HTML 5 Workshop

Nathaniel T. Schutta
@ntschutta

The Plan

- What is HTML 5?
- Why do I care?
- What can I do?
- How do I do it?

Assume you have:
browser, text editor.

Assume you have:
browser, text editor.

Copy zip.

Or clone it from github.

git clone git://github.com/ntschutta/html5_workshop.git

Extract to...somewhere ;)

Shout if you have ??s

What is it?

There was an HTML 4?

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
```

Remember XHTML?

Lack of features.

Browsers are forgiving.

Web flourished
because of it.

Lots of “broken” pages.

Draconian error handling.

Not backwards compatible.

Syntax was adopted.

HTML little long in the tooth...

And wasn't designed for applications.

Pushing boundaries.

De facto standards...

XHR anyone?

Standards aren't always clear cut.

Can contradict.

Not a conspiracy against developers.

Well not entirely.

Evolve over time.

A conversation.

Browser implementors, designers, standardistas.

Often a reaction to what we're doing.

We say we want standards...

Really want browser consistency.

Pain isn't standards, it's implementation.

2004 W3C workshop.

What should we do?

Evolving HTML lost.

Formed a new group.

WHAT Working Group.

Web apps!

Reversed engineered, and documented, parsing.

Web forms.

Canvas, audio, video tags.

Work continued on XHTML 2.0.

You probably didn't notice.

WHAT WG had momentum.

W3C joined the effort.

Thus was born HTML5.

So what is it again?

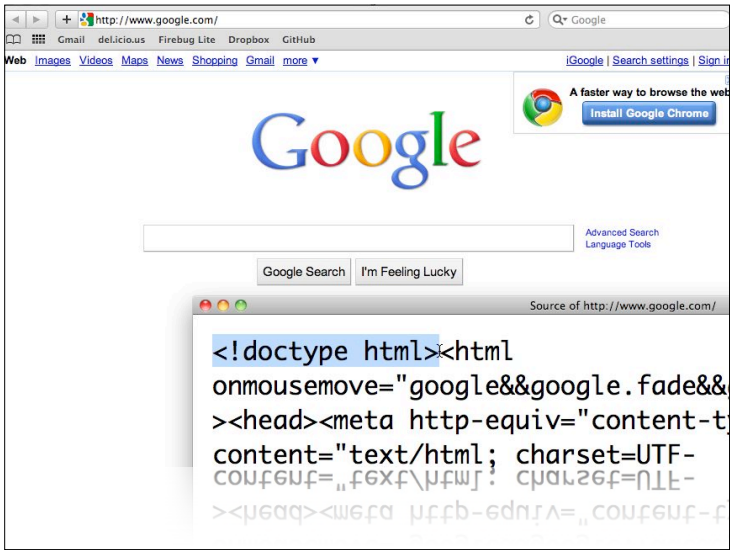
HTML5 is a collection of features...

Paving of cowpaths.

Evolutionary step.

Why should I care?

Market is moving.



Customers.

Mobile.

Next big thing.

Millions of mobile devices...more daily.

i* - iPad, iPhone etc.

Not just Apple...

Don't want to build a native app?

Don’t want to support multiple OS?

Android fragmentation, Black Berry, Palm...

Great! HTML5.

Mobile browsers excellent HTML5 support.

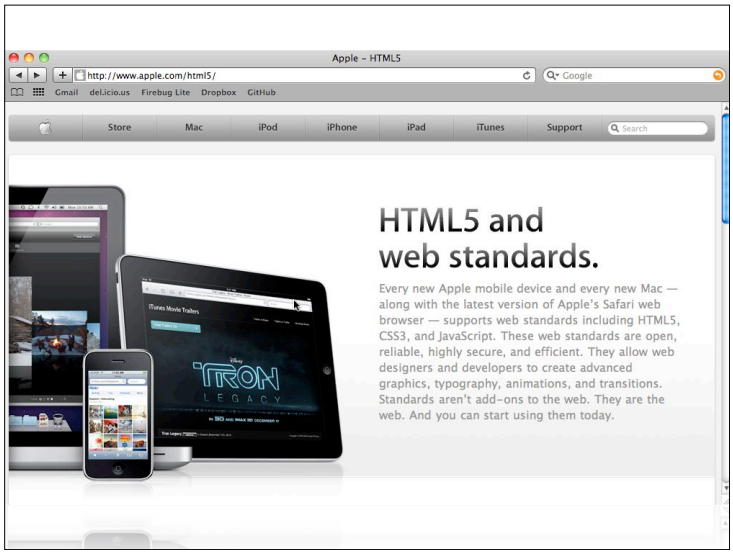
More likely than desktop peers.

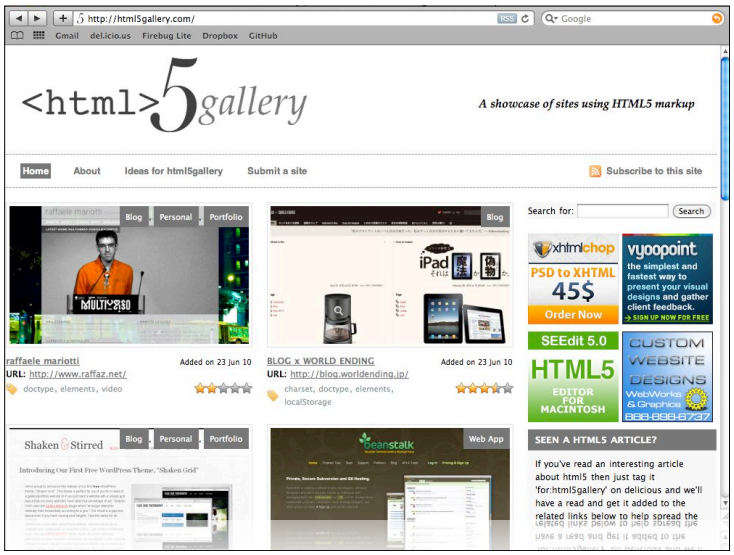
Flash is dying.

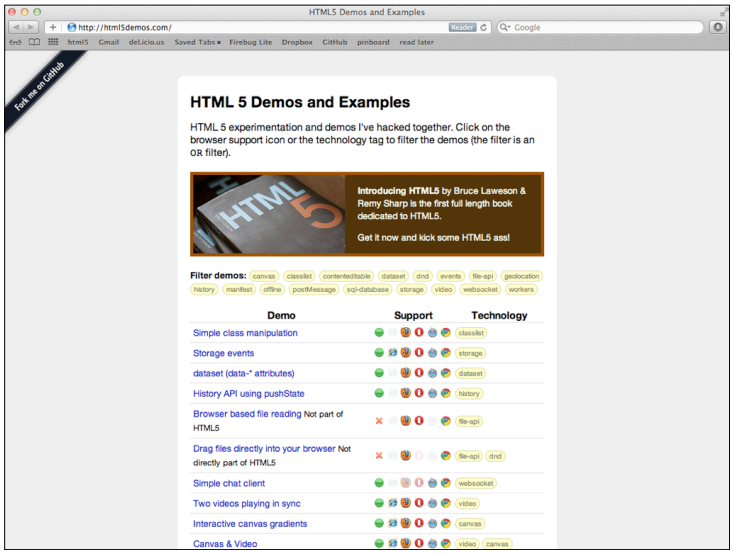
Lead largely by Apple and...Google.

YouTube

HTML5 gaining steam.







Customers want sites that work on today's devices.

HTML5 is the answer.

Respects what we're actually doing.

Forms, controls... well worn hacks.

Helps us build applications!

What can I do?

Browser support isn't universal.

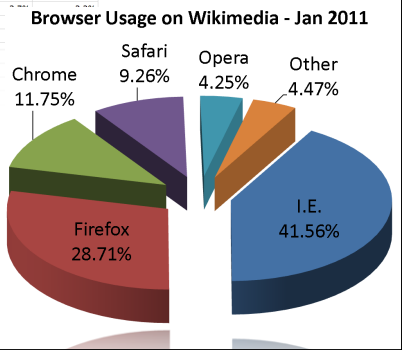
Shocking.

Older browsers pervasive.

%	8.9 %	plorer	Firefox	Chrome	Safari	Opera
%	9.6 %	26.5 %	42.4%	24.1%	4.1%	2.5%
%	10.2 %	26.6 %	42.8%	23.8%	4.0%	2.5%
		plorer	Firefox	Chrome	Safari	Opera
		27.5 %	43.5%	22.4%	3.8%	2.2%
		28.6 %	44.0%	20.5%	4.0%	2.3%
		29.7 %	44.1%	19.2%	3.9%	2.2%
		31.1 %	45.1%	17.3%	3.7%	2.2%
		30.7 %	45.8%	17.0%	3.5%	2.3%
		30.4 %	46.4%	16.7%	3.4%	2.3%
		31.0 %	46.6%	15.9%	3.6%	2.1%
		32.2 %	46.9%	14.5%	3.5%	2.2%
		33.4 %	46.4%	13.6%	3.7%	2.2%
		34.9 %	46.2%	12.3%		
		35.3 %	46.5%	11.6%		
		36.2 %	46.3%	10.8%		

http://www.w3schools.com/browsers/browsers_stats.asp

http://en.wikipedia.org/wiki/Usage_share_of_web_browsers



IE 6 is dying...

Even MS wants it gone.

SharePoint 2010 - no IE 6.

http://blogs.msdn.com/sharepoint/archive/2009/05/07/announcing-sharepoint-server-2010-preliminary-system-requirements.aspx

Google Apps...sorry IE 6.

http://googledocs.blogspot.com/2010/01/web-browser-support-for-docs-and-sites.html

Agent sniffing?

NO!

Browsers lie...

This browser's user agent:
Mozilla/5.0 (Macintosh; U; Intel Mac OS X

This browser's user agent:
Mozilla/5.0 (Macintosh; Intel Mac OS X

This browser's user agent:
Mozilla/5.0 (Macintosh; Intel Mac OS X

This browser's user agent:
Opera/9.80 (Macintosh; Intel Mac OS X

And you can always mimic another user agent...

Feature detection.

The DOM is your answer key.

Four approaches.

1. Ask the global object if a property exists.

```
function supports_geolocation() {
    return !!navigator.geolocation;
}
```

Create an element:

2. Look for a property.

```
function supports_canvas() {
    return !!document.createElement('canvas').getContext;
}
```

3. Look for a method.

```
function supports_video() {
    return !!document.createElement('video').canPlayType;
}
```

4. Set a property and see if the value sticks.

```
function supports_input (input_type) {
    var input = document.createElement("input");
    input.setAttribute("type", input_type);
    return input.type !== "text";
}
```

Modern browsers are evolving with spec.

Support is quite good.

Lab time!

Feature Detection

- Using the four detection techniques, add the proper code to the four empty methods
- ${extract}/html5_workshop/labs/detection.html

Modernizer.

http://www.modernizr.com/

```
function check_geo_modernizer() {
    if (Modernizr.geolocation) {
        va   function supports_geolocation() {          location!";
    } else       return !!navigator.geolocation;
        me   }
    }
    show_results(message);
}
```

```
function check_canvas_modernizer() {
    if (Modernizr.canvas) {
        var message = "modernizer says you've got canvas!";
    } else {
        message = no_support_mod;
    }
    show_results(message);
}
```

```
  function check_video_formats() {
f     if (Modernizr.video) {
          var message = "yep, we can do some kind of video!";
          if (Modernizr.video.ogg) {
              message = "actually, we can play ogg";
          } else if (Modernizr.video.h264) {
              message = "we prefer 264 thanks";
          }
      }
}     show_results(message);
  }
```

Ogg? 264? WebM?

Patents...

http://daringfireball.net/2010/03/gif_h264_patents

http://daringfireball.net/2010/03/on_submarine_patents

Browser support falls on philosophical lines.

