

# HTML5 Workshop

Nathaniel T. Schutta  
@ntschutta

# HTML5 Workshop

Nathaniel T. Schutta  
@ntschutta

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

Want more?

HTML5 for  
Developers  
LiveLessons



<http://www.informit.com/store/product.aspx?isbn=0132761718>

HTML5 Workshop

Nathaniel T. Schutta  
@ntschutta

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?

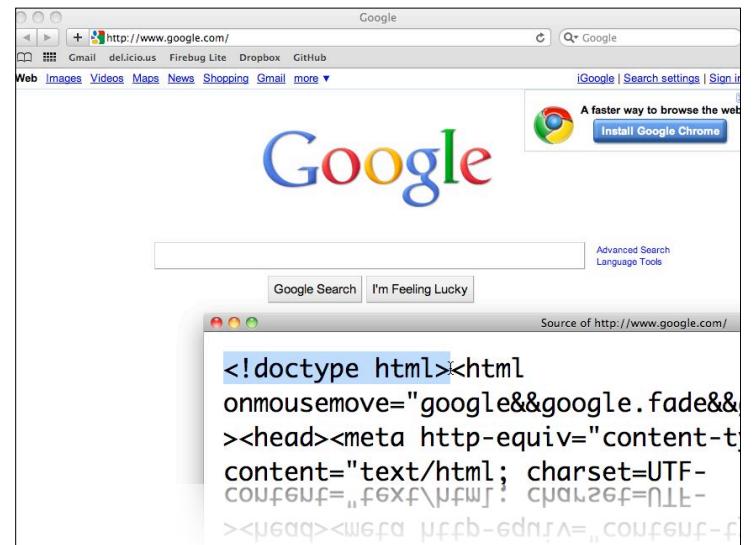
HTML5 is a collection  
of features...

Paving of cowpaths.

Evolutionary step.

Why should I care?

Market is moving.





Customers.

Mobile.

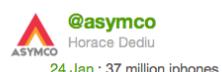
Next big thing.

Android activations  
outpacing baby births.

[http://www.telegraph.co.uk/technology/ces/9013487/  
CES-2012-Android-activations-outpacing-baby-births.html](http://www.telegraph.co.uk/technology/ces/9013487/CES-2012-Android-activations-outpacing-baby-births.html)



@lukew  
Luke Wroblewski  
There are more iPhones sold per day  
(402k) than people born in the World per  
day (300k). [twitter.com/#!/asymco/stat...](https://twitter.com/#!/asymco/stat...)



@asymco  
Horace Dediu  
24 Jan : 37 million iphones  
[http://news.cnet.com/8301-17938\\_105-57365767-1/  
apple-makes-more-iphones-than-humans-make-babies/](http://news.cnet.com/8301-17938_105-57365767-1/apple-makes-more-iphones-than-humans-make-babies/)  
<https://twitter.com/#!/lukew/statuses/161943568024469504>

Might be a natural cap  
there somewhere...

Horace Dediu  
ASYMCO

Android will reach a billion users in half the time it took Facebook.

[Reply](#) [13](#) [Retweet](#) [Favorite](#)

[Follow](#)

<https://twitter.com/asymco/statuses/254124178024833026>

iOS - a billion units in late 2014 or early 2015.

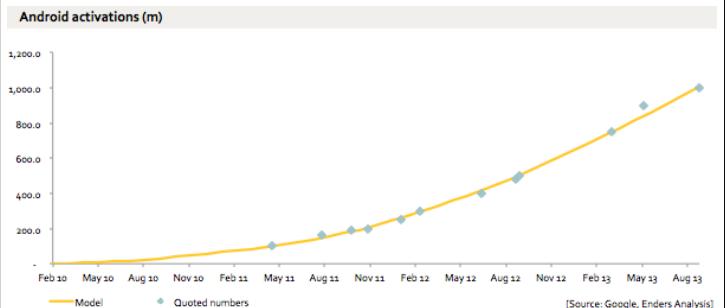
<http://www.asymco.com/2012/09/17/projecting-ios-devices-through-itunes-account-growth/>

> 700 million iOS devices sold.

<http://www.macworld.com/article/2048491/cook-more-than-700-million-ios-devices-sold.html>

1 billion Android activations.

<http://ben-evans.com/benedictevans/2013/9/8/androids-route-to-1bn>



<http://ben-evans.com/benedictevans/2013/9/8/androids-route-to-1bn>

To put that in perspective...

1 billion Windows PCs.

2.2 billion Internet users.

What about Windows 8?

<http://www.lukew.com/ff/entry.asp?1626>

2012, more Android devices sold than Windows devices.

Don't want to build a native app?

<http://techpinions.com/the-late-microsoft-windows-8/14882>

Don't want to support  
multiple OS?

Great! HTML5.

Mobile browsers excellent  
HTML5 support.

More likely than  
desktop peers.

Flash is dead.

HTML5 gaining steam.

**HTML5 and web standards.**

Every new Apple mobile device and every new Mac — along with the latest version of Apple's Safari web browser — supports web standards including HTML5, CSS3, and JavaScript. These web standards are open, reliable, highly secure, and efficient. They allow web designers and developers to create advanced graphics, typography, animations, and transitions. Standards aren't add-ons to the web. They are the web. And you can start using them today.

**html5 Gallery**  
A showcase of sites using HTML5 markup & API's

**SPONSORS**

**LOOKING FOR A JOB?**

- exoprivee** <http://exoprivee.com/> Added on September 16, 2013
- magpress** <http://magpress.net/> Added on September 16, 2013
- Wonder** [http://wonder.solutionsdigitales.com](#)
- Fab**: Junior Front-End Developer - Email
- Jola**: Digital Project Manager, Magento
- Curious LLC**: Web Designer
- Comcast**: Mobile Build Engineer

**HTML 5 Demos and Examples**

HTML 5 experimentation and demos I've hacked together. Click on the browser support icon or the technology tag to filter the demos (the filter is an OR filter).

**Filter demos:** canvas classlist contenteditable dataset dnd events file-api file geolocation getUserMedia hidden history manifest offline postMessage sd-database storage svg video websocket workers xhr2

Demo	Support	Technology
Stream video and filter with canvas		getUserMedia canvas
Stream video to the browser		getUserMedia
Also works on Opera Mobile 12		
Drag and drop and XHR upload		file dnd xhr2
Hidden property		hidden
Simple class manipulation		classlist
Storage events		storage

Customers want sites that work on today's devices.

