The Web Can Do That!?

Adventures Into HTML5

Eric Bidelman
Google Chrome Team

Watch the video!

Who is this Guy?

Senior Developer Programs Engineer

Google Chrome Team

+Eric Bidelman

@ebidel

ericbidelman.com

HTML CAN NOT

DO THAT!!!1!!

Legend

This slide deck is alive!

 Close to being ready. Keep it on your radar.

 Relevant Chrome/WebKit bug.

 Bug has been fixed/resolved.

 Specification link

#1 CSS For Web Apps

Floats, Tables, & Positioning...OH MY!

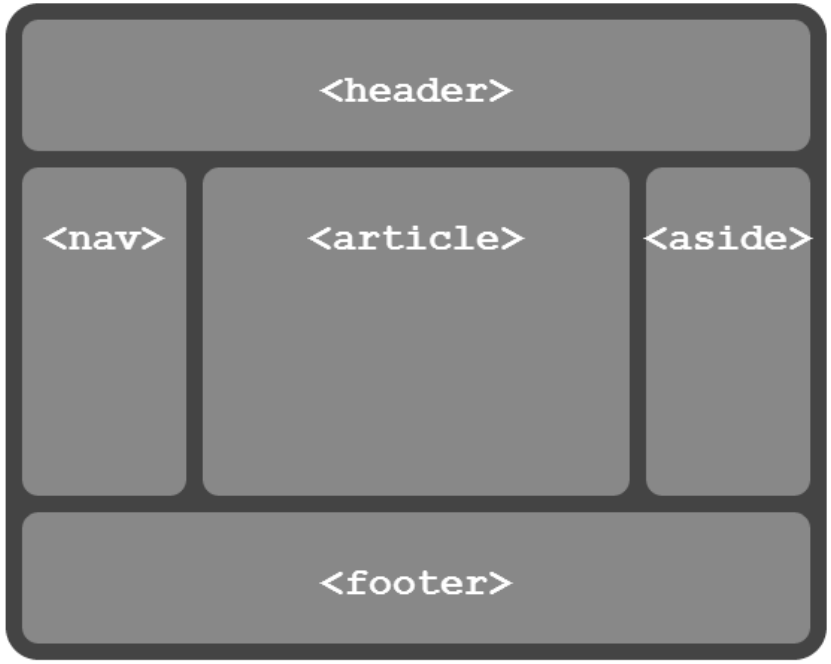
- Floats stack horizontally to allow simple placement.
- Tables lack repeatability and semantic meaning.
- Using absolute positioning is difficult. Makes adaptive display hard.
- CSS3 specifications address layout cases with regions, exclusions, grid and flexible box.

2012: The Rise of CSS for Web Apps

- CSS Regions Module Level 3
- CSS Flexible Box Layout Module (da new one)
- CSS Grid Layout
- CSS Hierarchies
- CSS Filter Effects 1.0
- Web Animations 1.0
- CSS Variables Module Level 1

So why is flexbox so awesome?

Achieve That "Holy Grail" Layout!

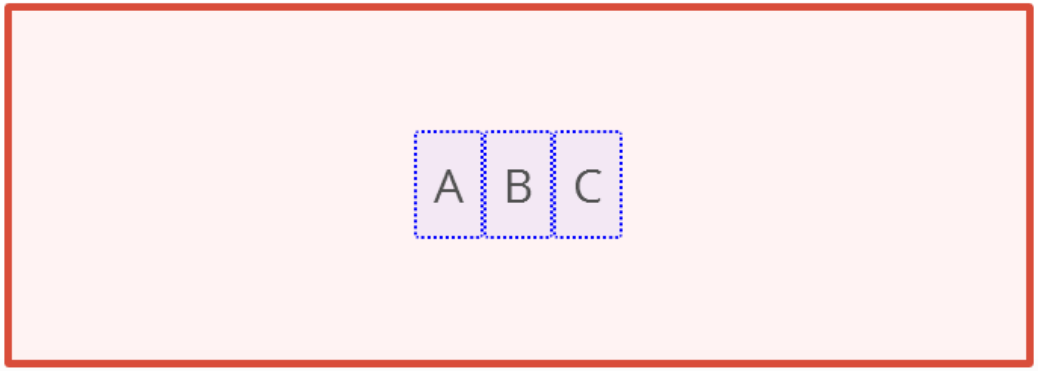


Alignment

Vertically centered content is easy peasy!

