HTML5 & CSS3

Barry Borsboom
Leiden Institute of Advanced Computer Science, Leiden University
barry.borsboom@gmail.com

David Graus
Leiden Institute of Advanced Computer Science, Leiden University
dpgraus@gmail.com

Sander Hutman
Leiden Institute of Advanced Computer Science, Leiden University
sjhutman@xs4all.nl

ABSTRACT

In this paper we concisely describe HTML5 and CSS3, the history and development process of these markup languages, we highlight some of their new features and functionalities, and try to offer an understanding of the application of the languages, as well as the implications these iterations of the markup languages have for the world wide web of the future.

1. PURPOSE, CONTEXT AND HISTORY

The HyperText Markup Language (‘HTML’) was primarily designed for semantically describing scientific documents. Since its birth in 1990, it has seen many revisions. The initial intended application for describing scientific documents has changed as well.

HTML5 is being developed to better suit the modern day web, in all its multi-medial and interactive glory, while also improving on the semantic structure of web content. Next to that, the development of HTML5 is the first attempt to formally document and unify the different features and standards of the markup language[http://diveintohtml5.org/introduction.html].

Examples of how HTML5 reaches the goals of improved semantic structure or how it caters to more modern day websites are some of the new elements and attributes. For example, web content in blocks is now largely handled by context-specific block elements, such as <nav> <header> and <footer> blocks, as opposed to <div>. Multimedia is better supported by means of integrated <video> and <audio> tags, integrating multimedia in the markup language, making embedding via external software such as Adobe Flash Player or the Microsoft Silverlight Player redundant.

Backwards compatibility with older technologies and increasing consistency of the language are main concerns of HTML5. The earlier HTML developers largely worked separately, decreasing coherence in the markup language. By properly supervising and controlling the development process, the two organizations involved in the development process – the Web Hypertext Application Technology Working Group (‘WHATWG’) and the World Wide Web Consortium (‘W3C’) - have a better possibility of designing the markup language in a more coherent and consistent manner.

Another important aspect is error handling. Other than HTML (which accepts and displays ‘broken’ HTML documents) and XHTML (which has so-called ‘Draconian Error Handling’, displaying an error to the user when the browser encounters an error in the page, instead of displaying the page).

CSS

Cascading StyleSheets (‘CSS’) are related to all of these markup languages (HTML, XHTML, HTML5). The stylesheets define the way the elements are displayed within an HTML document. As an external file, it can change the layout, style and appearance of an HTML document. So where HTML structures an online document, CSS handles the formatting and appearance.

CSS3 offers features that make some forms of graphic design redundant. Features such as rounding borders (by using the border-radius attribute), creating drop shadows under ‘boxes’ (by using the box-shadow attribute), changing opacity of elements previously had to be done with images in photoshop, but can now be used for dynamic content.

Context & History

- **1990** - Hyper-Text Markup Language (‘HTML’), the markup language of web documents has been developed by Tim Berners Lee from CERN. It was further developed throughout the years, by different instances and organizations.
- **1995 to 1998** – During these years, the development of HTML was largely in hands of the World Wide Web Consortium (‘W3C’). The W3C was founded by Tim Berners Lee to develop standards for the world wide web. In 1998, the W3C ceased development of HTML, and started a new effort in developing the so-called eXtensible Hypertext Markup Language, or XHTML.
- **1998** – In December 1998, the W3C published a draft called ‘Reformulating HTML in XML’. This draft was also known as XHTML 1.0, and it was in effect not much more than a reformulation of HTML in XML. No new features or attributes were added.
- **2000** – Early 2000, XHTML 1.0 became a W3C

Recommendation. It was finalized and went public. Later that year, XHTML 1.1 followed, and in 2001 version 1.2 was a W3C Recommendation.

