engage

Web Components: build your own HTML tags like <domino-view>

Knut Herrmann

Leonso

engage

Knut Herrmann

- Senior Software Architect
  Leonso GmbH
- Notes Domino developer since version 2
- Web application developer
- StackOverflow contributor

IBM CHAMPION
2018 and 2019

E-Mail: knut.herrmann@leonso.de

Twitter: @KnutHerrmann

Skype: knut.herrmann

engage

Why

Web Components

engage

Frameworks for Web Application Development

Server-side rendering

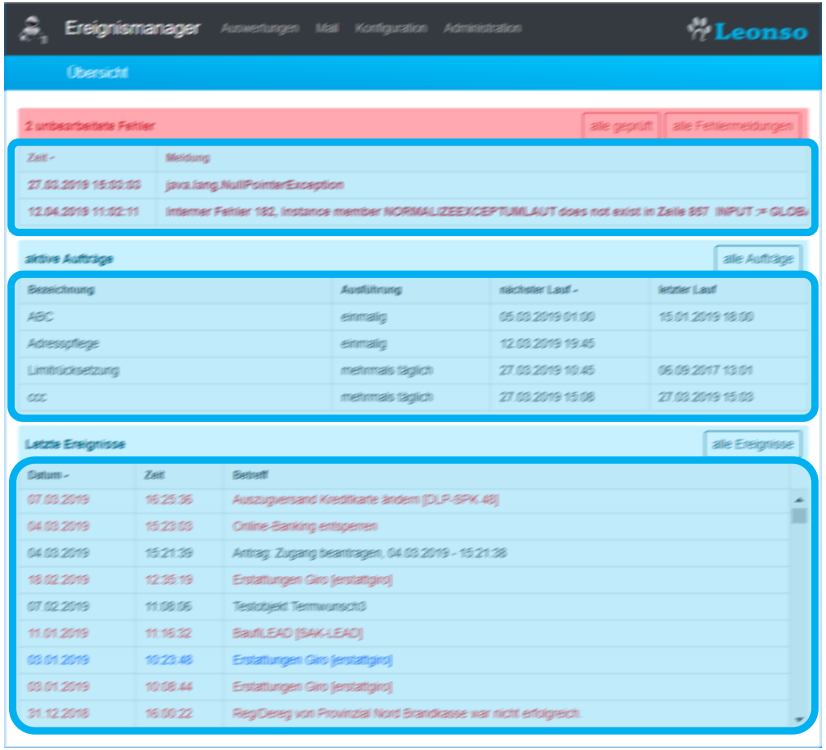
Client-side rendering

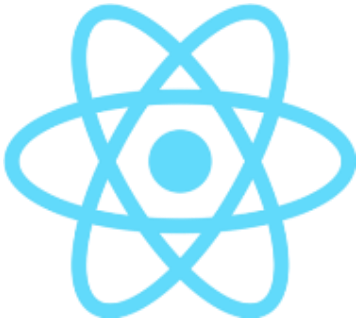
XPages

vaadin

php

engage



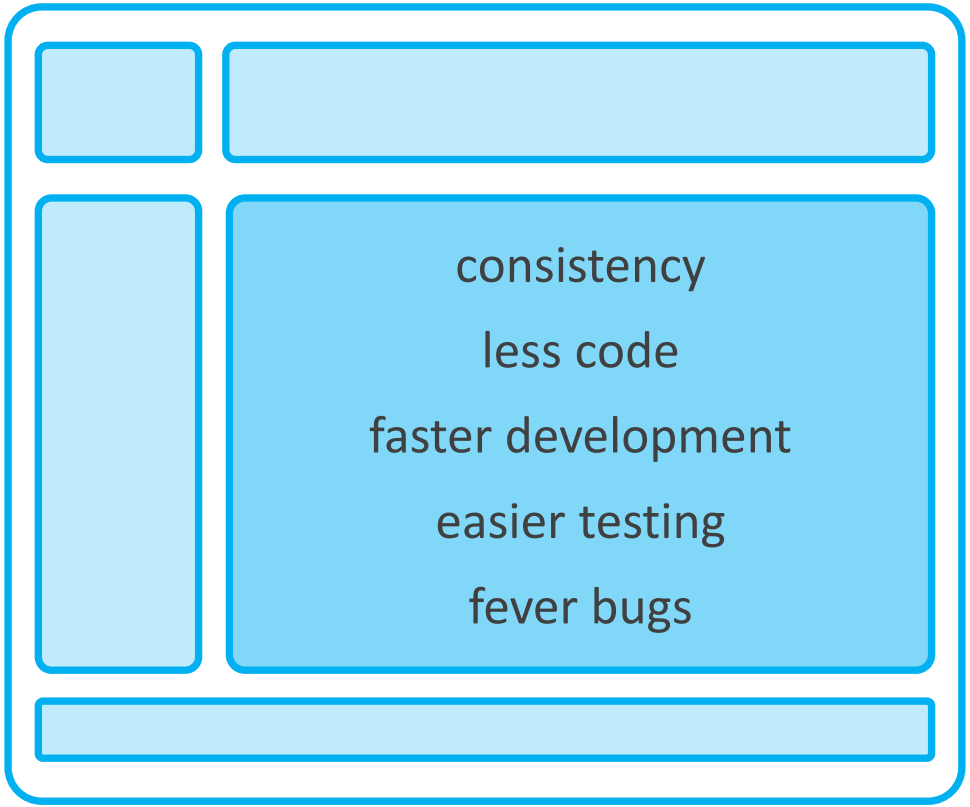


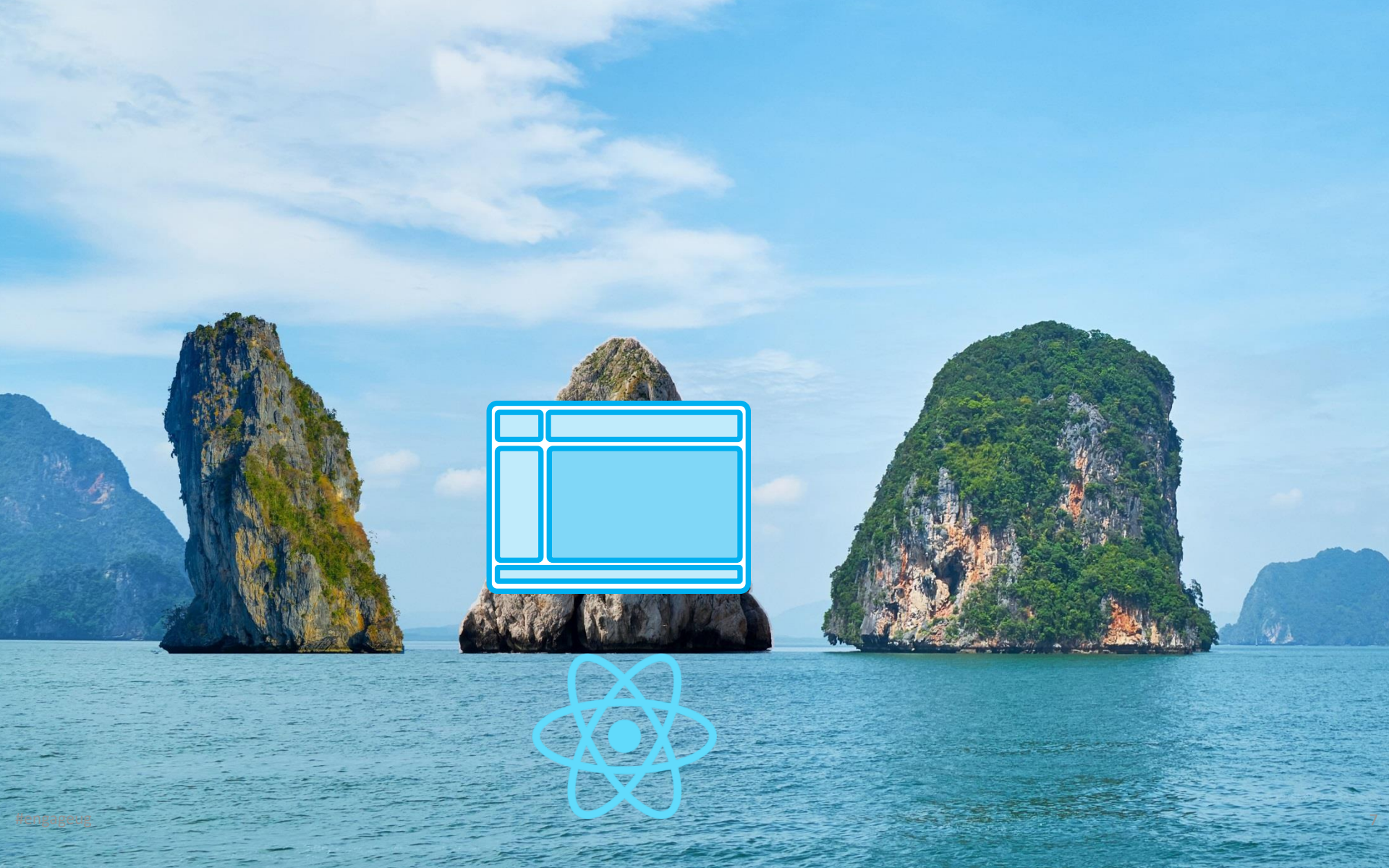
React Frontend

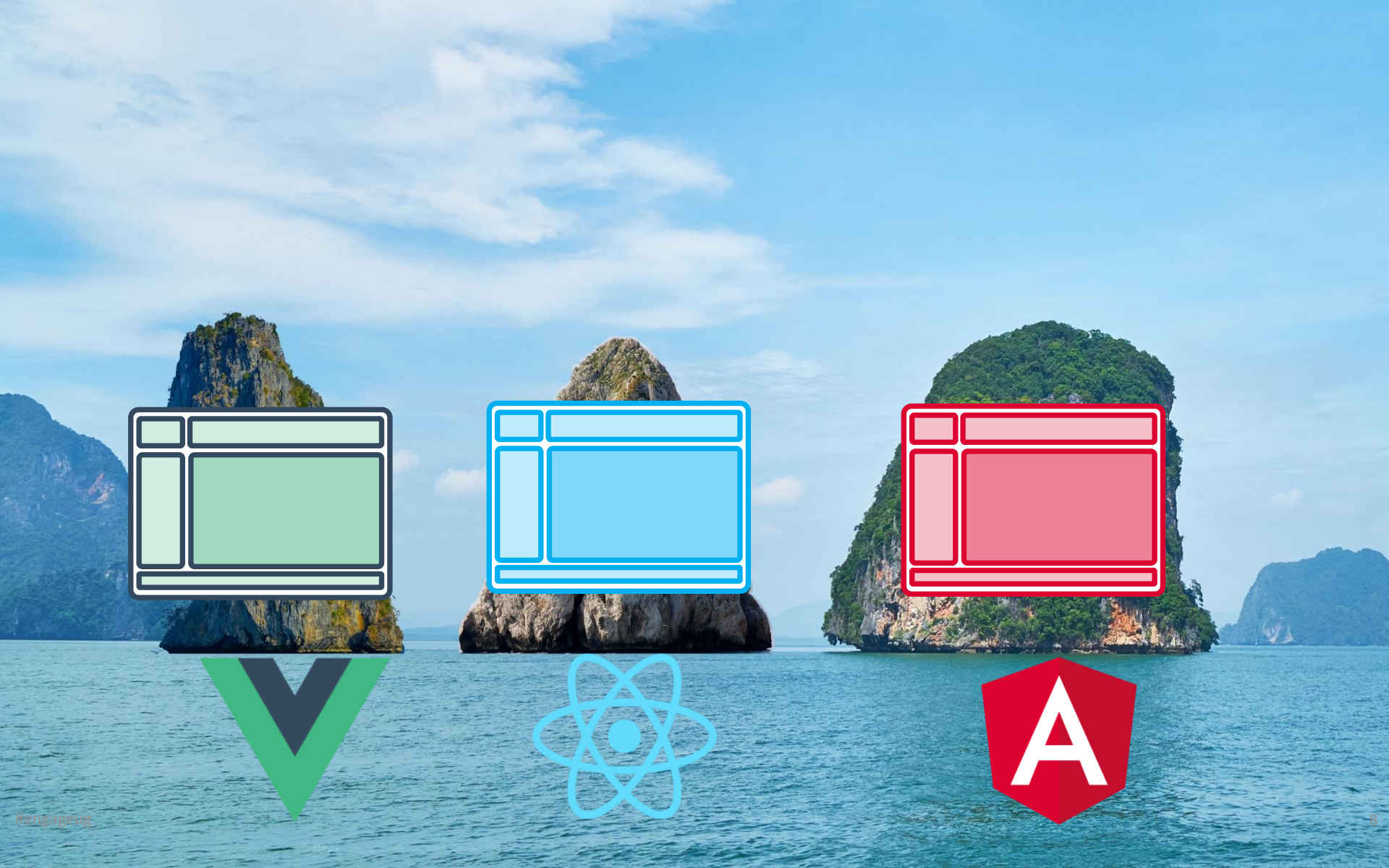
Domino Backend

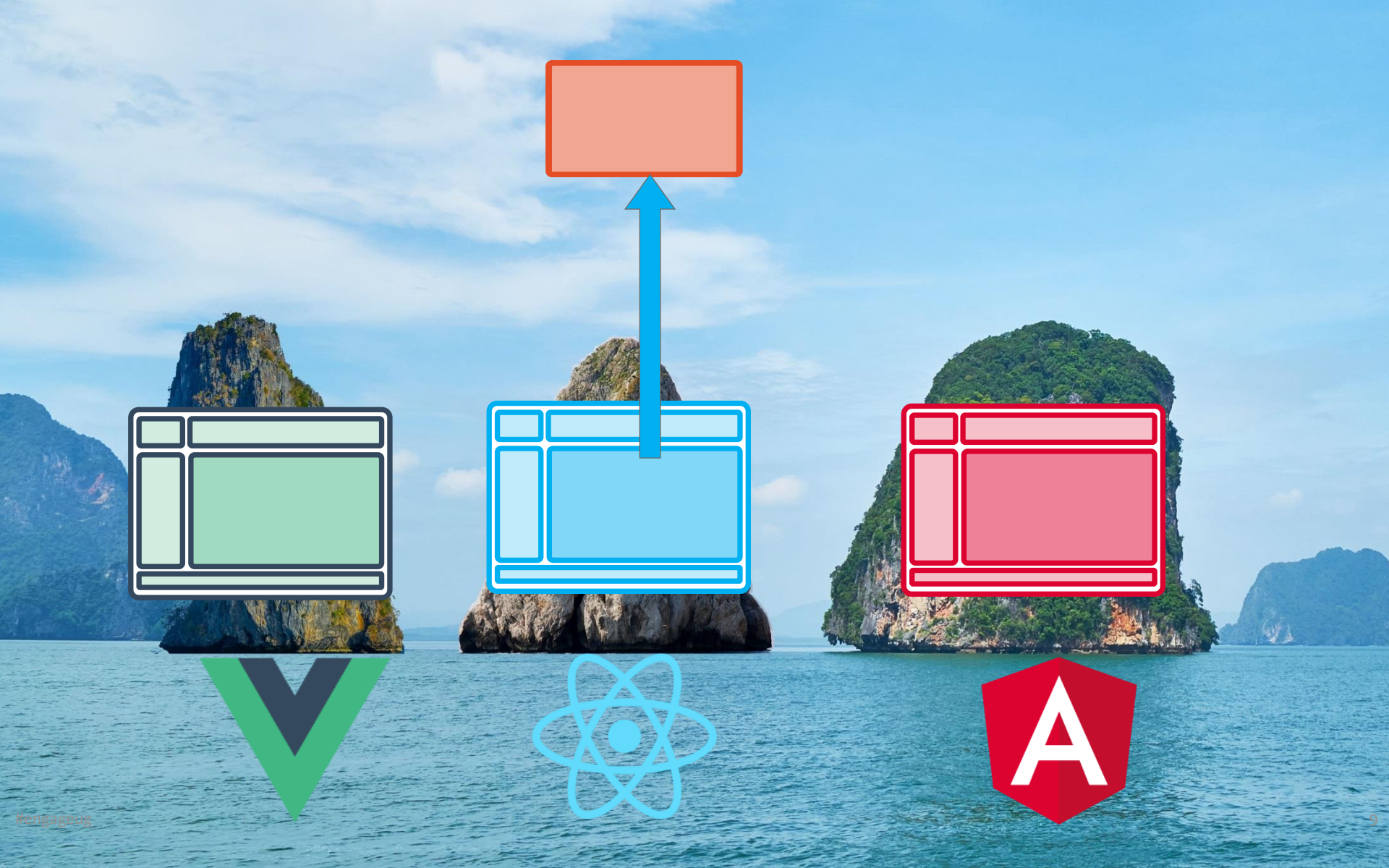
engage

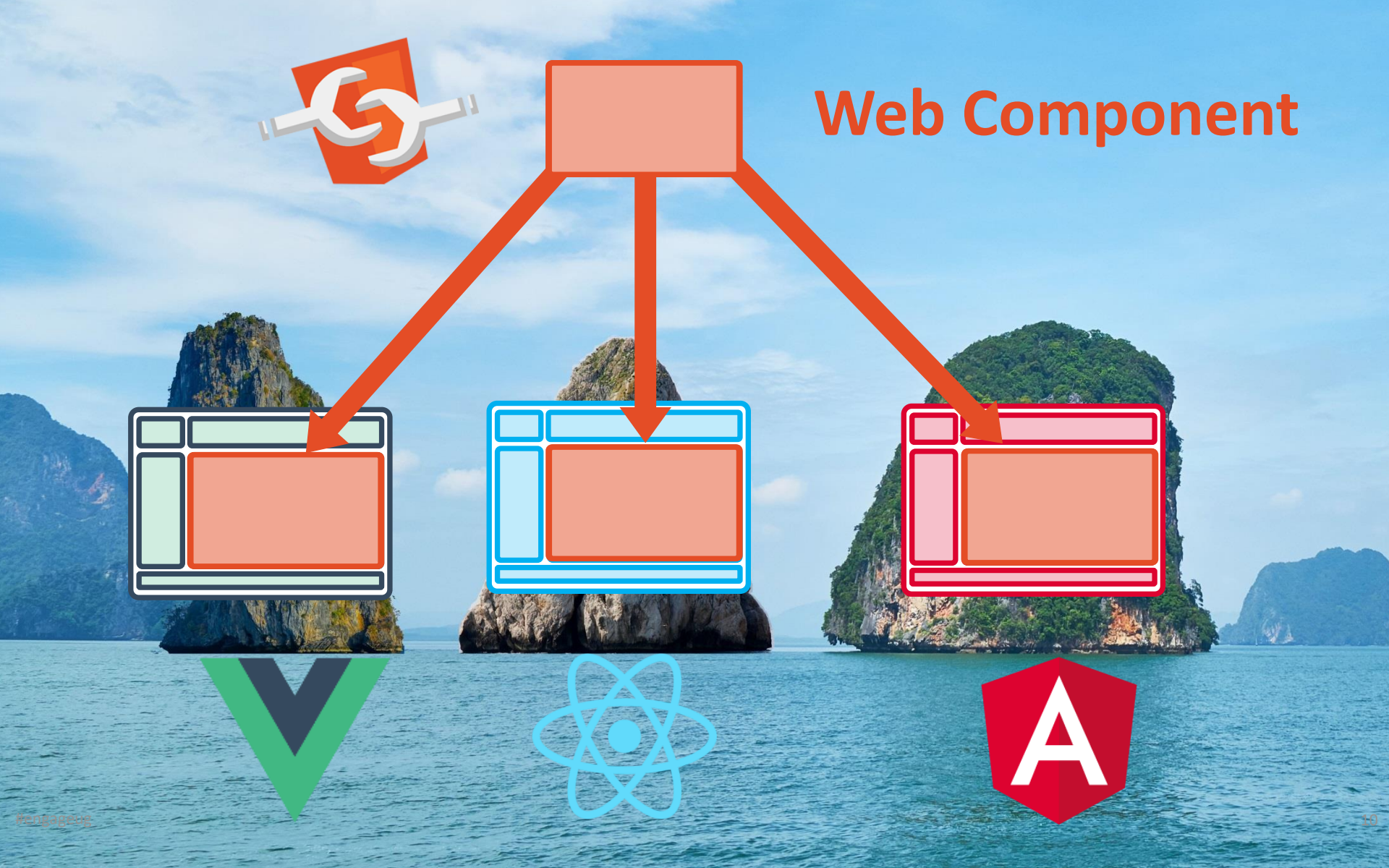