CSS

```
.box {
  display: +flex;
  +justify-content: center      ;
  +align-items: center    ;
 }
```



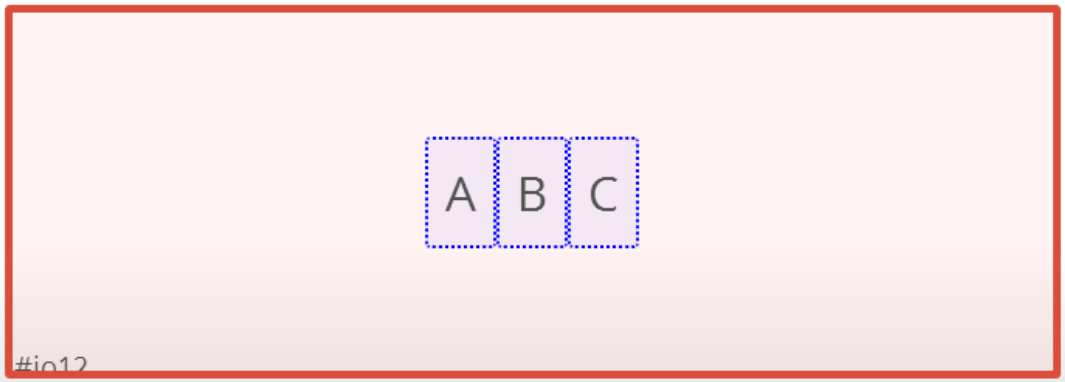
Ordering & Orientation

Order independent from source

- Content is laid out from lowest to highest numbered ordinal group.
- Items with the same order group are laid out per source document.

CSS

```
.box {
  +flex-direction:  row           ;
}
.box > :nth-child(2) {
  +order:  0 ;
}
```



Same Height Columns!

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla varius tortor ut sem volutpat eu tincidunt ligula feugiat. Pellentesque sollicitudin turpis egestas lorem tincidunt ultricies in at metus. Fusce nec nibh justo.

height: 196px

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

height: 74px

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla varius tortor ut sem volutpat eu tincidunt ligula feugiat.

