Open and Accessible Presentations

Erik Wilde
School of Information
UC Berkeley, USA
Email: dret@berkeley.edu

***Abstract*—E-learning often is perceived as something that, on the technical level, can be addressed by designing an *e-learning system*, which often is equipped with a Web-based interface. We argue that this traditional approach of e-learning system design should be reversed in today's Web-oriented environment, in the sense that e-learning applications should be designed as well-behaving Web citizens and expose their services through nothing else but the Web's loose coupling principles. This article presents a system for Web-based presentations which follows this approach in publishing presentation material in a way that is as Web-friendly as possible. We show how such a system can be used as one building block in an e-learning infrastructure; replacing the traditional view of monolithic e-learning systems with an open and loosely coupled ecosystem of cooperating e-learning Web applications.**

I. Introduction

E-learning has come a long way from using standalone computers as basic training tools, progressing to the use of networked settings as an environment with better communications facilities, and is now facing the challenge of making the transition from the traditional understanding of e-learning applications as a class of standalone applications, to understanding e-learning as a discipline regarding the production, management, structuring, distribution, usage, and embedding of educational services and content within the context of the Web. E-learning thus is facing the same fundamental challenges as many other information-intensive disciplines: How to make the leap from continuing to build essentially closed systems with a Web interface, to building true Web-based systems where the fundamental approach is to design, build, and deploy services as part of the Web. This follows the general idea of publishing *Linked Data* on the Web [1], where data is made available in a form that is linked to other data, and that makes it easy for other data to link to it.

This article presents a solution to one specific but fundamental part of e-learning on the Web: how to create and present presentation material. In many cases, this is only one part of an e-learning environment, but it often is the hub from which supplemental resources are made available. The approach presented in this article focuses on structuring presentation material on the semantic level. The result are Web-based presentations which can be easily published, are embedded *into* the Web rather than just being published *on* the Web, but still maintain the logical structure of the presentation context, and contain additional metadata that allows applications to better access and reuse the presentation material. This approach can serve as a blueprint for other educational content as well. Ideally, e-learning applications should be designed to "blend into the Web" so that they can benefit from the Web, and that the Web can benefit from them.

In this article, we first investigate the fundamental challenges e-learning faces today. These challenges mostly arise from the dynamics and the loosely coupled nature of today's Web [2]. We then progress to investigating these challenges in more detail, looking at services and content as the two main problems in bridging the gap between e-learning systems and the Web as an information system. We then describe our solution for structuring presentation material, and continue with describing the way of how this content can be used for presenting content with today's browsers.

II. Web Information Management

E-learning started out as *computer-assisted learning*, by simply programming computers as training tools providing more functionality than paper or books. With the advent of computer networks, e-learning has followed two mainstream approaches, based on different assumptions about the setting. One approach is *distance education*, where learning material is provided to a dispersed set of users. The second approach is using computers to augment educational settings in which people still meet in person. Both approaches have a lot in common, but in distance education there typically is a heavier focus on electronic means to foster communications and collaboration.

The difference in terms of technologies is that distance education settings often have less control over the computers being used, whereas in-house systems might be able to rely on a standardized set of classroom computers. Increasingly, however, this assumption also is not true anymore, because students have their own computers and want to be able to access e-learning applications and content using their personal computers. The result is that both scenarios, distance education and in-house approaches, are facing the same challenge, which is how to build e-learning systems that are designed for the Web, and can be used with as few assumptions about the educational setting (and the technical environment) as possible.

Many e-learning systems resemble a specialized *Content Management System (CMS)*, with the ability to create, manage, and distribute educational content. This basic task of managing educational content is often augmented with additional *functional modules*, for example bulletin boards, chat tools, mail groups, and wikis. Overall, many e-learning tools closely resemble the popular notion of a *portal*, which typically is a flexible, configurable, and personalizable interface for a collection of managed content accessed through it.

This paper looks at a different approach: Instead of looking at an e-learning system as a portal-like CMS with a customizable interface, we look at it as a systematic way of how to publish Web content, or, more generally speaking, provide Web services. E-learning then moves from "using the e-learning system" to simply "using the Web," and the central question is how to make the Web more e-learning friendly. This approach is Web-centric and thus abandons the idea of a centralized e-learning system or platform altogether. Instead, an e-learning system is a set of loosely coupled Web services and Web content, which can be used in the same way as any Web service or Web content, but also is augmented to facilitate e-learning by specifically supporting the needs of the e-learning setting when designing the services and the content.

III. Web Services

A number of disciplines are currently in a transition phase from standalone or closed client/server approaches, to designing systems in a more open way. Typical examples for disciplines in the midst of that transition are *Geographic Information Systems (GIS)* or e-learning. These disciplines have a rich history of designing and implementing systems that are solving the specific problems they are investigating. Technologically, they are usually based on the state of the art in implementing user-facing software.