HTML5 is the answer.

Respects what we're actually doing.

Forms, controls...  
well worn hacks.

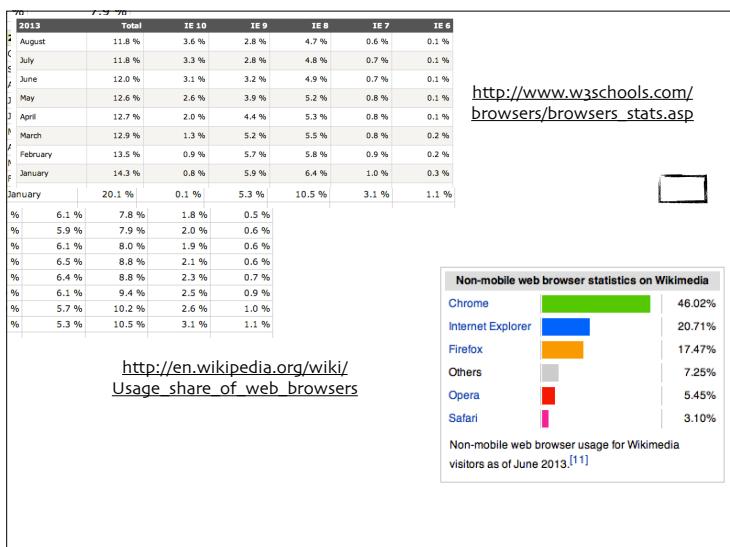
Helps us build  
applications!

What can I do?

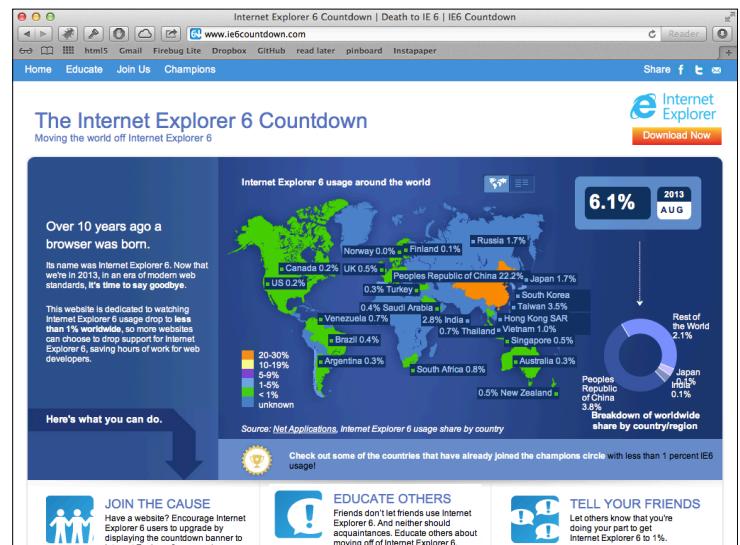
Browser support isn't  
universal.

Shocking.

Older browsers pervasive.



IE 6 is dead...



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:  
Mozilla/5.0 (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/](http://www.modernizr.com/)

```
function check_geo_modernizer() {
  if (Modernizr.geolocation) {
    var message = "yep, we can do geolocation!";
  } else if (Modernizr.geolocation) {
    message = "actually, we can't do geolocation";
  }
  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() {
  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 h264 thanks";
    }
  }
  show_results(message);
}
```

Ogg? 264? WebM?

Patents...

[HTTP://DARINGFIREBALL.NET/2010/03/GIF\\_H264\\_PATENTS](http://daringfireball.net/2010/03/gif_h264_patents)

Browser support falls on philosophical lines.

[HTTP://DARINGFIREBALL.NET/2010/03/ON\\_SUBMARINE\\_PATENTS](http://daringfireball.net/2010/03/on_submarine_patents)

