HTML5

An introduction

- ★ W3C stopped developing HTML in 1999
- ★ W3C thought the future would be XML
- ★ XHTML2 is a beautiful and pure spec, but has no relation to what we actually do, and none to what browsers do.
- ★ A group of people from Mozilla, Opera and Apple, under the lead of *Ian Hickson*, started the working group **WHATWG** and created a spec called **Web Applications 1.0**

The web is moving from a paged document environment to a rich, interactive and forward-thinking web

HTML5

An application centered language

HTML5 is a spec

About 900 pages

HTML5 spec

Don’t try to read or print it, because it is mostly for UA implementors

- **Replaces everything** that came before (HTML4 and XHTML 1.0)
- Replacement of DOM2HTML (DOM)
- No longer looking at a DOM, but at **a number of API's**

- HTML5 **starts where we are today**, not where some academics who created XHTML2 want us to be
- **Expanding HTML**, to prevent people turning to plugins like Flash and Silverlight, simply because HTML didn’t provide an answer to that

DOCTYPE

<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

DOCTYPE

<!DOCTYPE html>

- ★ Is the shortest notation to put the browser in standards mode
- ★ No DTD
- ★ No version number
- ★ Stick to uppercase ‘DOCTYPE’

- ★ HTML5 uses syntax compatible with both HTML4 and XHTML 1.0
- ★ HTML syntax is served as text/html
- ★ HTML5 can use XML syntax (like XHTML 1.0)

This is valid HTML5:

```
<!DOCTYPE html>
<meta charset=UTF-8>
<title>Page title</title>
<p>Look at my sloppy code!</p>
```

WTF?

Where’s the html, head and body tag?

- If you look under the hood, with Firebug, DragonFly or the Web Inspector, then you'll see that the browser inserts these tags automatically

- You are, however, not advised to write HTML like that, but just know you can...

- ★ The XHTML validator cared about uppercase/ lowercase, quotes around attributes and trailing slashes.
- ★ In reality: the browsers never cared

XHTML notation

Still, I would personally advise you to use the stricter XHTML notation, because of its consistency and cleaner code.

HTML5 elements

The new kids in town

HTML5 elements

for better structure

<section>

- ★ The section element represents a generic section of a document or application.

- ★ A section, in this context, is a **thematic grouping of content**, typically with a **heading**.

<article>

- a **self-contained composition** in a document, page, application, or site and that is intended to be **independently distributable** or reusable, e.g. in syndication.

- For example: a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, ...

<nav>

- ★ The nav element represents a section of a page that links to other pages or to parts within the page: a section with navigation links.

- ★ Only sections that consist of **major navigation blocks** are appropriate for the nav element.

<aside>

- ★ Represents a section of a page that consists of content that is **tangentially related to the content** around the aside element, and which could be **considered separate** from that content.
- ★ Can be used for typographical effects like pull quotes or sidebars, for advertising, for groups of nav elements, and for other content that is considered separate from the main content of the page.

<hgroup>

- Represents the heading of a section. The element is used **to group a set of h1–h6 elements** when the heading has multiple levels, such as subheadings, alternative titles, or taglines.

<header>

- ★ Represents **a group of introductory or navigational aids**.

- ★ Is intended to usually contain the section's heading (an h1–h6 element or an hgroup element), but this is not required.

- ★ Can also be used to wrap a section's table of contents, a search form, or any relevant logos.

<footer>

- ★ Represents a footer for its nearest ancestor sectioning content or sectioning root element.

- ★ Typically contains **information about its section** such as who wrote it, links to related documents, copyright data, and the like.

- ★ Footers don’t necessarily have to appear at the end of a section, though they usually do.

<address>

- ★ Represents the **contact information** for its nearest article or body element ancestor.
- ★ Must not be used to represent arbitrary addresses (e.g. postal addresses), unless those are in fact the relevant contact information.
- ★ Typically, it would be included along with other information in a footer element.