The current shift from *building systems* to *exposing content and services*,[1] however, requires these historically systems-oriented disciplines to look at completely approaches of how they expose their functionality. We see three possible stages of development for e-learning system design, and the system presented in this paper is a contribution to the third stage, which we think is the future of e-learning system design.

1) E-learning systems can be designed as closed systems which perform all their data management internally. This does not exclude the possibility to have Web-based user interfaces or import and export capabilities, but it essentially means that the system only fully supports data that is managed within the system. This design allows the implementation of functionality more easily, because everything can be managed locally, but is increasingly at odds with the abundance of information that is available on the Web, and the fact that a hard boundary between the e-learning system and an information resource as rich as the Web makes it hard for users to work with all this information as easily and fluidly as possible.
2) A more open approach is to design it as a system with programming interfaces, so that other programs can interact with it. This encourages the evolution of an ecosystem of applications interacting with the e-learning system, so that content and functionality available outside of the e-learning system can still be accessed and used by the e-learning system. This raises the question which technology such an API to build on, because in a typical educational environment, there is a multitude of other applications which might be candidates for interacting with the e-learning system.[2]
3) A RESTful approach to designing e-learning systems is to design all services in an e-learning system as *Web services* (in the REST sense of the word), so that for example there is no specific API for uploading a course syllabus to a central course catalog, but instead there is a syllabus document format, and the central course catalog fetches syllabi from individual course sites as required. Such an approach is more loosely coupled and open, and effectively promotes the evolution of an educational institution's Web as its "e-learning system."

[1]Indicators for this shift are *cloud computing* and *service-orientation*, both approaches to shift the focus away from building systems to providing serivces.

Our work focuses on a RESTful design approach to e-learning. The central premise in such a RESTful e-learning approach is to identify relevant resource types, and to design link-enabled representations for them. The main goal in a Web-based environment then is to have a representation that is semantically rich enough to allow repurposing of educational content in various contexts, and to have a representation which is usable by the end-user of such a system. On today's Web, this often translates to designing a vocabulary using the *Extensible Markup Language (XML)* which encodes the required semantics (our approach is described in Section V), and to define a transformation from that vocabulary into an HTML-based presentation which can be used interactively (Section VI describes how to present in HTML). As a general foundation of our specific content type, Section IV briefly describes how Web content fits into the REST model and what today's content support on the Web looks like in terms of features and constraints.

IV. Web Content

One of the basic assumptions of Web Architecture is that resources should be identifiable by URI. For the types of resources considered in this paper this translates to the requirement that presentations and parts within them should have URIs, so that users can refer to presentations or parts or even individual "slides" by URI, tagging them, annotating them, linking them with other resources, and generally using and reusing them to build their own network of resources relevant to the presentation material.[3]

In terms of Web technologies, HTML is used for publishing structured documents on the Web, but has only limited structuring and presentation support. *Cascading Stylesheets*

[2]In a university, for example, there are IT systems for student registration, for facility management for booking rooms, for library services, for collaborative tools for classes, and many other systems which often constitute a very diverse landscape of services.

[3]Google's recently launched *Sidewiki* is an example for a collaborative Web-based application that can be easily applied for content that is made available through well-identified Web resources.

(CSS) provide a language for controlling the presentation of documents, and modern browsers support a rather sophisticated set of CSS features. For the specific scenario discussed in this paper, however, there is only little support in today's CSS, because CSS so far has focused on scrollable documents. However, the increased standards-compliance of browsers makes it possible to use *JavaScript* for implementing functionality which is not (yet) supported by HTML or CSS. For presentation material, the first publicly available approaches to build toolkits for HTML-based presentations were the *Simple Standards-Based Slide Show System* (S^5) and *Slidy*. Both approaches focus on creating a single Web page for a presentation.

The missing part in that landscape was the ability to start from a structured description of a presentation, and use that for generating one or more presentations, which can then be generated and presented in a way which not only allows richer structures within one presentation to be exposed (such as presentation parts and links to them), but also associations between individual HTML representations of one presentation (references between lecture notes in the context of a course), links to resources closely associated with the presentation (such as images and examples), and of course links to any other resources available on the Web.

Creating a richer presentation format for presentations not only allows better support for generating Web content for users, it also allows the generation of better metadata, for example linking all presentations with document relationships, and linking all presentation material (presentations plus the associated resources) as one compound document. This means that instead of algorithmically determining the boundaries of compound documents on the Web [3], such a model makes it easier for other applications to retrieve declarative information about the boundaries of compound documents.