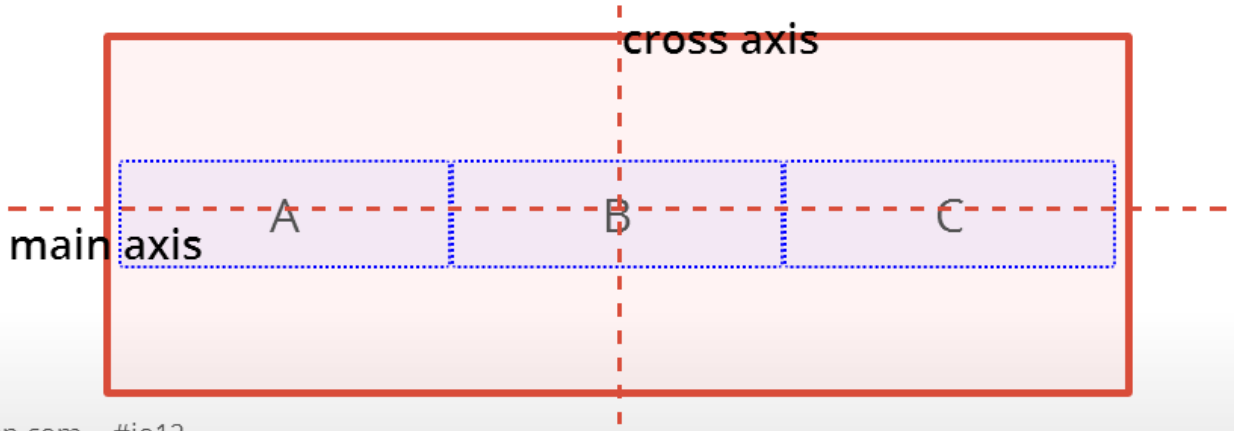
height: 123px

Flexibility

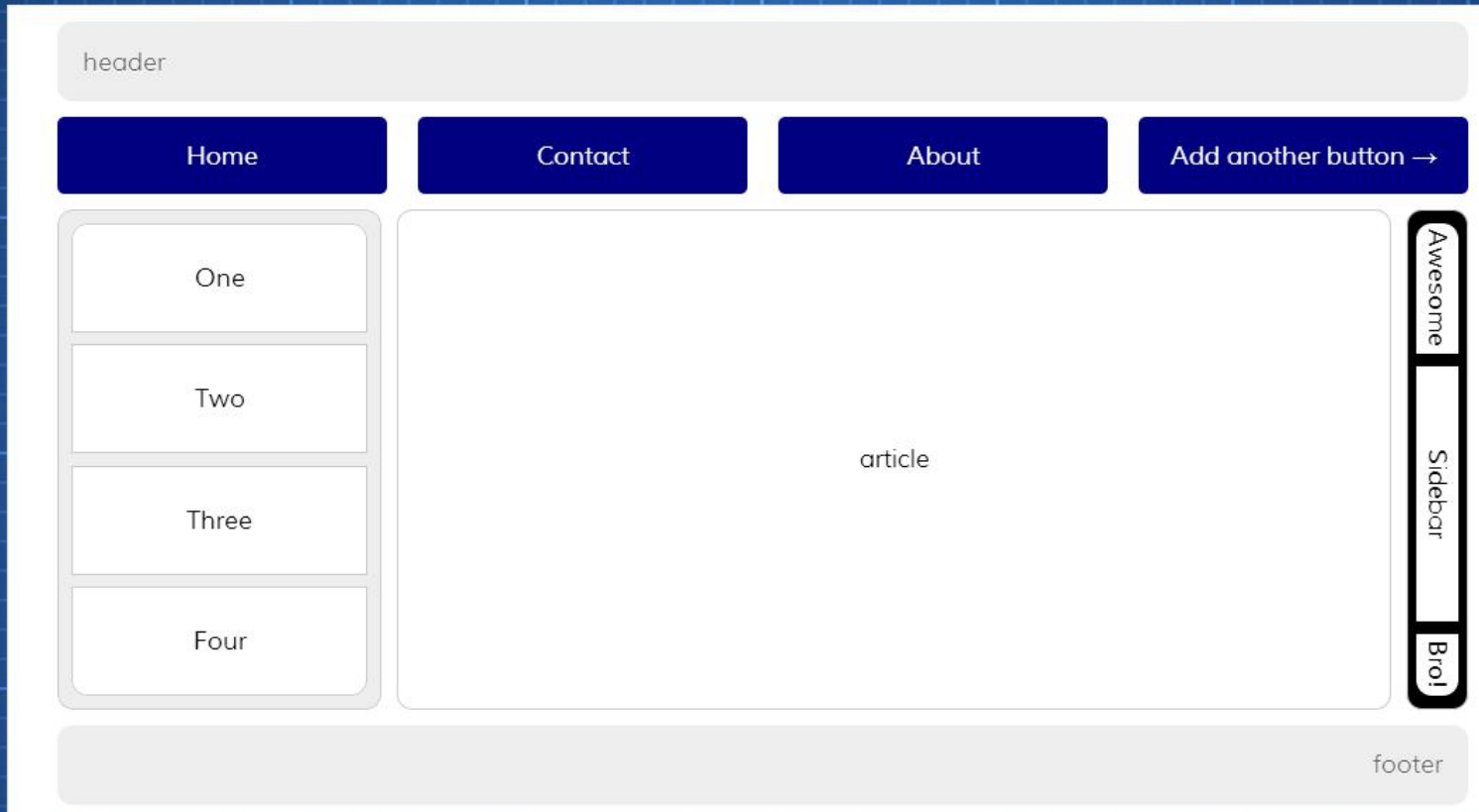
- Flexbox items alter their width/height to fill available space.
- Items grow/shrink proportional to their positive/negative flexibility
- 3rd argument is preferred size

CSS

```
.box > * {
  +flex: 1 0 0px;
}
.box > :nth-child(2) {
  +flex: 1  0  0 px;
}
```



Demo: Holy Grail Layout



Browser Support

CSS Flexbox (da new one)


#2 Dynamic CSS
CONFIRMED

calc()

CSS

```
.circle {
  width: 300px;
  height: 300px;
}
div {
  display: +flex;
  width: +calc(100% - 4em);
  height: +calc(100% - 4em);
  border-radius: 50%;
  +align-items: center;
  +justify-content: center;
}
div:hover {
  border-radius: 0;
}
```

calc()

CSS

```
.circle {
  width: 300px;
  height: 300px;
}
div {
  display: +flex;
  width: +calc(100% - 4em);
  height: +calc(100% - 4em);
  border-radius: 50%;
  +align-items: center;
  +justify-content: center;
}
div:hover {
  border-radius: 0;
}
```

Browser Support

CSS calc()


#3 Data binding
CONFIRMED

Old Hat for JS Frameworks

Angular (angularjs.org) example:

HTML

```
<div ng-app ng-init="val=25">
  Volume: <input type="range" min="0" max="100" ng-model="val">
  {{val}}/100
</div>
```

Volume: 25/100

One-way Data Binding: `data-*` Attributes

"Poor man's data binding"

- Model: data attribute
- Use `attr()` to get the value(s)
- View: render generated content to `:before`/`:after` pseudo elements
- Hook listeners that watch for changes.

