HTML5 – 201

Nolan Erck

About Me

- Independent Consultant – Southofshasta.com
- Co-Manager - SacInteractive User Group
- Stuff I've used: HTML, ColdFusion, .NET, PHP, C++, Java, JavaScript, jQuery, SQL, etc.
- Stuff I've done: CycleGear, Cast Images, Grim Fandango, SimPark, SimSafari, Star Wars Episode 1, .
- Music junkie.

I'M REAL PROUD OF MY RECORD COLLECTION

HTML5

- The spec has been finalized...finally!
- Browser support is getting pretty good
  - (except for those IE people)
- BUT...
  - Just because the spec is final doesn't mean browsers support everything yet.
    - Example: SVG fonts barely work anywhere
    - CanIUse.com

HTML5

- Several technologies, being marketed together.
- <HTML5>
- CSS3
- JavaScript
- SVG, etc

Basic HTML5 Things

- First stuff we used
  - Easier DOCTYPE tags
  - Rules about <tags /> aren't as strict
  - <section>, <aside>, etc.
  - Yes, <canvas> too.
  - New HTML form elements
    - (browser support still in progress).
  - New CSS
    - Media queries, shadows, transparency/opacity, importing fonts, etc.

Stuff we'll cover today

- Autofocus and Placeholder
- SVG
- Audio and Video
- Geolocation
- Drag and Drop
- Data Sets
- Custom elements
- Other resources, questions, etc.

Autofocus

No more JavaScript needed to set focus on a form field!

```
<!DOCTYPE html>
<body>
<form action="submit.cfm">
  First name: <input autofocus type="text" placeholder=”Your First Name”
                     name="fname" /><br />
  Last name: <input type="text" name="lname"
                    placeholder=”Your Last Name” /><br />
  Age: <input type="text" />
  <input type="submit" />
</form>
</body>
</html>
```

(Demo 1)

SVG – Scalable Vector Graphics

- Draw vector graphics in HTML “on the fly”
  - No JPGs or PNGs required.

```
<svg>
        <circle cx="50" cy="50" r="30" />
        <rect x="0" y="0" width="100" height="100" />
</svg>
```

(Demo 2)

SVG – Scalable Vector Graphics

- Can do favicons in SVG for higher resolution images.
- Currently only works in Firefox.
  - For cross browser stuff, use normal .ICO files.

Audio

```
<audio>
    <source src="chopin.mp3" type="audio/mpeg" />
</audio>
```

- Supports “graceful degradation”.
- WAV files don't work in Firefox yet.
- MP3 isn't an "open" standard, support may vary.
- See also: Web Audio API for high-level processing
  - (No IE support yet, but Edge works.)

(Demo 3)

Video

- Works same as <audio>

```
<video width="400" controls>
    <source src="UnavailableFile.ogg" type="video/ogg">
    <source src="Buckethead.mp4" type="video/mp4">
    Your browser does not support HTML5 video.
</video>
```

- Same as MP3/audio, MP4 isn't open standard
  - Support may vary

(Demo 4)

Geolocation

- Determining where a user is located.

```
<script>
navigator.geolocation.getCurrentPosition(success, error);

function success() {  /* found the user's location */ }

function error() { /* can't find location */ }
</script>
```

(Demo 5)

Data Sets

- Can specify extra “meta data” to describe an element in your HTML.
- Old way:

```
<div id=”CustID_123_City_London_Age_69”>David Bowie</div>
```

- What if we don't know the city?

```
<div id=”CustID_123_City__Age_69”>David Bowie</div>
```

- Data is inconsistent, have to write various hacks to deal with all the variations of missing data.

Data Sets

- New way...with data sets!

  ```
  <div id=”myUser” data-custid=”123” data-city=”London” data-age=”69”>
          David Bowie
  </div>
  ```

- Can name a property anything you want, just prefix it with “data-”. Treated as valid HTML markup.
- To access it via JavaScript:

  ```
  var el = document.querySelector('#myUser');
  el.dataset.age = 64;
  el.dataset.city = “New York City”;
  ```

(Demo 6)

Drag and Drop

- Make an element draggable:
  - <img id=”myHeadshot” draggable="true">
- Then, specify what should happen when the element is dragged:
  - ondragstart="drag(event)"

Drag and Drop

```
function drag(ev) {
    ev.dataTransfer.setData("userID", ev.target.id);
}
```

The dataTransfer.setData() method sets the data name and the value of the dragged data.

