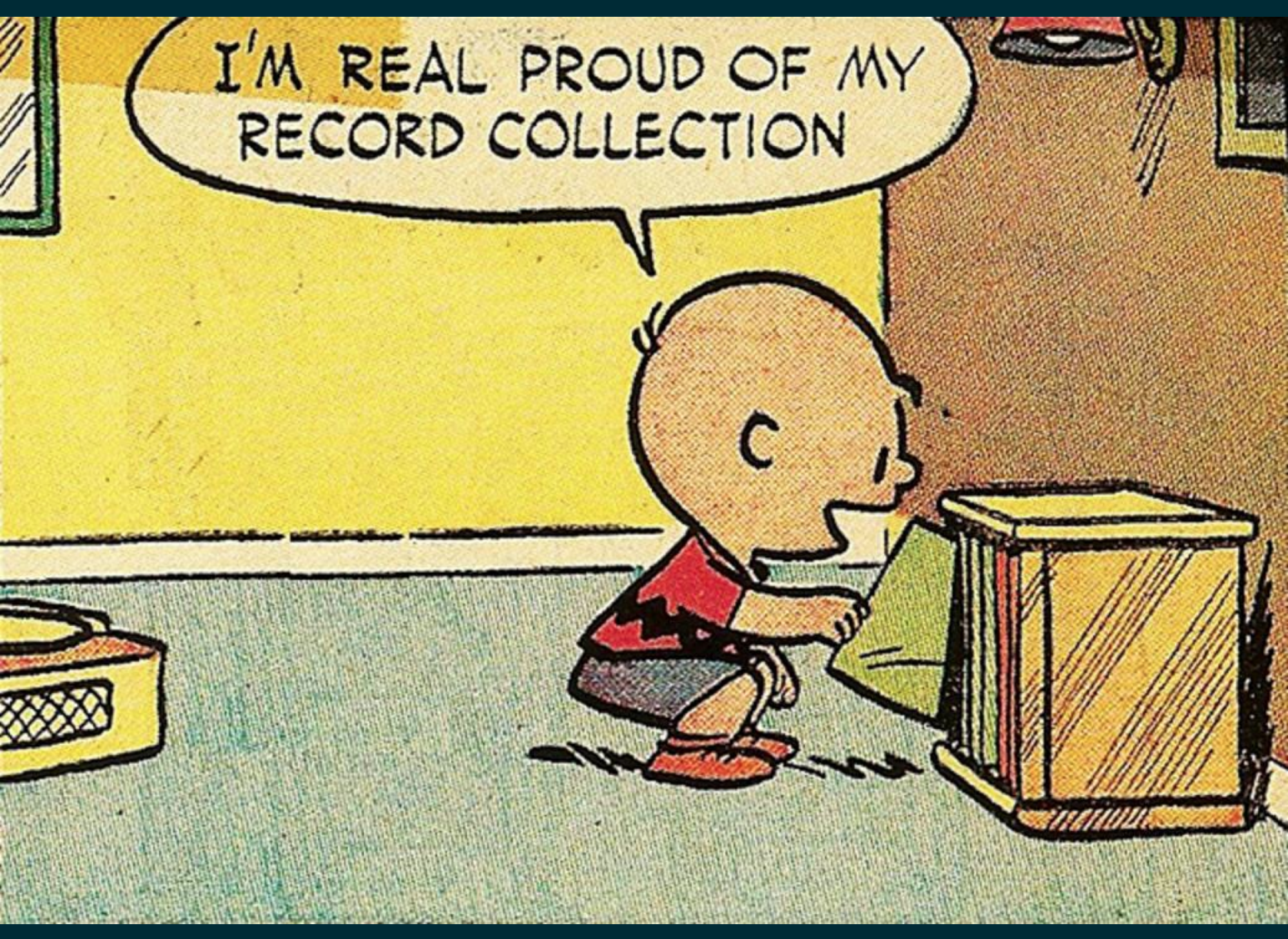
Web Components and CFML

Nolan Erck

South of Shasta

About Me

- Owner / Director, South of Shasta
- VP of Engineering, e-MissionControl
- Manager, Sac Interactive Tech Meetup
- Reformed Video Game Developer - Grim Fandango, SimPark, StarWars Rogue Squadron, etc...
- 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)
- 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
- Other resources

What is a Web Component?

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

Web Components

- Not framework specific
- Built on vanilla JavaScript
- Can also use them w/ *all* the SPA frameworks
- And they work in CFML too!

Web Components

- Require knowing a little OOP
- Based on JavaScript classes
- Doesn't require Node, TypeScript, 3rd party libaries, etc.
- Vanilla JavaScript
- Can connect to *any* server (CFML, Java, PHP, Perl, whatever sends JSON)

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

Your First Web Component (JavaScript)

- 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()

Lifecycle Methods

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

Lifecycle Methods

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

Lifecycle Methods

- 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)

Lifecycle Methods

- 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 basically
- Need better functionality for this to be really usable
- 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]

Now for the CFML!

Web Components + CFML

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

Web Components + CFML

- [5 CFML Vars Demo]

CFML + APIs or 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 patch (run PHP *and* COBOL)
- All the browser cares about is the JSON output

Getting JSON back from CFML

- Several ways to do it
- 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

- [6 CFML + JSON Demo]

Getting JSON back from CFML

- [7 CFML + JSON Demo via cfhttp]

Migration Uses

- Can use Web Components instead of full blown SPA frameworks
- Or as a migration plan for 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 Compoents
- A handful of lifecycle methods
- A little machinery/boilerplate
- Everything else is up to how fancy you want to get with JavaScript
- Can integrate any other JS libs you want to use
- (Depends how well that lib is written, YMMV)

Some Useful Tips

- Attributes are *lowercase* or they break!
- Stick to a standard prefix for your naming convention (the hyphen is req'd so use it to your advantage)
- Plan and organize first
- Form elements can be tricky. Use FormData Interface or ElementInternals or a Shim (YMMV)
- Attributes for input
- CustomEvent() for output

Other Resources

- Lit Framework
- Web Components in Space on YouTube (Ben Farrell)
- custom-elements-everywhere.com
- webcomponents.org
- Tons of libaries on GitHub

Learning in 30 minutes a day.

Questions? Comments?

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

Thanks!