Though philosophies shift...

```
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://caniuse.com)

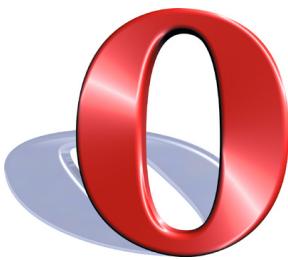
[HTTP://WWW.FINDMEBYIP.COM/#TARGET-SELECTOR](http://www.findmebyip.com/#target-selector)

Another view...

[HTTP://HTML5TEST.COM/INDEX.HTML](http://html5test.com/index.html)



The screenshot shows a web browser window displaying the results of the HTML5 Test. The title bar reads "The HTML5 test – How well does your browser support HTML5?". The main content area displays a large score of "378 AND 8 BONUS POINTS" out of a total of 500 points. Below the score, it says "your browser scores". A sidebar on the right provides a brief explanation of the test's purpose: "The HTML5 test score is an indication of how well your browser supports 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." At the bottom, there is a section titled "Parsing rules" with a score of "10" and a note about supporting SVG and MathML. The browser interface includes tabs for "your browser", "other browsers", "compare", "news", and "about the test".



The HTML5 test - How well does your browser support HTML5? [html5test.com/index.html](https://html5test.com/index.html)

your browser scores **442** AND 9 BONUS POINTS out of a total of 500 points

You are using Opera 16.0 on Mac OS X 10.8.5 Correct? ✓ ✎

**Parsing rules** +2 bonus points **10**

<!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.

**SPONSORS**

[CREATE A FREE HTML5 WEBSITE](#) [Click Here](#)



The HTML5 test - How well does your browser support HTML5? [html5test.com/index.html](https://html5test.com/index.html)

your browser scores **414** AND 10 BONUS POINTS out of a total of 500 points

You are using Firefox 24.0 on Mac OS X 10.8.5 Correct? ✓ ✎

**Parsing rules** +2 bonus points **10**

<!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.

**SPONSORS**

[Free HTML5 Templates](#) [PSD FILES INCLUDED](#) [DOWNLOAD](#)



The HTML5 test - How well does your browser support HTML5? [html5test.com/index.html](https://html5test.com/index.html)

your browser scores **463** AND 13 BONUS POINTS out of a total of 500 points

You are using Chrome 29 on Mac OS X 10.8.5 Correct? ✓ ✎

**Parsing rules** +2 bonus points **10**

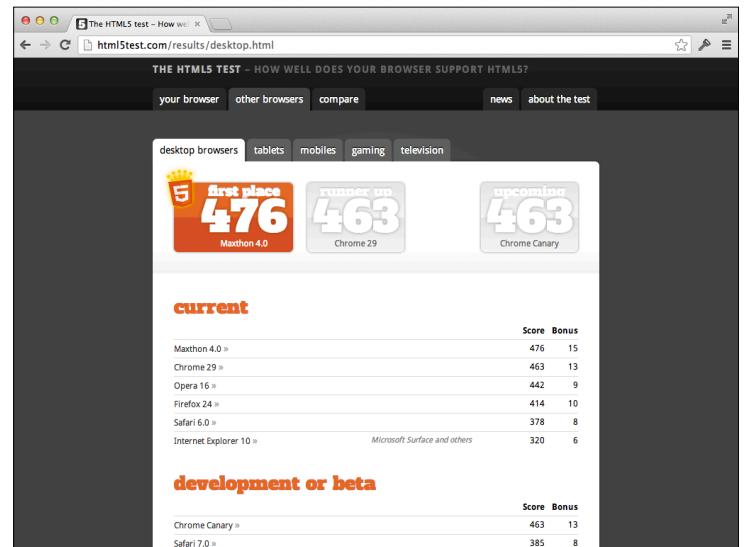
<!DOCTYPE html> triggers standards mode	Yes ✓
HTML5 tokenizer	Yes ✓
HTML5 tree building	Yes ✓

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.

**SPONSORS**

[GOJS Canvas Library](#)

And other browsers?



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?

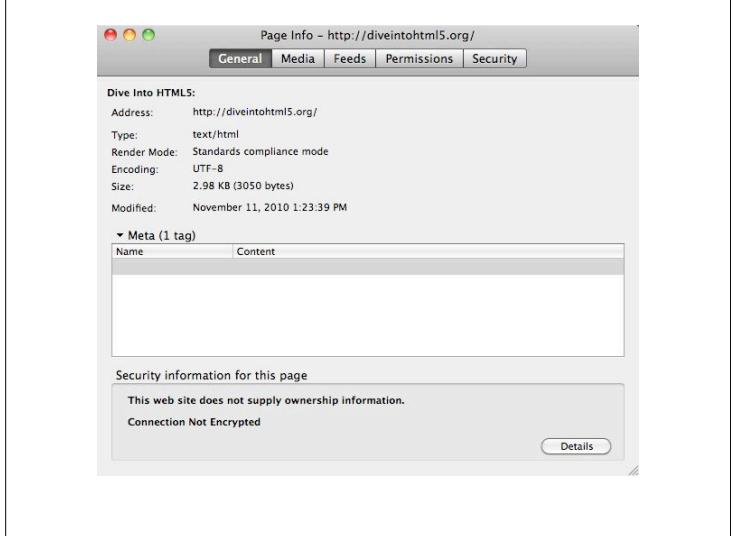
Quite a mouth full...

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...

JavaScript is the default  
scripting language.

So long *type=!*

```
<script src="../javascript/fib.js"></script>
<script src="../javascript/jquery-1.7.js"></script>
<script>
$(document).ready(function(){
  $("#callFib").click(calculateFib);
  $("#callFibworker").click(calculateFibWW);
});

function calculateFib() {
  var input = $("#num").val();
  var result = fib(input);
  $("#outcome").html(result.toString());
}
```

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.

About us	Client portfolio	What we do	How we work	News & events	ThoughtWorks family	ThoughtWorks UK
› Our history	› View all	› Services	› Technology	› Upcoming events	› Martin Fowler	ThoughtWorks Ltd
› People		› Products	› Agile & lean	› Past events	› ThoughtWorks Studios	9th Floor Berkshires House 168-173
› Leaders			› Offshore	› News archive	› opensource.thoughtworks.com	High Holborn
› Revolutionizing IT			› Customer experience	› Blogs	› testing.thoughtworks.com	London, WC1V 7AA
			› Testing		› offshore.thoughtworks.com	T +44 (0) 20 7497 4500
					› agile-translations.thoughtworks.com	F +44 (0) 20 7497 4501
					› thoughtworker.com	E info.uk@thoughtworks.com
						Other global offices >

© 2010 ThoughtWorks, Inc. | Privacy policy

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" ;)`

Doesn'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](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 datetim="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.

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/](http://REMYSHARP.COM/2009/01/07/HTML5-ENABLING-SCRIPT/)

What's all the fuss about?

Divs work, right?

Document outline.

Before, headings were  
our only hope.

[HTTP://GSNEEDERS.HTML5.ORG/OUTLINER/](http://GSNEEDERS.HTML5.ORG/OUTLINER/)

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...

## HTML5 Adds New Input Types

Telephone:

URL:

Email:

```
<!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

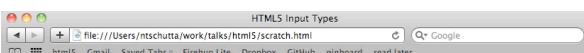
Search:



```
<!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?

A screenshot of a web browser window titled "HTML5 Input Types". The URL is "file:///localhost/Users/ntschutta/work/talks/html5/scratch.html". The page content includes the title "HTML5 Adds New Input Types" and two input fields: a text input with the value "0" and a range slider with a value of 0.

```
<!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 Chrome.

And you may not like it.

**HTML5 Input Types**

Search:

Search (Placeholder):

Number:

Range:

Color:

Telephone:

URL:

Email:

Date:

Month:

Week:

Time:

Date/Time:

[View \(1000\)](#)

```
<!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>
```

**Really Simple Address Book**

Name:

Email:

Phone:

Street:

City:  State:  Zip:

Website:

IM:

Birthday:

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!

But it's only in  
Opera and Chrome.

Bummer.

Uses native picker.

## HTML5 Adds New Input Types

Search:

```
<!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.

Today we use JavaScript.

