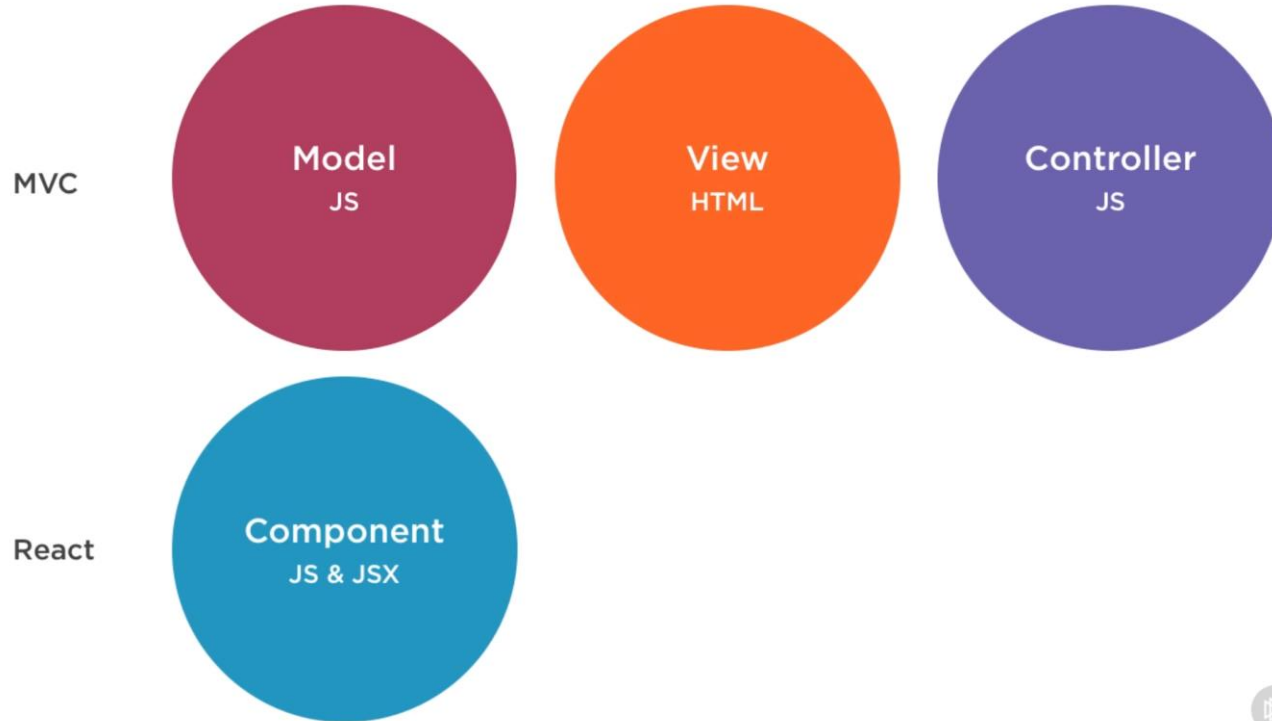




# The magical, reusable web component compiler

# Problem I: Separation of Concerns

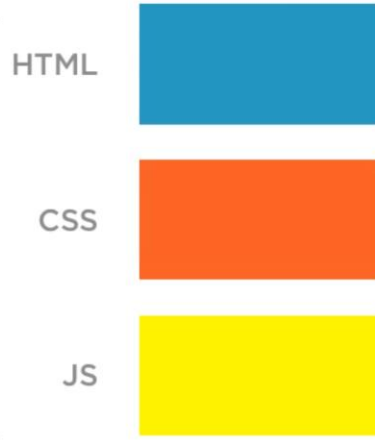
# MVC vs Components



# Separation of concerns

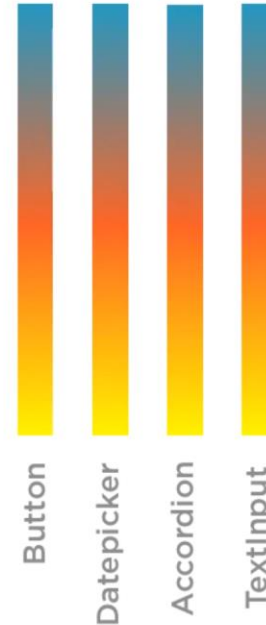
MVC [e.g. Ionic Page]

Separate technologies,  
but *intertwined* concerns.



