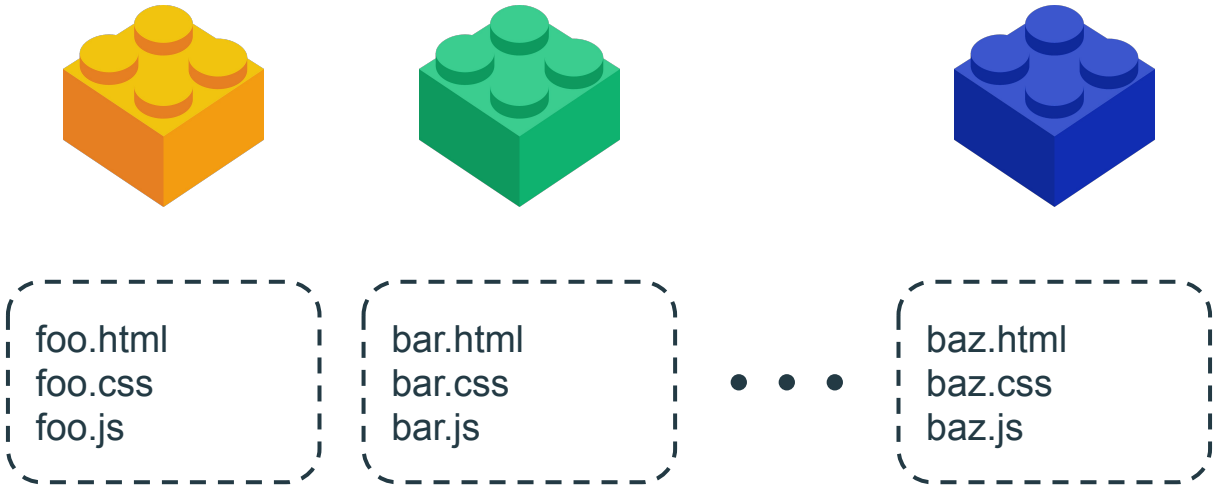
Hello Web Components!
Starring - StencilJS

Christina Kayastha
@christikaes

Web Components?


foo.html
foo.css
foo.js
bar.html
bar.css
bar.js
baz.html
baz.css
baz.js

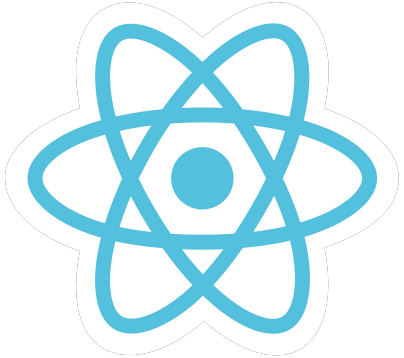

foo.html
foo.css
foo.js
bar.html
bar.css
bar.js
baz.html
baz.css
baz.js

foo.component.html
foo.component.css
foo.component.ts

bar.component.html
bar.component.css
bar.component.ts

• • •

baz.component.html
baz.component.css
baz.component.ts



foo.jsx
(foo.css)

bar.jsx
(bar.css)

• • •

baz.jsx
(baz.css)


foo.vue
bar.vue
baz.vue

foo.html
foo.css
foo.js

foo.html
foo.css
foo.js

• • •

foo.html
foo.css
foo.js

Existing approach

Web Components

Web Components

HTML Templates

Shadow DOM

ES Modules

Custom Elements

Web Components

HTML Templates

Shadow DOM

ES Modules

Custom Elements

HTML Templates

```
<template>
  <h2>Flower</h2>
  <img src="img_white_flower.jpg">
</template>
```