```
<!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>
```

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!

## HTML5 Adds New Input Types

Email:

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...improving.

Safari, no error messages.

Just doesn't submit ;)

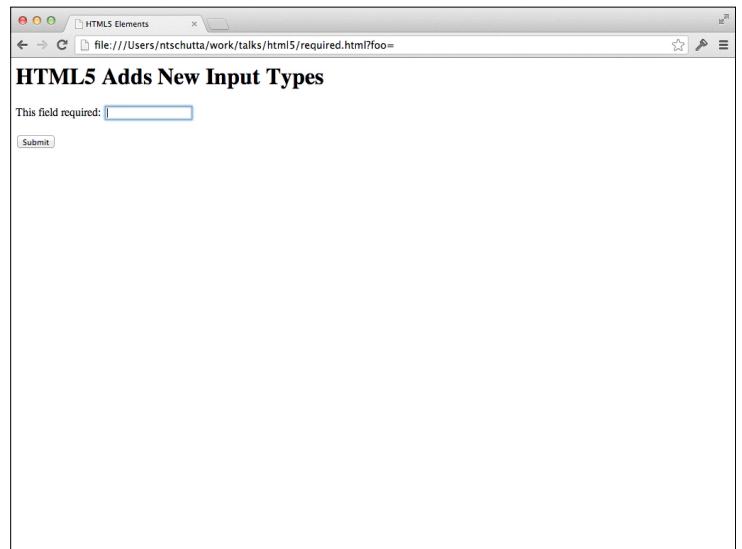
Very user friendly.

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 - border, 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!

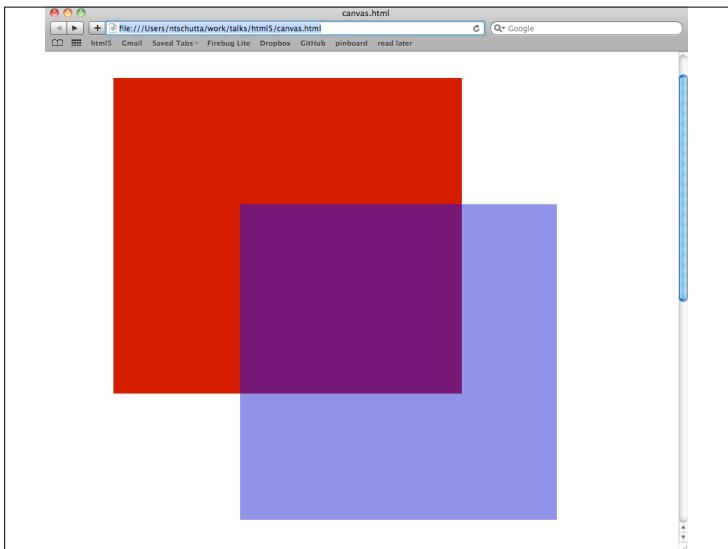
To draw, we need a context.

```
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, 500, 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.

- ➊ `beginPath` - creates path
- ➋ `closePath` - tries to close the shape
- ➌ `stroke` - draws an outlined shape
- ➍ `fill` - draws a solid shape
- ➎ `moveTo` - moves the “pen”, doesn’t draw



[HTTPS://DEVELOPER.MOZILLA.ORG/EN/  
CANVAS\\_TUTORIAL%3ADRAWING\\_SHAPES](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(75,75,40,0,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.

`arc(x, y, radius, startAngle, endAngle, anticlockwise)`

`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.

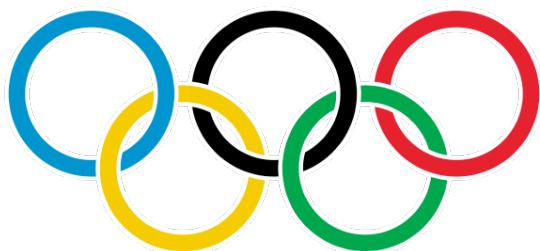
Animations!

Whew!

Lab time!

## Olympic Rings

- Using the various canvas methods, draw the Olympic Rings
- [\\${extract}/html5\\_workshop/labs/canvas\\_rings.html](#)



That's not all!

You can create text too.

Fairly low level API.

Give it the text to write,  
the font, weight, color, etc.

context.font  
context.textalign

bold 1.5em sans-serif

Stuff like that.

But hey, it's CSS.

context.fillText(text, x, y)

Also `strokeText`.

Lab time!

## Writing on a Canvas

- ➊ Using the various canvas text methods:
- ➋ Draw the text from an input box
- ➌ Change the color of the text based on user input
- ➍ Allow the user to specify italics
- ➎ `$(extract)/html5_workshop/labs/canvas_text.html`

Again, very rich space.

Want more?

<http://corehtml5canvas.com>

C O R E  
**HTML5**  
**CANVAS**

Graphics, Animation,  
and Game Development



Geolocation.

Where in the world  
are you?



[HTTP://WWW.OFFMAPS.COM/](http://WWW.OFFMAPS.COM/)

Very helpful on phones!

Technically not a  
part of HTML5.

Geolocation Working Group.

Wide browser support.

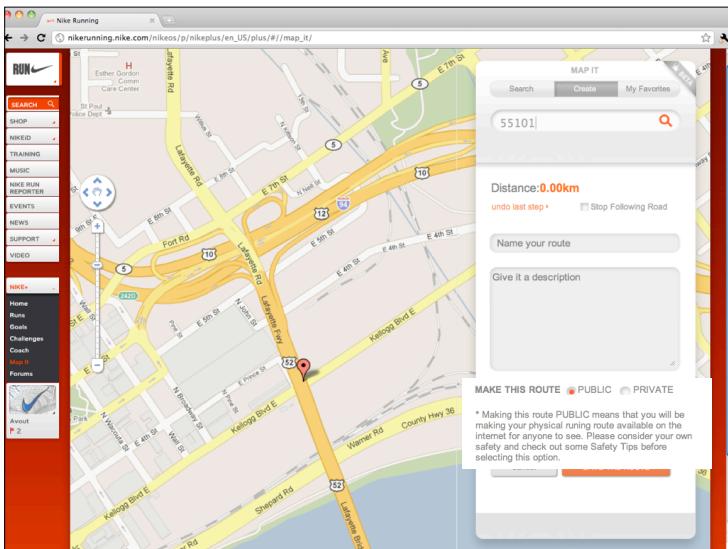
Your browser  
doesn't support it?