Type is "userID" and the value is the id of the draggable element ("myHeadshot").

Drag and Drop

- *Where* we're dragging this item *to*:
  - <div id="premireUsers" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
- ondragover event specifies where the dragged data can be dropped.
- By default, data/elements cannot be dropped INTO other elements. To allow a drop, we must prevent the default handling of the element.
  - event.preventDefault()

(Demo 7)

Editable Content

- Inline editable content on a web page.
- No need to swap <div> and <input> or have “read only” and “edit” templates.

```
<section id="editable" contenteditable="true">
```

- Listen for whatever JavaScript events make sense for your app, and save the data to local storage, ajax, whatever.

(Demo 8)

Custom Elements

- Create your own HTML tags.

  var MyTreehouseElement = document.registerElement('my-treehouse');

- Means I can do this in HTML:

  <my-treehouse> … </my-treehouse>

- x-treehouse is treated as a first class citizen (same as <aside>, <div>, whatever).
- Name of your custom element must contain a dash (-) so the browsers can determine between standard and custom HTML elements.
- (See also: <template> and Polymer).

(Demo 9)

Extending Existing Elements

```
var ThumbImage = document.registerElement('thumb-img', {
 prototype: ThumbImageProto,
   extends: 'img' });
```

- To use your custom element:

```
<img is="thumb-img">
```

- There are a number of callbacks that you can listen for when creating and managing your custom elements.

Extending Existing Elements

- Use callbacks to fire JavaScript via this event:
- createdCallback – Called when a custom element is created.
- attachedCallback – Called when a custom element is inserted into the DOM.
- detachedCallback – Called when a custom element is removed from the DOM.
- attributeChangedCallback(attrName, oldValue, newValue) – Called when an attribute on a custom element changes.

(No demo...browser support is pretty bad still.)

But wait there's more!

- Offline webapps via the “cache.manifest” file.
- Cryptography.
- IndexedDB – client side databases.
- Ambient light, websockets, animations, touch events, HTML templates, spellcheck, clipboard API and on and on and ON! Phew!
- Lots of smart people here at the conference – ask what they're doing!

Other resources

- W3schools.com
  - Examples of pretty much anything
- CanIUse.com
- Html5demos.com
- Html5shiv on GitHub
  - For VERY basic old-IE support
- Modernizr.com
  - For testing HTML5/CSS3 support
- DiveIntoHTML5.info
- Polymer project
  - Custom elements w/ better browser support.

Questions?

- Contact info:
  - Southofshasta.com
  - nolan@southofshasta.com
  - Twitter: @southofshasta

Thanks!

HTML5 – 201

Nolan Erck

About Me

- Independent Consultant – Southofshasta.com
- Co-Manager - SacInteractive User Group
- Stuff I've used: HTML, ColdFusion, .NET, PHP, C++, Java, JavaScript, jQuery, SQL, etc.
- Stuff I've done: CycleGear, Cast Images, Grim Fandango, SimPark, SimSafari, Star Wars Episode 1, .
- Music junkie.

I'M REAL PROUD OF MY RECORD COLLECTION

HTML5

- The spec has been finalized...finally!
- Browser support is getting pretty good
  - (except for those IE people)
- BUT...
  - Just because the spec is final doesn't mean browsers support everything yet.
    - Example: SVG fonts barely work anywhere
    - CanIUse.com

HTML5

- Several technologies, being marketed together.
- <HTML5>
- CSS3
- JavaScript
- SVG, etc

Basic HTML5 Things

- First stuff we used
  - Easier DOCTYPE tags
  - Rules about <tags /> aren't as strict
  - <section>, <aside>, etc.
  - Yes, <canvas> too.
  - New HTML form elements
    - (browser support still in progress).
  - New CSS
    - Media queries, shadows, transparency/opacity, importing fonts, etc.