```
function check_for_email_modernizer () {
function supports_input (input_type) {
    var input = document.createElement("input");
    input.setAttribute("type", input_type);
    return input.type !== "text";
}
}
```

Lab time!

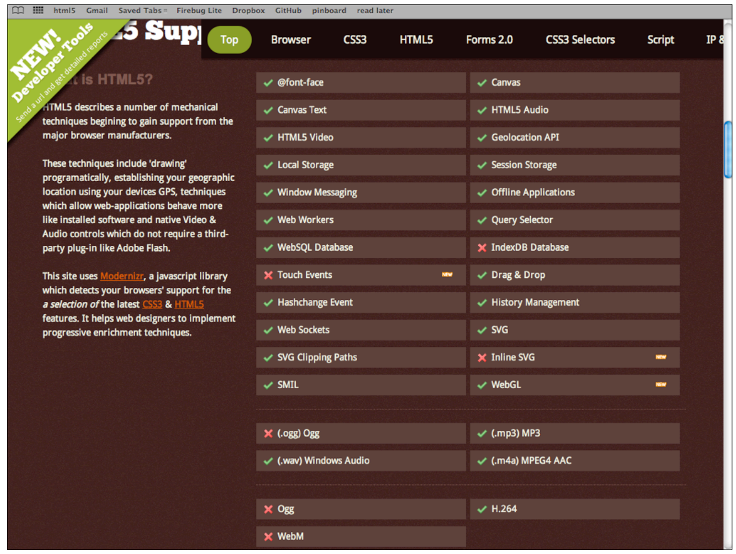
Feature Detection - 2

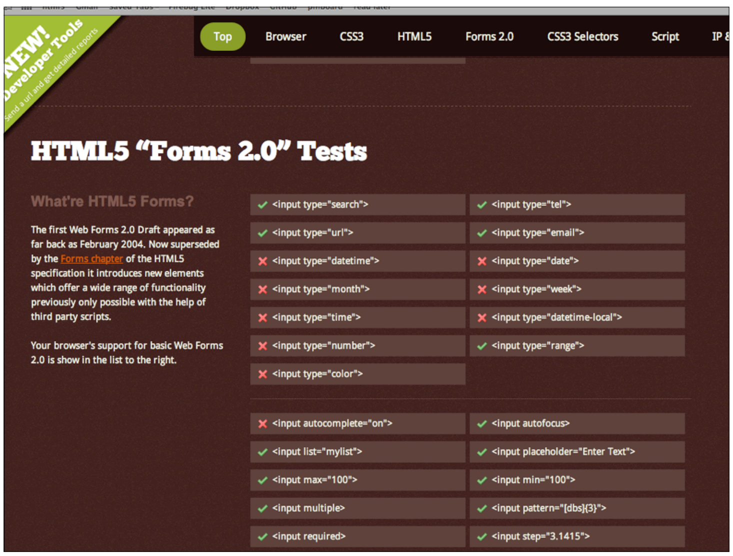
- Using Modernizr, add the proper code to the four empty methods
- ${extract}/html5_workshop/labs/detection_modernizr.html

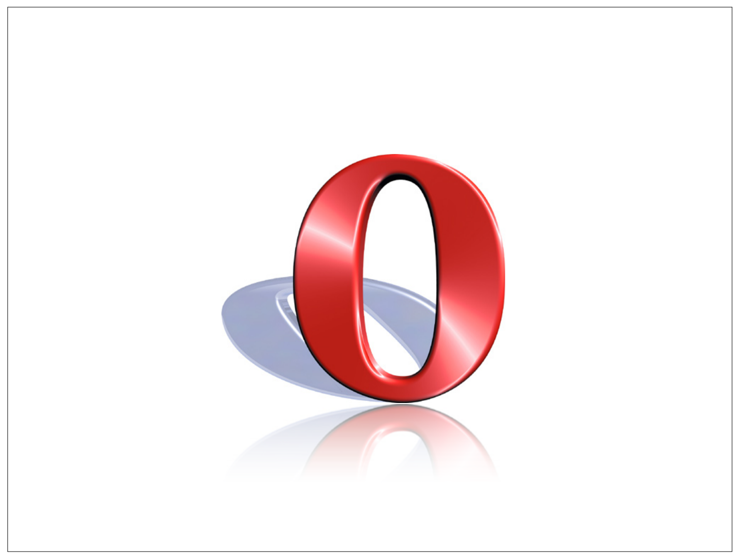
Not sure what your browser can do?

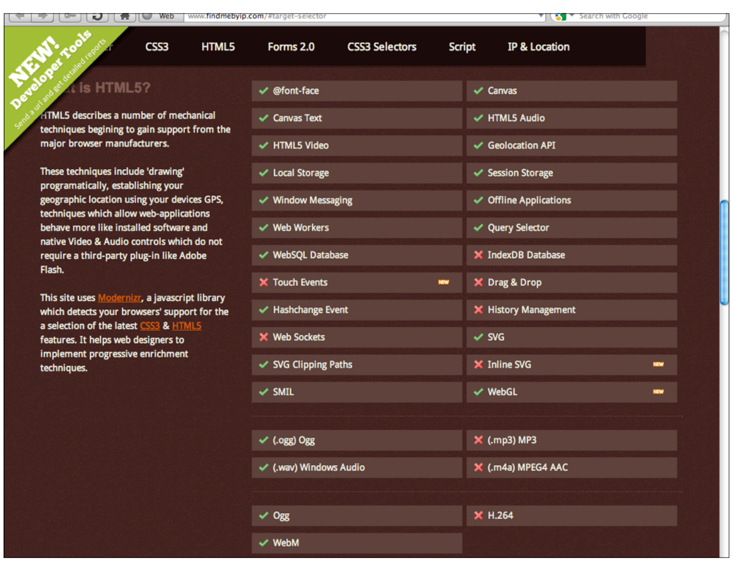
http://caniuse.com/

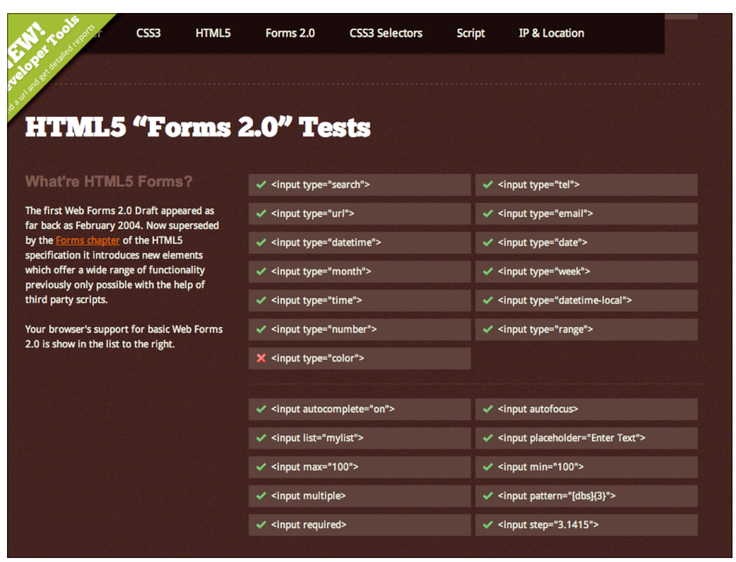
http://www.findmebyip.com/#target-selector

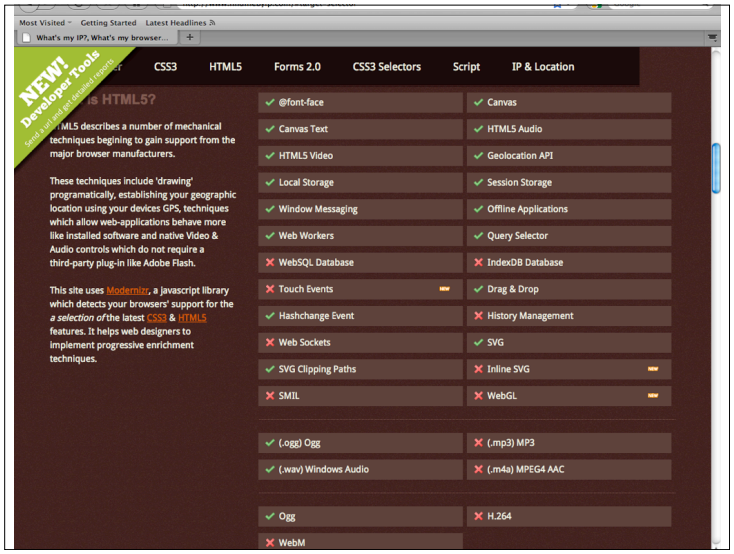

NEW! Developer Tools
Top
Browser
CSS3
HTML5
Forms 2.0
CSS3 Selectors
Script
HTML5 describes a number of mechanical techniques begining to gain support from the major browser manufacturers.
These techniques include 'drawing' programatically, establishing your geographic location using your devices GPS, techniques which allow web-applications behave more like installed software and native Video & Audio controls which do not require a third-party plug-in like Adobe Flash.
This site uses Modernizr, a javascript library which detects your browsers' support for the a selection of the latest CSS3 & HTML5 features. It helps web designers to implement progressive enrichment techniques.
@font-face
Canvas
Canvas Text
HTML5 Audio
HTML5 Video
Geolocation API
Local Storage
Session Storage
Window Messaging
Offline Applications
Web Workers
Query Selector
WebSQL Database
IndexDB Database
Touch Events
Drag & Drop
Hashchange Event
History Management
Web Sockets
SVG
SVG Clipping Paths
Inline SVG
SMIL
WebGL
(.ogg) Ogg
(.mp3) MP3
(.wav) Windows Audio
(.m4a) MPEG4 AAC
Ogg
H.264
WebM

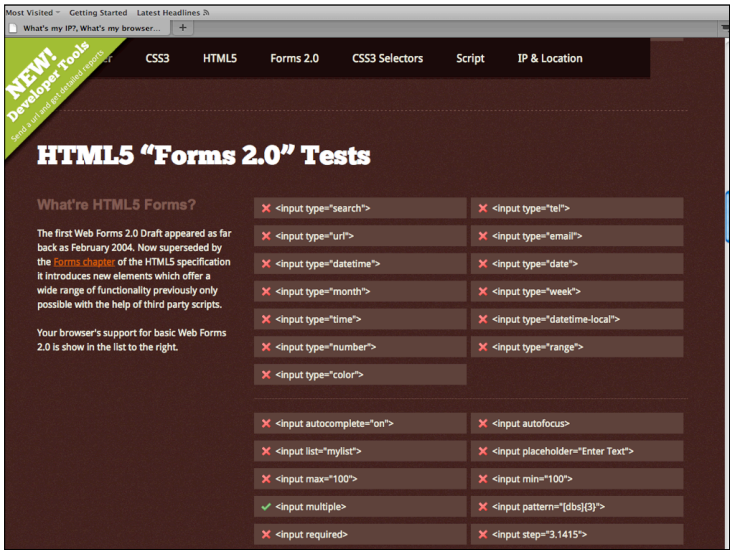

NEW! Developer Tools
Top
Browser
CSS3
HTML5
Forms 2.0
CSS3 Selectors
Script
HTML5 "Forms 2.0" Tests
What're HTML5 Forms?
The first Web Forms 2.0 Draft appeared as far back as February 2004. Now superseded by the Forms chapter of the HTML5 specification it introduces new elements which offer a wide range of functionality previously only possible with the help of third party scripts.
Your browser's support for basic Web Forms 2.0 is show in the list to the right.
<input type="search">
<input type="tel">
<input type="url">
<input type="email">
<input type="datetime">
<input type="date">
<input type="month">
<input type="week">
<input type="time">
<input type="datetime-local">
<input type="number">
<input type="range">
<input type="color">
<input autocomplete="on">
<input autofocus>
<input list="mylist">
<input placeholder="Enter Text">
<input max="100">
<input min="100">
<input multiple>
<input pattern="[dbs]{3}">
<input required>
<input step="3.1415">




NEW! Developer Tools
CSS3
HTML5
Forms 2.0
CSS3 Selectors
Script
IP & Location
HTML5 describes a number of mechanical techniques begining to gain support from the major browser manufacturers.
These techniques include 'drawing' programatically, establishing your geographic location using your devices GPS, techniques which allow web-applications behave more like installed software and native Video & Audio controls which do not require a third-party plug-in like Adobe Flash.
This site uses Modernizr, a javascript library which detects your browsers' support for the a selection of the latest CSS3 & HTML5 features. It helps web designers to implement progressive enrichment techniques.
@font-face
Canvas
Canvas Text
HTML5 Audio
HTML5 Video
Geolocation API
Local Storage
Session Storage
Window Messaging
Offline Applications
Web Workers
Query Selector
WebSQL Database
IndexDB Database
Touch Events
Drag & Drop
Hashchange Event
History Management
Web Sockets
SVG
SVG Clipping Paths
Inline SVG
SMIL
WebGL
(.ogg) Ogg
(.mp3) MP3
(.wav) Windows Audio
(.m4a) MPEG4 AAC
Ogg
H.264
WebM

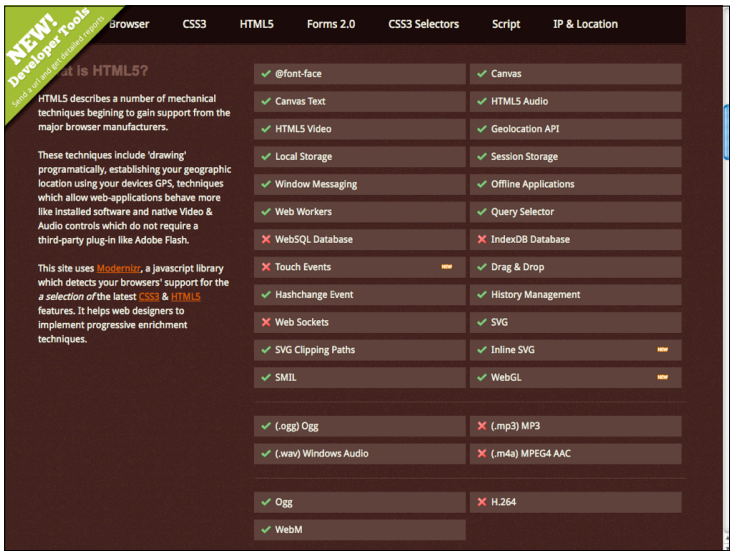

CSS3
HTML5
Forms 2.0
CSS3 Selectors
Script
IP & Location
HTML5 "Forms 2.0" Tests
What're HTML5 Forms?
The first Web Forms 2.0 Draft appeared as far back as February 2004. Now superseded by the Forms chapter of the HTML5 specification it introduces new elements which offer a wide range of functionality previously only possible with the help of third party scripts.
Your browser's support for basic Web Forms 2.0 is show in the list to the right.
<input type="search">
<input type="tel">
<input type="url">
<input type="email">
<input type="datetime">
<input type="date">
<input type="month">
<input type="week">
<input type="time">
<input type="datetime-local">
<input type="number">
<input type="range">
<input type="color">
<input autocomplete="on">
<input autofocus>
<input list="mylist">
<input placeholder="Enter Text">
<input max="100">
<input min="100">
<input multiple>
<input pattern="[dbs]{3}">
<input required>
<input step="3.1415">

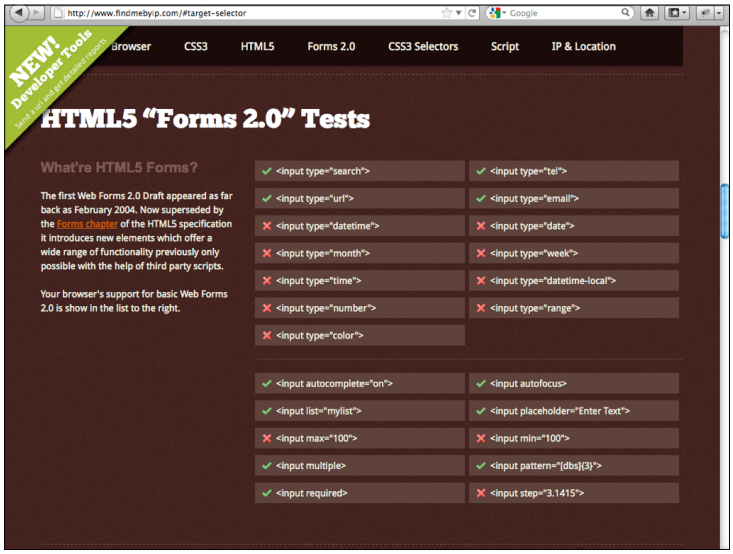


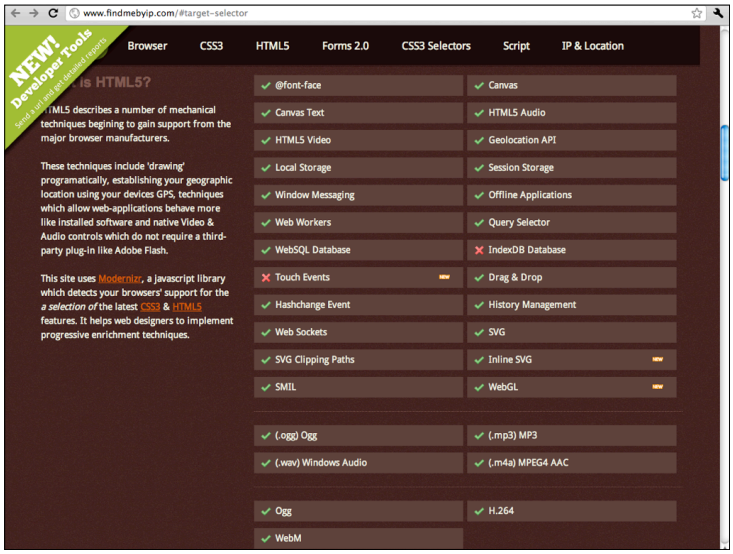


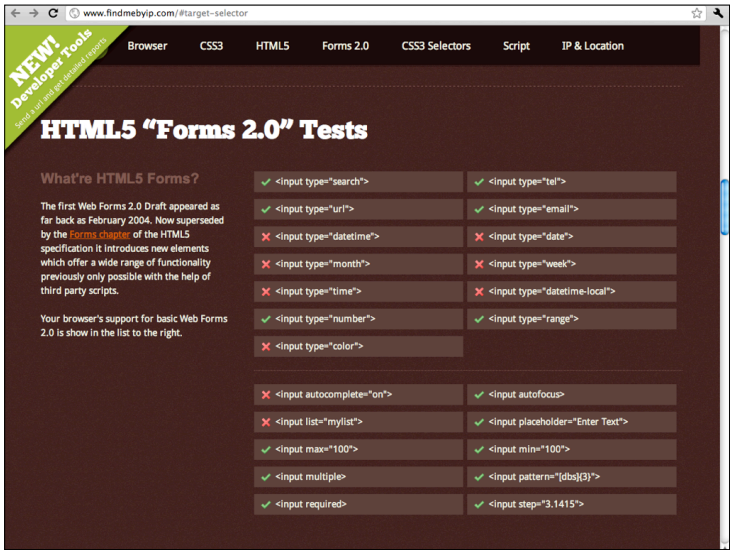
That's 3.6.13...

Here's 4...









Another view...

http://html5test.com/index.html

your browser | other browsers

UPDATED August 23, 2011

your browser scores

293

AND 8 BONUS POINTS

out of a total of 450 points

Parsing rules — 2 bonus points — 11

`<!DOCTYPE html>` triggers standards mode	Yes ✔
HTML5 tokenizer	Yes ✔
HTML5 tree building	Yes ✔

HTML5 defines rules for embedding SVG and MathML inside a regular HTML document. Support for SVG and MathML is not required though, so bonus points are awarded if your browser supports embedding these two technologies.

SVG in `text/html`	Yes ✔
MathML in `text/html`	Yes ✔

Canvas — 20

ABOUT THE TEST

The HTML5 test score is an indication of how well your browser supports the upcoming HTML5 standard and related specifications. Even though the specification isn't finalized yet, all major browser manufacturers are making sure their browser is ready for the future. Find out which parts of HTML5 are already supported by your browser today and compare the results with other browsers.