<figure>

- ★ Represents some flow content, optionally with a caption, that is self-contained and is typically referenced as a single unit from the main flow of the document.

- ★ Can be used to annotate illustrations, diagrams, photos, code listings, etc, that could, without affecting the flow of the document, be moved away from that primary content, e.g. to the side of the page, to dedicated pages, or to an appendix.

<figcaption>

★ Represents a **caption or legend** for the rest of the contents of the figcaption element's **parent figure element**, if any.

<small>

- ★ The small element represents **side comments** such as **small print**.

- ★ Small print typically features disclaimers, caveats, legal restrictions, or copyrights. Small print is also sometimes used for attribution, or for satisfying licensing requirements.

- ★ It is only intended for **short runs of text**.

<cite>

- ★ The cite element represents the **title of a work** (e.g. a book, a paper, an essay, a poem, etc).

- ★ This can be a work that is being quoted or referenced in detail (i.e. a citation), or it can just be a work that is mentioned in passing.

- ★ A person’s name is not the title of a work and the element must therefore **not be used to mark up people's names**.

<q *cite*>

- ★ The q element represents some **phrasing content quoted from another source**.
- ★ Content inside must be quoted from another source, whose address, if it has one, may be cited in the *cite* **attribute**. The source may be fictional, as when quoting characters in a novel or screenplay.

HTML5 elements

for other purposes

<time *datetime pubdate*>

- ★ Represents either a **time on a 24 hour clock**, or **a precise date** in the proleptic Gregorian calendar, optionally with a time and a time-zone offset.
- ★ Way to encode modern dates and times in a **machine-readable way**

<time *datetime pubdate*>

★ The *datetime* attribute, if present, gives the date or time being specified. Otherwise, the date or time is given by the element's contents.

★ The *pubdate* attribute is a **boolean attribute**. If specified, it indicates that the date and time given by the element is the publication date and time of the nearest ancestor article element, or, if the element has no ancestor article element, of the document as a whole.

<time *datetime pubdate*>

```
<time datetime="2007-10-05">October 5</time>

<p>I usually have a snack at
<time>16:00</time>.</p>

<article>
  <h1>Small tasks</h1>
  <footer>
    Published <time pubdate>2009-08-30</time>.
  </footer>
  <p>I put a bike bell on his bike.</p>
</article>
```

<mark>

- ★ Represents a **run of text** in one document **marked or highlighted** for reference purposes.

- ★ Can for example be used to highlight words that match some search string.

<ruby>

- ★ “Ruby” are **short runs of text** alongside the base text, typically used in East Asian documents to **indicate pronunciation or to provide a short annotation**.
- ★ So **not for embedding ruby code** (the programming language) into your document…

<wbr>

★ Represents a **line break opportunity**.

```
<p>And so they sang: &quot;RubyRubyRuby<wbr>Ruuubbbyyy&quot;</p>
```

<canvas>

- ★ A resolution-dependent bitmap canvas
- ★ Basically a rectangle in your page where you can use JavaScript to draw anything you want.
- ★ Can be used for rendering graphs, game graphics or other visual elements on the fly.
- ★ Has two attributes: *width* and *height*

<canvas>

```
<canvas width="400" height="150"></canvas>

<!-- provide alternative content -->
<canvas id="clock" width="150" height="150">
    <img src="img/clock.png" width="150" height="150" alt="Clock" />
</canvas>
```

```
// Reference the canvas object
var canvas = document.getElementById('clock');

// Check for support
if(canvas.getContext) {
    // Get the context
    var context = canvas.getContext('2d');
}
```

<canvas>

```
<html>
 <head>
  <script type="application/javascript">
    function draw() {
      var canvas = document.getElementById("canvas");
      if (canvas.getContext) {
        var ctx = canvas.getContext("2d");

        ctx.fillStyle = "rgb(200,0,0)";
        ctx.fillRect (10, 10, 55, 50);

        ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
        ctx.fillRect (30, 30, 55, 50);
      }
    }
  </script>
 </head>
 <body onload="draw();">
   <canvas id="canvas" width="150" height="150"></canvas>
 </body>
</html>
```

Video in HTML5

It’s rather complicated

<video>

- ★ For embedding video into a webpage
- ★ One <video> element can link to multiple video files, and the browser will choose the first video it can actually play.
- ★ It is up to you to know which browsers support which containers and codecs.