V. Structured Presentations

As mentioned earlier, the main goal of the work presented here is to expose information about presentation material in a Web-friendly way. Our approach for doing this uses a mix of HTML as the presentation language of the Web, and a custom XML vocabulary for expressing those concepts which we have identified as being relevant for presentation material. The language (and the publishing system for it) is called *Hotspot* and defines a rather small set of XML elements (about 20), which are used in combination with regular HTML markup.

Figure 1 shows an example of a Hotspot document. It is not fully representative for average Hotspot code, because it mainly shows Hotspot elements, whereas a large percentage of elements in an average Hotspot presentation will be HTML elements, using features such as paragraphs, lists, tables, and other basic HTML markup constructs. In addition to defining a basic set of elements for representing presentation semantics, Hotspot also allows to use custom elements and extensions (described in more detail in Section VII), making it possible to adapt it to scenarios that need additional semantics and/or features when publishing presentation material.

The following description looks at the document shown in Figure 1 in more detail. It explains some of the non-HTML elements of the documents, and how they represent semantics important for presentation material.

① Presentation structures are represented by nested elements, the `presentation` element identifies complete presentations, `part` elements can be further nested and represent logical parts of a presentation, and `slide` elements represent individual slides. These structures also provide support for linking to presentation structures.

② License information can be specified, which is useful for licensing the complete presentation as a whole (exceptions can be specified as well). The license information then can be exposed in a way which allows automated metadata discovery, for example by search engines looking for license information.

③ *Table of Contents (ToC)* can be generated in various ways, so that the structure of presentations are available in different formats. In our day-to-day use of Hotspot, we have one ToC in HTML for the course Web page, and one ToC in XML, which is used as import by an automated syllabus management system to get a list of dates and lectures for each course.

④ Using index pages and category information, it is possible to define various ways of making additional information available, or to provide alternative access paths to information. Typical examples for this are indices and glossaries, which both can be implemented by this mechanism.

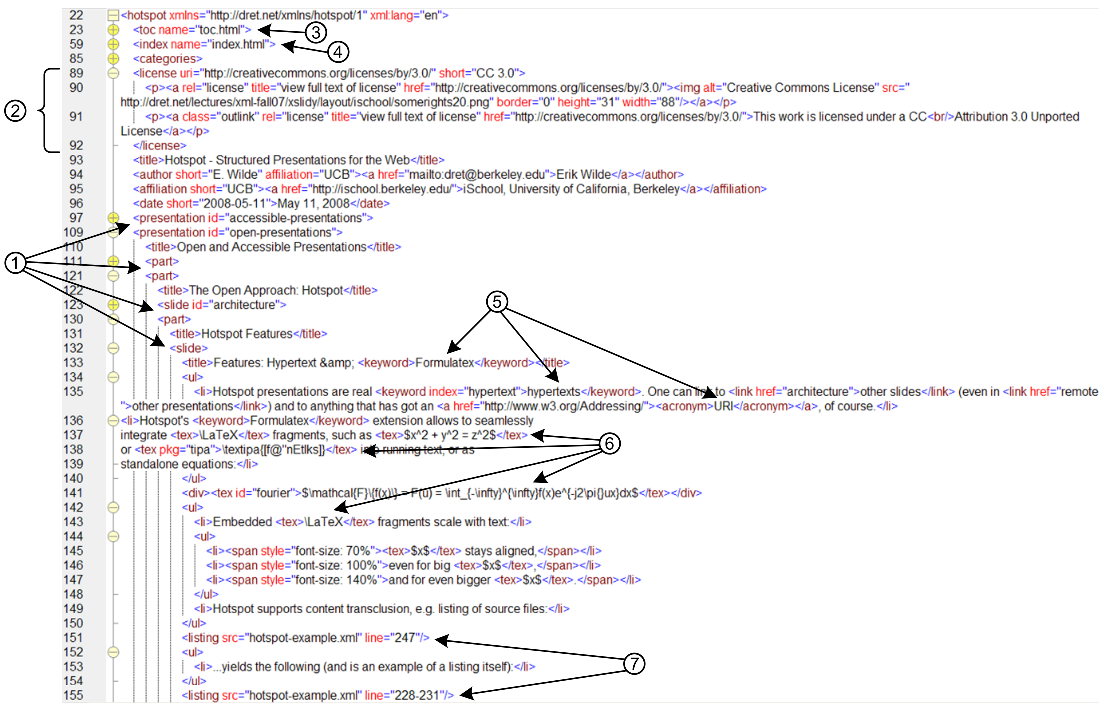
⑤ Specific elements can be combined with behavior and/or formatting, either simply for the sake of consistent formatting, or for additional processing, such as picking up content and compiling it for index or glossary pages.