SPONSORS

MAKE APPS TODAY WITH HTML5

Advertise Here

Video — 4 bonus points — 21/31

`video` element	Yes ✔
Subtitle support	No ✘
Poster image support	Yes ✔

The following tests go beyond the requirements of the HTML5 specification and are not counted towards the total score. If a browser supports one or more video codecs, two bonus points are awarded for each codec.

MPEG-4 support	Yes ✔
H.264 support	Yes ✔
Ogg Theora support	No ✘
WebM support	No ✘

Audio — 2 bonus points — 20

`audio` element	Yes ✔

The following tests go beyond the requirements of the HTML5 specification and are not counted towards the total score. If a browser supports one or more audio codecs, one bonus point is awarded for each codec.

PCM audio support	No ✘
MP3 support	Yes ✔
AAC support	Yes ✔
Ogg Vorbis support	No ✘
WebM support	No ✘

Elements — 22/28

Embedding custom non-visible data	Yes ✔

New or modified elements

all of the new features offered by HTML5, nor does it try to test the functionality of each feature it does detect. Despite these shortcomings we hope that by quantifying the level of support users and web developers will get an idea of how hard the browser manufacturers work on improving their browsers and the web as a development platform.

The score is calculated by testing for the many new features of HTML5. Each feature is worth one or more points. Apart from the main HTML5 specification and other specifications created the W3C HTML Working Group, this test also awards points for supporting related drafts and specifications. Some of these specifications were initially part of HTML5, but are now further developed by other W3C working groups. WebGL is also part of this test despite not being developed by the W3C, because it extends the HTML5 canvas element with a 3d context.

The test also awards bonus points for supporting audio and video codecs and supporting SVG or MathML embedding in a plain HTML document. These test do not count towards the total score because HTML5 does not specify any required audio or video codec. Also SVG and MathML are not required by HTML5, the specification only specifies rules for how such content should be embedded inside a plain

Forms — 55/98

Field types

`input type=search`	Yes ✔
`input type=tel`	Yes ✔
`input type=url`	Yes ✔
`input type=email`	Yes ✔
`input type=datetime`	Partial ○
`input type=date`	Partial ○
`input type=month`	Partial ○
`input type=week`	Partial ○
`input type=time`	Partial ○
`input type=datetime-local`	Partial ○
`input type=number`	Yes ✔
`input type=range`	Yes ✔
`input type=color`	Partial ○
`input type=checkbox`	Yes ✔
`input type=image`	Partial ○
`textarea`	Partial ○
`select`	Yes ✔
`fieldset`	Partial ○
`datalist`	Partial ○
`keygen`	Yes ✔
`output`	Yes ✔
`progress`	No ✘
`meter`	No ✘

Fields

Field validation	Yes ✔

goalpost.

SPONSORS

Advertise Here

DEVELOPMENT

The HTML5 test is created by Niels Leenheer. Visit my website at rakaz.nl or follow me on twitter.

FOLLOW ME ON twitter

The HTML5 test is being developed at Github. Please file an issue there if you find any bugs or think of any improvements to this test. Please note that the HTML5 test is not affiliated with the W3C or the HTML5 working group.

Thanks to Henri Sivonen for allowing me to reuse his HTML5 parser tests and all other contributors.

August 23, 2011 - version 2.2.4

Hosting provided by:

sights.

Fields

Field validation	Yes ✔
Association of controls and forms	Partial ○
Other attributes	Partial ○
CSS selectors	Yes ✔
Events	Yes ✔

Forms

Form validation	Partial ○

User interaction — 34/36

Drag and drop

Attributes	Partial ○
Events	Yes ✔

HTML editing

Editing elements	Yes ✔
Editing documents	Yes ✔
APIs	Yes ✔

History and navigation — 5

Session history	Yes ✔

Microdata — 0/15

Microdata	No ✘

Web applications — 15/20

Application Cache	Yes ✔
Custom scheme handlers	No ✘
Custom content handlers	No ✘
Custom search providers	No ✘

Security — 5/10

Sandboxed `iframe`	Yes ✔
Seamless `iframe`	No ✘

Related specifications

Geolocation — 15

Geolocation	Yes ✔

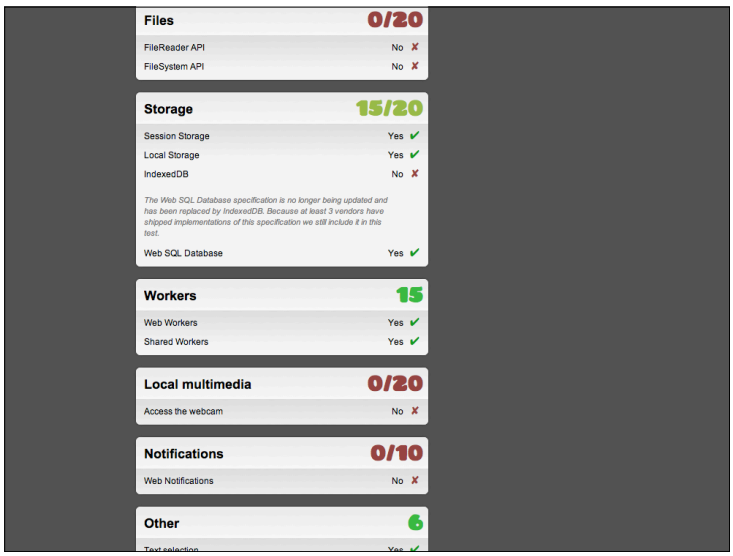
WebGL — 9/25

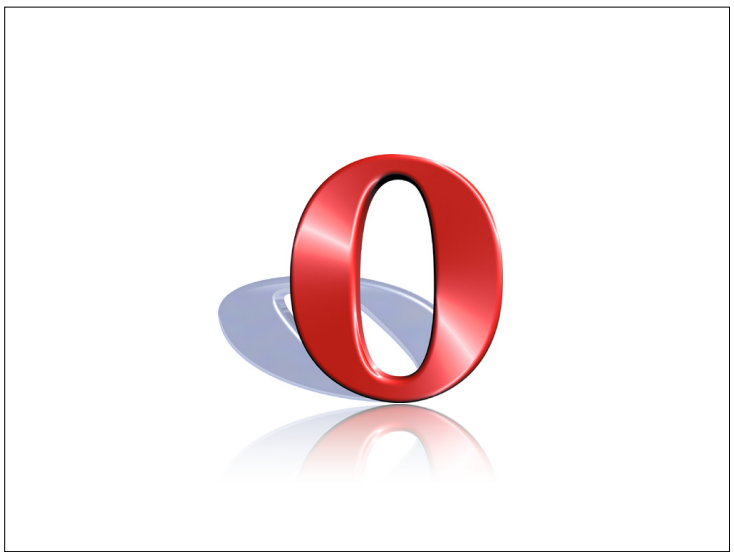
3D context	No ✘
Native binary data	Partial ○

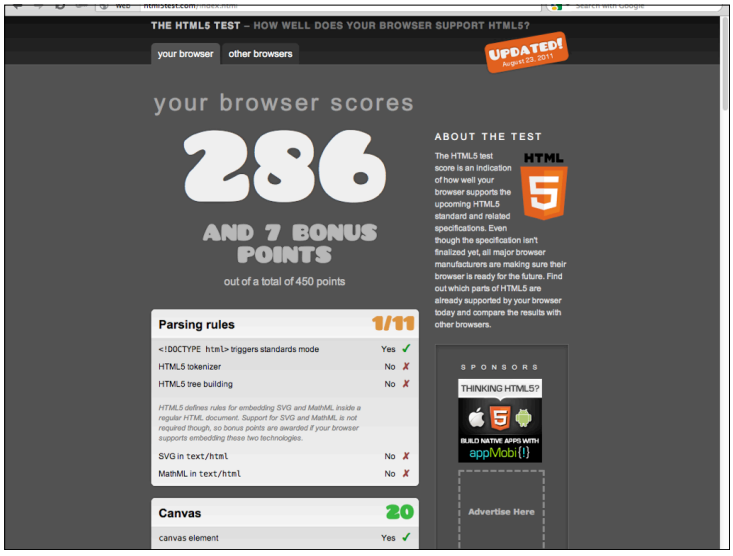
Communication — 25

Cross-document messaging	Yes ✔
Server-Sent Events	Yes ✔

Both Mozilla and Opera do support the WebSocket protocol in their latest browsers, but have disabled it due to a fundamental security issue with the protocol. Once the protocol has been updated it is expected they will re-enable this feature.

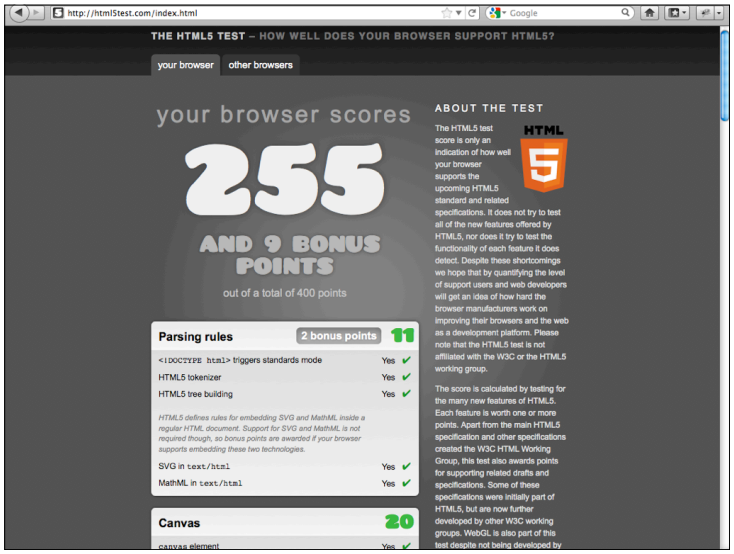




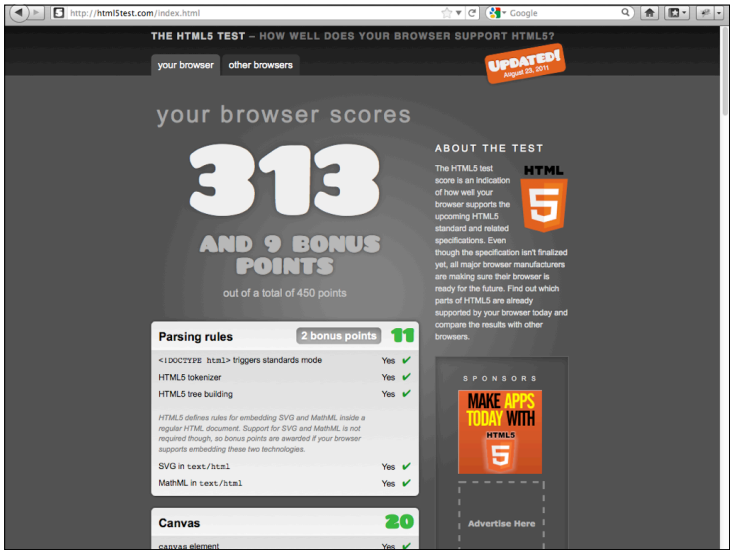


Again, that's 3.6.13.

Here's 4.


http://html5test.com/index.html
THE HTML5 TEST – HOW WELL DOES YOUR BROWSER SUPPORT HTML5?
your browser
other browsers
your browser scores
255
AND 9 BONUS POINTS
out of a total of 400 points
ABOUT THE TEST
Parsing rules
2 bonus points
11
<!DOCTYPE html> triggers standards mode
HTML5 tokenizer
HTML5 tree building
SVG in text/html
MathML in text/html
Canvas
20

And 6...


http://html5test.com/index.html
THE HTML5 TEST – HOW WELL DOES YOUR BROWSER SUPPORT HTML5?
your browser
other browsers
UPDATED!
August 23, 2011
your browser scores
313
AND 9 BONUS POINTS
out of a total of 450 points
ABOUT THE TEST
Parsing rules
2 bonus points
11
<!DOCTYPE html> triggers standards mode
HTML5 tokenizer
HTML5 tree building
SVG in text/html
MathML in text/html
Canvas
20
SPONSORS
MAKE APPS TODAY WITH HTML5
Advertise Here


THE HTML5 TEST – HOW WELL DOES YOUR BROWSER SUPPORT HTML5?
your browser
other browsers
UPDATED!
August 23, 2011
your browser scores
341
AND 13 BONUS POINTS
out of a total of 450 points
ABOUT THE TEST
Parsing rules
2 bonus points
11
<!DOCTYPE html> triggers standards mode
HTML5 tokenizer
HTML5 tree building
SVG in text/html
MathML in text/html
Canvas
20
canvas element
2D context
Text
SPONSORS
MAKE APPS TODAY WITH HTML5
Advertise Here

Let's go!

Doctype.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
```

