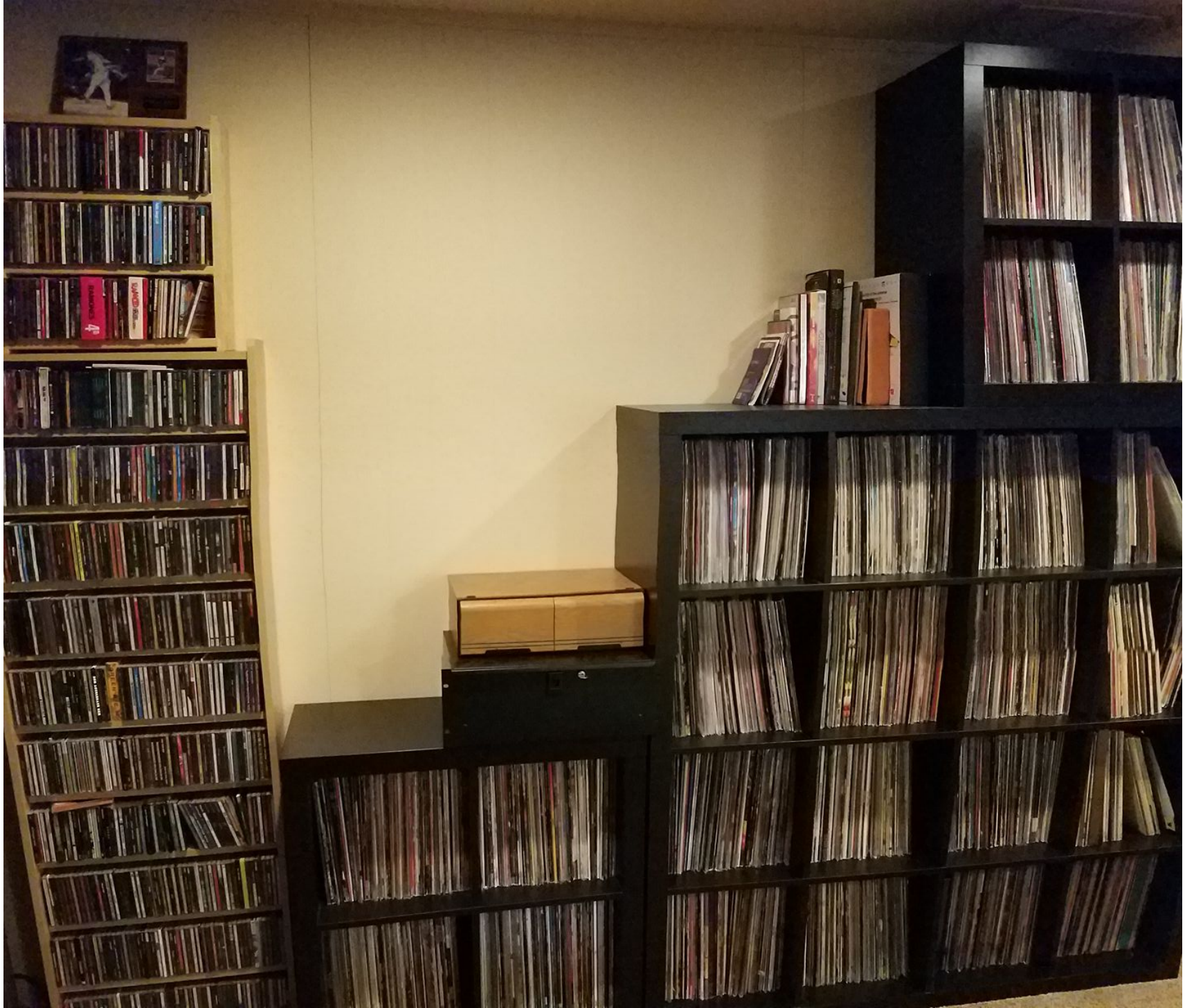
Web Components and CFML

Nolan Erck

South of Shasta

About Me

- Owner / Director, South of Shasta
- Manager, Sac Interactive Tech Meetup
- Reformed Video Game Developer - Grim Fandango, SimPark, StarWars Rogue Squadron, etc...
- CTO at FSK Audio
- Music Junkie




I'M REAL PROUD OF MY RECORD COLLECTION

Code And Slides

github.com/nolanerck

web-components-and-cfml

Prerequisites

- Comfortable with modern JavaScript
  - (including basic OOP concepts)
- Familiar with the DOM
- Experience using CF Custom Tags
- APIs, JSON, Ajax, etc.

Today's Agenda

- What are Web Components?
- Some basic examples and concepts
- Web Components with CFML
- Tips and Other Goodies
- More Resources

What is a Web Component?