Web Components

HTML Templates

Shadow DOM

ES Modules

Custom Elements

Web Components

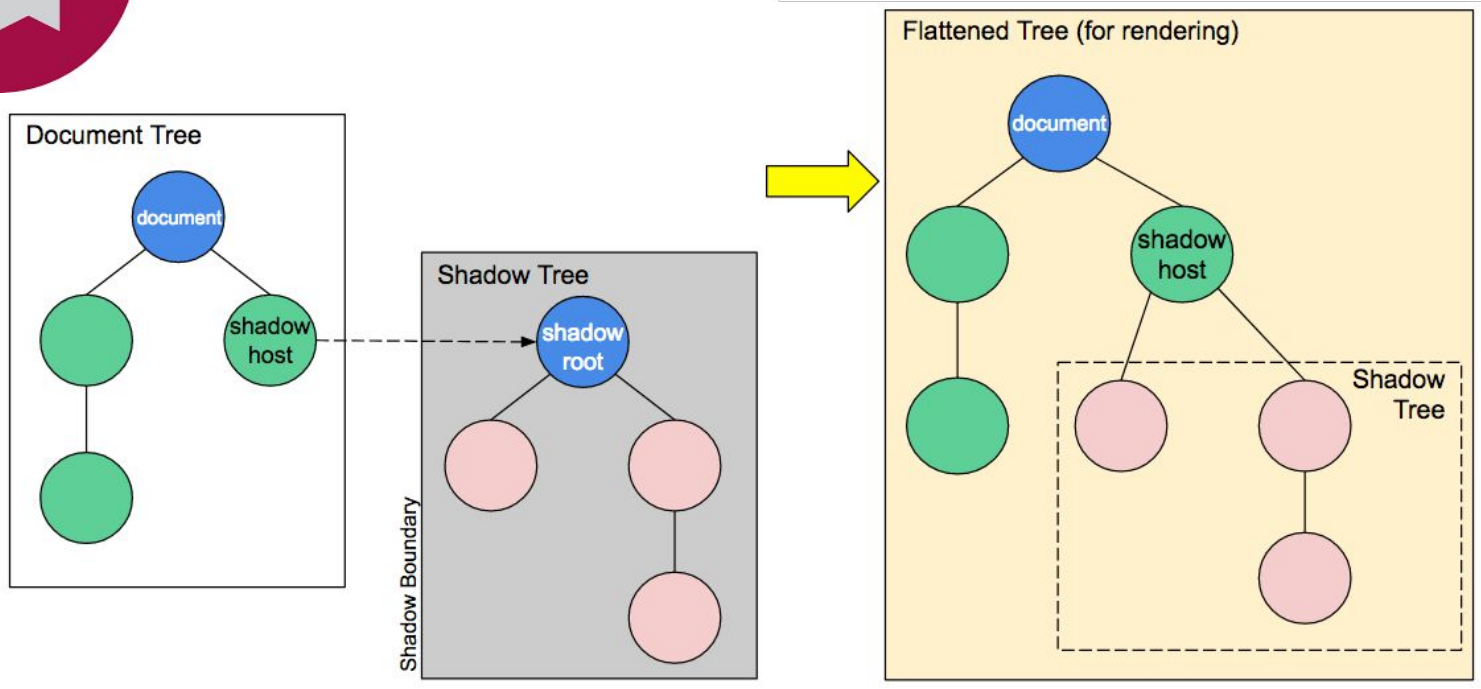
HTML Templates

Shadow DOM

ES Modules

Custom Elements

Shadow Dom


Document Tree
document
shadow host
Shadow Tree
shadow root
Shadow Boundary
Flattened Tree (for rendering)
document
shadow host
Shadow Tree

Web Components

HTML Templates

Shadow DOM

ES Modules

Custom Elements

Web Components

HTML Templates

Shadow DOM

ES Modules

Custom Elements

Web Components

HTML Templates

Shadow DOM

ES Modules

Custom Elements

Custom Elements

```
<my-button></my-button>
```

Web Components

HTML Templates

Shadow DOM

ES Modules

Custom Elements

Browser support	CHROME	OPERA	SAFARI	FIREFOX	EDGE
HTML TEMPLATES	STABLE	STABLE	STABLE	STABLE	STABLE
CUSTOM ELEMENTS	STABLE	STABLE	STABLE	STABLE	POLYFILL DEVELOPING
SHADOW DOM	STABLE	STABLE	STABLE	STABLE	POLYFILL DEVELOPING
ES MODULES	STABLE	STABLE	STABLE	STABLE	STABLE


stencil

The magical, reusable web component compiler

GET STARTED LEARN MORE

Watch launch video

Decorators

Decorators are a pure compiler-time construction used by stencil to collect all the metadata about a component, the properties, attributes and methods it might expose, the events it might emit or even the associated stylesheets. Once all the metadata has been collected, all the decorators are removed from the output, so they don't incur in any runtime overhead.