⑥ For content that is not well-supported by browsers, extensions can be used which allow special content to be processed and embedded into presentations. In this example, an extension for processing $\LaTeX$ content is used to embed mathematical formulas and IPA characters into a presentation (see Section VII for details).

⑦ Transclusion is supported by a special element supporting the inclusion of external files (completely or as line ranges) while maintaining the information about their origin. This ensures that there are no inconsistencies between external files and the presentation.

Preparing and managing presentations in the format shown in Figure 1 is not trivial, but this is also due to the fact that presentation material by its very nature can be quite complex. For example, for a full semester course on XML technologies (27 lectures), the XML document is 400kB, using about 100 images and 100 transcluded examples. We currently do not have a specialized editor, so the source has to be edited by hand. This limits the potential user base, but we prefer to be able to easily tweak the vocabulary and its semantics over having to update a potentially complex editor whenever making changes to the system. We plan to add a specialized editor in the future, and since our vocabulary uses mostly HTML elements, we plan to adapt an existing HTML editor and extend it with support for the custom elements.

Once a presentation has been prepared in XML, it is



```
<hotspot xmlns="http://dret.net/xmlns/hotspot/1" xml:lang="en">
  <toc name="toc.html">
  <index name="index.html">
  <categories>
  <license uri="http://creativecommons.org/licenses/by/3.0/" short="CC 3.0">
    <p><a rel="license" title="view full text of license" href="http://creativecommons.org/licenses/by/3.0/"><img alt="Creative Commons License" src="
http://dret.net/lectures/xml-fall07/xslidy/layout/ischool/somerights20.png" border="0" height="31" width="88"/></a></p>
    <p><a class="outlink" rel="license" title="view full text of license" href="http://creativecommons.org/licenses/by/3.0/">This work is licensed under a CC<br/>Attribution 3.0 Unported
License</a></p>
  </license>
  <title>Hotspot - Structured Presentations for the Web</title>
  <author short="E. Wilde" affiliation="UCB"><a href="mailto:dret@berkeley.edu">Erik Wilde</a></author>
  <affiliation short="UCB"><a href="http://ischool.berkeley.edu/">iSchool, University of California, Berkeley</a></affiliation>
  <date short="2008-05-11">May 11, 2008</date>
  <presentation id="accessible-presentations">
  <presentation id="open-presentations">
    <title>Open and Accessible Presentations</title>
    <part>
    <part>
      <title>The Open Approach: Hotspot</title>
      <slide id="architecture">
      <part>
        <title>Hotspot Features</title>
        <slide>
          <title>Features: Hypertext &amp; <keyword>Formulatex</keyword></title>
          <ul>
            <li>Hotspot presentations are real <keyword index="hypertext">hypertexts</keyword>. One can link to <link href="architecture">other slides</link> (even in <link href="remote
">other presentations</link>) and to anything that has got an <a href="http://www.w3.org/Addressing/"><acronym>URI</acronym></a>, of course.</li>
<li>Hotspot's <keyword>Formulatex</keyword> extension allows to seamlessly
integrate <tex>\LaTeX</tex> fragments, such as <tex>$x^2 + y^2 = z^2$</tex>
or <tex pkg="tipa">\textipa{[f@"nEtlks]}</tex> into running text, or as
standalone equations:</li>
          </ul>
          <div><tex id="fourier">$\mathcal{F}\{f(x)\} = F(u) = \int_{-\infty}^{\infty}f(x)e^{-j2\pi{}ux}dx$</tex></div>
          <ul>
            <li>Embedded <tex>\LaTeX</tex> fragments scale with text:</li>
            <ul>
              <li><span style="font-size: 70%"><tex>$x$</tex> stays aligned,</span></li>
              <li><span style="font-size: 100%">even for big <tex>$x$</tex>,</span></li>
              <li><span style="font-size: 140%">and for even bigger <tex>$x$</tex>.</span></li>
            </ul>
            <li>Hotspot supports content transclusion, e.g. listing of source files:</li>
          </ul>
          <listing src="hotspot-example.xml" line="247"/>
          <ul>
            <li>...yields the following (and is an example of a listing itself):</li>
          </ul>
          <listing src="hotspot-example.xml" line="228-231"/>
```

Fig. 1. XML Representation of a Presentation

processed using *XSL Transformations (XSLT)*, transforming the XML/HTML document into HTML only. This HTML, however, then uses specialized formatting and scripting which allows browser-based presentations, providing support for remote presentations, timed logging of presentations, embedded presentations, and improved structuring and navigation.