Component

Each *component* is a separate concern.



# Component-Based-Architecture

The screenshot displays the Pluralsight website interface. On the left, a sidebar navigation menu is highlighted with a blue box, containing links for Home, NavLink, Browse, Paths, Mentors, Channels, Bookmarks, and Notes. The main content area features an author profile for Cory House, which includes a profile picture, name, title, social media links, and a bio. This profile section is enclosed in a blue box. To the right of the profile, a list of courses authored by Cory House is shown. The first course, 'Building a JavaScript Development Environment', is highlighted with a green box. Below it, the 'Building Applications with React and Redux in ES6' course is highlighted with a blue box, and its star rating is specifically highlighted with an orange box. The 'StarRating' label is placed next to this rating. The entire list of courses is enclosed in a green box. The bottom of the page shows the 'bbv Software' logo.

PLURALSIGHT

What do you want to learn?

LIBRARY

Home

NavLink

Browse

Paths

Mentors

Channels

Bookmarks

Notes

AuthorPhoto

Cory House  
Pluralsight Author

Cory is an independent consultant with over 15 years of experience in software development. He is a Microsoft MVP, ASP Insider, and a member of the Telerik developer experts program. As a software architect at Autotrader, Cory specializes in creating single page applications for the automotive industry using C#, .NET, Node, and JavaScript. Cory coaches teams around the world on clean coding practices, software architecture, and front-end development. He also speaks regularly at regional and international conferences like NDC, Fluent, and Codemash. Cory lives in Kansas City where he blogs at bitnative.com and is active on Twitter as @housecor.

AuthorSummary

Courses authored by Cory House

AuthorCourses

Building a JavaScript Development Environment

by Cory House Beginner 5h 19m 10 November 2016 ★★★★★ (163)

CourseSummary

Building Applications with React and Redux in ES6

by Cory House Intermediate 6h 13m 20 May 2016 ★★★★★ (558)

StarRating

Building Applications with React and Flux

by Cory House Intermediate 5h 8m 12 August 2015 ★★★★★ (967)

HTML5 Web Component Fundamentals

by Cory House Beginner 5h 3m 09 January 2015 ★★★★★ (370)

Becoming an Outlier: Reprogramming the Developer Mind

by Cory House Intermediate 2h 33m 24 April 2014 ★★★★★ (724)

Architecting Applications for the Real World in .NET

by Cory House Intermediate 2h 52m 07 January 2014 ★★★★★ (1532)

Clean Code: Writing Code for Humans

by Cory House Intermediate 3h 10m 07 October 2013 ★★★★★ (1590)

bbv Software

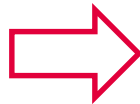
## Best practices: components

1. Small, focused components
2. Use container components

# Best practices: components

- small, focused components [style]
- use container components [data-binding]

```
class Sidebar extends React.Component {  
  componentDidMount() {  
    fetch('api.com/sidebar')  
      .then(res => {  
        this.setState({  
          items: res.items  
        })  
      });  
  }  
  
  render() {  
    return (  
      <div className="sidebar">  
        {this.state.items.map(item => (  
          <div className="sidebar__item">{item}</div>  
        ))}  
      </div>  
    )  
  }  
}
```



```
class SidebarContainer extends React.Component {  
  componentDidMount() {  
    fetch('api.com/sidebar')  
      .then(res => {  
        this.setState({  
          items: res.items  
        })  
      });  
  }  
  
  render() {  
    return (  
      <Sidebar>  
        {this.state.items.map(item => (  
          <SidebarItem item={item} />  
        ))}  
      </Sidebar>  
    )  
  }  
}
```