"Poor Man's Data Binding"

HTML/JS

```
<style>
input::after {
  content: attr(data-value) '/' attr(max);
  position: relative;
  left: 135px; top: -20px;
}
</style>
<input type="range" min="0" max="100" value="25">
<script>
var input = document.querySelector('input');
input.dataset.value = input.value; // Set an initial value.

input.addEventListener('change', function(e) {
  this.dataset.value = this.value;
});
</script>
```

Volume:  25/100

One-way Data Binding: <datalist>

HTML

```
Browsers: <input list="browsers">
<datalist id="browsers">
  <option value="Chrome">
  <option value="Firefox">
  <option value="Internet Explorer">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

- Specifies a list of pre-defined options for an <input> element.
- `list` attribute "binds" data list to an <input>
- Useful for "autocomplete" on <input> elements.

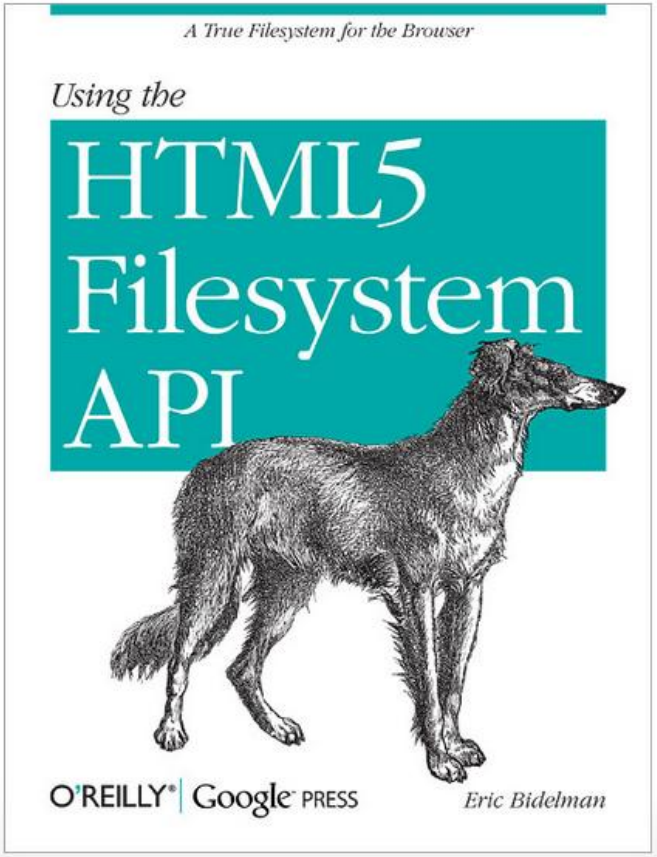
Browsers:

Browser Support

Data binding with `data-*` attrs / `<datalist>`

#4 Access a Filesystem

I Got So Excited...I Wrote a Book :)



HTML5 Filesystem API

Persist data to files/folders

Opening a filesystem:

JS

```
window.requestFileSystem(
  TEMPORARY,          // persistent vs. temporary storage
  1024 * 1024,        // size (bytes) of needed space
  initFs,             // success callback
  opt_errorHandler    // opt. error callback, denial of access
);
```

- Security: sandboxed per origin
- Power: native apps can do files/folders,...web apps should too!
- Benefit: dynamically cache and manage assets.

Example: Caching Remote Files

JS

```
var xhr = new XMLHttpRequest();
xhr.open('GET', '/path/to/image.png', true);
xhr.responseType = 'arraybuffer';
xhr.onload = function(e) {









};

xhr.send();
```

Example: Caching Remote Files

JS

```
var xhr = new XMLHttpRequest();
xhr.open('GET', '/path/to/image.png', true);
xhr.responseType = 'arraybuffer';
xhr.onload = function(e) {
  window.requestFileSystem(TEMPORARY, 1024 * 1024, function(fs) {
    fs.root.getFile('image.png', {create: true}, function(fileEntry) {

      fileEntry.createWriter(function(writer) {
        writer.onwriteend = function(e) { ... };
        writer.onerror = function(e) { ... };
        writer.write(new Blob([xhr.response], {type: 'image/png'}));
      }, onError);

    }, onError);
  }, onError);
};

xhr.send();
```

Using Local Assets

- Stored files/folders get their own (`filesystem:`) URL:

```
filesystem:http://example.com/temporary/image.png
```

- Use for a src/href value:

JS

```
var img = document.createElement('img');
img.src = fileEntry.toURL();
document.body.appendChild(img);
```

- Get back to a `FileEntry` from its `filesystem:` URL:

JS

```
window.resolveLocalFileSystemURL(img.src, function(fileEntry) { ... });
```

Moar Callbackzzzzzz!

JS

```
var xhr = new XMLHttpRequest();
xhr.open('GET', '/path/to/image.png', true);
xhr.responseType = 'arraybuffer';
xhr.onload = function(e) {
  window.requestFileSystem(TEMPORARY, 1024 * 1024, function(fs) {
    fs.root.getFile('image.png', {create: true}, function(fileEntry) {

      fileEntry.createWriter(function(writer) {
        writer.onwriteend = function(e) { ... };
        writer.onerror = function(e) { ... };
        writer.write(new Blob([xhr.response], {type: 'image/png'}));
      }, onError);