<video>

- ★ There is **no single combination** of containers and codecs **that works in all HTML5 browsers**.
- ★ This is not likely to change in the near future.
- ★ To make your video watchable across all of these devices and platforms, you're going to **need to encode your video more than once**.

<video>

```
<!-- Basic embedding -->
<video src="somefile.ogv"></video>

<!-- Specify video dimensions and enable controls -->
<video src="somefile.ogv" width="320" height="240" controls></video>

<!-- Video will start downloading (but not playing) as soon as page loads -->
<video src="somefile.ogv" width="320" height="240" preload></video>

<!-- Enable autoplay -->
<video src="somefile.ogv" width="320" height="240" autoplay></video>
```

<video>

```
<!-- Realistic example -->
<video width="320" height="240" controls>
    <source src="somefile.mp4" type="video/mp4; codecs='avc1.42E01E, mp4a.40.2'" />
    <source src="somefile.webm" type="video/webm; codecs='vp8, vorbis'" />
    <source src="somefile.ogv" type="video/ogg; codes='theora, vorbis'" />
    <!-- provide Flash player fallback here -->
</video>
```

- ★ The benefit of specifying the *type* **attribute** is that the browser will **check it first**.

- ★ If the browser decides it **can't play** a particular video, it **won't download** the file.

HTML5 forms

Wassup with that?

New input types

Backwards compatible

<input type=email>

- ★ Tells the browser not to submit the form, unless the user has entered a valid email address

- ★ *multiple* attribute is allowed, which means the field can be a list of comma-seperated valid email addresses

<input type=url>

- ★ Causes the browser to ensure that the entered field is a correct URL.

- ★ Browser may offer the user assistance and show a list of recently visited URLs.

`<input type=date>`

- ★ Dates were tricky for a user to enter and therefore we relied on JavaScript solutions for displaying a datepicker.

- ★ Browsers will display a (popup) calendar widget.

- ★ Browser will now also know what you mean. Previously, datepickers were just a collection of `<div>`s, `<span>`s, and links with lots of JavaScript behavior attached.

Date / time related input types

```html
<!-- for calendar date pickers -->
<input type="date" />

<!-- for months -->
<input type="month" />

<!-- for weeks -->
<input type="week" />

<!-- for timestamps -->
<input type="time" />

<!-- for precise, absolute date/timestamps -->
<input type="datetime" />

<!-- for local dates and times -->
<input type="datetime-local" />
```

<input type=number>

- ★ Throws an error if the user doesn't enter a numeric value.
- ★ Works with *min*, *max* and *step* attributes.
- ★ Browsers (such as Opera) can render it as a spinner control.

`<input type=range>`

- ★ Renders as a slider.
- ★ Used to be JavaScript driven elements, but now the responsability of creating **accessibile sliders** is moved from the developer to browser vendors.

<input type=search>

- ★ Expects a search term.
- ★ Attribute *placeholder* can be provided. This text will only display when the search field has no focus.
- ★ In Safari on Mac, it takes the default OS search box appearance. This can be overwritten:

```
input[type="search"] { -webkit-appearance: textfield; }
```

<input type=tel>

- ★ Expects a telephone number.
- ★ Doesn’t enforce numeric-only input.
- ★ As mobile phones “know” their own number, most of them might autocomplete this field.
- ★ The iPhone brings up a telephone input keyboard.

<input type=color>

- ★ Allows the user to input a color value via a picker.
- ★ Current support is very limited.

New attributes

To spice your elements

The list attribute

- ★ Hooks up with a new element <datalist>
- ★ Reminiscent of a select box, but user can input own values, if they don't want to choose one on the predefined options.
- ★ The id of the <datalist> is referenced in the vaue of the *list* attribute.
- ★ It's possible to **dynamically repopulate** the options as the user types and thus recreating a **Google Suggest** functionality.