There were some important consequences to the use of XHTML over HTML. The most important practical consequence was that XHTML used so called 'Draconian Error Handling'. This means that when a document contains erratic XHTML code, or lacks code, the web browser would stop processing the document, and display an error to the user. HTML did not, it was much more lenient towards errors in the document, browsers would try to display the document as best as they could. With an estimated 99% of all websites containing errors in their markup, the disadvantages of the error handling in XHTML becomes clear. What happened was that XHTML as W3C envisioned was not used. The W3C made it obligatory for web designers to give a new application/xhtml+xml MIME type to the header of the document, as opposed to the older text/html MIME type used for HTML. This made it possible for web designers to use the new features and syntax of XHTML, while not being prone to the harsh error handling.

- **2004** - During a W3C workshop several individuals presented the idea to continue evolving the HTML markup language, instead of carrying on development of the XHTML language. The idea was rejected by the W3C, who chose to continue developing XHTML.

  But that was not the end of it. The companies behind the individuals who announced the idea of improving HTML - Apple, Mozilla and Opera - assembled themselves in an organization called the Web Hypertext Application Technology Working Group ('WHATWG'). The WHATWG set out to properly supervise and lead the development process of HTML5.

- **2006 -** Finally In 2006, the W3C changed their view and announced they would work together with the WHATWG in their effort to develop 'HTML5'[http://dev.w3.org/html5/spec/Overview.html].

- **2008** – The W3C publishes its First Public Working Draft of the HTML5 specifications. These are specifications only, an ongoing process of documenting different features and functionalities of HTML5. These features however can already be finished, implemented and working in different web browsers[http://en.wikipedia.org/wiki/HTML5].

2. OPERATING PRINCIPLES

HTML5

The operating principles of HTML5 are similar to that of HTML. The markup language consists out of elements and attributes. All HTML5 documents start with a <!DOCTYPE HTML> declaration, to tell the browser how to read the document. The syntax of HTML5 is the same as that of HTML. It is also fully backwards compatible, meaning documents will not break when displayed by an older browser.

However, the most apparent changes to HTML or XHTML to web designers are the new structural elements. Instead of creating different content in a website specified by <div>-elements, with HTML5 there are context-specific structural elements, such as <header> for the (physical) header of a website, <nav> for the navigation of a website, <article> for an independent item of content, <aside> for a sidebar and displaying content related to the main document, and the <footer> element for marking up the footer of a website. Advantages of using these elements instead of current <div> elements is that these content-specific elements can support browsers to generate tables of contents/outlines or better assist users with disabilities in browsing the website. By putting content in its standardized proprietary element, data is easier to find, access and process. It is another example of standardization, what could be accomplished by using arbitrarily named <div>-elements is standardized in HTML5, into a set amount of content-specific elements.

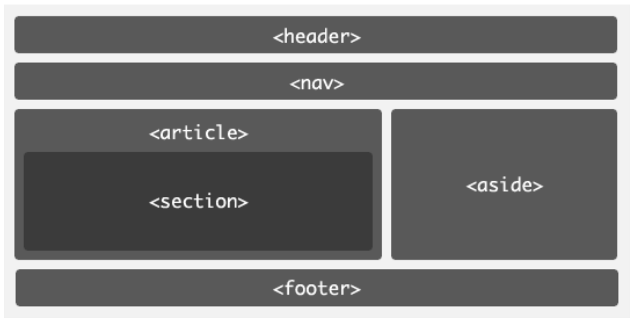

Figure 1.
[http://www.alistapart.com/articles/previewofhtml5]

Some more advanced features are supported by the APIs that HTML5 uses. An API is essentially an interface of one software program to interact with another piece of software; essentially a bridge between the two. We will further go into the use of the APIs in the Strengths and weaknesses section.

Another part of HTML5 which is notably upgraded are the forms. By specifying an <input type= > the browser knows a specific input box is used. Now there are specific input fields for email, urls, search queries, there are date pickers and color pickers. This is another example of standardization. All web forms can use the same specific input fields, opening the possibility to let the browser automatically fill in web forms for the user. These web forms will also display in older browsers, because all <input type= > tag with an unknown type (for example <input type=email>) will be treated as an <input type=text> by default[Pilgrim, M. Dive into HTML5. O'Reilly Media, http://www.diveintohtml5.com]!