Trans-what-now?

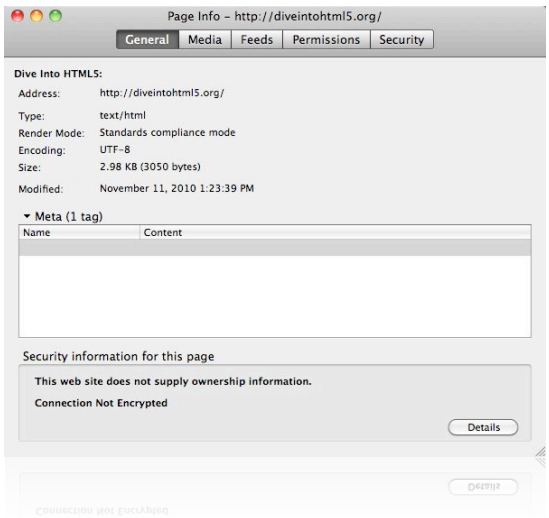
Quite a mouth full...

People designed expecting poor rendering.

Browsers standards support improved.

And they broke the web.

Quirks mode vs. standards mode.



And almost standards mode.

Seriously.

Let developers opt in.

HTML5 simplifies life.

```
<!DOCTYPE HTML>
```

It's almost too easy!

And you can type it.

Oh, must be the first line.

Blank line can kick you into quirks mode...

New semantic elements.

HTML5 adds new elements.

- section
- nav
- article
- aside
- hgroup
- header
- footer
- time
- mark

Defines things we've been doing for years.

With divs and ids.

It works, but
lacks meaning.

Common markup.

Again, nod to what
we're actually doing.

More meaningful than divs!

So what do these
elements mean?

<section>

Thematic grouping of content.

Might have heading or an outline.

Chapters, tabs.

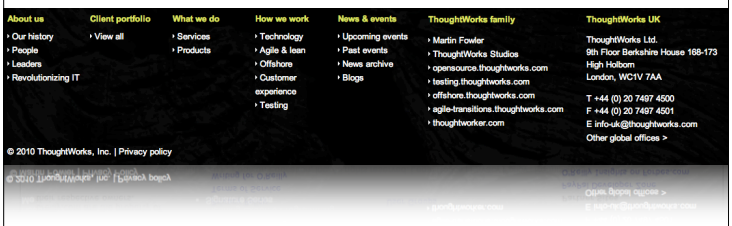
Intro, part 1, part 2...part N, conclusion.

<nav>

Section with links.

Major navigation blocks.

Common in footers.



Nothing tells you that's navigation though.

Common yes...

Accessibility.

Screen readers,
keyboard only users.

<article>

Reusable or
distributable.

Post, blog entry, comment.

Think syndication.

<aside>

Tangential content.

Sidebars, pull quotes.

<hgroup>

Group of set of headings (h1-h6).

<header>

Introduction.

Could contain hgroup or headings.

Does't create a new section.

Not a new scope for headers/footers.

```
<div id="header">
...
</div>
```

```
<header>
...
</header>
```

```
<footer>
```

Usually at the bottom of a section.

Often contains copyright, contact info, help, privacy, etc.

Whatever lives in the `div id="footer"` ;)

Does't create a new section.

`<time>`

Encode time/date for machine use.

Meetings, birthdays, anniversaries ;)

```
<time datetime="2011-02-22" pubdate>February 22, 2011</time>
```

3 parts.

1. Machine readable.


```
<time datetime="2011-02-22" pubdate>February 22, 2011</time>
```


YYYY-MM-DD

Quite flexible.

http://www.whatwg.org/specs/web-apps/current-work/multipage/common-microsyntaxes.html#valid-global-date-and-time-string

Want time?