Reusable Components






A


A


Web Component

Web Components

based on web standards

framework agnostic

engage

Let’s create a Web Component

```
<domino-view
    db="myDb"
    view="myView">
</domino-view>
```

Custom Elements

create new HTML tags or extend existing
with vanilla JS/HTML/CSS

</> Custom Elements

domino-view.js

```
class DominoElement extends HTMLElement {…code…}

customElements.define('domino-view', DominoElement);
```

dash!

index.html

```
<domino-view></domino-view>

<script type="module" src="domino-view.js"></script>
```

engage

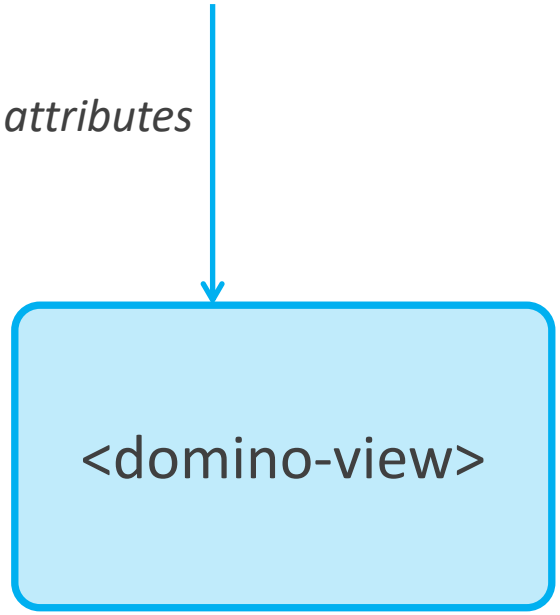
</> Custom Elements

live cycle

```
class DominoElement extends HTMLElement {
  constructor() {
    super();
    ...
  }
  connectedCallback() {
    ...
  }
  disconnectedCallback() {
    ...
  }
}
```

