conclurer

Web Components

The Lego bricks of web developers

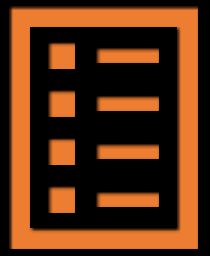
Aaron Czichon

conclurer

- Aaron Czichon
- Head of Software Development
- NodeJS & Web Components

The basic problem of HTML elements

conclurer



Building a login form...

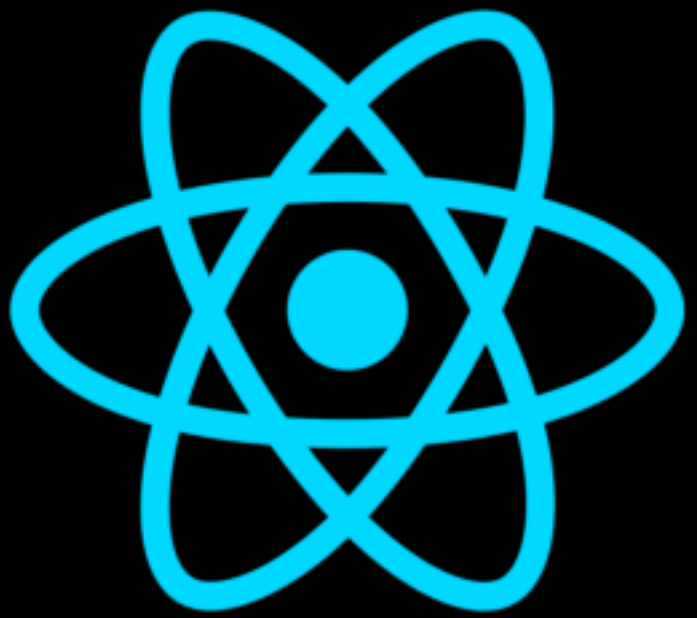
```
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Dev Days Europe 2019</title>
        <meta name="description" content="Example for Web Components">
        <meta name="author" content="Aaron Czichon">
        <link rel="stylesheet" href="./assets/edel-elements.css">
    </head>

    <body>
    <!-- CONTENT HERE  -->
    </body>
</html>
```

conclurer

```
<div class="login-card card">
        <div class="card-meta">
                <h3>Login</h3>
                <p>Please login first</p>
        </div>
        <div class="login-card-form">
                <div class="form form--grid">
                        <div class="form-item">
                                <label>Username or email</label>
                                <input placeholder="Username or email" type="text">
                        </div>
                        <div class="form-item">
                                <label>Password</label>
                                <input placeholder="Super Secret Password" type="password">
                        </div>
                        <div class="form-item">
                        <button type=“submit“ class="button button--primary">Login</button>
                        </div>
                </div
        </div>
</div>
```

Want using it in multiple applications?

conclurer



Multiple Framework Options

Downsides

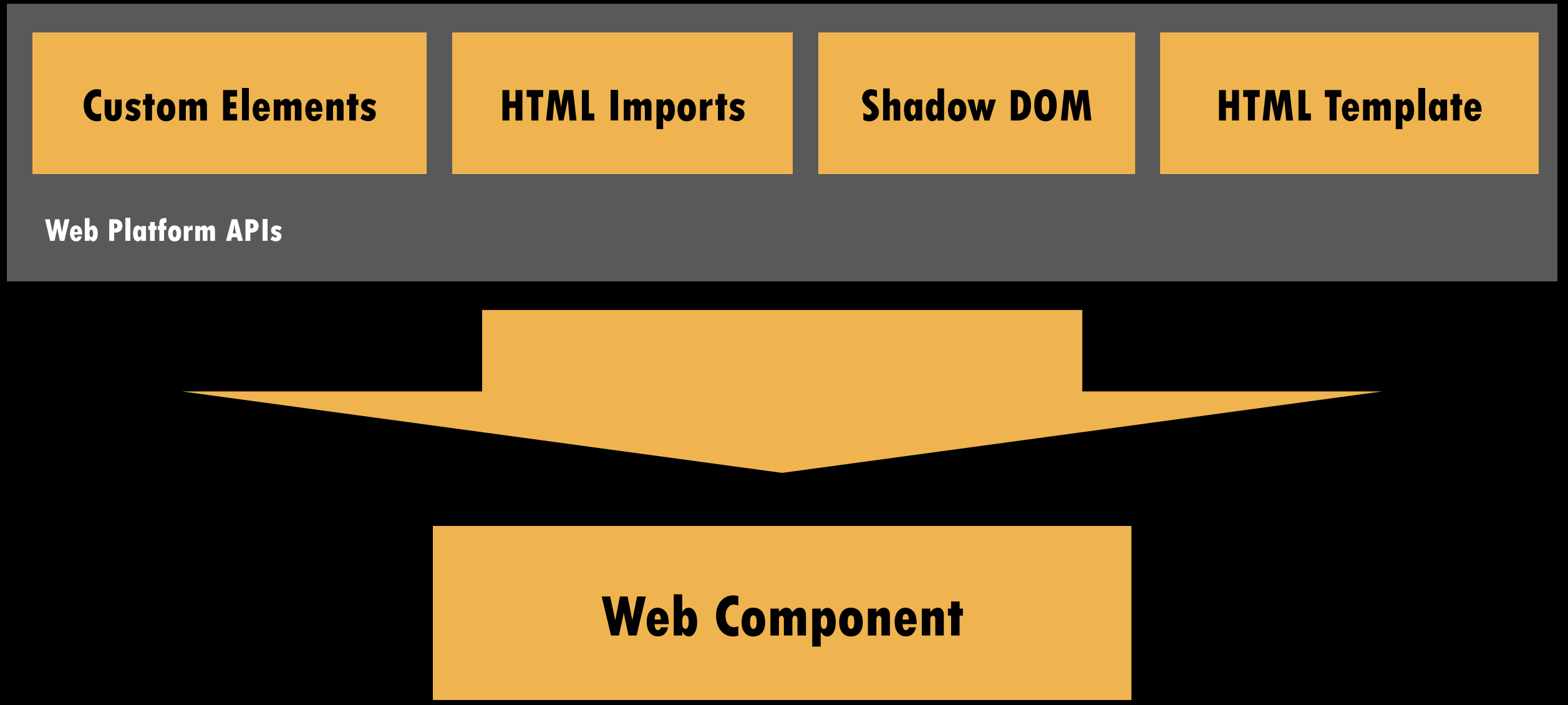
- Mostly only usable inside the framework
- (Angular has export options for web components)
- Sometimes a lot of overhead which is not needed

