WEB COMPONENTS

BUILDING BLOCKS FOR THE WEB

COMPONENTS?

A term we as developer use a lot, but fail to come up with a single definition. ;)

- Something which encapsulates a functionality
- Something that fosters (black box) reuse
- Something you can mix and match
- A unit of independent deployment

*Components let you split the UI into **independent**, **reusable** pieces, and think about each piece in **isolation**. [..] They accept arbitrary inputs (called "props") and return React elements describing what should appear on the screen.*

Components are one of the most powerful features of Vue. They help you ***extend basic HTML*** *elements to* ***encapsulate reusable code****. At a high level, components are* ***custom elements*** *that Vue's compiler attaches* ***behavior*** *to. In some cases, they may also appear as a native HTML element extended with the special is attribute.*

*Components are the most basic **building block** of an UI in an Angular application. An [..] application is a **tree of [..] components**. [..] Unlike directives, components always have a **template** and only one component can be instantiated per an element in a template.*

"One thing can be stated with certainty:
Components are for composition.
[..] Beyond that trivial observation, much is unclear."
(Clemens Szypersky)

WEB COMPONENTS

A set of well-supported specs:

- Custom Elements
- HTML Templates
- Shadow DOM
- HTML Imports

Browser support	CHROME	OPERA	SAFARI	FIREFOX	EDGE
TEMPLATES	STABLE	STABLE	STABLE	STABLE	STABLE
CUSTOM ELEMENTS	STABLE	STABLE	STABLE	POLYFILL DEVELOPING	POLYFILL CONSIDERING
SHADOW DOM	STABLE	STABLE	STABLE	POLYFILL DEVELOPING	POLYFILL CONSIDERING
<SCRIPT TYPE="MODULE">	STABLE	STABLE	10.1	FLAG IN 54	FLAG IN 15
HTML IMPORTS	STABLE	STABLE	POLYFILL ON HOLD	POLYFILL ON HOLD	POLYFILL CONSIDERING

CUSTOM ELEMENTS

Allow the developer to create new HTML tags (autonomous custom elements) or extend the behaviour of existing elements (customized built-in elements).

Provide a component life cycle.

```
class MyCustomElement extends HTMLElement {
  constructor() {
   super();
  }

  connectedCallback() {}
  attributeChangedCallback(attrName, oldVal, newVal) {}
  disconnectedCallback() {}
  adoptedCallback(oldDocument, newDocument) {}
  static get observedAttributes() {
    return ["attr1", "attr2"];
  }
}
```

CUSTOM ELEMENTS REGISTRY

```
customElements.define("my-custom-tag", MyCustomElement);

customElements.get("my-custom-tag");

customElements.whenDefined("my-custom-tag");
```

HTML TEMPLATES

The template element is used to declare fragments of HTML that can be cloned and inserted in the document by script.

The content of a template

- is inert until activated
- won't have side effects (no scripts executed, no images loading) until it is used
- is considered not to be in the document

```
<template id="mytemplate">
  My HTML goes here
</template>
```

```
const t = document.querySelector('#mytemplate');

const clone = document.importNode(t.content, true);

document.body.appendChild(clone);
```

SHADOW DOM

Shadow DOM enables to

- author self-contained components with isolated DOM
- bundle CSS with markup
- scope and simplify CSS
- hide implementation details

```
const root = element.attachShadow({mode: 'open'});
root.appendChild(node);
```

HTML IMPORTS

Intended to be the packaging mechanism for web components

```
<link rel="import" href="my-component.html">
```

BUILDING WEB COMPONENTS WITH

STENCIL

STENCIL JS

- Build time tool for developing 100% standard-compliant plain JavaScript web components
- Borrows ideas from popular frameworks without itself being yet another one
- Utilizes: TypeScript, JSX / TSX, SCSS

DECORATOR BASED API

- @Component()
- @Prop()
- @State()
- @Watch()
- @Method()
- @Listen()
- @Event()
- @Element()

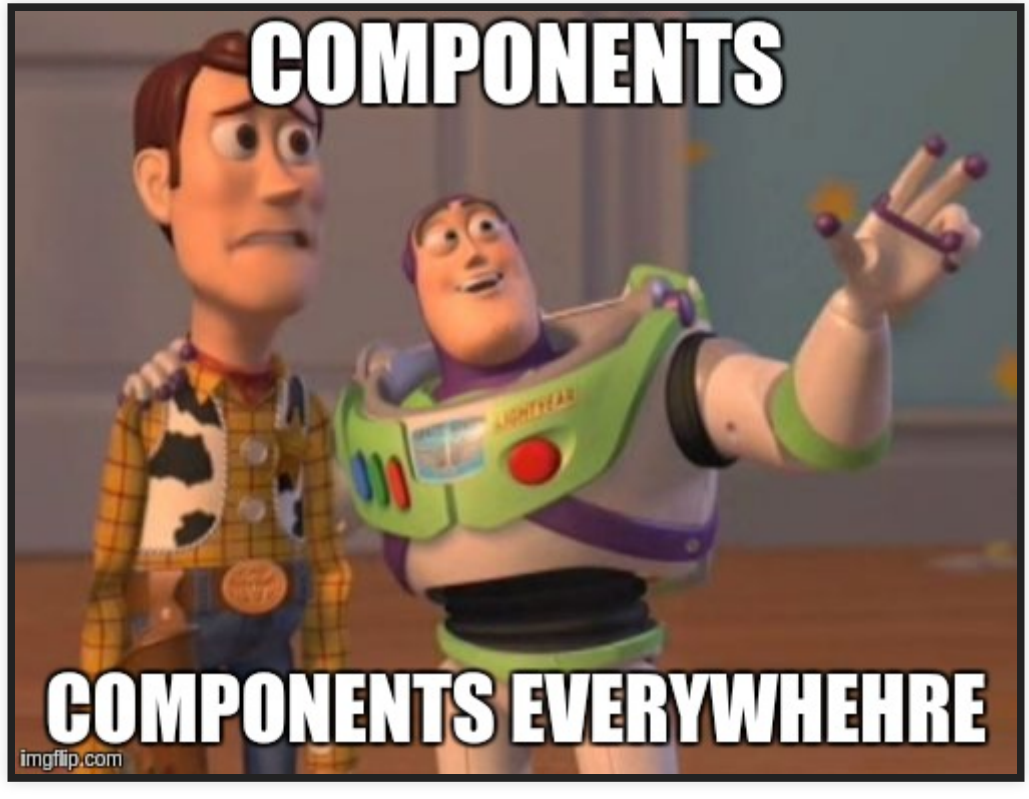
JSX/TSX FOR TEMPLATES / RENDERING

```
render() {
 return (
  <li>
   I'm an item  <slot></slot>
  </li>);
}
```

LIFE CYCLE HOOKS

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

QUESTIONS?



THANK YOU