GETTING STARTED

TUTORIAL

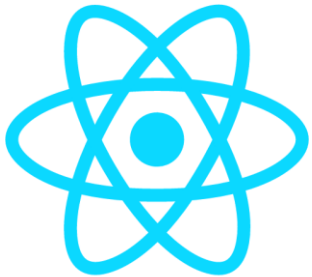
FUNDAMENTALS

Architecture



## Introduction to components

A *component* controls a patch of screen called a *view*. For example, individual components views from the [Tutorial](#):



# React





# Problem II: Dependency/Framework Hell



# Angular Google Maps (AGM)

ANGULAR 2+ COMPONENTS FOR GOOGLE MAPS

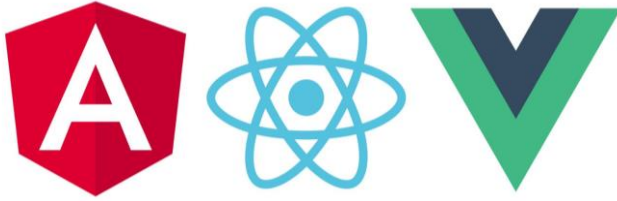
```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, Component } from '@angular/core';

import { AgmCoreModule } from '@agm/core';

@Component({
  selector: 'app-root',
  styles: [
    agm-map {
      height: 300px;
    }
  ],
  template: `
    <agm-map [latitude]="lat" [longitude]="lng"></agm-map>
  `
})
export class AppComponent {
  lat: number = 51.678418;
  lng: number = 7.809007;
}

@NgModule({
  imports: [
    BrowserModule,
    AgmCoreModule.forRoot({
      apiKey: 'YOUR_GOOGLE_MAPS_API_KEY'
    })
  ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule {}
```

# Frameworks vs WebComponents



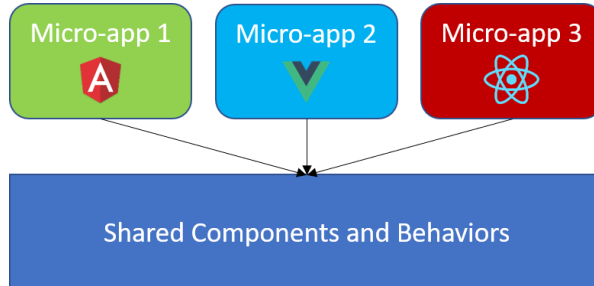
Which framework do we choose?

It doesn't matter.

Use **WebComponents**!

=set of web platform APIs:

<https://www.webcomponents.org/>



# WebComponents with Stencil

# WebComponents ↔ Stencil

## WebComponents

The Web Components Specifications



Custom Elements



Shadow DOM



HTML Templates



HTML Imports

## Stencil

Reactive Data-binding

Virtual DOM

Typescript Support

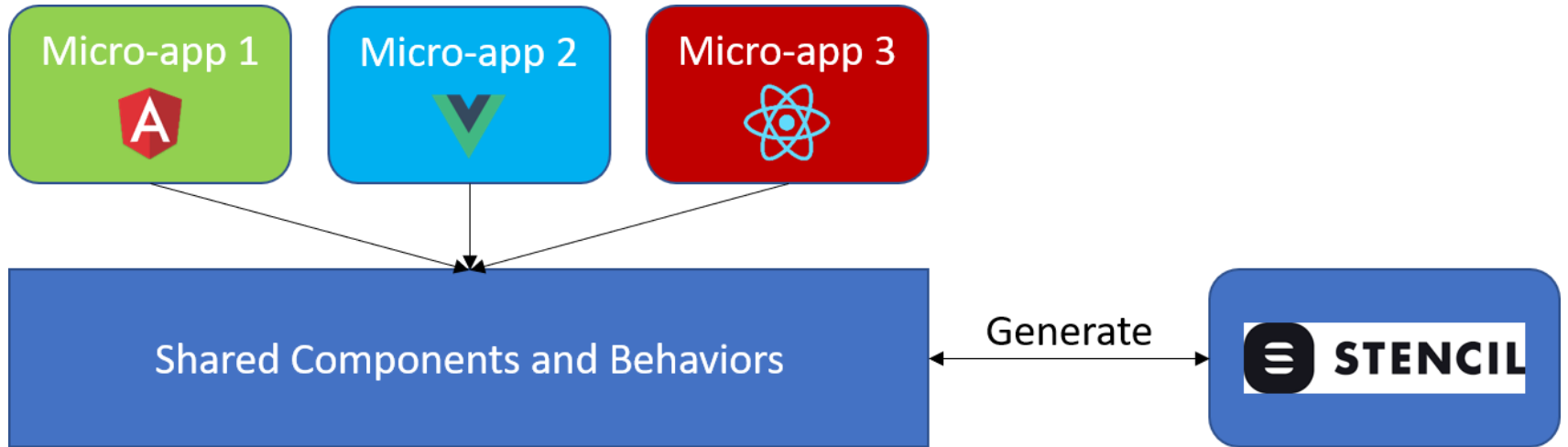
Async rendering ala React Fiber

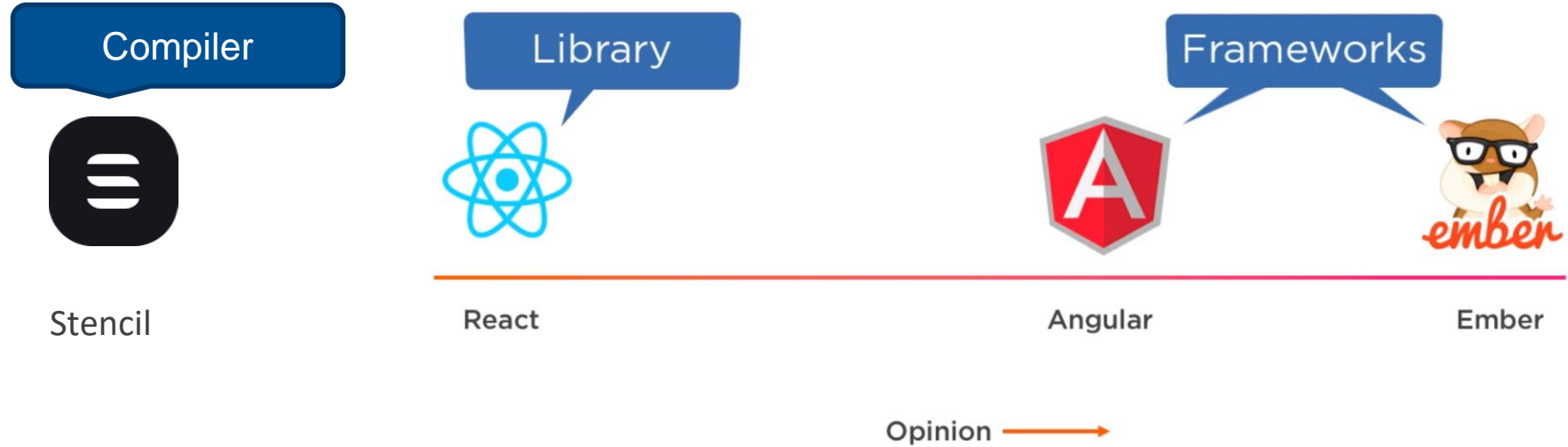
JSX (react syntax)

SSR, pre-compilation

⇒ Stencil is a compiler - not a framework

⇒ Outputs vanilla web components







```
function HelloWorld(props) {  
  return (  
    <div>  
      Hello {props.message}  
    </div>  
  )  
}
```

If I pass in "world"  
for props.message...