Web Components!

„Web components are a set of web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags“

Source: webcomponents.org

conclurer


Custom Elements
HTML Imports
Shadow DOM
HTML Template
Web Platform APIs
Web Component

conclurer

Custom Elements

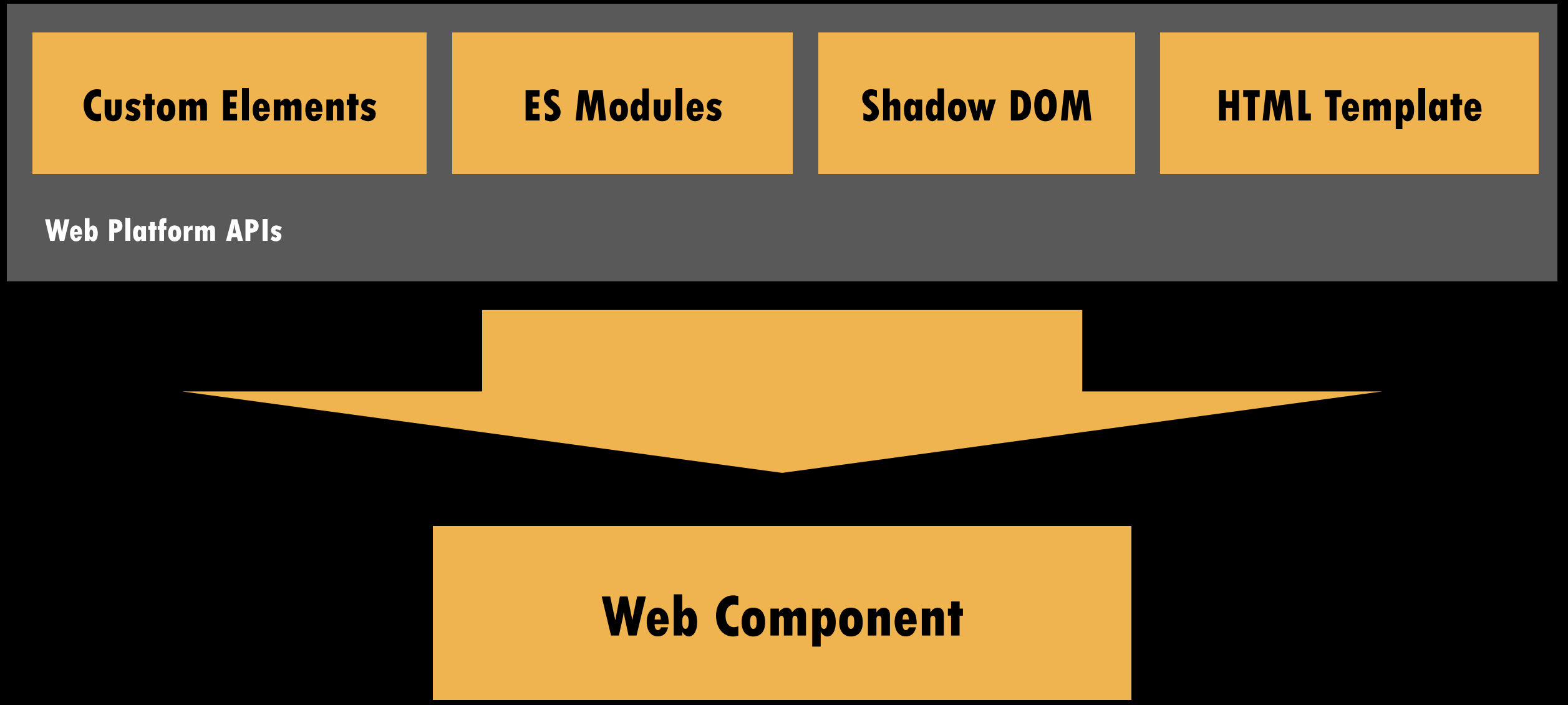
Shadow DOM

HTML Template

Web Platform APIs

Web Component

conclurer


Custom Elements
ES Modules
Shadow DOM
HTML Template