VI. Presenting in HTML

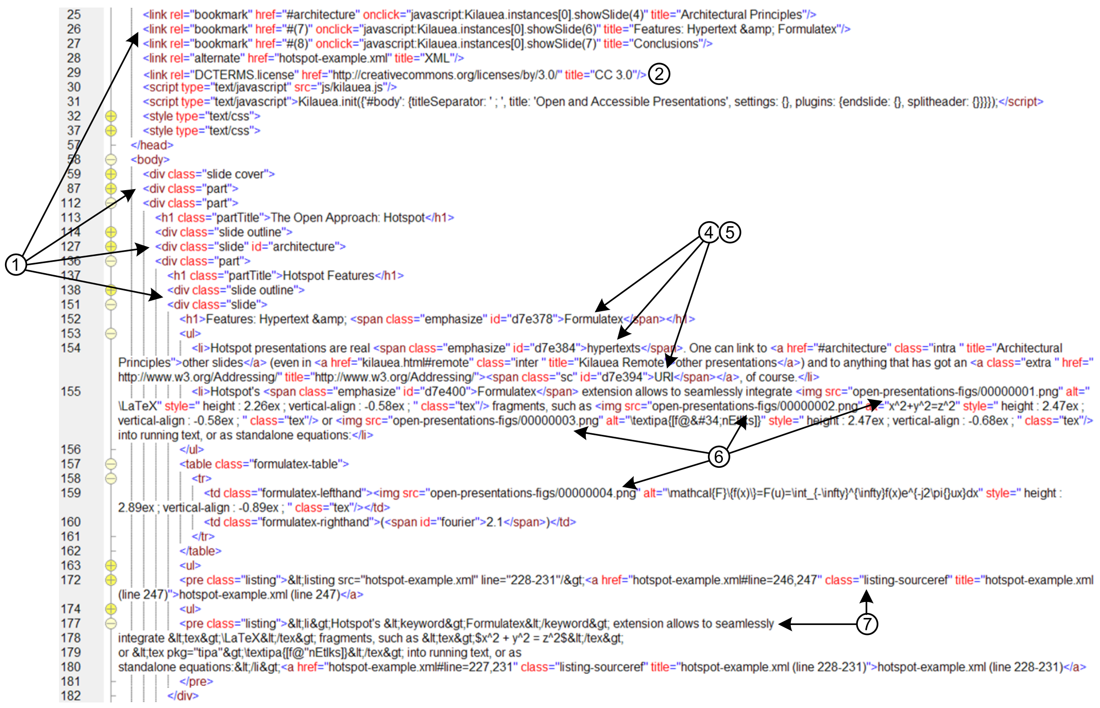
The XML/HTML document described in the previous section serves as input to a transformation which produces HTML for Web-based presentations. Because of the explicit presentation semantics in the source format, however, it would also be possible to author other transformations, for example producing a version that is targeted at high-quality print production of lecture notes.

Figure 2 shows the HTML that is generated from the document shown in Figure 1. The presentation structures (①) are represented by HTML container elements which can be picked up by styling and scripting. Additionally, HTML document relationship information is produced which can be used for navigating the presentation structures. The license information (②) is available as DC metadata and can be picked up by crawlers. Semantically marked up terms (⑤) appear with special formatting and can, if required, link back to index or glossary pages (④). Special content such as formulas (⑥) is represented by embedded images which scale with the font size. Transcluded content (⑦) is included in the HTML and is linked to the original, so that it is easy to navigate to the transcluded content.

The HTML shown in Figure 2 can be used in any modern browser to create an interactive, paged presentation, complete with all the features one would expect from on-screen presentations (zooming, paging back and forth, switching between paged and continuous display, following links). The HTML uses a JavaScript library to achieve the presentation effects. CSS-based templates can be used to apply a predefined layout for presentations. Figure 3 shows the rendering of the HTML from Figure 2, in this case using the Firefox browser. All major browsers can be used, but IE is limited by its weaknesses in JavaScript and CSS standards-compliance. Firefox, on the other hand, is well-known for being resource-intensive, so for large presentations (having hundred or more images), the more resource-aware Opera browser has proven to be a better choice.

Once loaded by the browser, scripting creates a paged presentation and allows users to navigate pages using keyboard or mouse navigation. The presentation structures (①) are used for formatting in the layout's header area, as well as for additional metadata which can be used by the browser to make the presentation more accessible.[4] The license information (②) is available directly through the information in the slide footer, but is also detectable as HTML metadata. An alternative way to using the HTML ToC, which is a separate HTML page, is accessing the ToC information through the HTML document relationships (③). Content that does not display well across browsers (such as mathematical formulas) is rendered as images (⑥) with carefully adjusted spacing and scaling so that it always displays correctly. Transcluded content (⑦) is styled in a way which makes its origins clear; it is also always linked