Add T, time in 24 hour, timezone offset.

```
datetime="2011-02-22T11:21:37-07:00"
```

2. Human readable.

```
<time datetime="2011-02-22" pubdate>February 22, 2011</time>
```

Text doesn't have to match the datetime attribute.

It's human readable!

Next Sunday, tomorrow, in three days...

Could even be empty.

pubdate flag.

```
<time datetime="2011-02-22" pubdate>February 22, 2011</time>
```

Boolean.

Says timestamp is publication date.

For article or the document...

<mark>

Think highlight.

Call attention to something.

And if your browser doesn't support it?

Unknown elements rendered inline.

However, many of these elements are block.

In older browsers...

Style them as block.

HTML5 Reset.

http://html5doctor.com/html-5-reset-stylesheet/

Oh, before 9, IE won't style unknown elements.

Despite your CSS.

Also affects the DOM.

The workaround?

Create the element in JavaScript.

IE will allow you to style it.

Don't want to do that yourself?

No worries.

HTML5 enabling script.

http://remysharp.com/2009/01/07/html5-enabling-script/

What's all the fuss about?

Divs work, right?

Document outline.

http://gsnedders.html5.org/outliner/

Before, headings were our only hope.

Sectioning content (article, aside, nav, section)...

Create new nodes.

Each has its own hierarchy.

Aids composability.

Lab time!

Semantic Elements

- Take the sample web page and "convert" it to use HTML5 semantic elements
- View the page in various browsers
- ${extract}/html5_workshop/labs/semantic_elements.html

New Input Types.

We've spent a lot of time developing apps.

With a really limited palette.

Text box, text area, drop down, radio button...

Pretty limited.

Libraries help!

But why doesn't the browser do more?

Now it can!

HTML5 adds 13 new types.

And if your browser doesn't support it?

No worries.

Unknown types treated as text.

Even works in IE 6!

So what's been added?

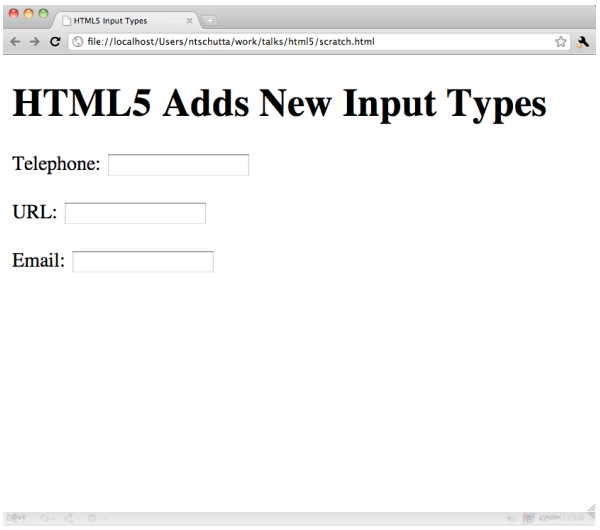
- search
- spinner
- slider
- color picker
- telephone number
- url
- email
- date, month, week, timestamp
- datetime

What do they do?

The spec doesn't say.

In many cases, they look just a text box.

For example...




```
<!DOCTYPE HTML>
<html>
<head>
    <title>HTML5 Input Types</title>
</head>
<body>
    <h1>HTML5 Adds New Input Types</h1>

    Telephone: <input type="tel"> <br/> <br/>
    URL: <input type="url"> <br/> <br/>
    Email: <input type="email"> <br/> <br/>

</body>
</html>
```

Impressed?

Yeah...

So what's the point?

What about the iPhone?

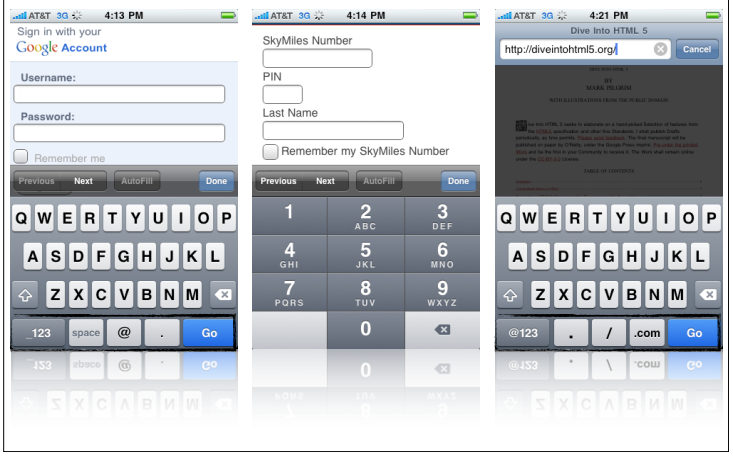
No keyboard.

"Need a keyboard."

Really?

Can't reconfigure a physical keyboard.

But when it's software...



That's useful!

Frustrating when sites don't.

And it costs you nothing.

Search.

Speaking of inputs that don't look much different...

HTML5 Adds New Input Types

Search:

HTML5 Adds New Input Types

Search:

HTML5 Adds New Input Types

Search:

HTML5 Adds New Input Types

Search:

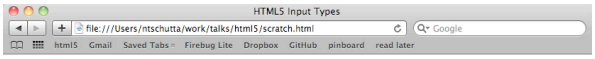
```
<!DOCTYPE HTML>
<html>
<head>
    <title>HTML5 Input Types</title>
</head>
<body>
    <h1>HTML5 Adds New Input Types</h1>

    Search: <input type="search" autofocus> <br/><br/>

</body>
</html>
```

Rounded corners!

Oh, and an X...



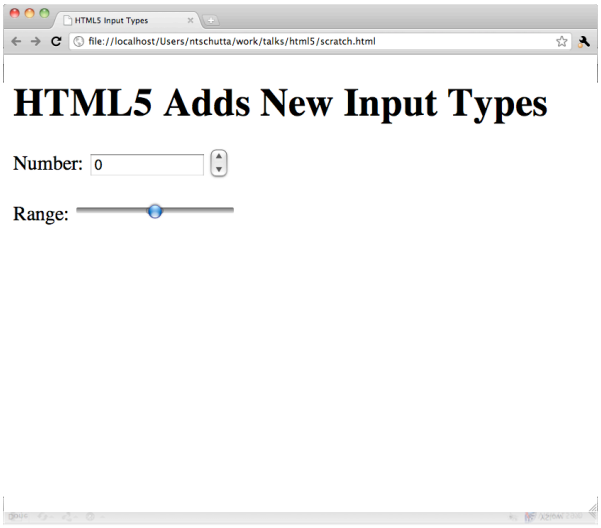
HTML5 Adds New Input Types

Search:

Numbers.

Like email addresses & URLs, they're special.

How about spinners and ranges?




```
<!DOCTYPE HTML>
<html>
<head>
    <title>HTML5 Input Types</title>
</head>
<body>
    <h1>HTML5 Adds New Input Types</h1>

    Number: <input type="number"
                   step="2"
                   value="0"
                   min="0"
                   max="10"> <br/> <br/>

    Range: <input type="range"> <br/> <br/>

</body>
</html>
```

Attributes are optional.

Default step value is 1.

Ranges.

Slider control.

Shades of thick clients!

```
<!DOCTYPE HTML>
<html>
<head>
    <title>HTML5 Input Types</title>
</head>
<body>
    <h1>HTML5 Adds New Input Types</h1>

    Number: <input type="number"
                   step="2"
                   value="0"
                   min="0"
                   max="10"> <br/> <br/>

    Range: <input type="range"> <br/> <br/>

</body>
</html>
```

Also includes some handy JavaScript.

stepUp(n)
stepDown(n)

Increase/decrease
the value by n.

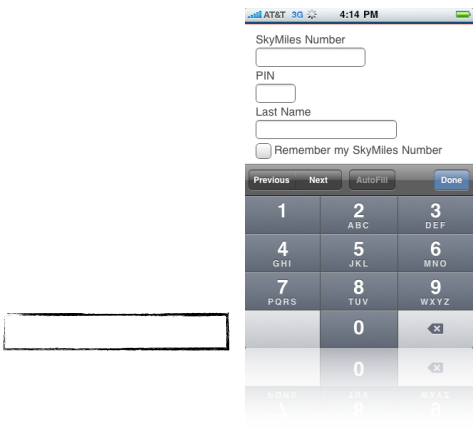
valueAsNumber

Returns value
as a number!

The value attribute
is a string...

Useful?

Back to the iPhone...



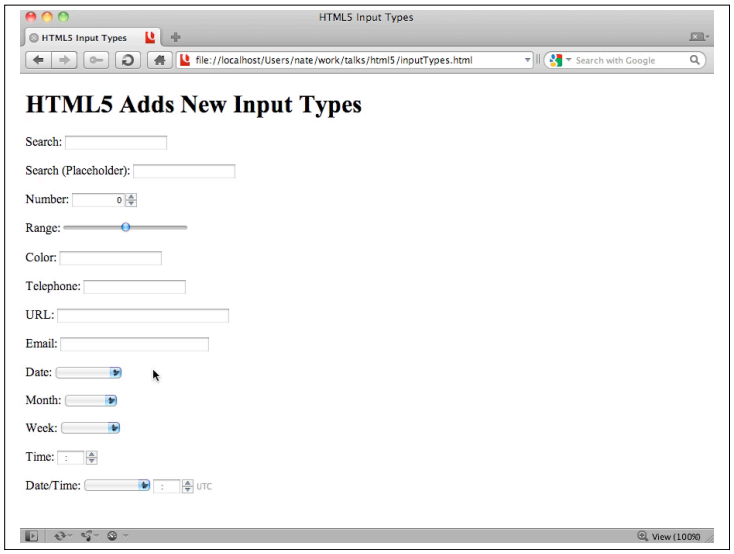
Date pickers.

Why don't we have a native date picker?

Now we do!

But it's only in Opera.

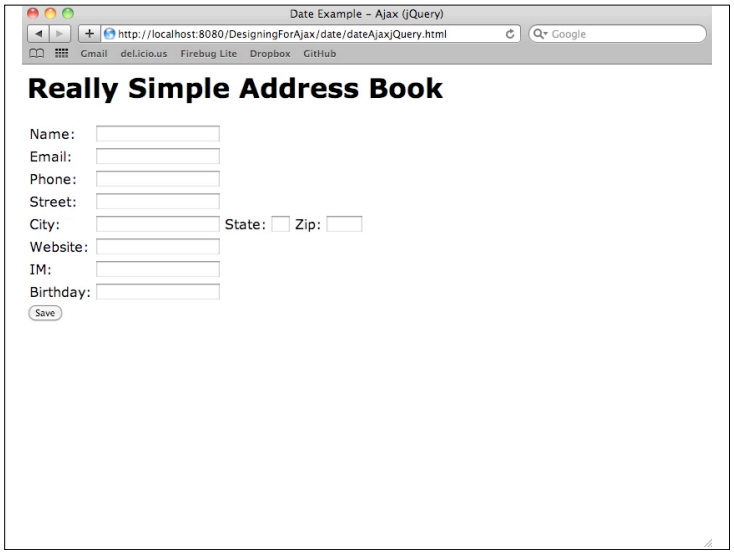
And you may not like it.




```
<!DOCTYPE HTML>
<html>
<head>
    <title>HTML5 Input Types</title>
</head>
<body>
    <h1>HTML5 Adds New Input Types</h1>

    Date: <input type="date"> <br/> <br/>
    Month: <input type="month"> <br/> <br/>
    Week: <input type="week"> <br/> <br/>
    Time: <input type="time"> <br/> <br/>
    Date/Time: <input type="datetime"> <br/> <br/>

</body>
</html>
```



CSS could be improved.

But which would your users' prefer?

Fallback to a library.

Speaking of pickers.

Color!

Pick a color, get
a hex value!