Web Platform APIs
Web Component

conclurer

Custom Element

```
customElements.define('star-wars-character', StarWarsCharacter, { extends: 'p' });
```

conclurer

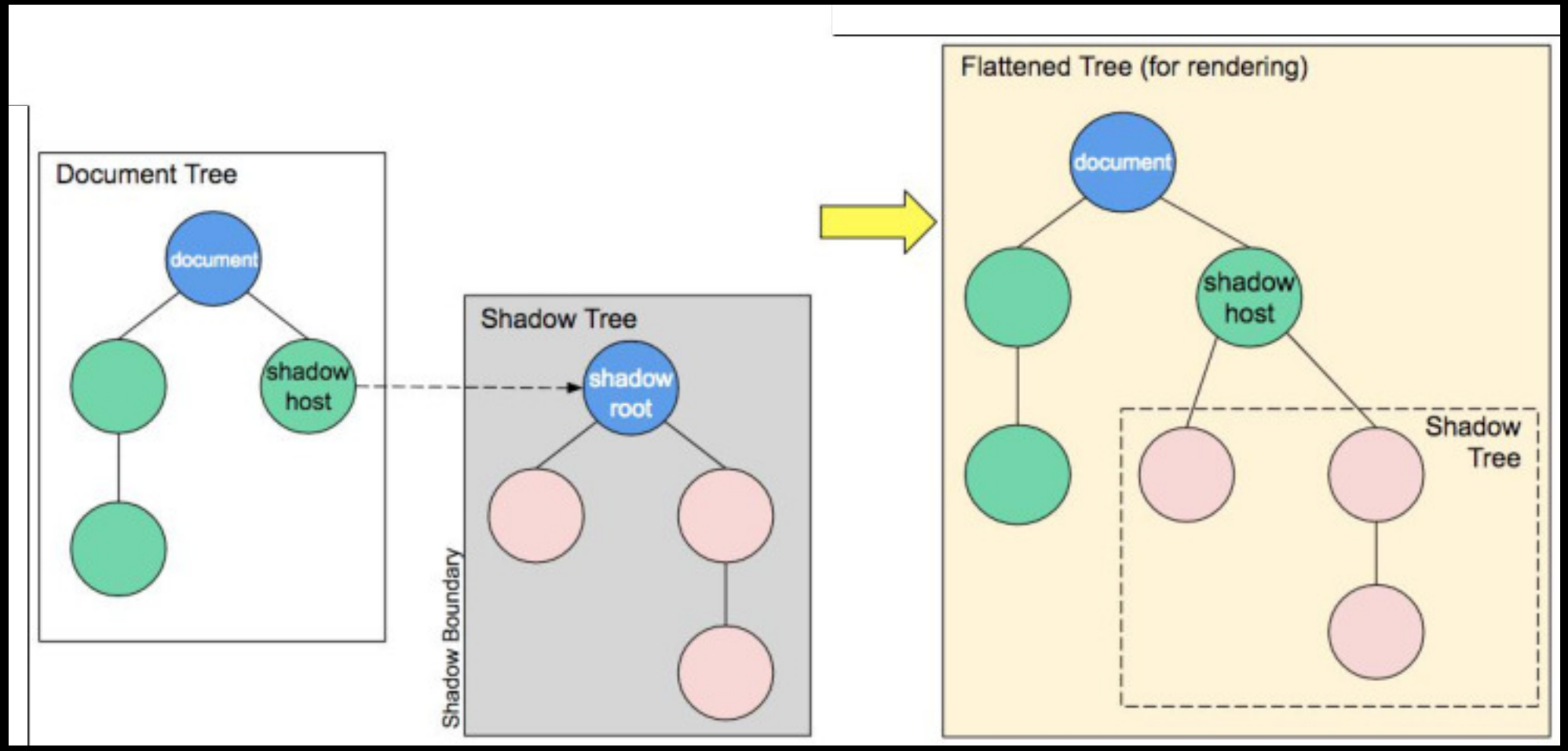
Custom Element

```
class StarWarsCharacter extends HTMLParagraphElement {
    constructor() {
      // Call super for trigger Praagraph constructor
      super();

      // Element implementation here
    }
}
```