[4]In this example, the `cmSiteNavigation` extension of Firefox is used to make document relationships navigable.



```
    <link rel="bookmark" href="#architecture" onclick="javascript:Kilauea.instances[0].showSlide(4)" title="Architectural Principles"/>
    <link rel="bookmark" href="#(7)" onclick="javascript:Kilauea.instances[0].showSlide(6)" title="Features: Hypertext &amp; Formulatex"/>
    <link rel="bookmark" href="#(8)" onclick="javascript:Kilauea.instances[0].showSlide(7)" title="Conclusions"/>
    <link rel="alternate" href="hotspot-example.xml" title="XML"/>
    <link rel="DCTERMS.license" href="http://creativecommons.org/licenses/by/3.0/" title="CC 3.0"/>
    <script type="text/javascript" src="js/kilauea.js"/>
    <script type="text/javascript">Kilauea.init({'#body': {titleSeparator: ' ; ', title: 'Open and Accessible Presentations', settings: {}, plugins: {endslide: {}, splitheader: {}}}});</script>
    <style type="text/css">
    <style type="text/css">
  </head>
  <body>
    <div class="slide cover">
    <div class="part">
    <div class="part">
      <h1 class="partTitle">The Open Approach: Hotspot</h1>
      <div class="slide outline">
      <div class="slide" id="architecture">
      <div class="part">
        <h1 class="partTitle">Hotspot Features</h1>
        <div class="slide outline">
        <div class="slide">
          <h1>Features: Hypertext &amp; <span class="emphasize" id="d7e378">Formulatex</span></h1>
          <ul>
            <li>Hotspot presentations are real <span class="emphasize" id="d7e384">hypertexts</span>. One can link to <a href="#architecture" class="intra " title="Architectural
Principles">other slides</a> (even in <a href="kilauea.html#remote" class="inter " title="Kilauea Remote">other presentations</a>) and to anything that has got an <a class="extra " href="
http://www.w3.org/Addressing/" title="http://www.w3.org/Addressing/"><span class="sc" id="d7e394">URI</span></a>, of course.</li>
            <li>Hotspot's <span class="emphasize" id="d7e400">Formulatex</span> extension allows to seamlessly integrate <img src="open-presentations-figs/00000001.png" alt="
\LaTeX" style=" height : 2.26ex ; vertical-align : -0.58ex ; " class="tex"/> fragments, such as <img src="open-presentations-figs/00000002.png" alt="x^2+y^2=z^2" style=" height : 2.47ex ;
vertical-align : -0.58ex ; " class="tex"/> or <img src="open-presentations-figs/00000003.png" alt="\textipa{[f@&#34;nEtlks]}" style=" height : 2.47ex ; vertical-align : -0.68ex ; " class="tex"/>
into running text, or as standalone equations:</li>
          </ul>
          <table class="formulatex-table">
            <tr>
              <td class="formulatex-lefthand"><img src="open-presentations-figs/00000004.png" alt="\mathcal{F}\{f(x)\}=F(u)=\int_{-\infty}^{\infty}f(x)e^{-j2\pi{}ux}dx" style=" height :
2.89ex ; vertical-align : -0.89ex ; " class="tex"/></td>
              <td class="formulatex-righthand">(<span id="fourier">2.1</span>)</td>
            </tr>
          </table>
          <ul>
          <pre class="listing">&lt;listing src="hotspot-example.xml" line="228-231"/&gt;<a href="hotspot-example.xml#line=246,247" class="listing-sourceref" title="hotspot-example.xml
(line 247)">hotspot-example.xml (line 247)</a>
          <ul>
          <pre class="listing">&lt;li&gt;Hotspot's &lt;keyword&gt;Formulatex&lt;/keyword&gt; extension allows to seamlessly
integrate &lt;tex&gt;\LaTeX&lt;/tex&gt; fragments, such as &lt;tex&gt;$x^2 + y^2 = z^2$&lt;/tex&gt;
or &lt;tex pkg="tipa"&gt;\textipa{[f@"nEtlks]}&lt;/tex&gt; into running text, or as
standalone equations:&lt;/li&gt;<a href="hotspot-example.xml#line=227,231" class="listing-sourceref" title="hotspot-example.xml (line 228-231)">hotspot-example.xml (line 228-231)</a>
          </pre>
        </div>
```

Fig. 2. HTML Generated from XML shown in Figure 1

to the source material, allowing direct access to all resources.

Presentations created by Hotspot should be well-behaving Web citizens, using all the principles of good Web design in the sense of making as much information available through standardized structures as possible. We believe this already is a big step forward from the usual practice of posting presentations as PDF or other proprietary and opaque formats, which are harder to mine for information.