    }, onError);
  }, onError);
};

xhr.send();
```

“Callback spaghetti is hard,... let's go shopping!”

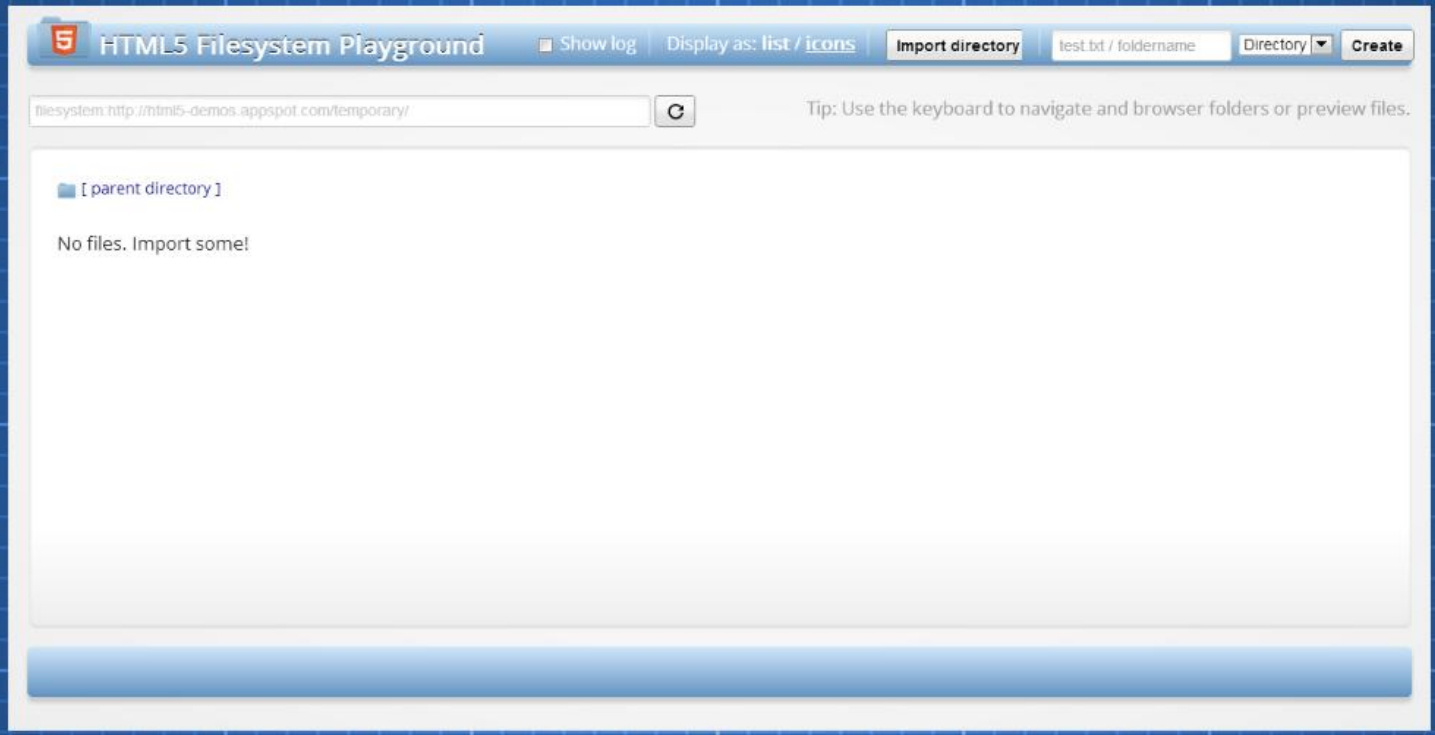
filer.js (github.com/ebidel/filer.js)

Wrapper lib that implements common UNIX cmds (ls, cp, mv)

JS

```
var filer = new Filer();

filer.init({persistent: false, size: 1024 * 1024}, function(fs) {...}, onError);

filer.ls('path/to/some/dir/', function(entries) { ... }, onError);

filer.cp('file.txt', '/path/to/folder', 'newname.txt', function(entry) {
  // entry.fullPath == '/path/to/folder/newname.txt'
}, onError);

var b = new Blob(['body { color: red; }'], {type: 'text/css'});
filer.write('styles.css', {data: b, type: b.type}, function(entry, writer) {
    ...
}, onError);
```

Demo: Filesystem Playground



But kind Sir, this marvel is only in Chrome!

OH YEAH?

Browser Support

HTML5 Filesystem API - thanks to idb.filesystem.js shim!

#5 sudo make me a sandwich

#6 Serverless Downloads

Client-side Downloads

OLD Force downloads with a server:

```http
Content-Disposition: attachment; filename="MyLogo.png";
```

HTTP

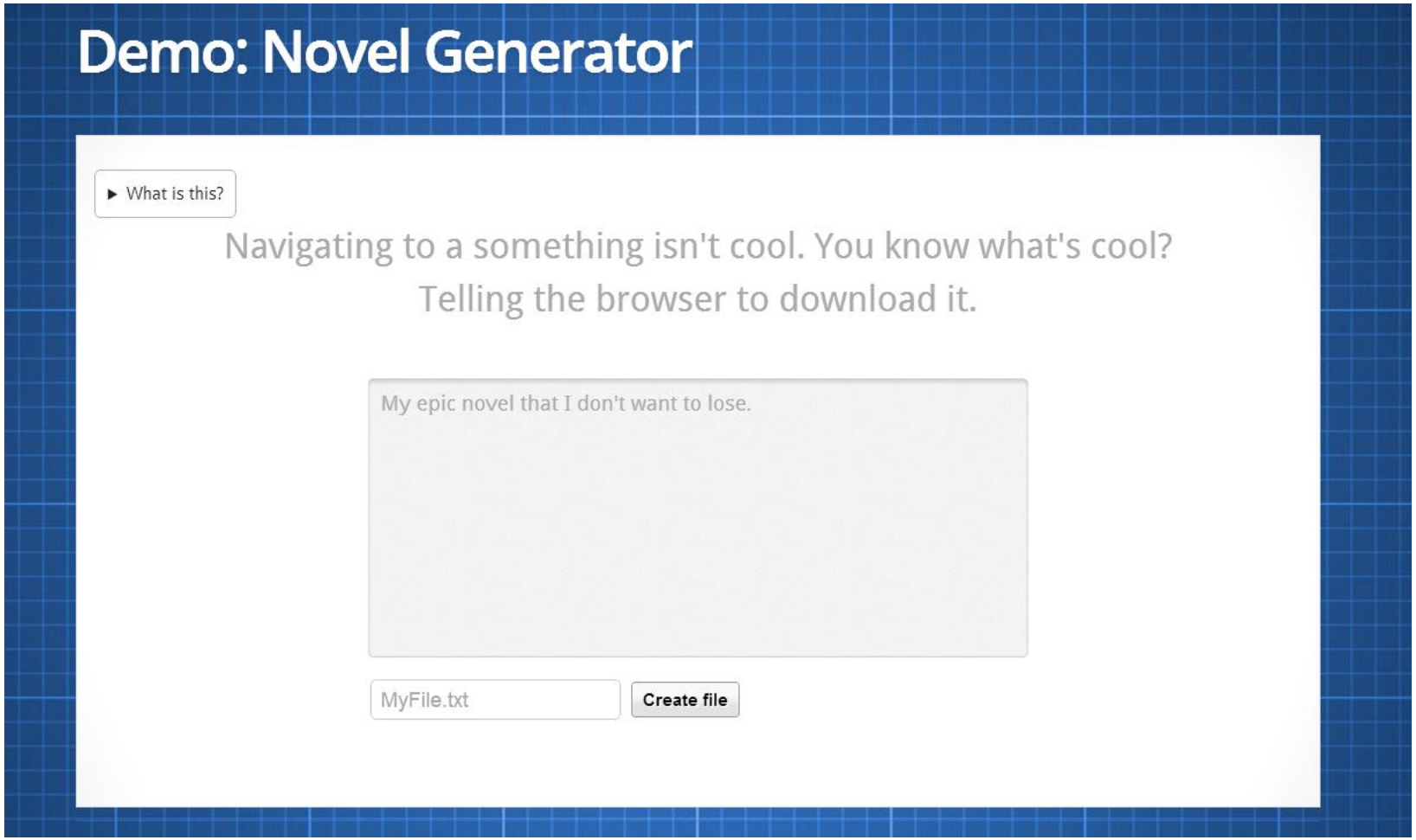
- Need a way for users to download data created in the app.

NEW Use `download` to download resources rather than navigate to them:

```html
<a href="http://google.com/logo.png" download="Logo.png">download me</a>
```

HTML

Demo: Novel Generator



Example: Downloading a Generated File

JS

```
function downloadLink(name, content, mimetype) {
  var a = document.createElement('a');
  a.href = window.URL.createObjectURL(new Blob([content], {type: mimetype}));
  a.download = name;
  a.textContent = 'Download ready';

  document.body.appendChild(a);
}

downloadLink('MyNovel.txt', document.querySelector('textarea').textContent, 'text/plain
```


#7 Efficiently Transfer Data
CONFIRMED
AIR SERVICE

Evolution of postMessage()

1. Basic strings