Device specific options.

Privacy issue?

Absolutely.

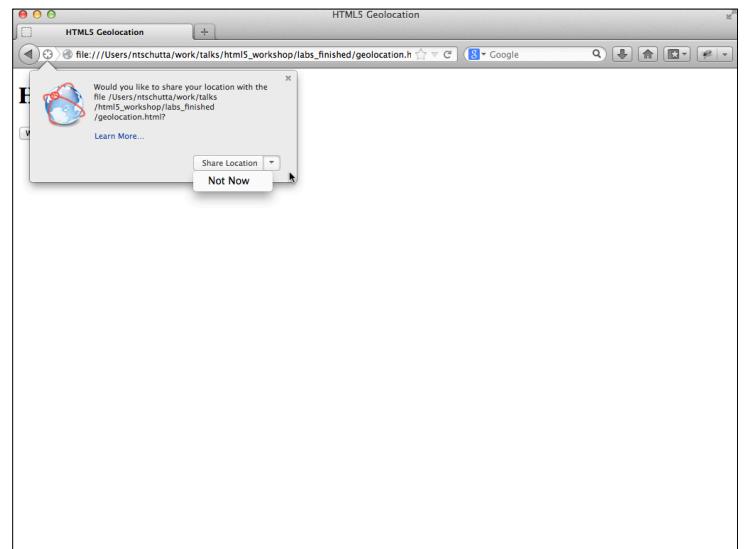
If you know where  
the device is...



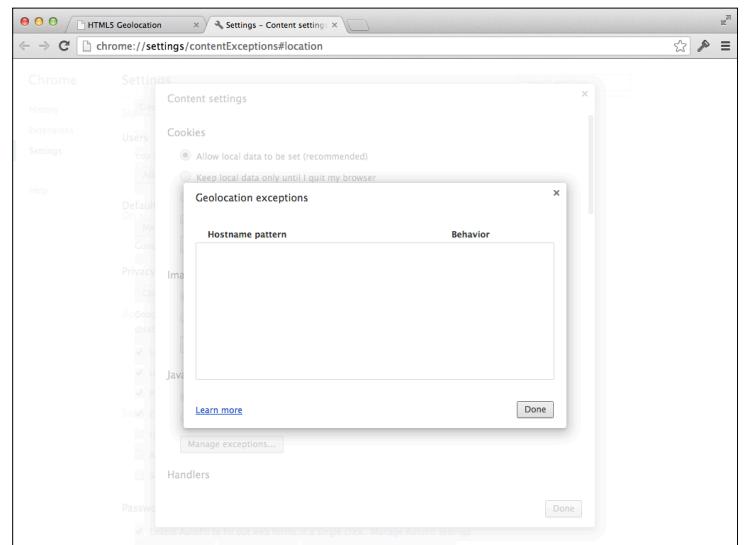
Opt-in.

[HTTP://WWW.W3.ORG/TR/GEOLOCATION-API/#SECURITY](http://www.w3.org/TR/geo-location-api/#security)

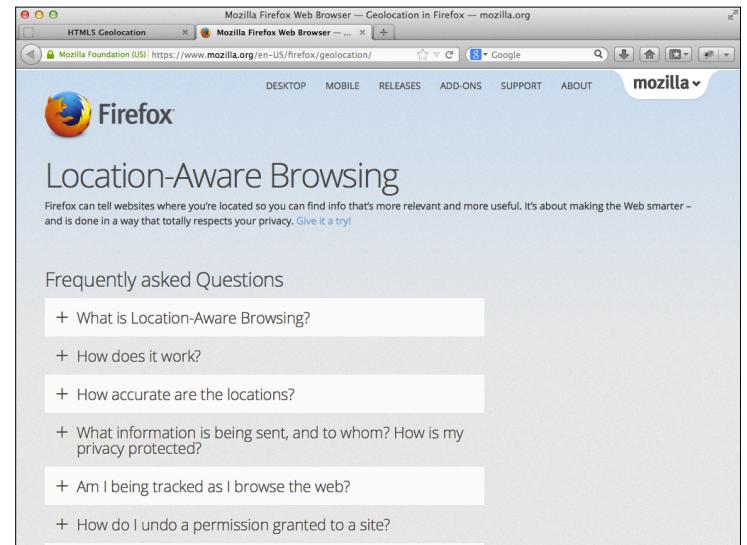
Browsers tell you  
before data is sent.



Some block it by default.



Some provide more info.



Infobars are smart.

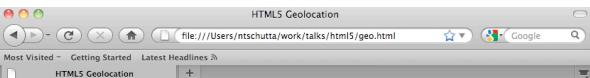
Often give link to further information.

Aren't modal.

Tab specific.

Blocks.

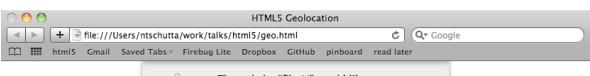
You can go about your business in other tabs.



## HTML5 Adds Geolocation

Where am I?

How does this work?