- **@Component()** declares a new web component
- **@Prop()** declares an exposed property/attribute
- **@State()** declares an internal state of the component
- **@Watch()** declares a hook that runs when a property or state changes
- **@Element()** declares a reference to the host element
- **@Method()** declares an exposed public method
- **@Event()** declares a DOM event the component might emit
- **@Listen()** listens for DOM events

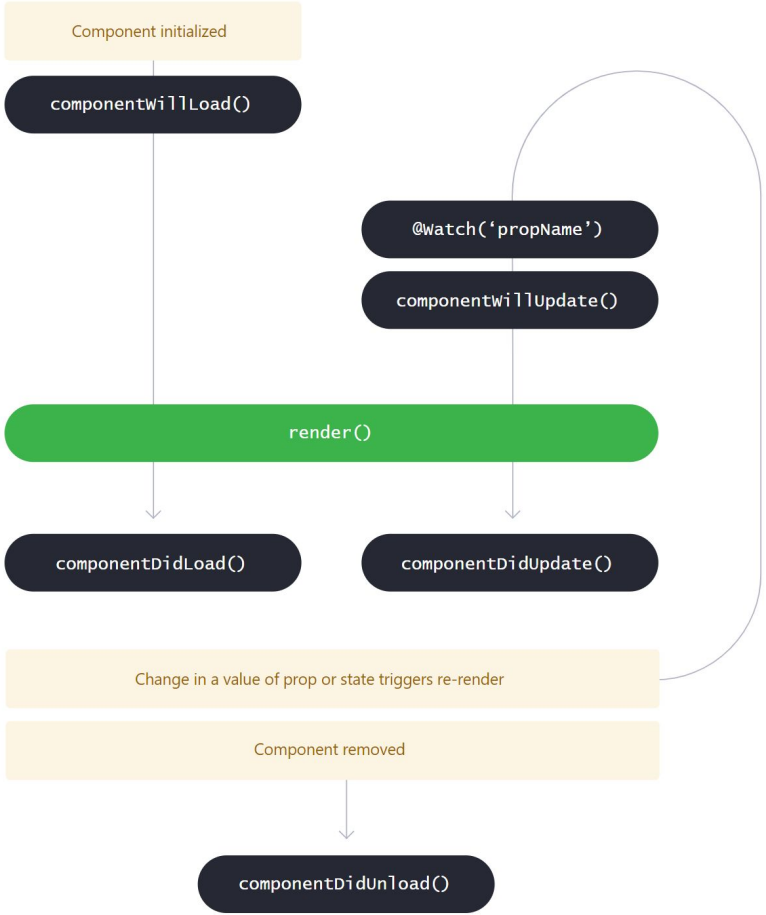
Decorators

Decorators are a pure compiler-time construction used by stencil to collect all the metadata about a component, the properties, attributes and methods it might expose, the events it might emit or even the associated stylesheets. Once all the metadata has been collected, all the decorators are removed from the output, so they don't incur in any runtime overhead.

- `@Component()` declares a new web component
- `@Prop()` declares an exposed property/attribute
- `@State()` declares an internal state of the component
- `@Watch()` declares a hook that runs when a property or state changes
- `@Element()` declares a reference to the host element
- `@Method()` declares an exposed public method
- `@Event()` declares a DOM event the component might emit
- `@Listen()` listens for DOM events