```JS
worker.postMessage('Hello World');
window.postMessage(JSON.stringify({msg: 'Hello World'}), '*');
```

2. JSON

```JS
worker.postMessage({msg: 'Look ma, no strings!'});
```

Evolution of postMessage()

3. Complex types (`File`, `Blob`, `ArrayBuffer`) via structured cloning

JS

```
var data = [1, 2, 3, 4];

// Blob() in Chrome 20, FF 13
worker.postMessage(new Blob(data.toString(), {type: 'text/plain'}));

var uint8Array = new Uint8Array(data);
worker.postMessage(uint8Array);
worker.postMessage(uint8Array.buffer);
```

- Problem: all of these methods are copies. Not efficient!
- We can do better...

Transferable Objects

a.k.a. workers that "don't suck"

Same old friend, different semantics (note the vendor prefix):

```JS
worker.postMessage(arrayBuffer, [arrayBuffer]);
window.postMessage(arrayBuffer, targetOrigin, [arrayBuffer]);
```

- Zero-copy
  - Ownership of data (`ArrayBuffer`) is transferred to new context (e.g. worker/window). Source buffer becomes unavailable.
  - ~50x faster than structured cloning (Chrome 13/FF).
- 2nd argument is list of what to transfer.

Demo: Transferable Objects

▶ Yo dawg, what is this?

Transferable Objects are lightning fast! The prefixed `[window|worker].webkitPostMessage()` now supports sending an `ArrayBuffer` as a transferable.

Run test

```
11:52:47:341 thread: USING TRANSFERABLE OBJECTS :)
11:52:47:342 thread: READY!
11:52:47:626 worker: READY! [11:52:47:622]
```

Feature Detection

- Feature detecting is a PITA b/c of false positives.
- If transferred, buffer's `.byteLength` goes to zero.

JS

```
var ab = new ArrayBuffer(1);
worker.postMessage(ab, [ab]);
if (ab.byteLength) {
  alert('Transferables are not supported in your browser!');
} else {
  // Transferables are supported.
}
```

Browser Support

Transferable Objects


#8 Access Native Hardware
CONFIRMED

"Device APIs"

Device APIs WG: www.w3.org/2009/dap/

- Geolocation API
- Device Orientation API (accelerometer)
- WebGL (GPU)
- HTML5 Filesystem API (sandboxed filesystem)
- `navigator.onLine` / `navigator.connection` (network connectivity)
- Battery API
- Gamepad API
- WebRTC (voice & video input) / Web Audio API (core audio)

First Step to Speech Enabled Apps!

```HTML
<input type="text" x-webkit-speech>
```

Camera & Microphone Access

Plugin-free acess to camera/microphone.

JS

```
navigator.getUserMedia({audio: true, video: true}, function(stream) {
  var video = document.querySelector('video');
  video.src = window.URL.createObjectURL(stream);
}, function(e) {
  console.log(e);
});
```

HTML

```
<video autoplay controls></video>
```

Demos

- Photo booth
- Live Effects
- Augmented Reality
- WebGL Live TV
- Instant Camera
- Explode Video

Recording Content

HTML