The list attribute

```
<input id="salutation" type="text" list="salutations">

<datalist id="salutations">
    <option label="Mr" value="Mr">
    <option label="Ms" value="Ms">
    <option label="Prof" value="Professor">
</datalist>
```

★ Example on Google Suggest

Other attributes

```
<!-- placeholder attribute -->
<input type="search" placeholder="Zoeken..." />

<!-- required attribute -->
<input type="email" required />

<!-- multiple attribute (for multiple uploads or email addresses) -->
<input type="file" multiple  />

<!-- pattern attribute (match input to custom regular expression) -->
<!-- you can include a hint: placeholder="9AAA" -->
<input pattern="[0-9][A-Z]{3}" title="A digit follow by three uppercase letters" />
```

So, when can we start using HTML5?

So, when can we start using HTML5?

Today!

Using HTML5

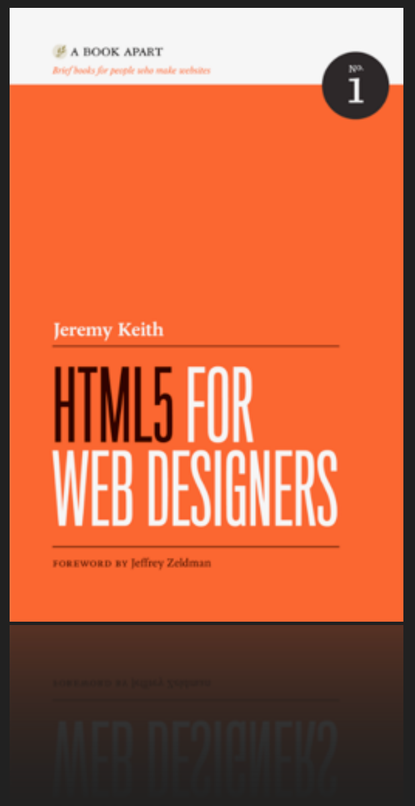
★ Modernizr

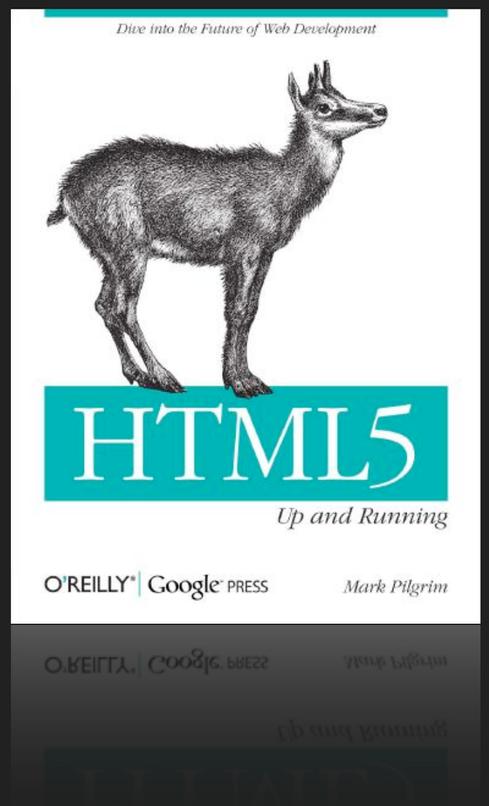
Modernizr is a small and simple JavaScript library that helps you take advantage of emerging web technologies (CSS3, HTML5) while still maintaining a fine level of control over older browsers that may not yet support these new technologies.

★ HTML5 Boilerplate

A rock-solid default for HTML5 awesome.

Interesting books





Interesting websites

- ★ http://html5doctor.com
- ★ http://diveintohtml5.org
- ★ http://html5demos.com
- ★ http://www.html5rocks.com
- ★ http://caniuse.com

The end

Questions?