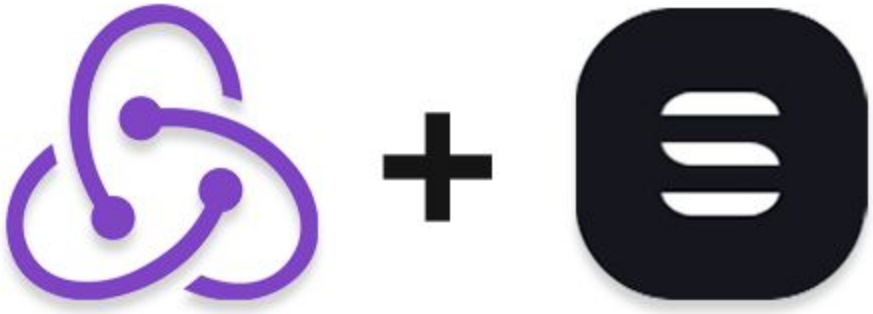
Lifecycle hooks

- componentWillLoad()
- componentDidLoad()
- componentWillUpdate()
- componentDidUpdate()
- componentDidUnload()




stencil

Redux + Stencil



Configure store

```
// src/store/index.ts
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk'; // Add-on you might want
import logger from 'redux-logger'; // Add-on you might want
import rootReducer from '../reducers/index';

const configureStore = (preloadedState: any) =>
  createStore(rootReducer, preloadedState, applyMiddleware(thunk, logger));

export { configureStore };
```

Configure reducers

```
// src/reducers/index.ts
import myReducer from './myReducer';

import { combineReducers } from 'redux';

const rootReducer = (combineReducers as any)({
  myReducer
});

export default rootReducer;
```

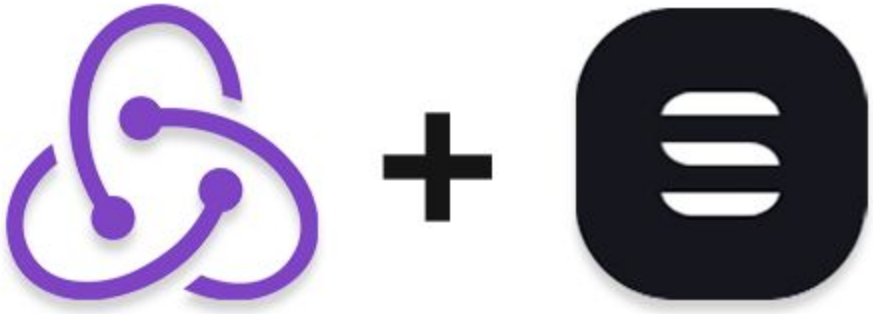
Configure Store in Root Component

```
import { Store } from '@stencil/redux';
import { configureStore } from '../../store/index'; // index required due t

@Component({
  tag: 'my-app',
  styleUrl: 'my-app.scss'
})
export class MyApp {
  @Prop({ context: 'store' }) store: Store;

  componentWillLoad() {
    this.store.setStore(configureStore({}));
  }
}
```

Map state and dispatch to props

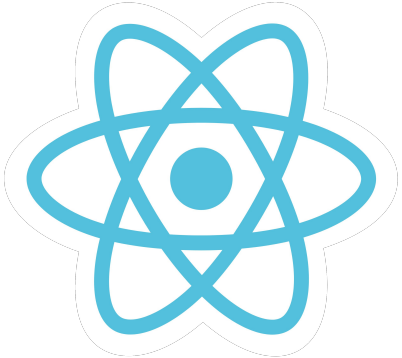
```
import { Store, Action } from '@stencil/redux';

@Component({
  tag: 'my-component',
  styleUrl: 'my-component.scss'
})
export class MyComponent {
  @Prop({ context: 'store' }) store: Store;

  @State() name: string;

  changeName: Action;

  componentWillLoad() {
    this.store.mapStateToProps(this, (state) => {
      const {
        myReducer: { name }
      } = state;
      return {
        name
      }
    });

    this.store.mapDispatchToProps(this, {
      changeName
    })
  }

  doNameChange(newName: string) {
    this.changeName(newName);
  }
}
```

Redux + Stencil



stencil

vs.



Hello Web Components!
Starring - StencilJS

Christina Kayastha
@christikaes