conclurer

Shadow DOM

-> Shadowed rendering of DOM Nodes



conclurer

Shadow DOM

-> Attach shadow to existing element (shadow host)

```
let shadow = elementRef.attachShadow({ mode: 'open' });
```

conclurer

Shadow DOM

-> Create new element and attach it to the shadowed element

```
const characterName = document.createElement('h1');
shadow.appendChild(characterName);
```

conclurer

HTML Imports

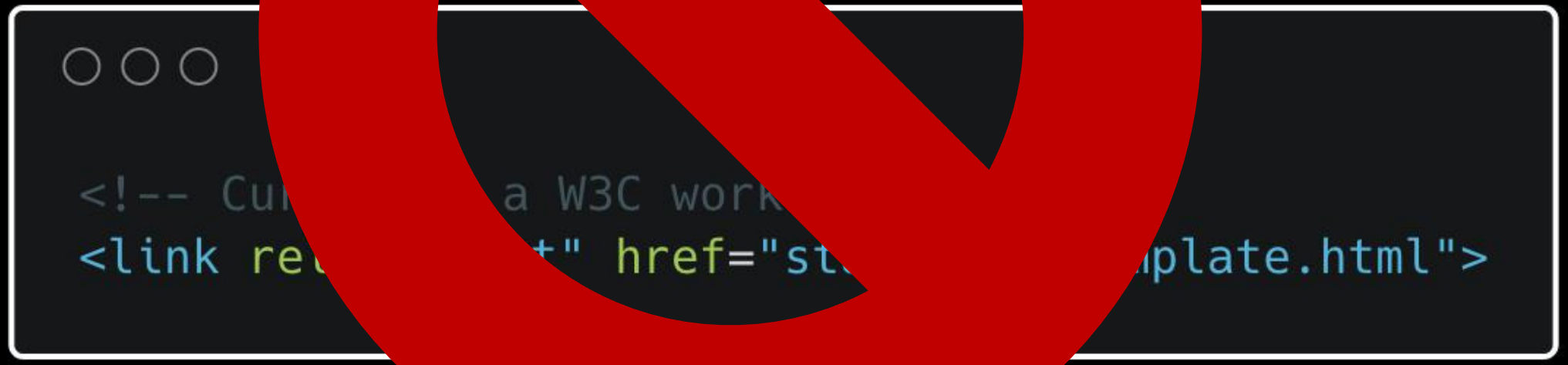
-> Importing HTMLmarkup like script files (W3C Working Draft)

```
<!-- Currently a W3C working draft -->
<link rel="import" href="star-wars-template.html">
```

conclurer

HTML Imports

-> Importing HT… …e script files (… …Draft)



conclurer

ES Module

-> Using your module inside your website/application

```
<script type="module" src="star-wars-character.js"></script>
```

conclurer

HTML Templates

-> Render templates on specific location with JS

```
<!-- Define HTML Template -->
<template id="sw-character-component">
  <h1>Yoda</h1>
</template>
```

conclurer

HTML Templates

-> Render templates on specific location with JS

```
// Render with JS
const template = document.getElementById('sw-character-component');
const clone = document.importNode(template.content, true);
document.body.appendChild(clone);
```

conclurer

Combine everything!

-> Create Javascript component

```
// component.js
class SwCharacter extends HTMLElement {
    constructor() {
        super();
        const template = document.querySelector('#star-wars-template');
        const clone = document.importNode(template.content, true);
        const shadowRoot = this.attachShadow({mode: 'open'});
        shadowRoot.appendChild(clone);
    }
}

customElements.define('sw-character-component', SwCharacter);
```

conclurer

Combine everything!