HTML5 is much like regular HTML and can be created with a slew of different software packages, all with different levels of support. From the simplest raw text editor such as Microsoft Windows' Notepad, to the more elaborate software packages such as Adobe's Dreamweaver.

CSS3

CSS has a different syntax to HTML, but the syntax from CSS2 to CSS3 did not change much. CSS3 is mostly defined by new features, it is more of an upgrade to a language than a reinvention of it.

CSS works by means of statements. The statement identifies the element within the html document, and 'tells' the browser how to display it. It does so by assigning values to (possibly multiple) properties. The syntax is as follows:

selector { property1: value1; property2: value2 }

The CSS can be either inside a html document (in the <head> part), or in an external .css file. This css file is linked inside the html <head> element by a link with a type="text/css" definition as follows: <link rel="stylesheet" type="text/css" href="url/style.css" />

Interesting to note is that a lot of the new selectors of CSS3 make graphic design redundant. Now CSS can round corners of boxes, create dropshadows for them (and for text), etc. These are features that normally would have to be done by a graphic designer, by the use of images instead of code.

3. STRENGTHS AND WEAKNESSES

The most obvious benefit of HTML5 are the new APIs and the opportunities they open up for the future of web apps. Google Gears gave us offline data storage and Flash introduced us to the power of application cache (Pandora uses it to save your log in information). With HTML5, these capabilities are now available to use right in the language and can easily be expanded with JavaScript.

- **Offline Capability API** Programs like Thunderbird, Mail and Outlook let you browse through your old mail data while staying offline. With HTML5, you'll have this same functionality in the browser. This is the first serious step towards bridging the gap between the desktop and the Web, and opens all sorts of doors for the future of Web apps.

Other notable new APIs are:

- **Canvas** Canvas consists of a drawable region defined in HTML code with height and width attributes. JavaScript code may access the area through a full set of drawing functions similar to other common 2D APIs, thus allowing for dynamically generated graphics. Some anticipated uses of the canvas include building graphs, animations and image composition.
- **Drag & drop** The drag and drop API defines an event-based drag and drop system. However, it never defines what "drag and drop" is. This API requires JavaScript to fully work as normal think drag and drop functionality.
- **Video & audio** The audio & video APIs are massive upgrades in media embedding. Although support is limited right now, something like video embedding has never been easier.
- **Geolocation** Geolocation can be used to programmatically determine location information through a device's user agent. This could prove very useful for mobile devices.

On top of these new APIs HTML5 also introduced the new semantic structure tags, these tags will offer a unified way to build a page. Which does not only increase the readability for humans but also for computers. Voice Over functionality of websites for instance will be increased. The new API's and semantic structure will work in all modern browsers as well as a mobile browser on an iPhone or iPad.

But because HTML5 is not yet standardized by the W3C, not all browsers support the same new functions and capabilities. To find out exactly which browser supports what there are various sources to check[http://html5readiness.com]. But a real problem lies with the users who keep using older browsers. On the other hand, since HTML5 is fully backwards compatible, it will not break websites for the user, but only make them 'lack' certain features.

However, arguably this backwards compatibility has a weakness as well, currently the only way to make sure older browsers display embedded videos too is to use an Adobe Flash embedded player or similar. A <video> tag will not break the page, but it will not contain a video for older browsers. The old way, by using external embeddable players will work for both new browsers, supporting HTML5 and old ones.

Another debate going on right now is the debate between which video codec should be the standard for the video tag. Chrome, Safari and Internet Explorer want to use the licensed H.264 codec, but Opera and Firefox want to use the open source OGG codec. The debate is partly about licenses, H.264 being a 'licensed' codec vs. OGG's 'open' codec. The latter companies are afraid 'hidden patents' exist and lawsuits will follow as soon as W3C and the

WHATWG pick OGG.