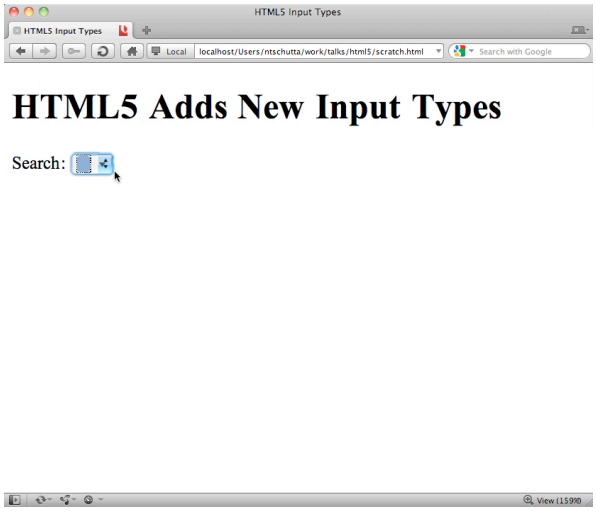
Cool!

Only works in Opera.

Bummer.

Uses native picker.

Except on Linux.




```
<!DOCTYPE HTML>
<html>
<head>
    <title>HTML5 Input Types</title>
</head>
<body>
    <h1>HTML5 Adds New Input Types</h1>

    Color Picker: <input type="color">

</body>
</html>
```

Autofocus.

```html
<!DOCTYPE HTML>
<html>
<head>
    <title>HTML5 Input Types</title>
</head>
<body>
<h1>HTML5 Adds New Input Types</h1>
Search: <input type="search" autofocus> <br/><br/>
Search (Placeholder): <input type="search" placeholder="Search"> <br/> <br/>
Number: <input type="number" step="2" value="0"> <br/> <br/>
Range: <input type="range"> <br/> <br/>
Color: <input type="color"> <br/> <br/>
Telephone: <input type="tel"> <br/> <br/>
URL: <input type="url"> <br/> <br/>
Email: <input type="email"> <br/> <br/>
Date: <input type="date"> <br/> <br/>
Month: <input type="month"> <br/> <br/>
Week: <input type="week"> <br/> <br/>
Time: <input type="time"> <br/> <br/>
Date/Time: <input type="datetime"> <br/> <br/>
</body>
</html>
```

Today we use JavaScript.

Edge cases.

Consistency.

Can be disabled.

Validation!

Email address.

Yes, you can do this
in JavaScript today.

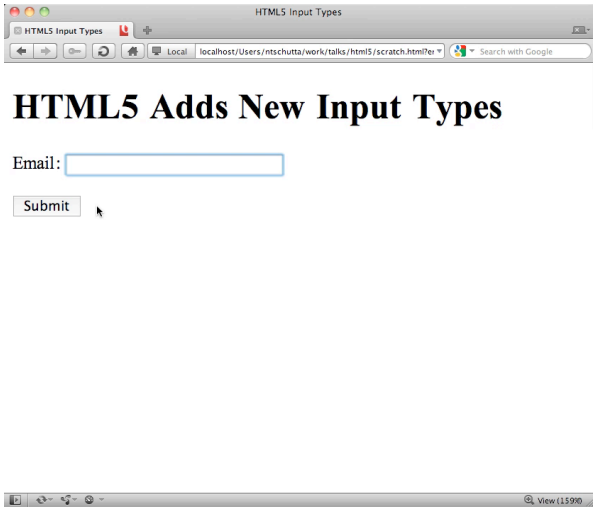
Maybe you already do.

It's hard!

What if JS is disabled?

And you're still validating
on the server right?

HTML5 to the rescue!



Validation is on by default.

Also works for url and numbers.

Even respects min/max.

Don't want to validate?

Use novalidate attribute.

Support is...soft.

Safari and Chrome,
no error messages.

Just doesn't submit ;)

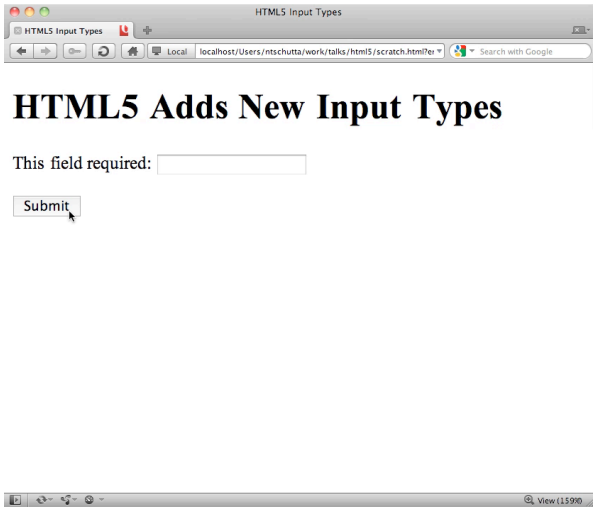
Very user friendly.

Required fields.

Add the required attribute!

Appearance varies by browser.

For example...




```
<!DOCTYPE HTML>
<html>
<head>
    <title>HTML5 Input Types</title>
</head>
<body>
    <h1>HTML5 Adds New Input Types</h1>
    <form>
        This field required: <input name="foo" required>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Lab time!

Forms

- Take the sample web page and "convert" it to use HTML5 form elements
- Make a field required
- Put focus in the first field
- Add placeholder text
- Submit the form in various browsers
- ${extract}/html5_workshop/labs/forms.html

Fun With Numbers

- Convert the text field to a number
- Write a function that adds 1 to the number
- Write a function that subtracts 1 from the number
- Write a function to display the type of the input field using valueAsNumber
- ${extract}/html5_workshop/labs/numbers.html

Canvas.

Graphics!

Graphs, shapes, animations, etc.

Controlled via scripting.

Pretty simple.

```
<canvas id="canvas" width="800" height="800"></canvas>
```

That's it?

Only two attributes:
width and *height*.

Optional, default is
300 pixels by 150 pixels.

CSS sizing as well.

Can be styled like an image - boarder, margin, etc.

Specifying fallback content.

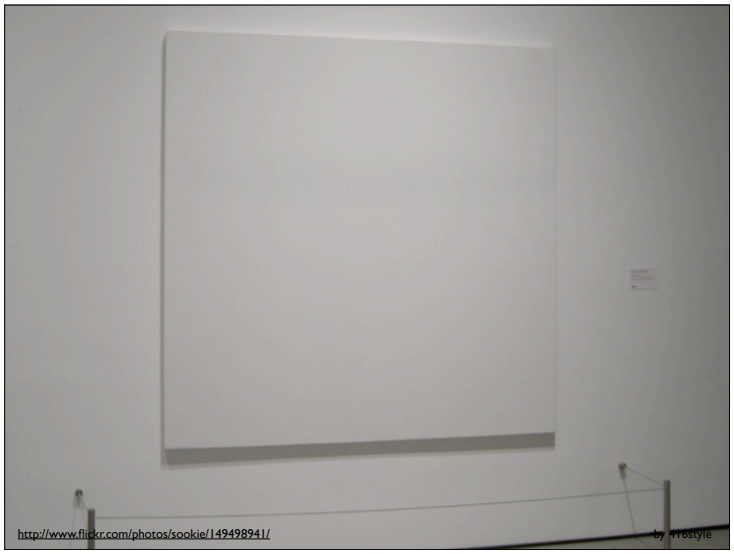
```
<canvas id="fallback" width="800" height="800">
   Put fallback content here...perhaps an image.
</canvas>
```

Content between tag.

Ignored by browsers supporting canvas...

Canvas tag is ignored by browsers lacking support.

Canvas starts like any canvas...



Up to you to fill it!

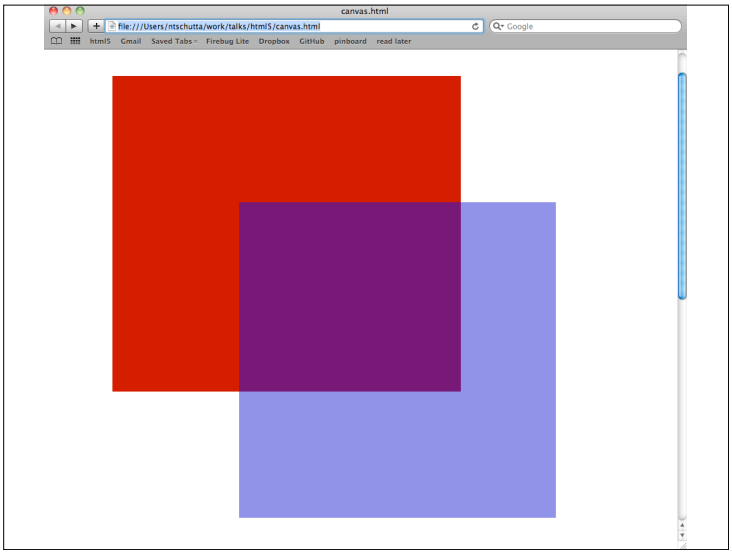
To draw, we need a context.

```
var ctx = canvas.getContext("2d");
```

For now, just 2D...

Likely 3D in the future.

Once we have a context, we can draw!




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

            ctx.fillStyle = "rgb(200,0,0)";
            ctx.fillRect(100, 100, 550, 500);

            ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
            ctx.fillRect(300, 300, 550, 500);
        }
    </script>
</head>
<body onload="draw()">
<canvas id="canvas" width="800" height="800"></canvas>

<canvas id="fallback" width="800" height="800">
   Put fallback content here...perhaps an image.
</canvas>
```

One primitive - rectangle.

Three methods.

```
fillRect(x, y, width, height);
strokeRect(x, y, width, height);
clearRect(x, y, width, height);
```

All take the same arguments...

x and y position of left corner of rectangle...

Width and height.

fillRect - filled rectangle.

strokeRect - outline.

clearRect - clears area, makes it transparent.

So that's it? Rectangles?

No!

Paths.

We can draw shapes.

- *beginPath* - creates path
- *closePath* - tries to close the shape
- *stroke* - draws an outlined shape
- *fill* - draws a solid shape
- *moveTo* - moves the "pen", doesn't draw

https://developer.mozilla.org/en/Canvas_tutorial%3aDrawing_shapes

```
function draw() {
    var canvas = document.getElementById("canvas");
    var ctx = canvas.getContext("2d");

    ctx.beginPath();
    ctx.arc(75,75,50,0,Math.PI*2,true); // Outer circle
    ctx.moveTo(110,75);
    ctx.arc(75,75,35,0,Math.PI,false);   // Mouth (clockwise)
    ctx.moveTo(65,65);
    ctx.arc(60,65,5,0,Math.PI*2,true);  // Left eye
    ctx.moveTo(95,65);
    ctx.arc(90,65,5,0,Math.PI*2,true);  // Right eye
    ctx.stroke();

}
```

Arc? Draws circles.

- x
- y
- radius
- startAngle - start point, radians
- endAngle - end point, radians
- anticlockwise - boolean, clockwise or not

lineTo(x, y) - Straight lines

Curves.

quadraticCurveTo,
bezierCurveTo

And of course you can combine these...

You can also use images.

Get an image - from page of from scratch.

drawImage(image, x, y)

Useful as backdrops...

Can also scale images.

Add width and height.

You can also crop...

And on and on!

Colors, gradients, line styles, patterns...

Rotating, scaling, transforms, compositing.

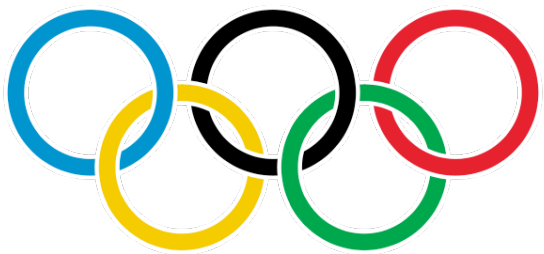
Animations!

Whew!

Lab time!

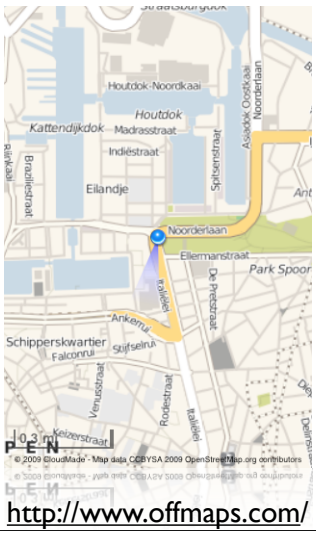
Olympic Rings

- Using the various canvas methods, draw the Olympic Rings
- ${extract}/html5_workshop/labs/canvas_rings.html



Geolocation.