</> Custom Elements

attributes

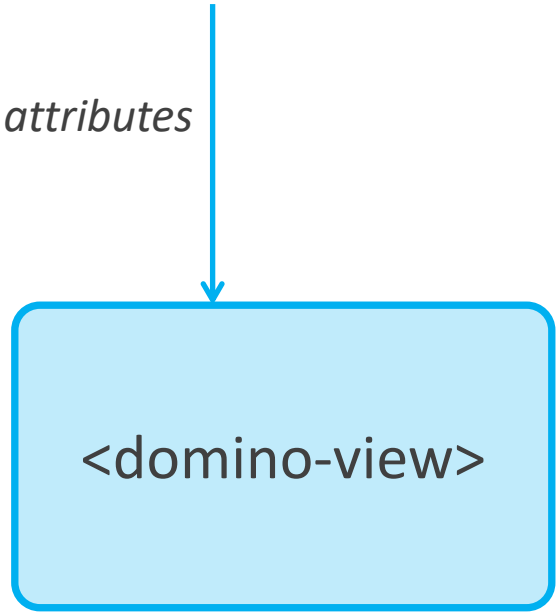



```
<domino-view
     db="myDb"
     view="myView">
</domino-view>
```

```
<script>
  document.querySelector('domino-view')
  .setAttribute('view', 'myOtherView')
</script>
```

</> Custom Elements

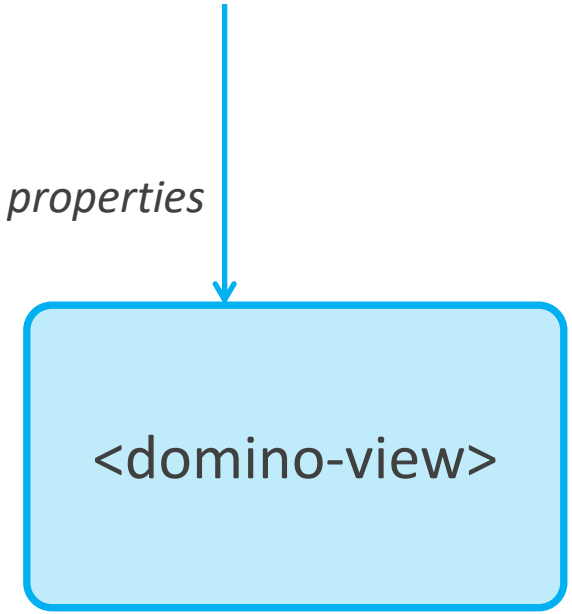
attributes




```
class DominoElement extends HTMLElement {

  static get observedAttributes() {
    return ["db", "view"];
  }

  attributeChangedCallback(attrName,
                           oldValue, newValue) {
    if (attrName === "db") {
      this._db = newValue;
    } else if (attrName === "view") {
      this._view = newValue;
    }
    this.render();
  }
```

</> Custom Elements

properties



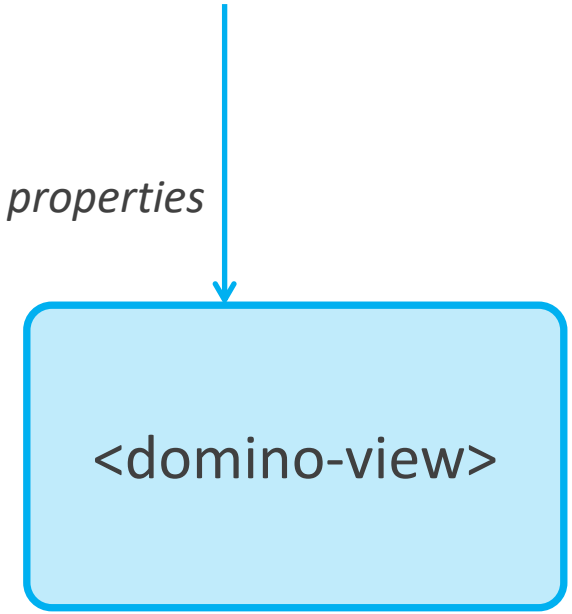

```
<domino-view></domino-view>
```

```
<script>
  const dominoView =
   document.querySelector('domino-view');
  dominoView.db = 'myDb';
  dominoView.view = 'myView';
</script>
```

</> Custom Elements

properties

```
class DominoElement extends HTMLElement {
```

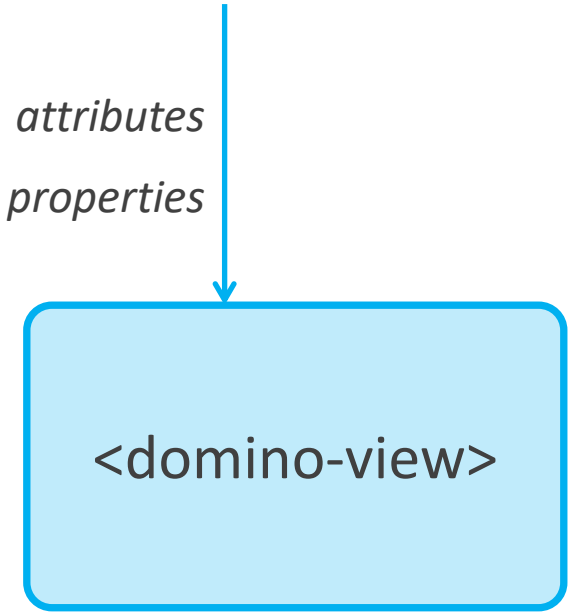



```
  set view(view) {
    this._view = view;
    this.render();
  }

  get view() {
    return this._view;
  }
```