Stuff we'll cover today

- Autofocus and Placeholder
- SVG
- Audio and Video
- Geolocation
- Drag and Drop
- Data Sets
- Custom elements
- Other resources, questions, etc.

Autofocus

No more JavaScript needed to set focus on a form field!

```
<!DOCTYPE html>
<body>
<form action="submit.cfm">
  First name: <input autofocus type="text" placeholder=”Your First Name”
                      name="fname" /><br />
  Last name: <input type="text" name="lname"
                     placeholder=”Your Last Name” /><br />
  Age: <input type="text" />
  <input type="submit" />
</form>
</body>
</html>
```

(Demo 1)

SVG – Scalable Vector Graphics

- Draw vector graphics in HTML “on the fly”
  - No JPGs or PNGs required.

```
<svg>
        <circle cx="50" cy="50" r="30" />
        <rect x="0" y="0" width="100" height="100" />
</svg>
```

(Demo 2)

SVG – Scalable Vector Graphics

- Can do favicons in SVG for higher resolution images.
- Currently only works in Firefox.
    - For cross browser stuff, use normal .ICO files.

Audio

```
<audio>
    <source src="chopin.mp3" type="audio/mpeg" />
</audio>
```

- Supports “graceful degradation”.
- WAV files don't work in Firefox yet.
- MP3 isn't an "open" standard, support may vary.
- See also: Web Audio API for high-level processing
  - (No IE support yet, but Edge works.)

(Demo 3)

Video

- Works same as <audio>

```
<video width="400" controls>
    <source src="UnavailableFile.ogg" type="video/ogg">
    <source src="Buckethead.mp4" type="video/mp4">
    Your browser does not support HTML5 video.
</video>
```

- Same as MP3/audio, MP4 isn't open standard
  - Support may vary

(Demo 4)

Geolocation

- Determining where a user is located.

```
<script>
navigator.geolocation.getCurrentPosition(success,
error);

function success() {  /* found the user's location */ }

function error() { /* can't find location */ }
</script>
```

(Demo 5)

Data Sets

- Can specify extra “meta data” to describe an element in your HTML.
- Old way:

```
<div id=”CustID_123_City_London_Age_69”>David Bowie</div>
```

- What if we don't know the city?

```
<div id=”CustID_123_City__Age_69”>David Bowie</div>
```

- Data is inconsistent, have to write various hacks to deal with all the variations of missing data.

Data Sets

- New way...with data sets!

  <div id=”myUser” data-custid=”123” data-city=”London” data-age=”69”>

  David Bowie

  </div>

- Can name a property anything you want, just prefix it with “data-”. Treated as valid HTML markup.
- To access it via JavaScript:

  var el = document.querySelector('#myUser');

  el.dataset.age = 64;

  el.dataset.city = “New York City”;

(Demo 6)

Drag and Drop

- Make an element draggable:
  - <img id=”myHeadshot” draggable="true">
- Then, specify what should happen when the element is dragged:
  - ondragstart="drag(event)"

Drag and Drop

```
function drag(ev) {
    ev.dataTransfer.setData("userID", ev.target.id);
}
```

The dataTransfer.setData() method sets the data name and the value of the dragged data.

Type is "userID" and the value is the id of the draggable element ("myHeadshot").

Drag and Drop

- *Where* we're dragging this item *to*:
  - `<div id="premireUsers" ondrop="drop(event)" ondragover="allowDrop(event)"></div>`
- ondragover event specifies where the dragged data can be dropped.
- By default, data/elements cannot be dropped INTO other elements. To allow a drop, we must prevent the default handling of the element.
  - event.preventDefault()

(Demo 7)

Editable Content

- Inline editable content on a web page.
- No need to swap <div> and <input> or have “read only” and “edit” templates.

  <section id="editable" contenteditable="true">
- Listen for whatever JavaScript events make sense for your app, and save the data to local storage, ajax, whatever.