Where in the world are you?



http://www.offmaps.com/

Very helpful on phones!

Technically not a part of HTML5.

Geolocation Working Group.

Wide browser support.

Your browser doesn't support it?

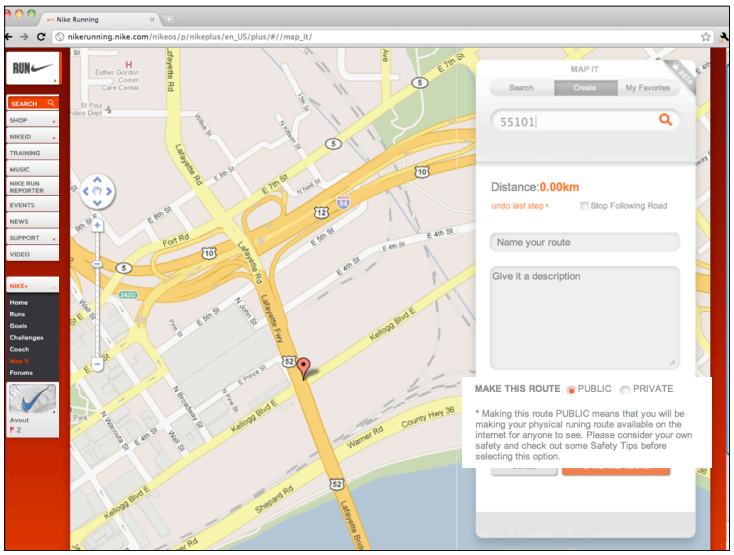
Device specific options.

Privacy issue?

Absolutely.

If you know where the device is...

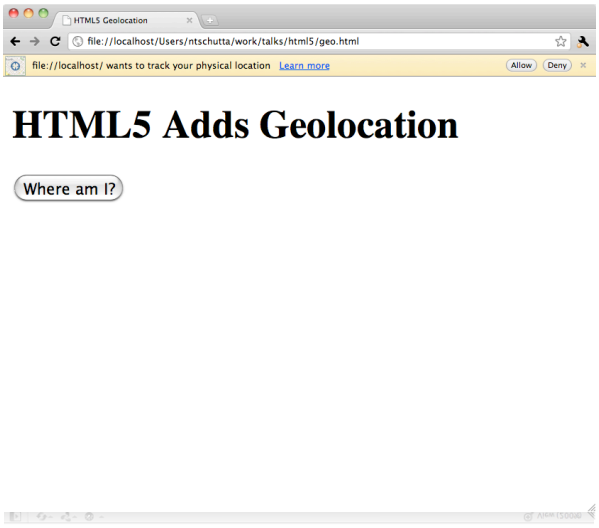
http://icanstalku.com/why.php

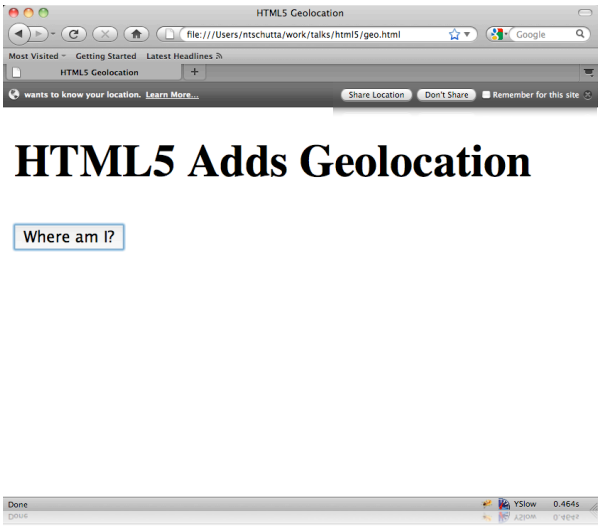


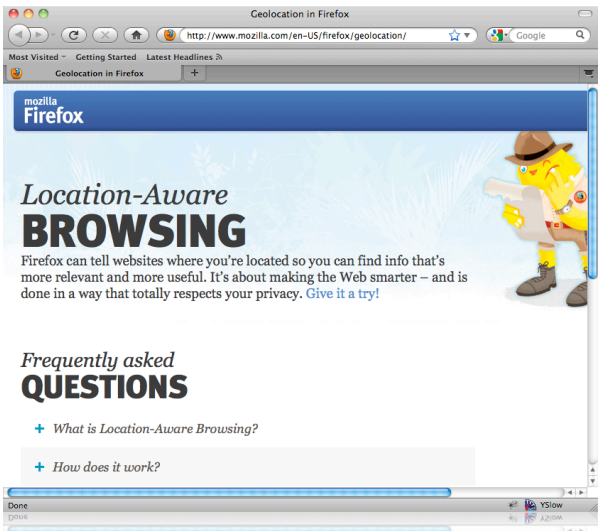
Opt-in.

http://www.w3.org/TR/geolocation-API/#security

Browsers tell you before data is sent.







Infobars are smart.

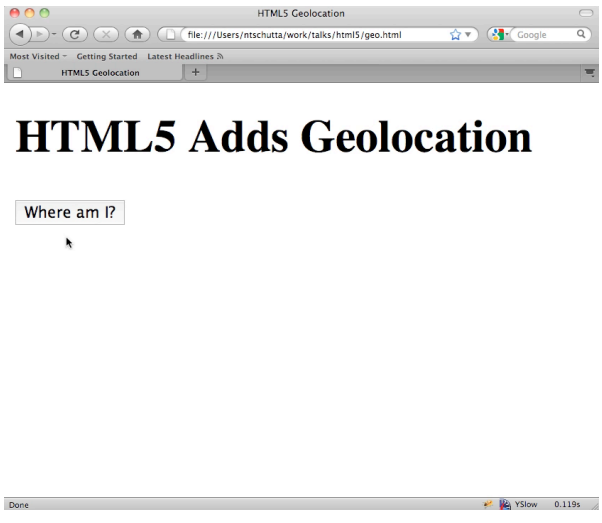
Often give link to further information.

Aren't modal.

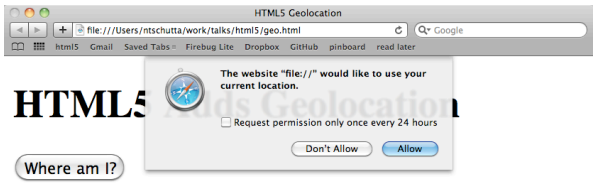
Tab specific.

Blocks.

You can go about your business in other tabs.



How does this work?




```
function whereAmI() {
  if (Modernizr.geolocation) {
    navigator.geolocation.getCurrentPosition(showLocation);
  } else {
    alert("this browser doesn't support geolocation, sorry!");
  }
}

function showLocation(position) {
  var latitude = position.coords.latitude;
  var longitude = position.coords.longitude;
  $("#location").html("lat: " + latitude + " and long: " + longitude);
}
```

New property.

navigator.geolocation

Simple API.

`getCurrentPosition()`

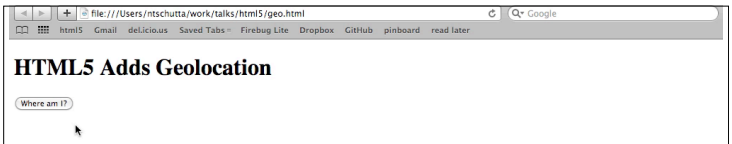
Browser "determines location," creates `Position`.

`Position` contains `Coordinates` & timestamp.

Coordinates

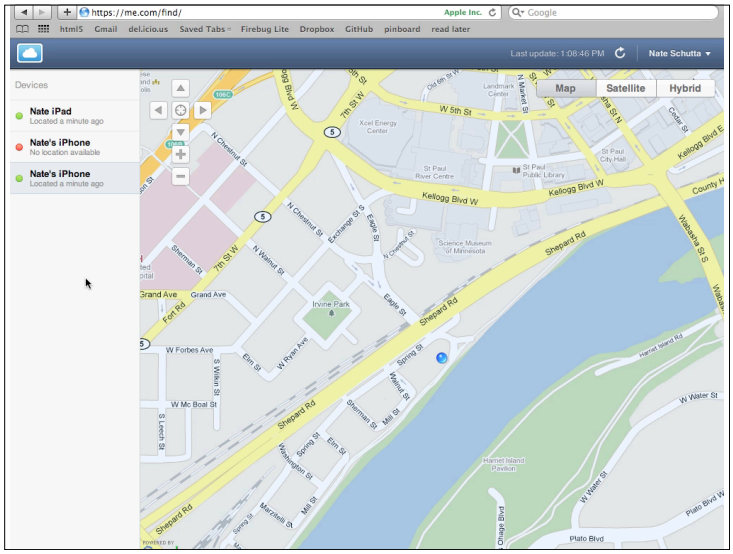
- latitude (double)
- longitude (double)
- altitude (double or `null`)
- accuracy (double)
- altitudeAccuracy (double or `null`)
- heading (double or `null`)
- speed (double or `null`)

You supply a callback function.



How accurate is it?

Can be *very* accurate.



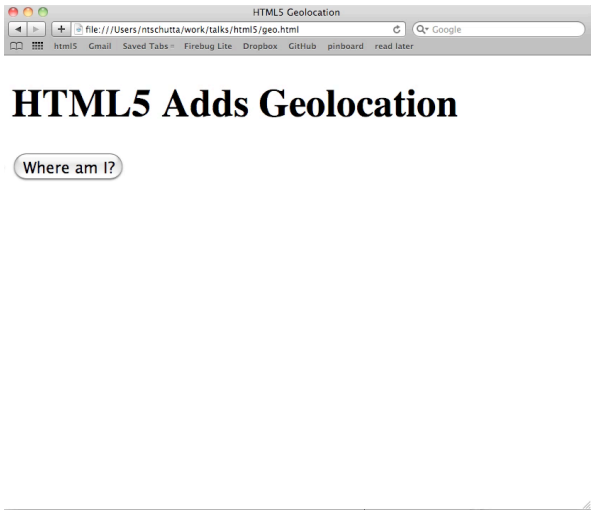
Your function receives a Position object with:

coords & timestamp

Can take some time ;)

getCurrentPosition()

Optional second arg:
error callback function.




```
function whereAmI() {
  if (Modernizr.geolocation) {
    navigator.geolocation.getCurrentPosition(showLocation, handleError);
  } else {
    alert("this browser doesn't support geolocation, sorry!");
  }
}

function showLocation(position) {
  var latitude = position.coords.latitude;
  var longitude = position.coords.longitude;
  $("#location").html("lat: " + latitude + " and long: " + longitude);
}

function handleError(error) {
  if(error.code == 1) {
    var message = "You want to keep your location private, that's OK!"
  }
  $("#location").html("Ooops! " + message);
}
```

Callback gets a
`PositionError` object.

Two attributes.

code & message

Code values...

- PERMISSION_DENIED (1)
- POSITION_UNAVAILABLE (2)
- TIMEOUT (3)
- UNKNOWN_ERROR (0)

getCurrentPosition()

Optional third argument.

PositionOptions

- enableHighAccuracy (boolean)
- timeout (long)
- maximumAge (long)

All attributes are optional.

Higher accuracy
may be slower.

Some devices have
separate permissions.

Timeout is based on
network time, not user.

Age allows you to
cache positions.

IE? Out of luck < 9.

Are other options.

Gears, device specific.

geo.js

http://code.google.com/p/geo-location-javascript/

Layer over various approaches.

Lab time!

Geolocation

- Using the geolocation API, create a function that displays your current latitude and longitude
- Handle the situation where a user opts out
- Handle the situation where location cannot be determined
- ${extract}/html5_workshop/labs/geolocation.html

Local storage.

Technically web storage.

Split into separate spec.

Some browsers call it DOM Storage.

Simple way to store *key/value* pairs.

Like cookies...

But bigger, stays local.

Very wide support.

Key is a *string*.

Data can be any JavaScript datatype...

But it's stored as a *string*.

Don't forget to parse...