- Write your own HTML tags!
- JavaScript based building block
- Kind of like CFML Custom Tags
- Processed in the browser, not server side
- No libraries required...they're fast!

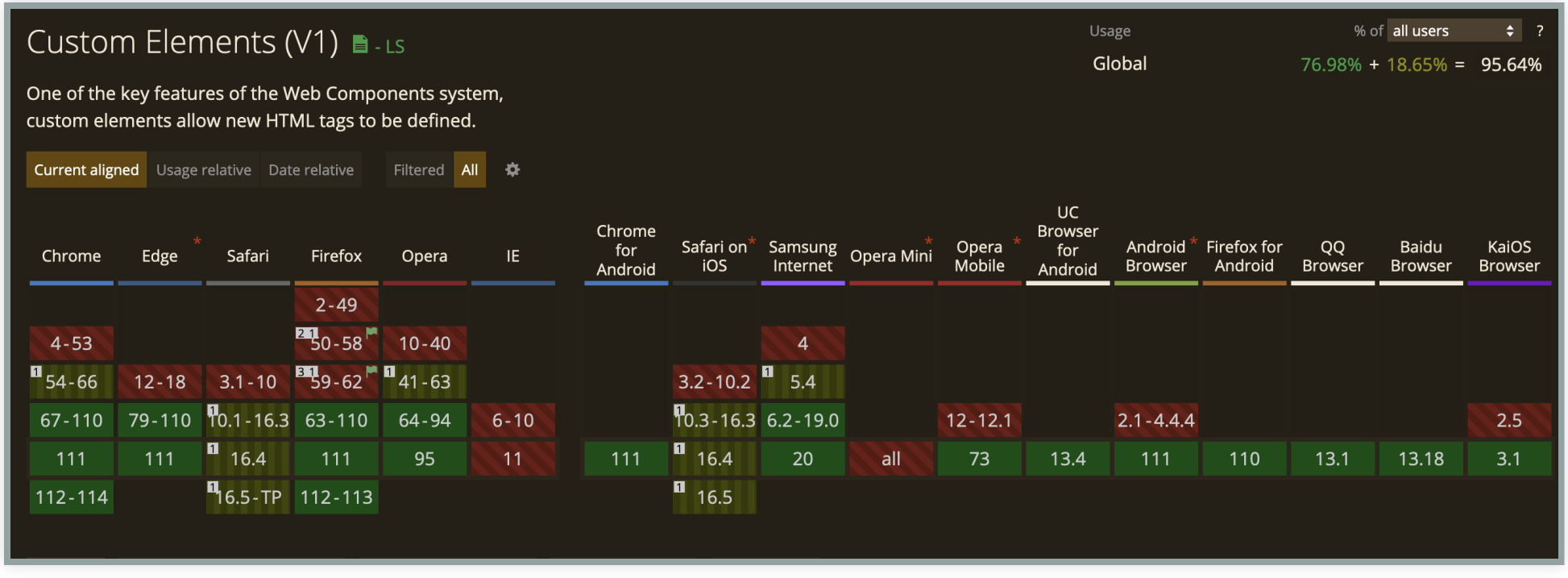
Web Components

- Not framework specific
- Vanilla JavaScript
- Can also use them w/ *all* the SPA frameworks
- ...or with *none* of the SPA frameworks!
- And they work in CFML too!

Web Components

- Require knowing a little OOP
- Basic knowledge of how the DOM works
- Doesn't require Node, TypeScript, 3rd party libaries, etc.
- Can connect to *any* server (CFML, Java, PHP, Perl, whatever sends JSON)

Browser Support



Your First Web Component

- [1 - Hello World Demo]

Your First Web Component

- Naming convention: hello-world
- The hyphen is *required*
- Prevents naming collision issues with future HTML tags
- Pick a project or module prefix for your code

A note about "extends HTMLElement"

- HTMLElement is the default for inheritance
- The spec says you can inherit from any HTML Tag (Table, Div, etc)
- Works fine in other browsers, not Safari
- Safari only supports "extends HTMLElement"
- Various Shims available if needed (unlikely)

The JavaScript Stuff...