## HTML5 Adds Geolocation

Where am I?

```
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

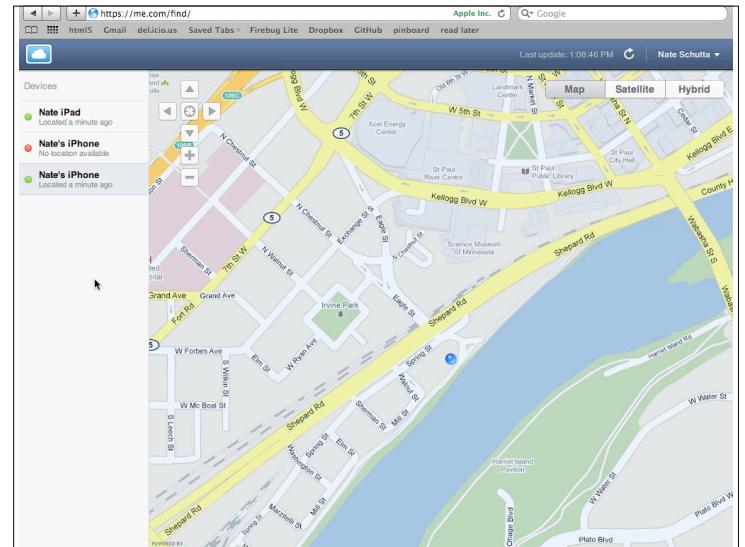
- latitude (double)
- longitude (double)
- altitude (double or null)
- accuracy (double)
- altitudeAccuracy (double or null)
- heading (double or null)
- speed (double or null)

You supply a  
callback function.

A screenshot of a web browser window. The address bar shows the URL `file:///Users/ntschutta/work/talks/html5/ggeo.html`. The page content is titled "HTML5 Adds Geolocation". It contains a single button labeled "Where am I?". The browser's toolbar at the top includes icons for back, forward, search, and other tabs.

How accurate is it?

A screenshot of a web browser window. The address bar shows the URL `https://me.com/find/`. The page content is titled "Can be \*very\* accurate.". It displays a map of a city area with several devices tracked. A legend on the left shows three entries: "Nate iPad" (green dot), "Nate's iPhone" (red dot), and "Nate's iPhone" (green dot). The map shows streets like Grand Ave, Sherman St, Elm St, W. Water St, and various numbered streets. Landmarks include the Xcel Energy Center, St Paul River Centre, St Paul Public Library, Science Museum of Minnesota, and Hamlin Island Pavilion.



Your function receives a Position object with:

coords & timestamp

Can take some time ;)

`getCurrentPosition()`

Optional second arg: error callback function.



**HTML5 Adds Geolocation**

[Where am I?](#)

```

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("Oops! " + 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

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  
overwrites 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

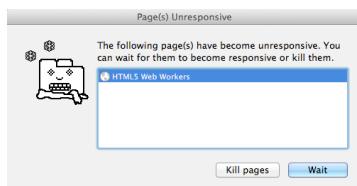
Web Workers.

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`

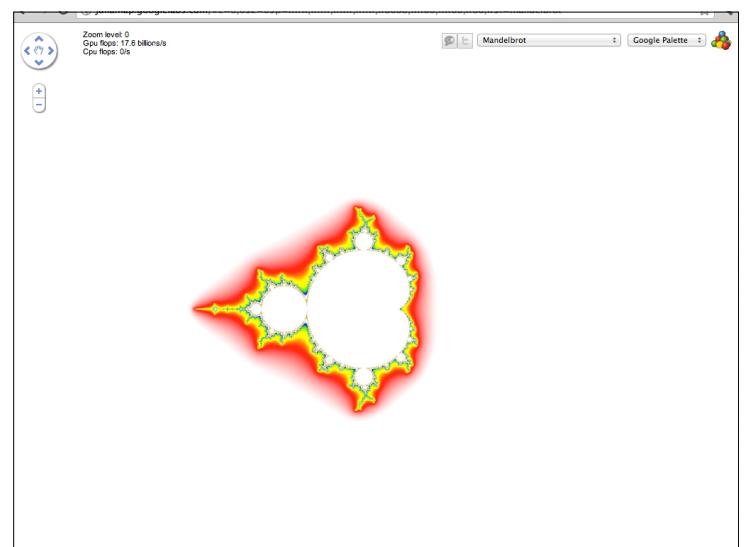
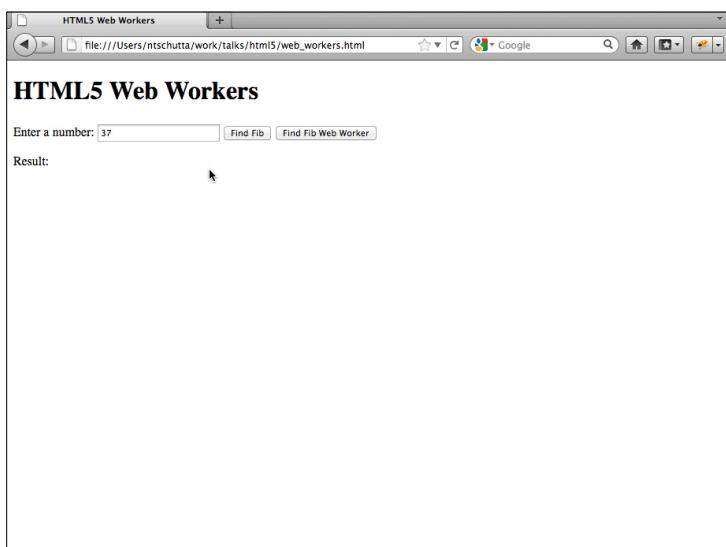
Error handler.

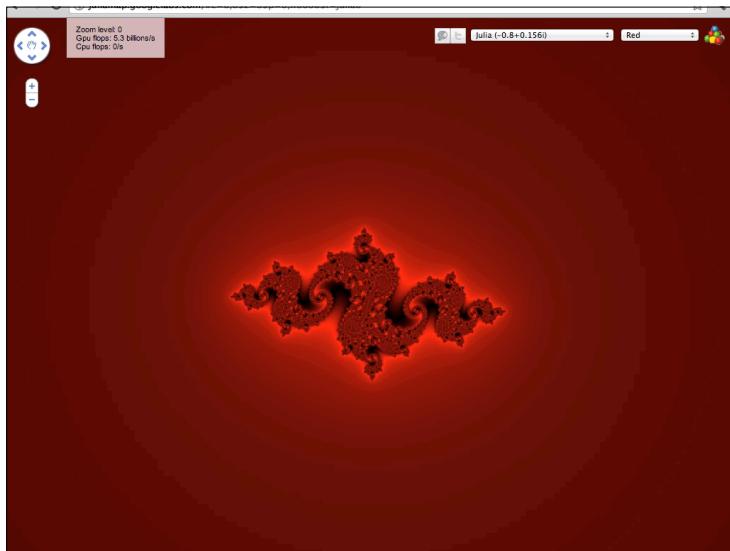
Workers can create workers!

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`

Selectors.

We've all written this...

`document.getElementById`

Ever fat finger that?

jQuery taught us a  
better way...

jQuery selectors.

CSS 1-3.

Plus more ;)

Very powerful.

How did we live  
without it for so long?

HTML5 borrows from  
jQuery selectors.

`document.querySelector`

Simple syntax.

Powerful selectors.

Can be very precise.

By id, class, tag, child elements, combinations, etc.