Simple interface.

getItem(key)
setItem(key, value)

setItem() silently overwrite old values.

getItem() with unused key returns null...

Can treat localStorage as an associative array.

In other words, bracket notation.

localStorage.getItem("key")
localStorage["key"]

Can remove items:
`removeItem(key)`

Called on a nonexistent key does nothing.

`localStorage.length` gets number of stored values.

`key(index)` retrieves key at that index.

Index out of bounds `returns null`.

Also a `storage` event.

Calls to setItem, removeItem or clear...

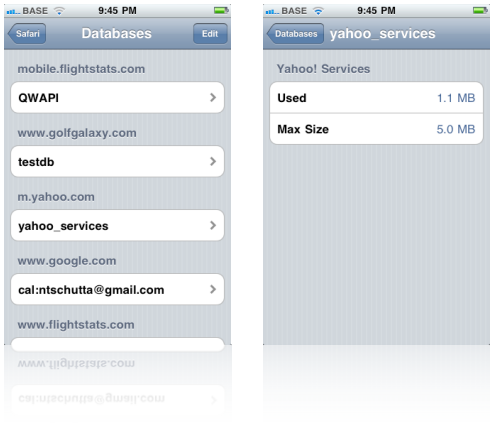
Provided *something changes.*

StorageEvent

- key
- oldValue
- newValue
- url

Can't cancel it.

Default max:
5 megabytes.



Exceed that...

QUOTA_EXCEEDED_ERR

Can you ask for more?

No.

Some browsers allow *the user* to control quota.

Lab time!

Local Storage

- Add a function that stores the current values locally
- Add a function that retrieves the values in local storage
- Add a function that displays the locally stored values
- Add a function that clears local storage
- ${extract}/html5_workshop/labs/localstorage_form.html

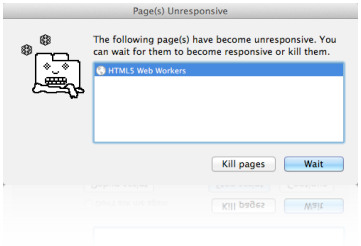
Web Workers.

JavaScript - single threaded.

Makes things easy.

Optimized DOM access.

But now we see the dreaded slow scripts.



And we do a lot in JavaScript today!

That's a problem.

Want a responsive UI.

How do we do that?

Run scripts in the background.

Independent of the UI.

Not interrupted
by user actions.

User continues on...

Do what we need to
do behind the scenes.

Relatively heavy weight.

Can hog resources!

Be careful...

How does it work?

First, no direct
access to the DOM.

Message passing.

How do we do it?

```
var foo = new Worker("foo.js");
```

Create a new worker.

foo.js contains
the worker code.

Pass message to worker.

```
foo.postMessage(input);
```

What can we send?

String, array, JSON object...

Cannot send a function.

How do we get messages?

Define a callback function.

foo.onmessage

Get an event object.

event.data = message from the worker.

event.target = which worker sent the message.

In the worker,
define *onmessage*.

Allows it to
receive messages.

```
onmessage = callFib;
```

Function handles
messages sent to worker.

Worker responds with
```
postMessage();
```

Messages between page
and worker are copied.

That's it!

Easy enough?

Couple more things...

```
foo.terminate();
```

Removes the worker.

Running scripts abort.

Can call `close()` from the worker itself.

`foo.onerror`

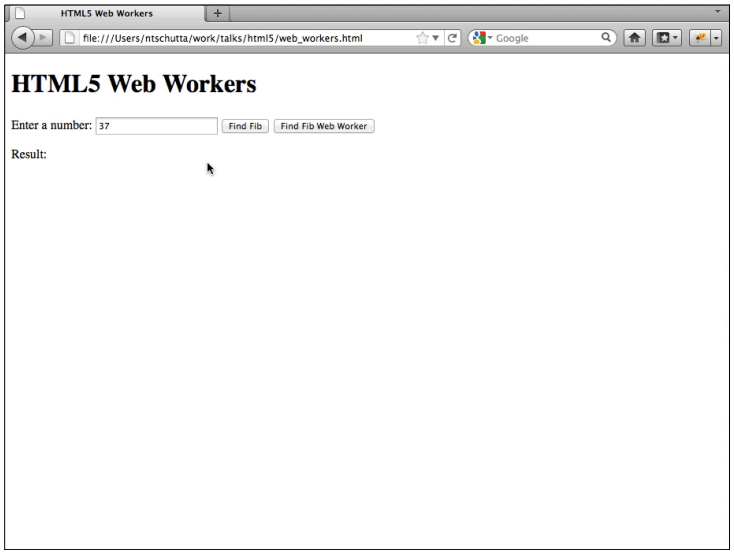
Error handler.

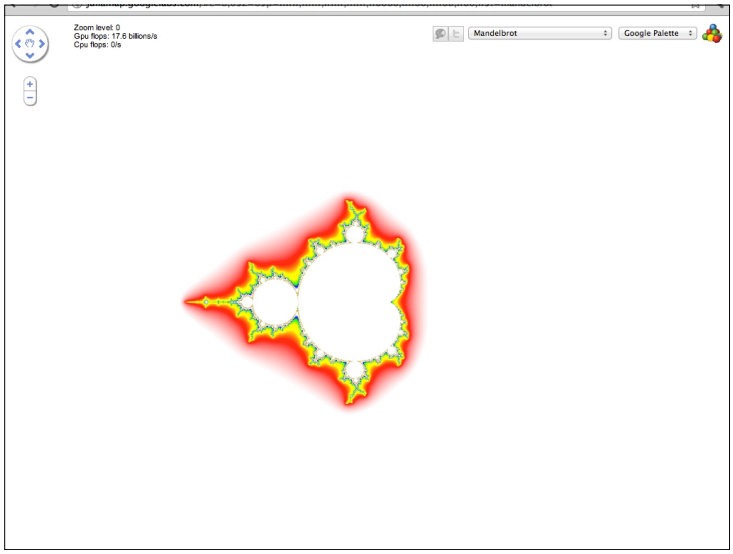
Workers can create workers!

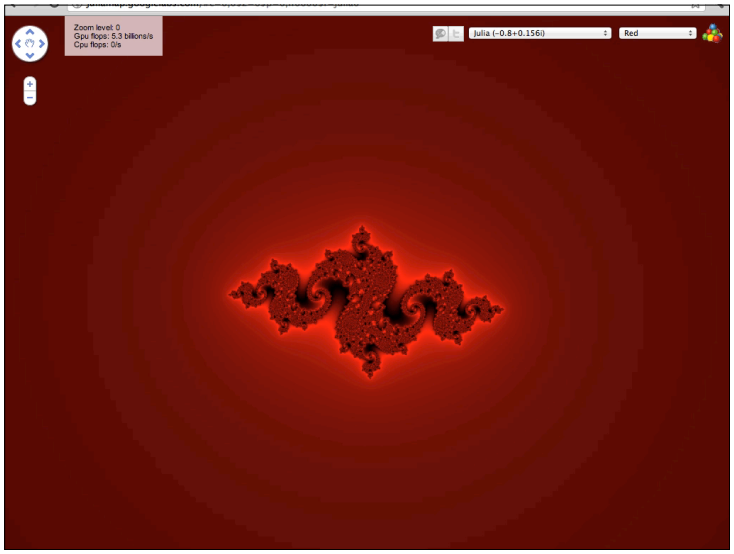
Same technique.

Just from the original worker.

Demo.







Lab time!

Web Workers

- Implement calculateFibWW
- Convert fibWW.js to be a web worker
- Use a web worker to call fibWW.js
- ${extract}/html5_workshop/labs/web_worker.html

There's more...

Offline.

Google Apps, GMail rock.

Some flights have wifi.

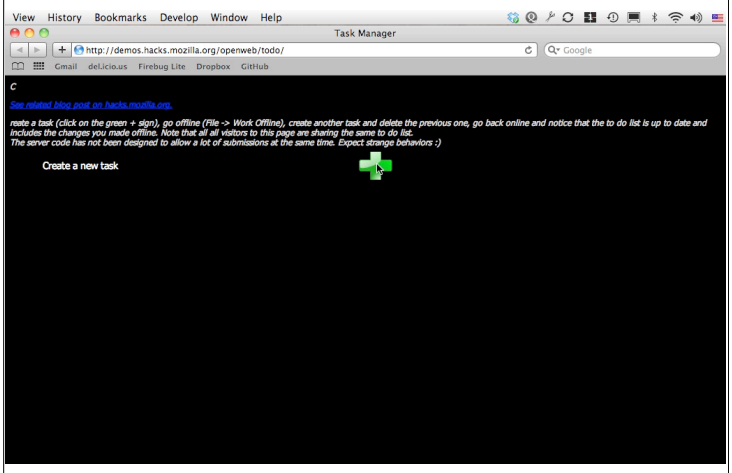
Not cheap.

Application cache.

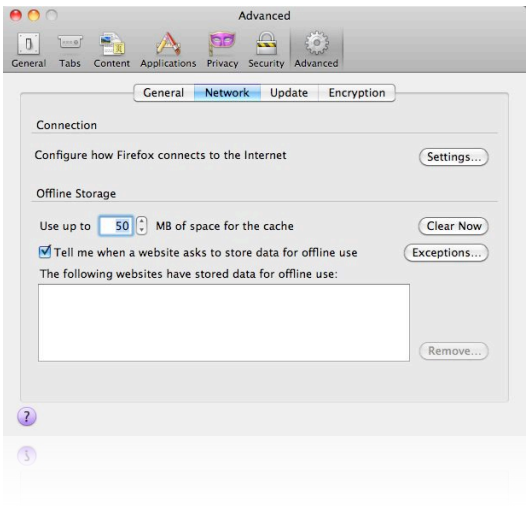
Web server tells client what it needs.

Application works when disconnected.

When it connects, changes are uploaded.



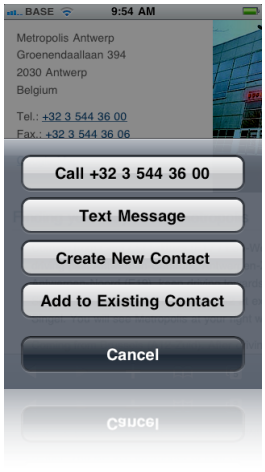
http://demos.hacks.mozilla.org/openweb/todo/



Microdata.

Add semantics
to your pages.

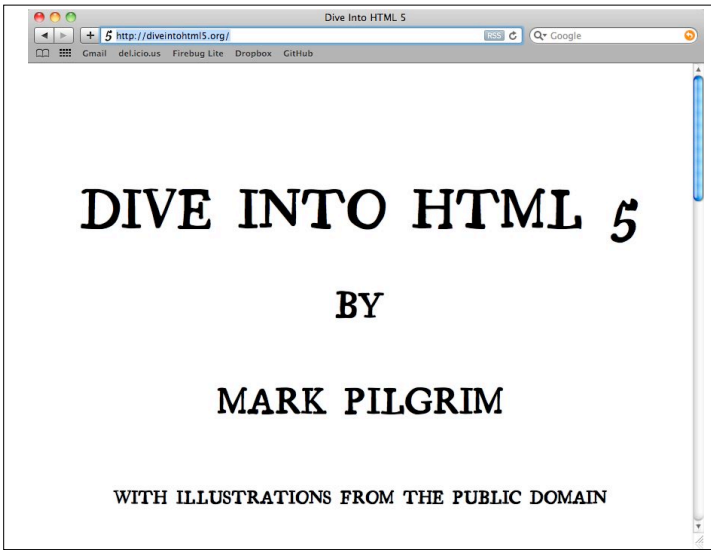
Licensing info, vCard.

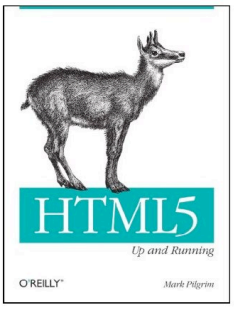


For search and browsers.

Books coming.

http://diveintohtml5.org/





http://books.alistapart.com/product/html5-for-web-designers

Plus *many* more.

Questions?!?

Thanks!

Please complete your surveys.