```
<input type="button" value="" onclick="record(this)">
<input type="button" value="" onclick="stop(this)">
```

JS

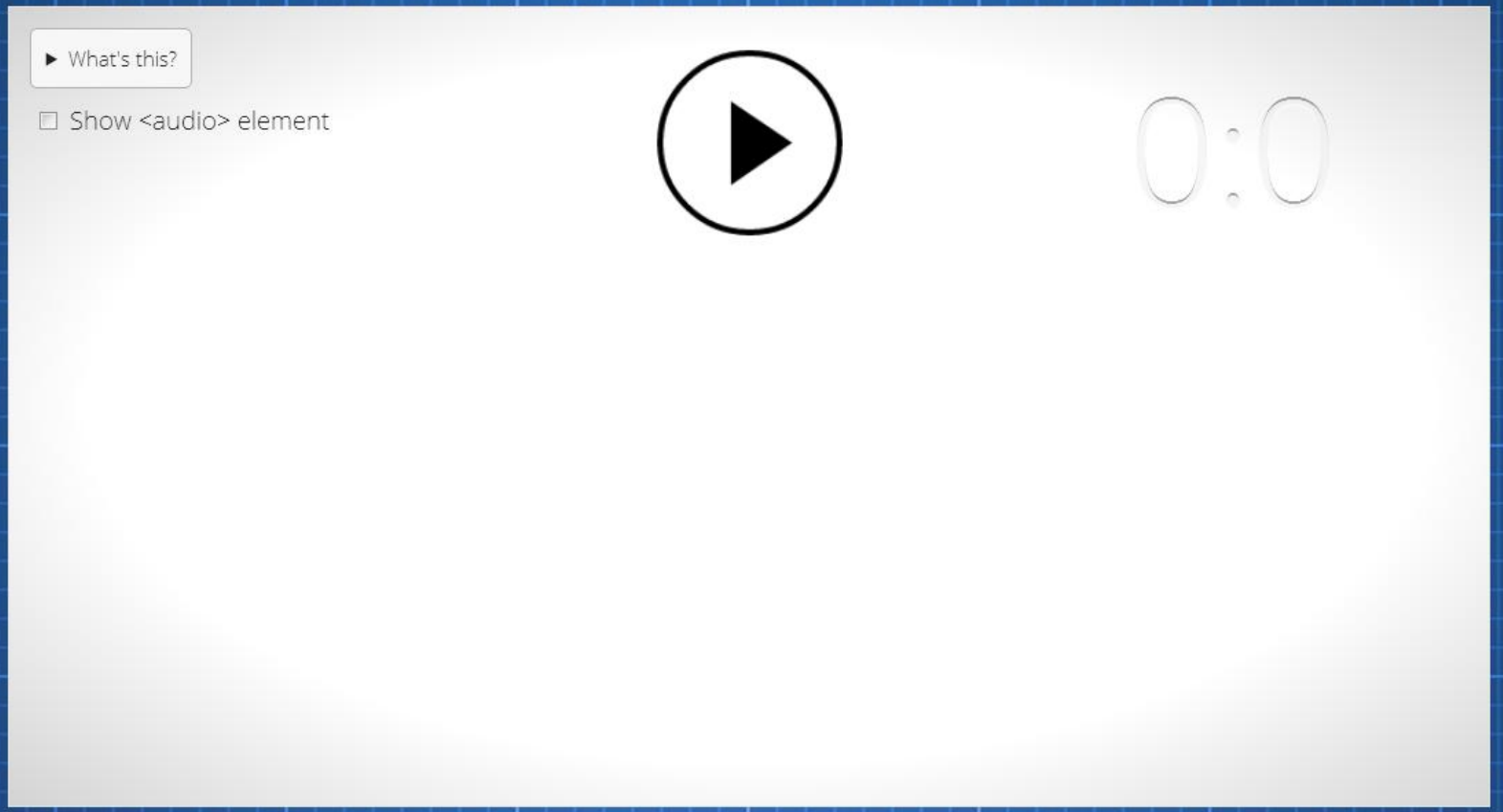
```
var localMediaStream, recorder;

var record = function(button) {
  recorder = localMediaStream.record();
};

var stop = function(button) {
  localMediaStream.stop();
  recorder.getRecordedData(function(blob) {
    // Upload blob using XHR2.
  });
};
```

Browser Support

`getUserMedia()`

#9 Make <audio> Sexy Again

Demo: HTML5 <audio> Visualizer



Web Audio API + <audio>/<video>

Use an `HTMLMediaElement` as the source to the API:

JS

```
var ctx = new window.AudioContext();
var audioElement = new Audio();
audioElement.src = 'sounds/dope_beats.mp3';
audioElement.controls = true;
audioElement.autoplay = true;

var source = ctx.createMediaElementSource(audioElement);
var analyser = ctx.createAnalyser();

source.connect(analyser);
analyser.connect(ctx.destination);
```

Note: There's no `source.noteOn(0)`. Play/pause is controlled by the `<audio>` element.


#10
CONFIRMED

Binary WebSockets

New look, same great taste

- WebSockets suffered the same limitations as Workers.
- Now send `File`, `Blob`, or `ArrayBuffer` types.

JS

```
var socket = new WebSocket('ws://example.com/sock', ['dumby-protocol']);
socket.binaryType = 'blob'; // or 'arraybuffer'
socket.onopen = function(e) {
  window.setInterval(function() {
    if (socket.bufferedAmount == 0) {
      socket.send(new Blob([blob1, blob2]));
    }
  }, 50); // rate limit us.
};
socket.onmessage = function(e) {
  document.querySelector('img').src = URL.createObjectURL(e.data);
};
```

Demos: Media Streaming

AUDIO STREAMER

INSTALL SCREENSHARE

Finito.

The Web Can Do Amazing Things...

- CSS For Web Apps
- Dynamic CSS
- Data binding
- Access a Filesystem
- sudo make me a sandwich
- Serverless Downloads
- Efficiently Transfer Data
- Access Native Hardware
- Make <audio> Sexy Again
- Stream Multimedia

< Thank You! >

Source: github.com/ebidel/html5can

g+ plus.ericbidelman.com
twitter @ebidel
www www.ericbidelman.com
github github.com/ebidel