</> Custom Elements

properties

```
class DominoElement extends HTMLElement {
```



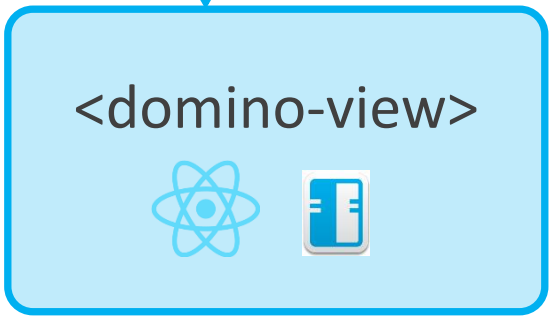

```
set view(view) {
  this.setAttribute("view", view);
}

get view() {
  return this.getAttribute("view");
}
```

</> Custom Elements

React & Domino

attributes

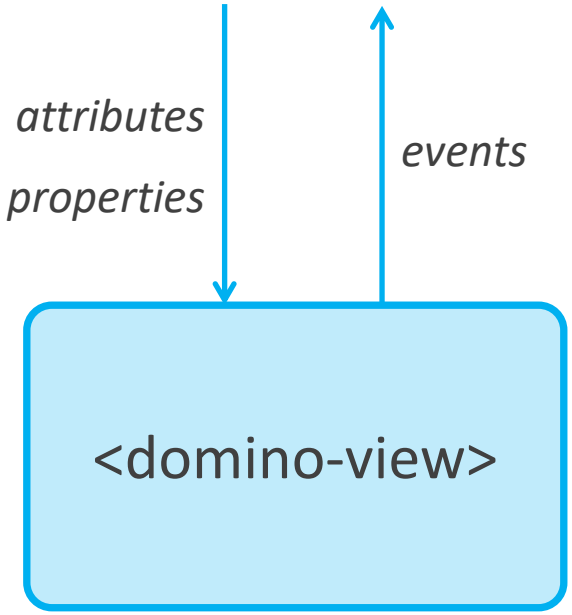
properties




```
import React from "react";
import ReactDOM from "react-dom";
import DominoView from "./DominoView";

class DominoElement extends HTMLElement {

  connectedCallback() {
    this._isConnected = true;
    this.render();
  }
  render = () =>
    this._isConnected &&
    ReactDOM.render(
      <DominoView db={this._db}
                  view={this._view} />,
      this
    );
```

</> Custom Elements

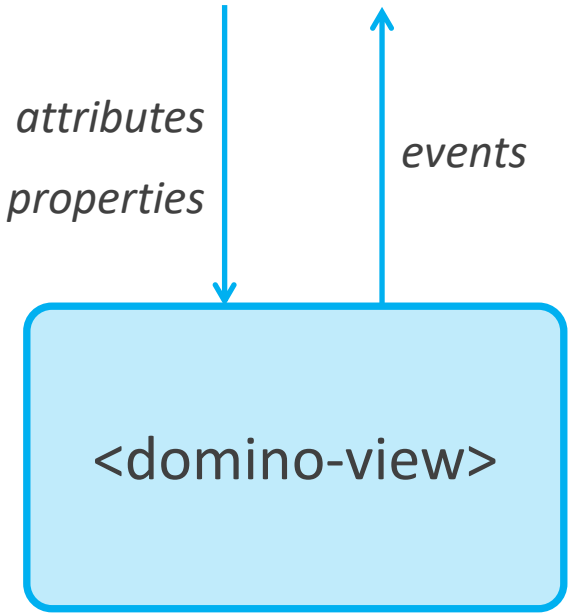
events




```
<domino-view
     db="myDb"
     view="myView">
</domino-view>

<script>
   document
      .querySelector("domino-view")
      .addEventListener(
        "handleRowClick",
        event => doSomethingWith(event.detail)
      );
</script>
```