(Demo 8)

Custom Elements

- Create your own HTML tags.

  var MyTreehouseElement = document.registerElement('my-treehouse');

- Means I can do this in HTML:

  <my-treehouse> … </my-treehouse>

- x-treehouse is treated as a first class citizen (same as <aside>, <div>, whatever).
- Name of your custom element must contain a dash (-) so the browsers can determine between standard and custom HTML elements.
- (See also: <template> and Polymer).

(Demo 9)

Extending Existing Elements

```
var ThumbImage = document.registerElement('thumb-img', {
 prototype: ThumbImageProto,
    extends: 'img' });
```

- To use your custom element:

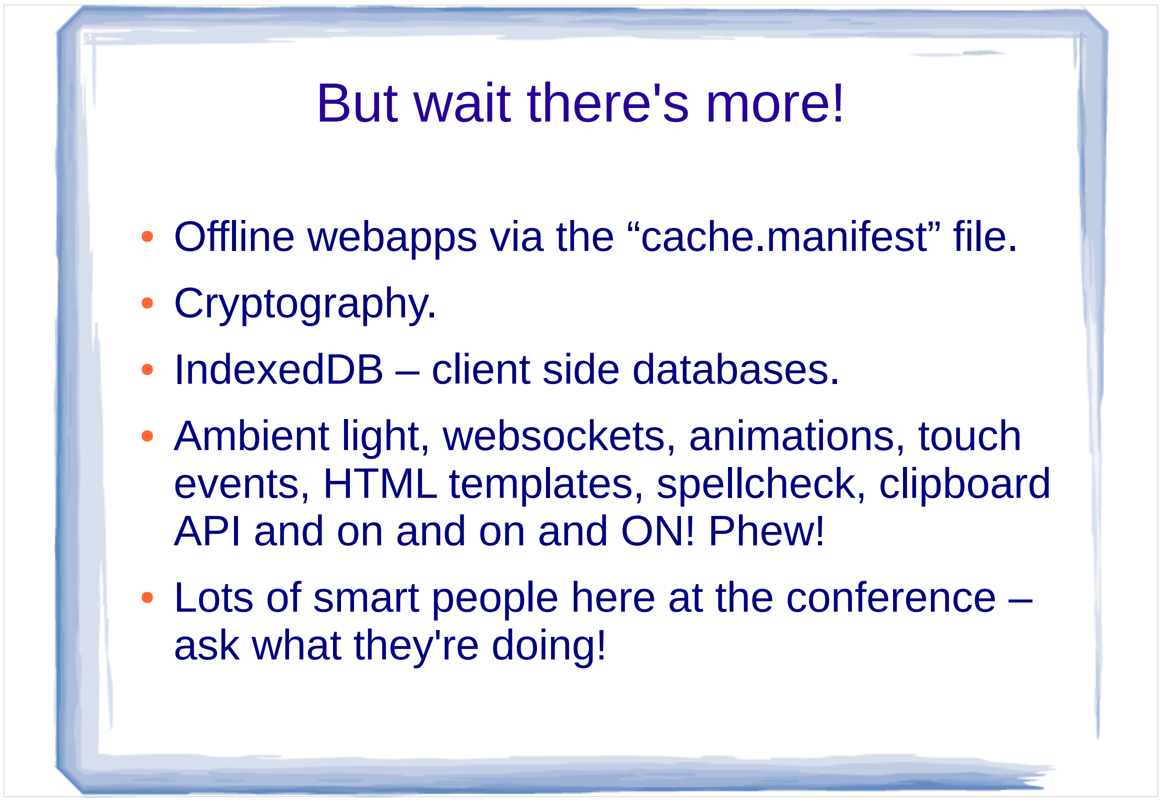
```
<img is="thumb-img">
```

- There are a number of callbacks that you can listen for when creating and managing your custom elements.