What about multiple matches?

Should return the first one.

What if you need a collection of elements?

`document.querySelectorAll`

Returns an array of elements that match the selector.

Very useful.

Lab time!

### jQuery Style Selectors

- ➊ Display the student's name retrieving the element by ID
- ➋ Display the student's age retrieving the element by class
- ➌ Display the first item in the list retrieving the element with a child selector
- ➍ Display the third item in the list retrieving all the li elements
- ➎ \${extract}/html5\_workshop/labs/selectors.html

Drag and Drop.

Many modern web apps include drag and drop.

Various libraries, numerous implementations.

Yet another nod to  
what we're doing!

Specify something  
as draggable...

Say where it can  
be dropped...

Supply handlers for  
the various events.

What can be draggable?

Almost anything.

`draggable="true"`

Easy to customize.

`dataTransfer` property -  
send data.

- `dragstart`
- `dragend`
- `dragenter`
- `dragleave`
- `dragover`
- `drop`

Very flexible.

Allows for some  
advanced techniques.

Drag items between  
windows...

Dragging files!

Desktop to browser...  
or vice versa.

More focussed on  
data transfer.

Very useful!

Libraries will eventually defer  
to native implementations.

Lab time!

## HTML5 Drag and Drop

- Make the two images draggable
- Make the text div draggable
- Make the ul draggable
- When dropped on the target, add the dragged item to the target
- `$(extract)/html5_workshop/labs/dragndrop.html`

Video.

Video on the  
web isn't new.

But it's always  
involved a plugin.

And those plugins haven't  
always been...all that safe.



It'd be nice if the browser could just play a video.

Now it can!

Now we have a video element.

Give it a height and width.

Source for the video.

Looks a lot like  
the image tag.

Codecs are still an issue.

Likely need to encode in  
more than one.

Sorry.

Though Firefox now  
supports H.264.

Lab time!

## Video

- Using the video element, show the video found in the images folder
- `${extract}/html5_workshop/labs/video.html`

History.

Forward/back buttons.

Back in the day...

Remove the chrome!

Hope users don't use keyboard shortcuts.

Why do we need  
back buttons?

The browser has one.

History API.

`window.history`

Want to change the URL?

`history.pushState()`

Browser won't  
load the URL.

Doesn't have to exist.

Three parameters.

1. State object.

JavaScript object.

Anything that's serializable.

Object associated with the new history element.

2. Title.

Typically ignored.

3. URL.

Can be relative.

Must be from the same origin.

Protocol, domain, port  
must remain the same.

Page doesn't load.

Creates an entry in  
browser history.

Backwards navigation  
throws `popstate` event.

Your app can  
respond accordingly.

`history.replaceState()`

Works like `pushState`...

Modifies current history entry.

Support is decent.

History.js.

<https://github.com/balupton/history.js>

Lab time!

## History API

- Add “dummy” pages to the browser’s history stack
- Implement the forward and back buttons to navigate the history stack
- Implement the length function to show how many items are in history
- `$(extract)/html5_workshop/labs/history.html`



Offline.

Google Apps, GMail rock.

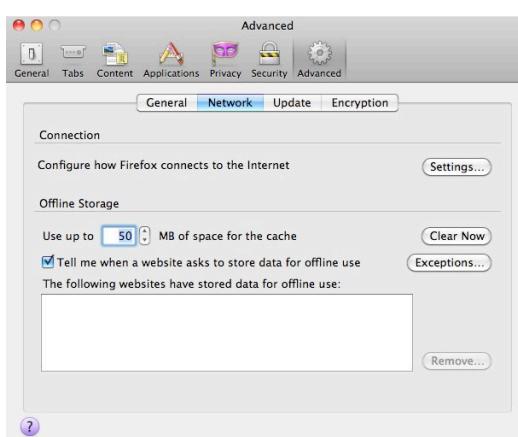
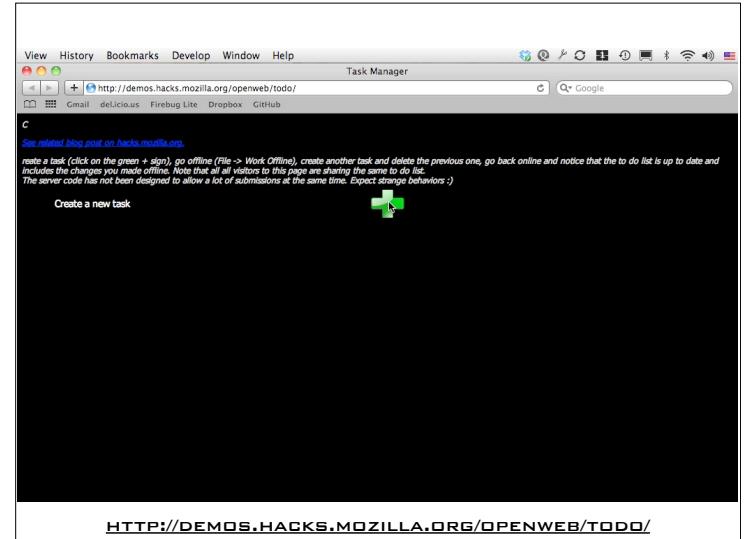
Not cheap.

Application cache.

Web server tells client what it needs.

Application works when disconnected.

When it connects, changes are uploaded.



New events:  
online and offline.

Hard part?

What to do when  
you're back online...

Lab time!

### Offline Detector

- Create a method that shows the network status
- Bind to the online and offline events triggering the method
- `#{extract}/html5_workshop/labs/offline.html`

There's more...

Microdata.

Semantic web.

Context, meaning.

Need more than just semantic markup.

Machine readable.

Search engines,  
mashups, smart apps.

License, contact info,  
location, resume, etc.

Not the first attempt.

XFN, RDFa, Microformat.

*class* and *rel* attributes.

Seen as hacks by many.

HTML5 gives us new syntax.

*itemscope* and *itemprop*  
attributes

Item is a name/value pair.

Add a descriptive  
property name.

Standard vocabularies.

[HTTP://SCHEMA.ORG/](http://schema.org/)

Search engines.

Book, recipe, event,  
person, place, review...

Add an `itemtype`.

Use the property names.

Custom attributes.