</> Custom Elements

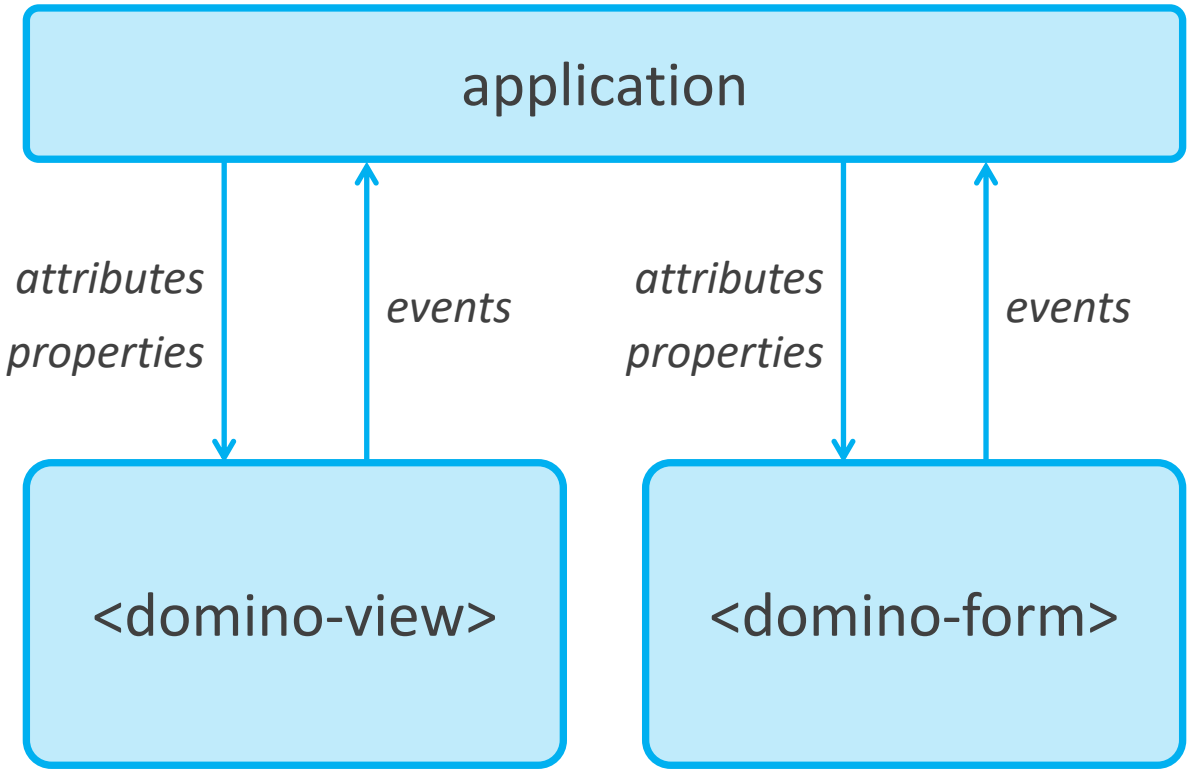
events




```
class DominoElement extends HTMLElement {

  handleRowClick = (rowData) =>
    this.dispatchEvent(
      new CustomEvent("handleRowClick",
                      {detail: rowData}));

  render = () =>
    this._isConnected &&
    ReactDOM.render(
      <DominoView
        db={this._db}
        view={this._view}
        handleRowClick={this.handleRowClick}
      />,
      this
    );
```

</> Custom Elements

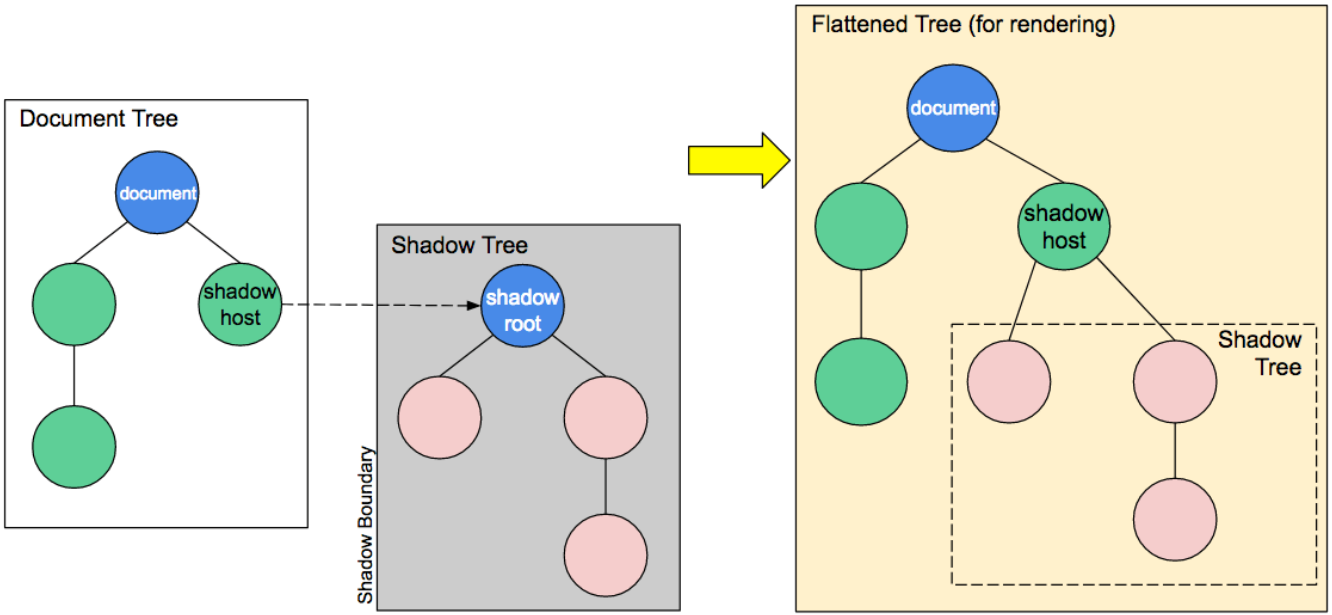
events



Shadow DOM

encapsulates custom components
by hidden separated DOM

Shadow DOM



https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM

Shadow DOM

isolated DOM

simple CSS selectors

cheap style checks

fast

Shadow DOM

```
class DominoElement extends HTMLElement {

  constructor() {
    super();
    this._root = this.attachShadow({mode: 'open'});
  }

  render = () =>
    this._isConnected &&
    ReactDOM.render(
      <DominoView
        db={this._db}
        view={this._view}
        handleRowClick={this.handleRowClick} />,
      this._root
    );
```

Shadow DOM

shadow host

```
<style>
    domino-view {
        height: 300px;
        color: blue;
        --header-color: green;
    }
</style>
<domino-view … ></domino-view>
```