Extending Existing Elements

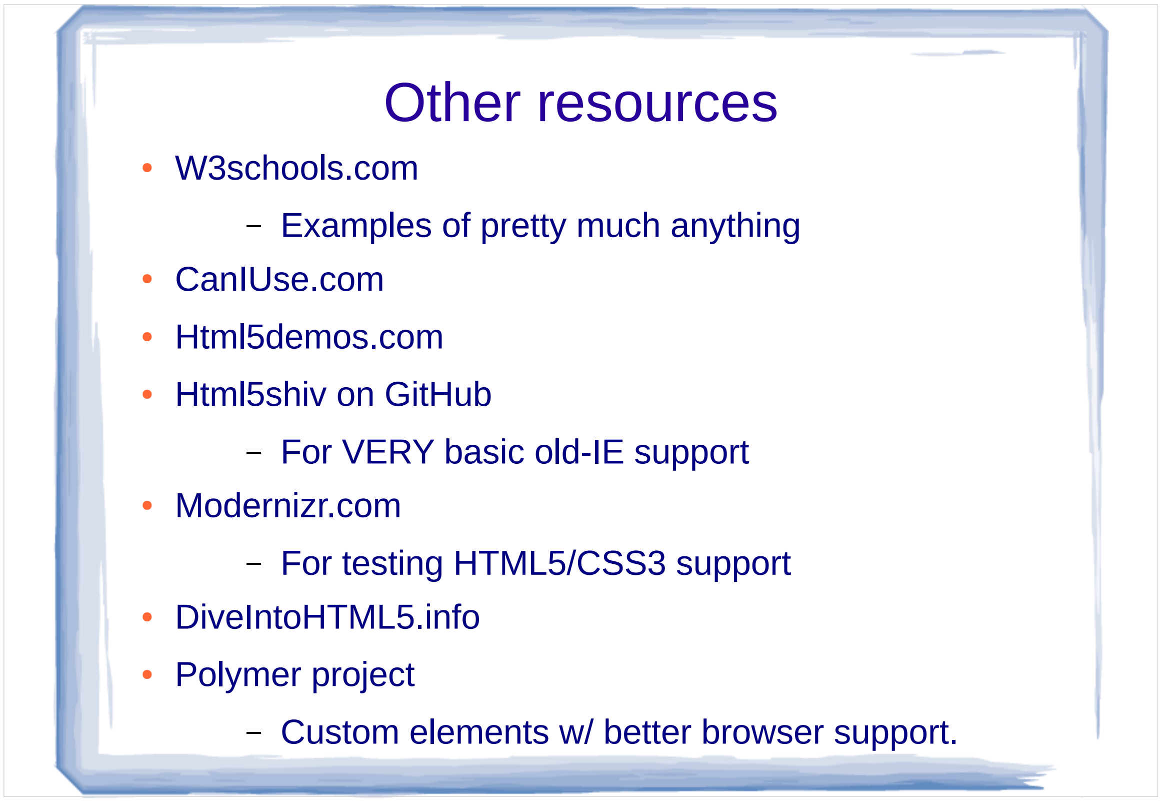
- Use callbacks to fire JavaScript via this event:
- createdCallback – Called when a custom element is created.
- attachedCallback – Called when a custom element is inserted into the DOM.
- detachedCallback – Called when a custom element is removed from the DOM.
- attributeChangedCallback(attrName, oldValue, newValue) – Called when an attribute on a custom element changes.

(No demo...browser support is pretty bad still.)



But wait there's more!

- Offline webapps via the “cache.manifest” file.
- Cryptography.
- IndexedDB – client side databases.
- Ambient light, websockets, animations, touch events, HTML templates, spellcheck, clipboard API and on and on and ON! Phew!
- Lots of smart people here at the conference – ask what they're doing!



Other resources

- W3schools.com
  - Examples of pretty much anything
- CanIUse.com
- Html5demos.com
- Html5shiv on GitHub
  - For VERY basic old-IE support
- Modernizr.com
  - For testing HTML5/CSS3 support
- DiveIntoHTML5.info
- Polymer project
  - Custom elements w/ better browser support.

Questions?

- Contact info:
  - Southofshasta.com
  - nolan@southofshasta.com
  - Twitter: @southofshasta

Thanks!