This debate might cause a delay for big content providers to switch their content from flash video to html5 ready video streaming. Another sound inside this debate is whether HTML5 should standardize any codec at all, or keep it open and support multiple codecs. As the debate is still going on, it's hard to say what way it will go, but it is interesting to see the different interests of the companies involved entangled. For example, Apple is part of the group of patent holders of the H.264 codec. They opposed to using the OGG Theora codec.

CSS

The major advantage of CSS is it keeps html documents cleaner. Imagine if you'd want all the <h1>-elements inside a html document to be displayed in blue. Without CSS you'd have to repeatedly write this property for each <h1> element in the HTML document. With CSS you define it once, and the browser applies it to each <h1> element. It keeps the html document cleaner and properly splits the markup for form and the markup for content.

4. INTENDED APPLICATIONS

Some well know examples of websites that already use HTML5 are GMail, Google Wave and Youtube. They make good use of some of the new functions and capabilities of HTML5. Like Drag & Drop, Offline mode and the new Video tag. More functional examples can be found on various websites[http://html5demos.com]. As HTML5 is not yet released as an official W3C recommendation, it remains to be seen how the markup language will be applied broadly. It is clear however that the main intended applications is building websites and web apps which are semantically transparent and structured (at least from the back-end) and interactive.

The iPhone browser makes some clever use of the new webforms functionality. By taking the input type into account, the virtual keyboard will be adapted towards providing easier input. For instance an <input type=email> will make the keyboard display a '@' and a smaller spacebar, <input type=website> will add a '.com' button to the keyboard.

The intended applications for HTML5 are broad, and largely undefined. HTML5 is such a thorough rework of the markup language that time will tell how else some of the new features and functionalities will be applied.

5. UNINTENDED APPLICATIONS

Unintended application are a bit harder to find. An example can be found with the usage of the canvas element[http://www.kesiev.com/akihabara/]. Although the canvas can perfectly be used for this sort of applications it is not directly intended as a programming environment for video games.

6. GETTING STARTED

To get started using HTML5 and CSS3 there are a few easy steps. First of all start a new HTML document with the following doctype:

```
<!DOCTYPE html>
```

After that you can use a the new HTML5 semantic structure tags, and make use of the new APIs. It is important to take into account what features the different browsers support as long as HTML5 is not published as an official W3C Recommendation yet. We can assume by the time HTML5 is published officially, all browsers will support the new syntax.

However in the mean time, plenty of new syntax can already be applied and used in websites. For example the new webforms, there is no reason not to use them yet as they wo not change and already will function no matter what browser you use. The same does not go for some of the APIs, <video> and <audio> elements can only be used experimentally so far, or with backup of a current solution for embedding media.

7. FINAL THOUGHTS

The features HTML5 and CSS3 bring are very interesting, but since both languages are still actively being developed, it's hard to say how exactly they will be applied. The possibilies seem broad enough, given the various projects which already entail using HTML5 in 'unexpected ways' (see note [http://www.kesiev.com/akihabara/]).

It is also interesting to see a language which is not 'officially published' yet already being put to use. With the different browsers already competing with each other on support for an unfinished language. The backwards compatibility is a major reason for this to be possible.

HTML5 introduces new elements and ways of structuring websites and applications, but its not until everyone widely migrates to using HTML5 that we can say what exactly the impact of the markup language is, and how it will change web apps. Offline capability already hints towards the fading of the border between online and offline content, and a potential paradigm-shift in the use of websites as we do today. The potential is there, but the way it will (re)shape the web remains to be seen.

REFERENCES

1. http://diveintohtml5.org/introduction.html
2. http://dev.w3.org/html5/spec/Overview.html
3. http://en.wikipedia.org/wiki/HTML5
4. http://www.alistapart.com/articles/previewofhtml5
5. http://html5readiness.com
6. http://html5demos.com
7. http://www.kesiev.com/akihabara/
8. Pilgrim, M. Dive into HTML5. O'Reilly Media,

http://www.diveintohtml5.com