VII. Extensions

The main idea of Hotspot is to create Web-based presentations for complex presentation material. While this by itself is a useful functionality and only rarely addressed, our main application is educational material, and therefore, it is important that the presentation material can be customized and put in context to improve the quality of the e-learning content. Hotspot provides an extension mechanism which allows additional code to be plugged into the system in a simple way, so that both the XML-to-HTML conversion as well as the HTML presentation can be customized.

One thing that HTML is notorious for is its lack of support for mathematical formulas.[5] For courses that use mathematical formulas, we have thus developed the *FormuLATEX* extension for Hotspot. It accepts mathematical formulas in LATEX notation, and renders them as images, item ⑥ shows how this works in all three figures. In addition to additional XSLT which extracts all formulas from presentations, the extension requires additional Perl code which reads all the formulas, renders them using LATEX, and then crops and scales them so that they properly scale with the textual content of the presentation.

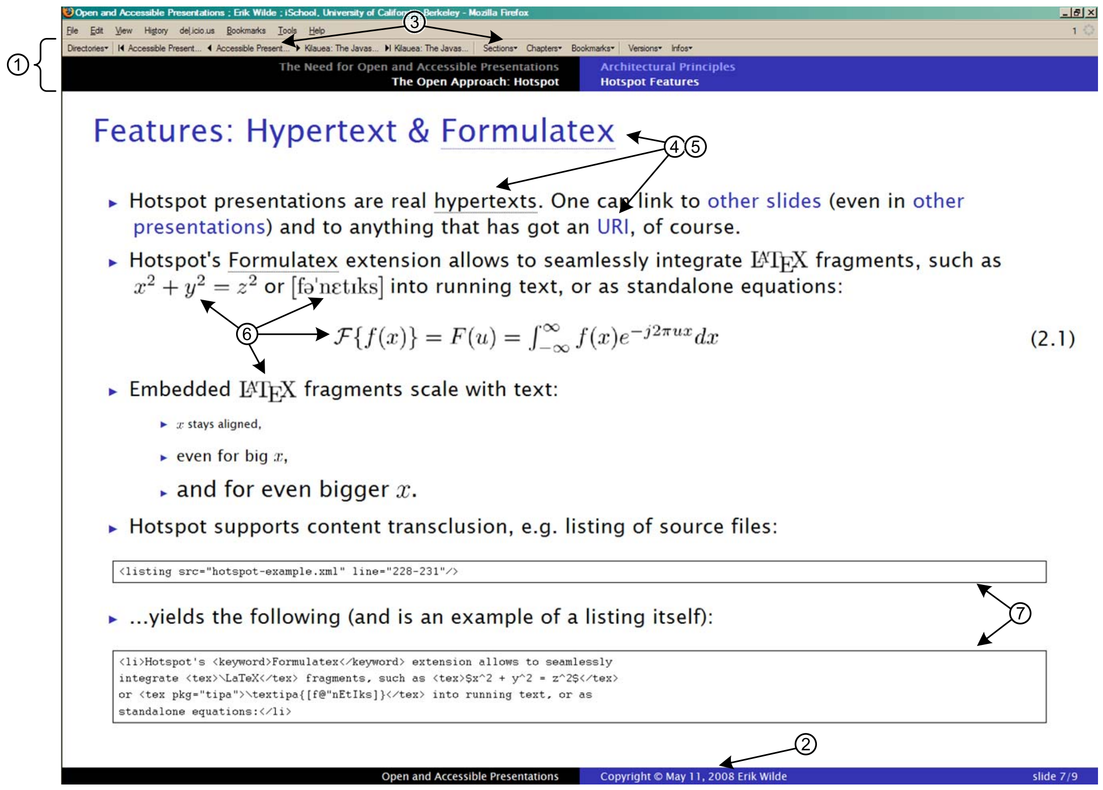
[5]This might start to change with the recently announced support for MathML [4] in WebKit.

VIII. Offline Presentations

One of the main properties of Hotspot is to rely on the Web as a platform. This means that along with the evolution of that platform, new functionality can be used by Hotspot. One example is the question of how offline presentations are possible with Hotspot. Traditionally, browsers access documents on HTTP servers or from the file system, so that offline presentations were possible by either running a local Web server, or copying all required files to the file system. However, on many of the newer devices (such as smartphones and tablet computers), installation of HTTP servers or even access to the file system is hard to do or impossible.

With the new extended functionality provided by the HTML5 family of technologies, one of the interesting opportunities is to publish *Offline Applications* [5] on the Web. Essentially, any Web resource can publish a list of required related resources, and browsers are then expected to locally store all these resources, so that the application becomes usable even when being offline. Since Hotspot is nothing but a set of resources required for presenting the material (essentially, HTML content, CSS styles, JavaScript code and auxiliary resources such as images or other media resources), it is easily possible to use the offline capabilities of new browsers to make presentations offline capable.