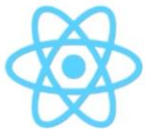
Reliable  
Deterministic  
No side-effects

<div>Hello World</div>



### "JS" in HTML

```
<div *ngFor="let user of users">  
  <div v-for="user in users">  
    {{#each user in users}}
```



### "HTML" in JS

```
{users.map(createUser)}
```

This function is built into JavaScript.

- ⇒ TSX/JSX compiles to JS
- ⇒ JavaScript vs. Framework specific syntax

# Stencil: TSX (JSX) + Angular component syntax

```
import { Component, Prop } from '@stencil/core';

@Component({
  tag: 'my-first-component',
  styleUrls: 'my-first-component.scss'
})
export class MyComponent {

  // Indicate that name should be a public property on the component
  @Prop() name: string;

  render() {
    return (
      <p>
        My name is {this.name}
      </p>
    );
  }
}
```

```
<my-first-component name="Max"></my-first-component>
```

**Implementation detail!**



```
<Button className="btn"></Button>  
<Button className="btn btn--primary"></Button>
```



**Implementation detail!**

```
<Button></Button>  
<Button primary></Button>
```



## Stencil Tools

natemoo-re.vscode-stencil-tools

Nate Moore



491



Repository

License

Helpful utilities for StencilJS Projects

Disable ▼

Uninstall

# Ionic v4

Reference Docs:

<https://github.com/ionic-team/ionic>

ionic-team / ionic

Watch

1,918

Star

35,245

Fork

11,826

<> Code

Issues 1,204

Pull requests 34

Projects 1

Insights

Branch: master ionic / core / src / components / button /

Create new file

Upload files

Find file

History

brandyscarney fix(button): add custom properties and remove --ion-color overrides (#...

Latest commit 3af4361 a day ago

|                      |   |              |
|----------------------|---|--------------|
| test                 | fix(button): add custom properties and remove --ion-color overrides (#... | a day ago    |
| usage                | fix(docs): use shape property to get round buttons (#15321)               | 12 days ago  |
| button.ios.scss      | fix(button): add custom properties and remove --ion-color overrides (#... | a day ago    |
| button.ios.vars.scss | refactor(all): cleanup mode font  | 14 days ago  |
| button.md.scss       | fix(button): add custom properties and remove --ion-color overrides (#... | a day ago    |
| button.md.vars.scss  | refactor(all): cleanup mode font  | 14 days ago  |
| button.scss          | fix(button): add custom properties and remove --ion-color overrides (#... | a day ago    |
| button.tsx           | fix(button): add custom properties and remove --ion-color overrides (#... | a day ago    |
| button.vars.scss     | refactor(components): update to use shadow DOM and work with css vari...  | 2 months ago |
| readme.md            | fix(button): add custom properties and remove --ion-color overrides (#... | a day ago    |

readme.md

## ion-button

Buttons provide a clickable element, which can be used in forms, or anywhere that needs simple, standard button functionality. They may display text, icons, or both. Buttons can be styled with several attributes to look a specific way.

### Expand

This attribute lets you specify how wide the button should be. By default, buttons are inline blocks, but setting this attribute will change the button to a full-width block element.

# Ionic PWA Toolkit Beta

<https://ionicframework.com/pwa/toolkit>

<https://github.com/ionic-team/ionic-pwa-toolkit>

DEMO: Stencil Poker

<https://github.com/blumk/bbv-planning-poker>

<https://bbv-poker.netlify.com/>



# Discussion

- WebComponents Compilers:
  - StencilJS?
  - Angular Elements
    - Performance (Angular 7?), Angular syntax?
  - SkateJs
- Living Styleguide?
  - <https://storybook.js.org/>
  - <https://frontify.com/de/>

**MAKING VISIONS WORK.**