`data-`

Define whatever you need.

Not intended to  
share information.

Valid markup.

You get what you need.

Web Sockets.

Bi-directional  
communication.

Browser and server.

Duplex channel.

Direct communication.

Upstream and downstream.

Single socket.

Haven't we done this before?

Sort of...

Hacks.

Comet.

aka Ajax Push, HTTP  
Streaming, Reverse Ajax...

Long polling.

Applets.

Flash.

CometD.

Dojo Foundation.

Bayeux Protocol.

All of these methods  
have various issues.

Some proved difficult  
for developers.

Others involved complex  
streaming protocols.

Many stumbled on  
proxies and firewalls.

Impacted servers.

Latency and throughput.

HTTP 1.1 spec itself...

Limited browser connections!

We could do it...but  
it wasn't easy.

Web sockets improve on  
these approaches.

Easier on servers.

Though it may impact  
your server architecture.

Connections long lived...

Detects proxies.

Greatly reduces  
network chatter.

Designed for low  
latency applications.

Real time applications.

Changes what we do on the server.

Many apps were designed for full page cycle.

Changes the app servers themselves.

Large pool of open connections.

Threading, non-blocking IO.

Not all web servers support web sockets.

But most do.

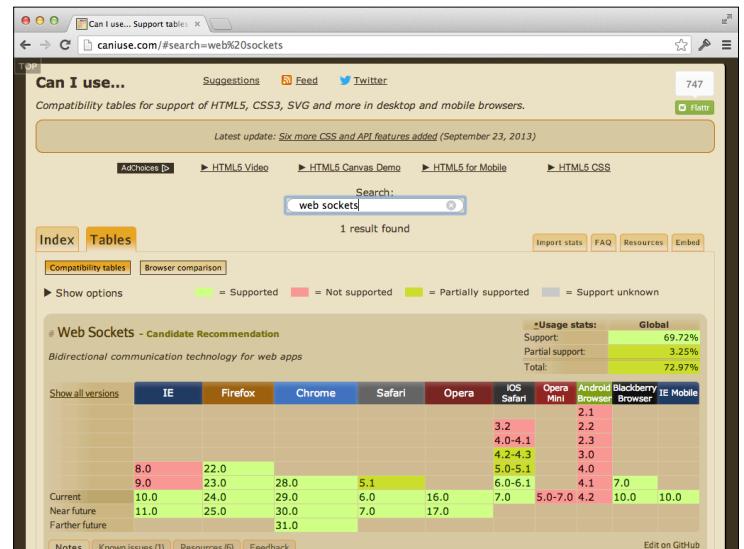
Jetty, Node, Socket.IO, Tomcat, JBoss, GlassFish.

Protocol is evolving.

New protocol:  
ws and wss

Requires browser and server support.

Support is...evolving.



Technically not a part of  
HTML5 spec.

Internet Engineering  
Task Force.

Still under the umbrella ;)

API.

Very simple.

`new WebSocket(ws://url);`

Notice the new URI  
schemes: ws and wss.

Throws a series of events.

`onopen;`  
`onmessage;`  
`onclose;`  
`onerror;`

Can associate callbacks  
to any event.

*send(...);*

Sends a message.

Payload options:

String, Blob, ArrayBuffer.

*readyState;*

Magic values.

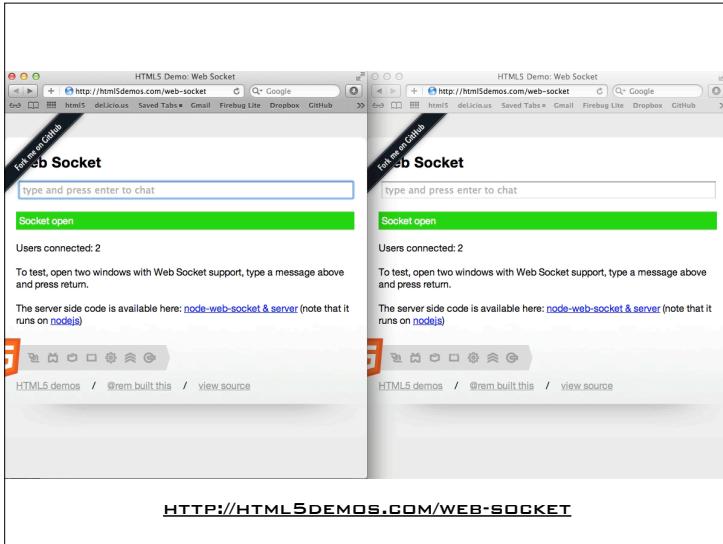
## Code values...

- CONNECTING (0)
- OPEN (1)
- CLOSED (2)

*close();*

Terminates the connection.

Demo.



Remember, support is soft.

Be prepared to fall back.

Or use a library.

Socket.IO for example.

File API.

[HTTP://SOCKET.IO/](http://socket.io/)

<http://www.w3.org/TR/FileAPI/>

Audio.

Device APIs.

WebGL.

Controllers (think video game consoles).

And on and on!

HTML5 is huge.



But you don't have to do it all to embrace it.

Pick and choose.

What scratches  
your itch?

What solves your pain.

Adopt that.

Specs are evolving.

So are the browsers.

Keep a weather eye out.

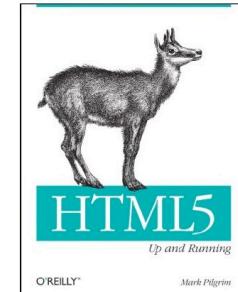
Be prepared.

Lots of books.

[HTTP://DIVEINTOHTML5.INFO/](http://diveintohtml5.info/)

DIVE INTO HTML5  
BY  
MARK PILGRIM

WITH CONTRIBUTIONS FROM THE COMMUNITY





[HTTP://BOOKS.ALISTAPART.COM/PRODUCT/HTML5-FOR-WEB-DESIGNERS](http://BOOKS.ALISTAPART.COM/PRODUCT/HTML5-FOR-WEB-DESIGNERS)

Plus \*many\* more.

HTML5 for  
Developers  
LiveLessons



<http://www.informit.com/store/product.aspx?isbn=0132761718>

Questions?!?

Thanks!

Nathaniel T. Schutta  
@ntschutta