-> Create Markup

```
<!-- index.html -->
<head>
    <script src="component.js"></script>
</head>

<body>
    <sw-character-component></sw-character-component>

    <template id="star-wars-template">
        <style> /* CSS HERE */</style>
        <h1>Yoda</h1>
    </template>
</body>
```

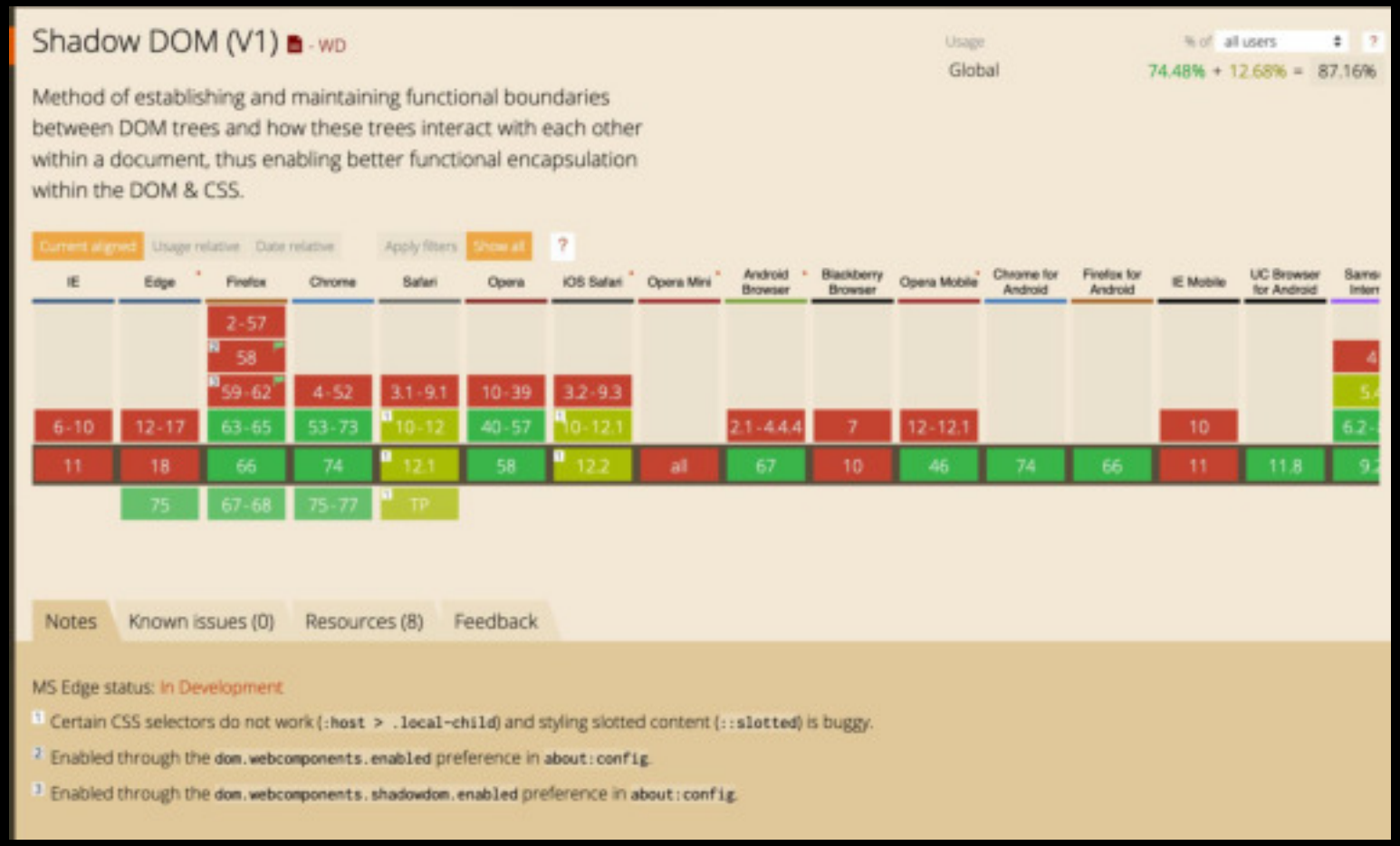
conclurer

Let‘s talk about browser support…

Support – HTML Templates

Support – Custom Elements

Support – Shadow DOM



Support – HTML Imports

conclurer

Use Polyfills

- Add support with Polyfills
- This extends browsers with new features
- Downside: Needs more bandwith (larger websites)

- -> Webcomponents.js on Github
  https://github.com/webcomponents/webcomponentsjs

conclurer

Easier development with libraries

POLYMER 2

STENCILJS

SKATEJS (WITH PREACT OR LIT-HTML)