- Class must extend HTMLElement
- connectedCallback() - code that runs when your Element is added to the DOM
- customElements.define(...) - registers the Element with the CustomElementRegistry
- ...Essentially makes it visible / usable to the outside world

Lifecycle Methods

- constructor()
- observedAttributes()
- attributeChangedCallback()
- connectedCallback()
- disconnectedCallback()
- adoptedCallback()
- (ala onClick, onSubmit, etc...)

constructor()

- Not *technically* required
- But OOP best practice
- Same as init() in a CFC

connectedCallback()

- Required for your Component to start doing things
- Runs when component is attached to the DOM

observedAttributes() and attributeChangedCallback()

- observedAttributes()
  - List of the attributes we care about
- attributeChangedCallback()
  - So we know which ones should / shouldn't trigger code to run in the component
- Optional (unless you have attributes)

disconnectedCallback()

- Runs when component is disconnected from the DOM
- Optional

adoptedCallback()

- Called when a Web Component is moved from one document to another
- I have no idea when this would be useful
- IFrames?
- Coffee if you can tell me a legit use case for it

Attributes

- <hello-world firstname="Nolan"></hello-world>
- Element must have a constructor that calls super()
- observedAttributes() - list of attributes to watch
- attributeChangedCallback() - what to do with an observed attribute

Attributes

- [2 - Attributes Demo]

Multiple Instances and HTML

- [3 - Multiple Instances Demo]

What Went Wrong?

- Can't add HTML tags via this.textContent, just plain text
- Need better functionality for this to be really usable
- this.innerHTML
- Also, want to put our CSS inside the Component so everything is organized, and safe from outside interference via other code.
- Enter ShadowDOM

Shadow DOM

- Solves our encapsulation problem
- Attaches a separated DOM to the Web Component
- this.attachShadow({ mode: 'closed' });
- Mode can be "open" or "closed"
- Open: Outside JS can access our Element's Shadow DOM
- Closed: Shadow DOM can only be accessed via our Element

Shadow DOM

- [4 Shadow DOM Demo]

Slots

- Content inside your Web Component
- <my-tag>where does this render?</my-tag>
- [5 Slots Demo]

A Real World Web Component

- <confirm-button>
- For "did you mean to click that?" situations
- [6 Confirm Button Demo]

Another Real World Example

- <slide-show>
- Stop downloading tons of JS plugins just for this!
- [7 Slideshow Demo]

Now for the CFML!

Web Components + CFML

- Can mix and match
- Ala jQuery or Custom Tags
- Neither required rewriting your entire CFML codebase
- Use them as needed
- Self contained modules

Web Components + CFML

- [8 CFML Vars Demo]

CFML + APIs + JSON

- JavaScript doesn't care what's on the other side of a URL
- CFML, PHP, COBOL, Perl, whatever
- If the server spits out JSON, your JavaScript can read it
- Can even mix and match (run CFML *and* COBOL!)
- All the browser cares about is the JSON output

Getting JSON back from CFML

- Several ways to do it
- ColdFusion built-in API features
- ColdBox REST API template
- ColdBox, return JSON instead of a /view file
- Framework-1 will return JSON
- Taffy.io
- Write a little code for older legacy apps

Getting JSON back from CFML

- [9 CFML + JSON Demo]

Getting JSON back from CFML

- [10 CFML + JSON Demo via cfhttp]

Migration Uses

- Web Components instead of full blown SPA frameworks
- Porting legacy apps to SPAs
- Move your business logic to Components
- They're compatible with all the frameworks
- And *also* work in .CFM files

Recap

- Very little syntax to Web Components
- A handful of lifecycle methods
- A little machinery / boilerplate
- Everything else is up to how fancy you want to get
- Can integrate any other libs
  (Depends how well that lib is written, YMMV)

Tips

- Attributes are *lowercase* or they break!
- Attributes for input, CustomEvent() for output
- Pick a prefix for your naming convention
  (the hyphen is req'd, use to your advantage)
- Form elements are trickier. Use FormData Interface or ElementInternals or a Shim (YMMV)
- Don't use for SEO-sensitive things

More Tips

- Keep the .js files organized
- Maybe 1 file / each component, or 1 for "related items"
- Yes, might mean a lot of files
- Can use a bundler like Webpack to squish them into 1 .js file better performance, faster loading
- Always load .js files at the bottom of your html page, not in <head>

Other Resources

- Raymond Camden's Blog
- Web Components in Space on YouTube (Ben Farrell)
- UnicornUtterances.com (Corbin Crutchley)
- custom-elements-everywhere.com
- webcomponents.org

Libraries To Explore

- Lit Framework (lit.dev)
- Stencil (Stenciljs.com)
- Shoelace (shoelace.style)
- Lit and Stencil need to be compiled
  Shoelace does not
- Tons of goodies on GitHub

Learning in 30 minutes a day.

Questions? Comments?

- southofshasta.com
- nolan@southofshasta.com
- Twitter: @southofshasta @nolanerck
- Github: nolanerck

Thanks!

Nested Web Component example Bootstrap styles Accessbility concerns? CustomEvent() example, both sending and receiving -- maybe use this and make a "notification bar" for the FuSE app? PHP Docker image for the CFML examples but in a different back-end environment Example of launching a modal (via Bootstrap?) "Spinner Button" example