User defined styles overwrite custom element's styles

CSS custom properties

shadow DOM

```
.domino-view__header-row {
    color: var(--header-color);
}
```

Shadow DOM

Inheritable styles
(background, color, font, line-height, etc.)
continue to inherit in shadow DOM

shadow DOM

unless

```
:host {
    all: initial;
    display: block;
    height: 100%;
}
:host([hidden]) {
    display: none;
}
```

Templates

placeholder for declaring the structure of a custom element

Templates

parsed once

instantiated during runtime any number of times

Templates

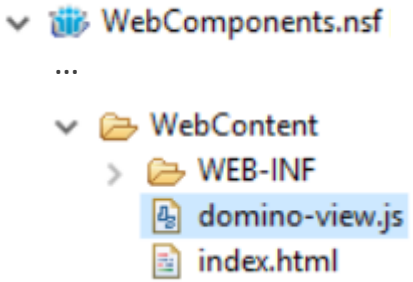
```
const template = document.createElement('template');
template.innerHTML = `
  <style>:host { ... }</style>
  <h2>This is an example for Templates.</h2>
  <slot></slot>
  <p>Content...</p>
`;

class DominoElement extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({mode: 'open'});
    this.shadowRoot.appendChild(
      template.content.cloneNode(true));
  }
```

Templates

```
<domino-view ...>
  <b>This will be placed into slot.</b>
</domino-view>
```

```
const template = document.createElement('template');
template.innerHTML = `
  <style>:host { ... }</style>
  <h2>This is an example for Templates.</h2>
  <slot></slot>
  <p>Content...</p>
`;
```

Deployment



… or publishing on NPM

```
<script type="module"
  src="https://server/WebComponents.nsf/domino-view.js">
</script>
```

engage

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

https://www.webcomponents.org/

Compatibility



Use Web Components	✓	✓*	✓
Build Web Components	✓	✓*	✓

* with workarounds/limitations now – better in future releases

https://custom-elements-everywhere.com/

engage

Tools for Web Component development

Polymer Project

https://www.polymer-project.org/

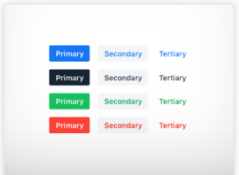
stencil

https://stenciljs.com/

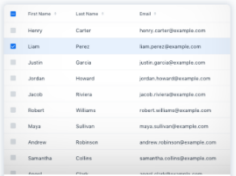
engage

Production ready **free** Web components

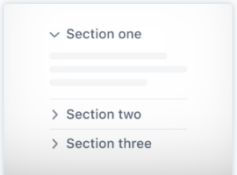
vaadin



Button
Latest v2.2.0

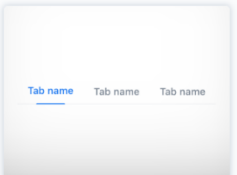


Grid
Latest v5.4.0

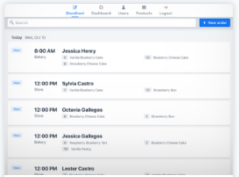


Accordion
Latest v1.0.0

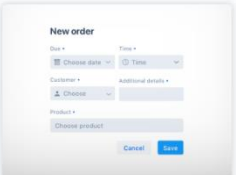
Progress Bar
Latest v1.1.0



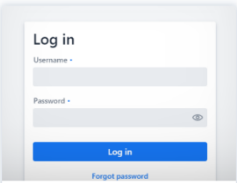
Tabs
Latest v3.0.0



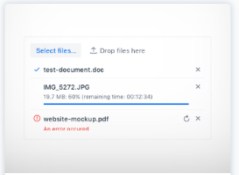
App Layout
Latest v2.0.0



Form Layout
Latest v2.1.2



Login
Latest v1.0.0



Upload
Latest v4.2.1

Date Picker
Latest v4.0.0

...

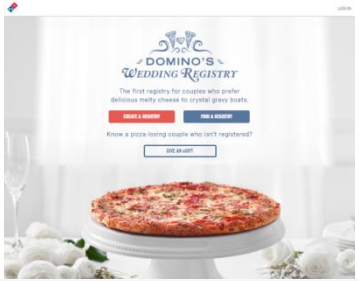
https://vaadin.com/components

Web Components

run on any platform or device

reusable, really

engage




```
▼<dom-boot>
  ▼#shadow-root (open)
    <style scope="dom-boot">:host {
      display: block;
            height: 100%;
    }</style>
   ▶<dom-loader>…</dom-loader>
   ▼<dom-app>
    ▼#shadow-root (open)
     ▶<style scope="dom-app-0">…</style>
     ▶<dom-configurable-path>…</dom-configurable-path>
     ▶<dom-user id="user">…</dom-user>
     ▶<app-location>…</app-location>
      <app-route pattern="/:page"> </app-route>
      <app-scrollpos-control id="scroll" reset></app-scrollpos-control>
     ▼<app-drawer-layout fullbleed force-narrow>
       ▶#shadow-root (open)
       ▼<app-drawer id="drawer" align="right" position="right" style="transition-duration: 200ms; touch-action: pan-y;">
         ▶#shadow-root (open)
         ▶<dom-nav-drawer selected="/">…</dom-nav-drawer>
        </app-drawer>
       ▶<app-header-layout fullbleed id="headerLayout">…</app-header-layout>
      </app-drawer-layout>
    </dom-app>
   ▶<template is="dom-if">…</template>
</dom-boot>
```

engage

Suggestion:

HCL or community creates a cool set of **Web Components**

- <domino-view>
- <domino-login>
- <domino-form>
- <domino-datetime-field>
- <domino-folder>
- <domino-richtext-field>
- <domino-agent>
- <domino-what-ever>
- <domino-for-ever>

engage

Questions?

engage

Thank you!