conclurer

Web component

```
class TodoItem extends HTMLElement {
    constructor() {
        super();
        this._root = this.attachShadow({ 'mode': 'open' });
        this._checked = false;
        this._text = '';
    }
    connectedCallback() {
        this._root.innerHTML = `
            <style></style>
            <li class="item">
                <input type="checkbox">
                <label></label>
                <button class="destroy">x</button>
            </li>
        `;
        this.$item = this._root.querySelector('.item');
        this.$removeButton = this._root.querySelector('.destroy');
        this.$text = this._root.querySelector('label');
        this.$checkbox = this._root.querySelector('input');
        this.$removeButton.addEventListener('click', (e) => {
            e.preventDefault();
            this.dispatchEvent(new CustomEvent('onRemove', { detail:
            this.index }));
        });
        this.$checkbox.addEventListener('click', (e) => {
            e.preventDefault();
            this.dispatchEvent(new CustomEvent('onToggle', { detail:
            this.index }));
        });
        this._render();
    }
    disconnectedCallback() {}
    static get observedAttributes() {
        return ['text'];
    }
    attributeChangedCallback(name, oldValue, newValue) {
        this._text = newValue;
    }
    set index(value) {
        this._index = value;
    }
    get index() {
        return this._index;
    }
    set checked(value) {
        this._checked = Boolean(value);
    }
    get checked() {
        return this.hasAttribute('checked');
    }
    _render() {
        if (!this.$item) return;
        this.$text.textContent = this._text;
        if (this._checked) {
            this.$item.classList.add('completed');
            this.$checkbox.setAttribute('checked', '');
        } else {
            this.$item.classList.remove('completed');
            this.$checkbox.removeAttribute('checked');
        }

    }
}

window.customElements.define('todo-item', TodoItem);
```

Polymer 2

```
<link rel="import"
href="../../bower_components/polymer/polymer-element.html">

<dom-module id="todo-item">
  <template>
    <style>
    </style>
    <li class$="item [[isCompleted(checked)]]">
      <input type="checkbox" value="{{checked}}" checked="{
      {checked::change}}">
      <label>{{text}}</label>
      <button class="destroy" on-click="handleOnRemove">x</button>
    </li>
  </template>

  <script>
    class TodoItem extends Polymer.Element {
      static get is() { return 'todo-item'; }
      static get properties() {
        return {
          checked: {
            type: Boolean,
            value: false
          },
          index: {
            type: Number,
          },
          text: {
            type: String,
            value: ''
          }
        };
      }
      handleOnRemove(e) {
        this.dispatchEvent(new CustomEvent('remove', { detail: this.index }
        ));
      }
      handleOnChecked(e) {
        this.dispatchEvent(new CustomEvent('toggle', { detail: this.index }
        ));
      }
      isCompleted(completed) {
        return completed ? 'completed' : '';
      }
    }

    window.customElements.define(TodoItem.is, TodoItem);
  </script>
</dom-module>
```

Angular Elements

```
import { Component, EventEmitter, Input, Output, ViewEncapsulation }
from '@angular/core';

@Component({
    selector: 'todo-item',
    template: `
    <li class="item" [class.completed]="checked">
        <input type="checkbox" [checked]="checked" (change)
        ="handleOnChecked()">
        <label>{{text}}</label>
        <button class="destroy" (click)="handleOnRemove()">x</button>
    </li>
    `,
    styles: [`
    `],
    encapsulation: ViewEncapsulation.Native
})
export class TodoItem {
    @Input() checked: boolean;
    @Input() text: string;
    @Input() index: number;
    @Output() onTodoItemChecked = new EventEmitter<number>();
    @Output() onTodoItemRemove = new EventEmitter<number>();

    handleOnRemove = () => this.onTodoItemRemove.emit(this.index);
    handleOnChecked = () => this.onTodoItemChecked.emit(this.index);
}
```

Vue-wrapper

```
<template>
  <li :class="['item', {'completed':checked}]">
    <input type="checkbox" :checked="checked" @click="handleOnToggle" >
    <label>{{text}}</label>
    <button class="destroy" @click="handleOnRemove">x</button>
  </li>
</template>

<script>
module.exports = {
  name: 'TodoItem',
  props: ['index', 'text', 'checked'],
  methods: {
    handleOnRemove() {
      this.$emit('onremove', this.index);
    },
    handleOnToggle() {
      this.$emit('ontoggle', this.index);
    },
  },
};
</script>
```

StencilJS

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

@Component({
  tag: 'todo-item',
  styleUrl: 'todo-item.scss',
  shadow: true,
})
export class TodoItem {
  @Prop() checked: boolean;
  @Prop() text: string;
  @Prop() index: number;
  @Event() onTodoItemChecked: EventEmitter;
  @Event() onTodoItemRemove: EventEmitter;

  handleOnRemove = () => this.onTodoItemRemove.emit(this.index);
  handleOnChecked = () => this.onTodoItemChecked.emit(this.index);

  render() {
    return (
      <li class={this.checked ? 'completed' : ''}>
        <input type="checkbox" checked={this.checked} onChange=
        {this.handleOnChecked} />
        <label>{this.text}</label>
        <button onClick={this.handleOnRemove}>x</button>
      </li>
    );
  }
}
```

SkateJS + Preact