However, while the new capabilities are promising and very exciting for any collaborative e-learning projects, because they


Open and Accessible Presentations ; Erik Wilde ; iSchool, University of California, Berkeley - Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help
Directories Accessible Present... Accessible Present... Kilauea: The Javas... Kilauea: The Javas... Sections Chapters Bookmarks Versions Infos
The Need for Open and Accessible Presentations
The Open Approach: Hotspot
Architectural Principles
Hotspot Features
1
2
3
4
5
6
7

Features: Hypertext & Formulatex

- Hotspot presentations are real hypertexts. One can link to other slides (even in other presentations) and to anything that has got an URI, of course.
- Hotspot's Formulatex extension allows to seamlessly integrate LaTeX fragments, such as $x^2 + y^2 = z^2$ or [fəˈnɛtɪks] into running text, or as standalone equations:

$$\mathcal{F}\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}dx \tag{2.1}$$

- Embedded LaTeX fragments scale with text:
  - x stays aligned,
  - even for big x,
  - and for even bigger x.
- Hotspot supports content transclusion, e.g. listing of source files:

```
<listing src="hotspot-example.xml" line="228-231"/>
```

- ...yields the following (and is an example of a listing itself):

```
<li>Hotspot's <keyword>Formulatex</keyword> extension allows to seamlessly
integrate <tex>\LaTeX</tex> fragments, such as <tex>$x^2 + y^2 = z^2$</tex>
or <tex pkg="tipa">\textipa{[f@"nEtIks]}</tex> into running text, or as
standalone equations:</li>
```

Open and Accessible Presentations Copyright © May 11, 2008 Erik Wilde slide 7/9

Fig. 3. Browser Presentation of HTML shown in Figure 2

combine the Web's interlinking capabilities with the ability for users to take content and applications offline, there are not yet well-developed user interfaces and ways for users to control and manage the offline content they want to use. The idea of "Web App Stores" is only starting to develop, and our hypothesis is that once those models and the browser controls associated with them have matured, they will become very capable implementation platforms for combined online/offline scenarios such as e-learning.

IX. Future Work and Conclusions

As a major step towards establishing this landscape of loosely couple e-learning resources described in Section II, we plan to extend Hotspot to be able to cooperate with other providers of timed and/or dated information resources. However, instead of integrating Hotspot with for example software for managing assignments and lab sessions, our approach is to establish cooperation by using a data format which will be a representation of the data which is shared, essentially enables the loose coupling between Hotspot and any other provider of timed and/or dated information resources. This design established cooperation around e-learning resources through their accessible "surface" available on the Web itself, rather than through hidden back channels that would introduce tight coupling.

Hotspot is not a complete e-learning system, but it already has helped to make presentation material easier to manage for teachers and easier to use for students. The approach of Hotspot can serve as a blueprint for a more fully-featured e-learning system, which reverses the traditional approach of building a closed system and then developing an interface for it, to building and deploying functional modules which can be freely used on the Web, and are then tied together by identifying and cooperating around common concepts and resources being used in that context. Creating structured content always take a certain amount of effort, no matter how good an editor is being used. However, educational material typically has value over a rather long timespan, and we found the ability to reuse and adapt content from previous years or other presentations, and to transclude content without the need to manually keep things consistent, something that payed off in the long run.

References

[1] M. Hausenblas, "Exploiting linked data to build web applications," *IEEE Internet Computing*, vol. 13, no. 4, pp. 68–73, 2009.

[2] C. Pautasso and E. Wilde, "Why is the web loosely coupled? a multi-faceted metric for service design," in *18th International World Wide Web Conference*, J. Quemada, G. León, Y. S. Maarek, and W. Nejdl, Eds. Madrid, Spain: ACM Press, April 2009, pp. 911–920.

[3] P. Dmitriev, "As we may perceive: Finding the boundaries of compound documents on the web," in *17th International World Wide Web Conference Posters*. Beijing, China: ACM Press, April 2008, pp. 1029–1030.

[4] R. Ausbrooks, S. Buswell, D. Carlisle, S. Dalmas, S. Devitt, A. Diaz, M. Froumentin, R. Hunter, P. Ion, M. Kohlhase, R. Miner, N. Poppelier, B. Smith, N. Soiffer, R. Sutor, and S. M. Watt, "Mathematical markup language (mathml) version 2.0 (2nd edition)," World Wide Web Consortium, Recommendation REC-MathML2-20031021, October 2003.

[5] A. van Kesteren and I. Hickson, "Offline web applications," World Wide Web Consortium, Note NOTE-offline-webapps-20080530, May 2008.