information, edit questionnaire, Submit.
<< User: Submit.
```

Both renderings represent the same questionnaire with only one element containing information (the *location* is set to "Europe" in both of them). The visual rendering displays all controls at the same moment to the user whereas the aural rendering serializes the interface into a sequential output.

To serialize the interface into a suitable sequential output, the aural rendering system looks for so called focus points. A focus point can be a form control, but also a container including other controls. Only focus points on the same level are rendered together. This gives a nested structure to the aural output which allows a better orientation for the user.

5 User Interface Adaptation

XForms provides an abstract language to author a form document that can be presented by a rendering system. The particular rendering system is very free in the way of rendering the form controls. E.g. a selection list might be rendered by a browser in one of the three ways shown in figure 6.

The concrete rendering of a form control can be influenced by specific attributes or a style-sheet. E.g. the attribute `appearance` of a selection list can be labeled as `full`, `compact` or `minimal` to get the left, the middle or the right

Listing 6 Different renderings of a selection list (from [9])

rendering shown in figure 6. But in the end it's on the rendering system to decide the concrete rendering - there is no guarantee.

This situation leads to the following points:

- ▷ The adaptation of an XFORMS document is mainly done at runtime. The author creates a single abstract form document that can be interpreted by different rendering systems. Especially in the case of a multi-modal approach like XFORMSMM, the author doesn't even know which modalities are available at runtime.
- ▷ The adaptation can be influenced at development time. By providing style-sheets for an XFORMS document or for different modalities, the rendering can be specified more concretely by the author.

XFORMS and especially XFORMSMM adapt in different ways. The most obvious way is the adaptation to a *specific modality* which is the purpose of XFORMSMM. But also the adaptation to a *specific device* or a *platform* is possible, due to the fact that the concrete rendering of the form controls is not prescribed. An adaptation to *different languages* is also possible. Because of the integration of existing XML technologies, it is possible to externalize strings into an XML file and load them via an XPATH reference. This enables an easy change of the provided language for an XFORMS document.

6 Discussion

XFORMS achieved many improvements in the field of form authoring compared with HTML. It forces a clean separation between data and view, integrates existing XML technologies and provides a higher abstraction level for controls. Among others, this solves a couple of problems related to multimodality:

- ▷ The need of scripting languages can be reduced using events and bindings.
- ▷ A document can be structured in a logical way.
- ▷ The view gets independent from the data behind and vice versa.

As a single-authoring approach, XFORMSMM makes a good use of the advantages of XFORMS. Based on the separation of data and model, XFORMSMM provides the ability to use multiple views simultaneously on the same data. The only effort for the developer is the creation of style-sheets for each modality. But this also means that the supported modalities must be *known at development time* although the adaption is done at runtime.

From a pragmatic point of view, XForms is probably a "dead technology". XForms was supposed to become a part of XHTML 2.0, but due to technical and political reasons the development of XHTML 2.0 was stopped - the W3C is now focusing on HTML5. Today, XForms is barely used[6] and therefore it's not supported be nearly every major browser. Hence, it is questionable how the future of this technology will look like.

7 Conclusion

Web forms are an established element of today's web-sites - however, their authoring has different problems. The W3C standard "XForms" addresses these issues, as for example the separation of data and view. Due to this, XForms is a suitable basis to create forms that can be presented in several modalities, as well as simultaneously or supplementary.

XFormsMM from Mikko Honkala and Mikko Pohja and their reference implementation for the XSmiles browser is one approach to use this basis. It uses modality dependent style-sheets to present a single-authoring document in different modalities. The communication between them is handled by an interaction manager synchronizing the different views.

As an alternative draft to HTML5, XForms suffered a loss of relevance in the last few years. Although its good approaches - especially for multimodality - it's rarely used today.

References

1. Dubinko, M.: XForms Essentials. O'Reilly (2003), http://xformsinstitute.com/essentials/
2. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Addison-Wesley (1994)
3. Google: Google trends for "xforms", http://www.google.de/trends/?q=xforms
4. Honkala, M., Pohja, M.: Multimodal interaction with xforms. International Conference on Web Engineering pp. 201–208 (July 2006), http://lib.tkk.fi/Diss/2007/isbn9789512285662/article9.pdf
5. Kugelman, J.: Formfaces (9 2009), http://sourceforge.net/projects/formfaces
6. Pohja, M.: Comparison of common xml-based web user interface languages. Journal of Web Engineering 9, 23 (2009)
7. W3C: The forms working group, http://www.w3.org/MarkUp/Forms/
8. W3C: Xforms requirements (2001), http://www.w3.org/TR/xhtml-forms-req
9. W3C: Xforms 1.0 recommendation (October 2003), http://www.w3.org/TR/2003/REC-xforms-20031014/
10. W3C: Xml schema (2010), http://www.w3.org/XML/Schema
11. W3C: Xml path language (xpath) 2.0 (2011), http://www.w3.org/TR/xpath20/
12. XSmiles: X-smiles.org (2008), http://www.x-smiles.org/

[6] E.g. Google Trends shows a dramatic decrease for the keyword *XForms* [3].