```
// @jsx h
import { props } from "skatejs/dist/esnext";
import { h } from "preact";
import { Component } from "./util"

export default class extends Component {
  static events = ["check", "remove"];
  static props = {
    checked: props.boolean,
    index: props.number
  };

  handleCheck = e => {
    this.onCheck({ index: this.index, value: e.target.checked });
  };
  handleRemove = () => {
    this.onRemove({ index: this.index });
  };

  render({ checked, handleCheck, handleRemove }) {
    return (
      <div>
        <style>{`
        `}</style>
        <li class={checked ? "completed" : ""}>
          <input type="checkbox" checked={checked} onChange={handleCheck} />
          <label>
            <slot />
          </label>
          <button onClick={handleRemove}>x</button>
        </li>
      </div>
    );
  }
}
```

SkateJS + lit-html

```
import { props } from "skatejs/dist/esnext";
import { html } from "lit-html/lib/lit-extended";
import { Component } from "./util";

export default class extends Component {
  static events = ["check", "remove"];
  static props = {
    checked: props.boolean,
    index: props.number
  };

  handleCheck = e => {
    this.onCheck({ index: this.index, value: e.target.checked });
  };
  handleRemove = () => {
    this.onRemove({ index: this.index });
  };

  render({ checked, handleCheck, handleRemove }) {
    return html`
      <style></style>
      <li class="${checked ? "completed" : ""}">
        <input type="checkbox" checked="${checked}" on-change="${handleCh
        <label>
          <slot></slot>
        </label>
        <button on-click="${handleRemove}">x</button>
      </li>
    `;
  }
}
```

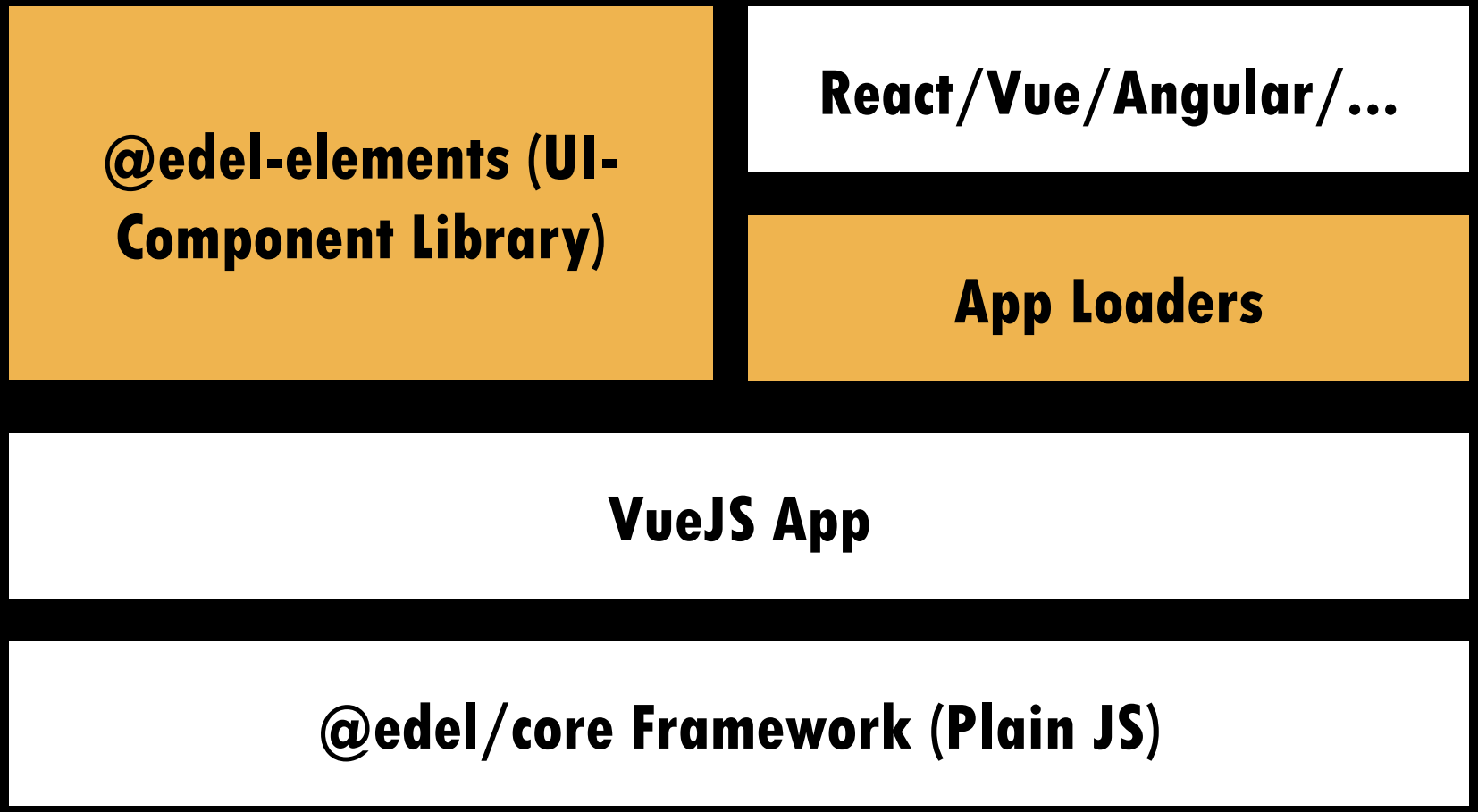
Building the login with Web Components

conclurer

How we use it for

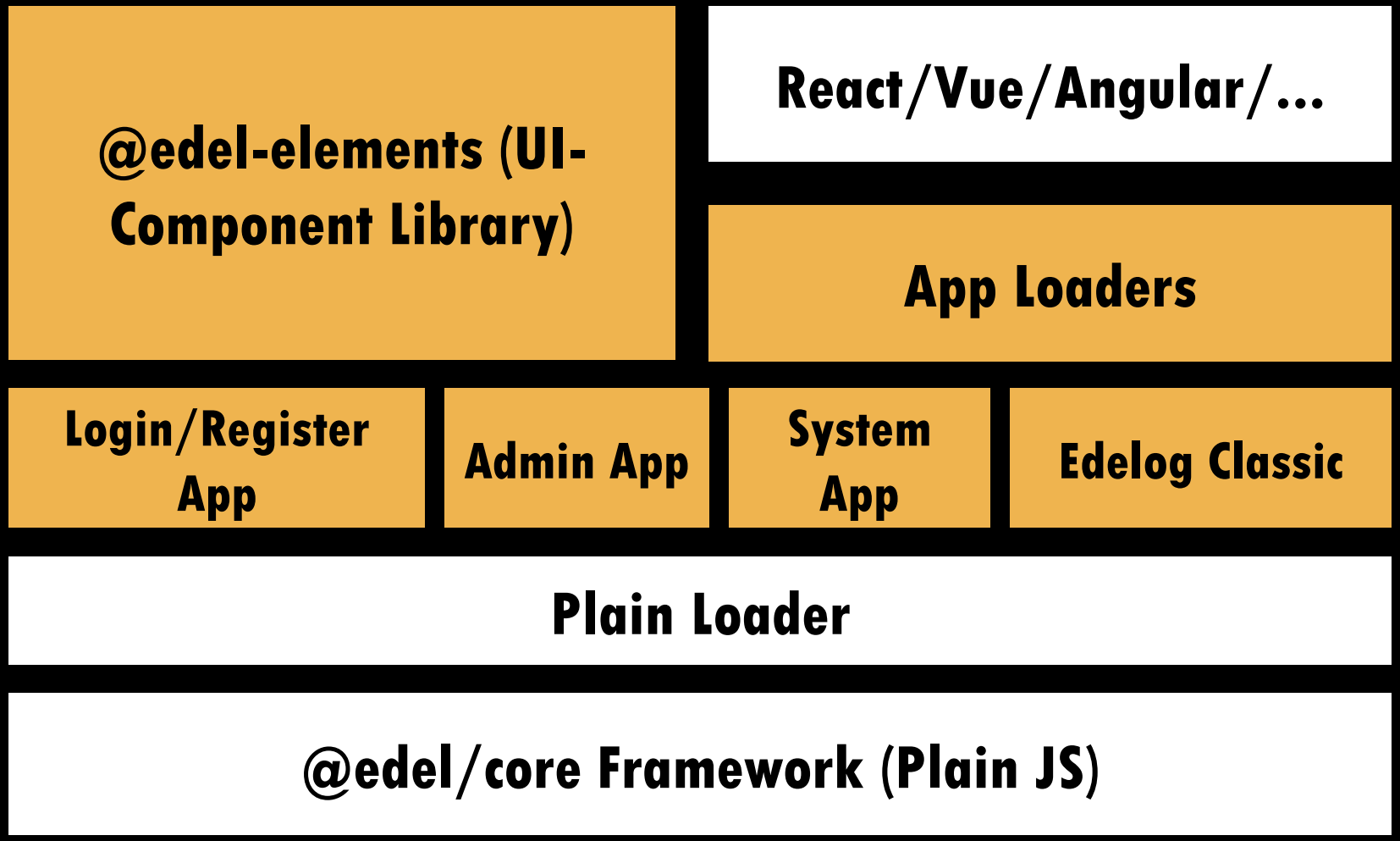
conclurer

How we use it



conclurer

How we use it



conclurer

Edelog & Edel-Elements

- www.edelog.com
- Publishing Edelog and Edel-Elements Open Source @ End 2019

conclurer

Thank you!

@lnoverse

online@aaronczichon.de