WebComponents

框架与类库

主题

Lit（Polymer3.0）

Stencil

Web Components

框架

知名框架

Lit (Google)

Fast (Microsoft)

LWC (Salesforce)

Stencil

Cybernetically enhanced web apps

Lit

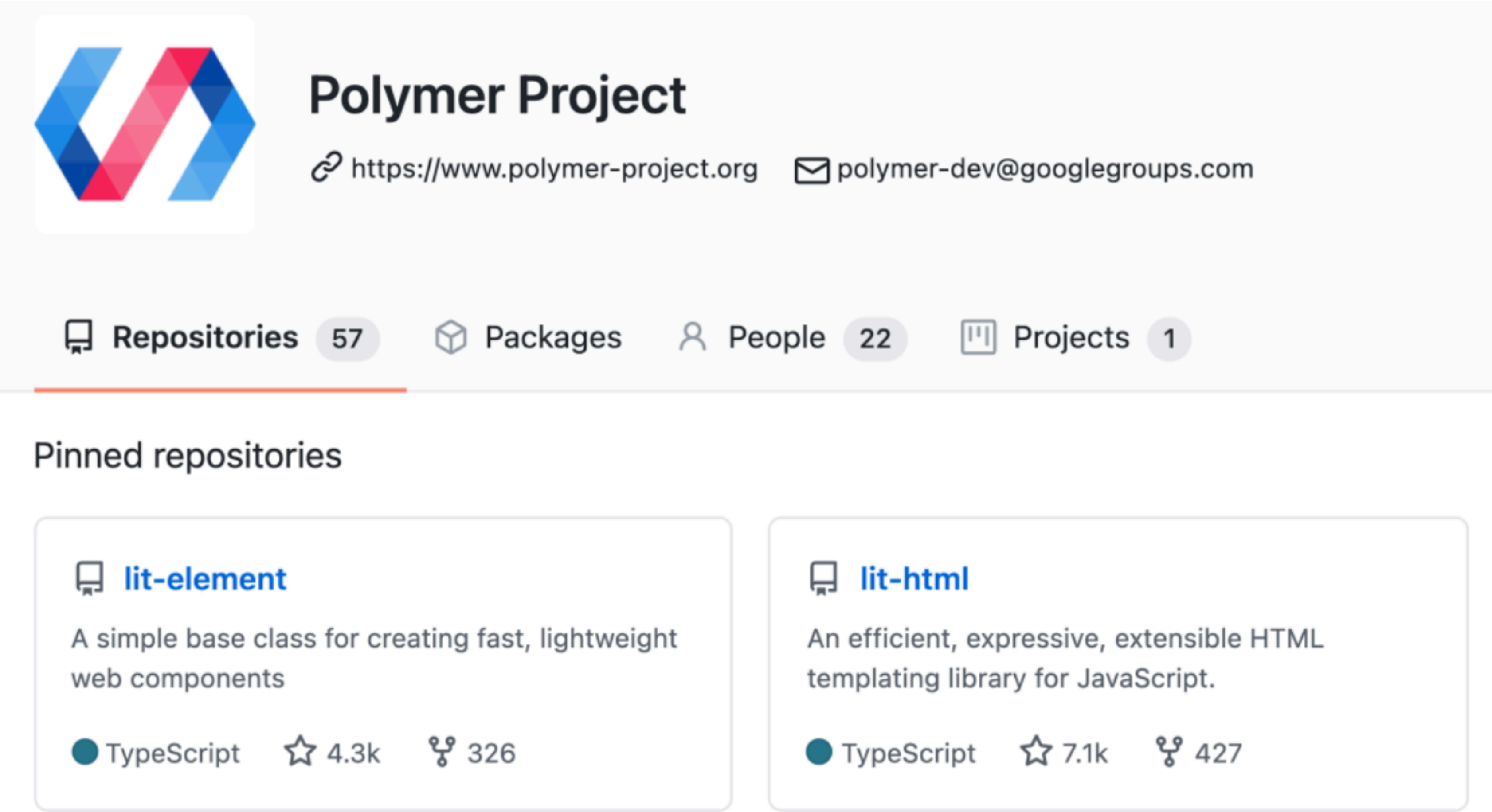
关于Polymer

Polymer是谷歌开发的一个开源项目，目的是用来构造Web组件。Polymer提供了一种为Web应用程序创建自定义元素的简单方法。

Polymer提供了组件库，称为Polymer元素，可用作Web应用程序的即用型构建基块，或用作扩展自定义元素的基础元素。

YouTube

Polymer的局限性



Polymer Project

https://www.polymer-project.org polymer-dev@googlegroups.com

Repositories 57 Packages People 22 Projects 1

Pinned repositories

lit-element

A simple base class for creating fast, lightweight web components

● TypeScript ☆ 4.3k 326

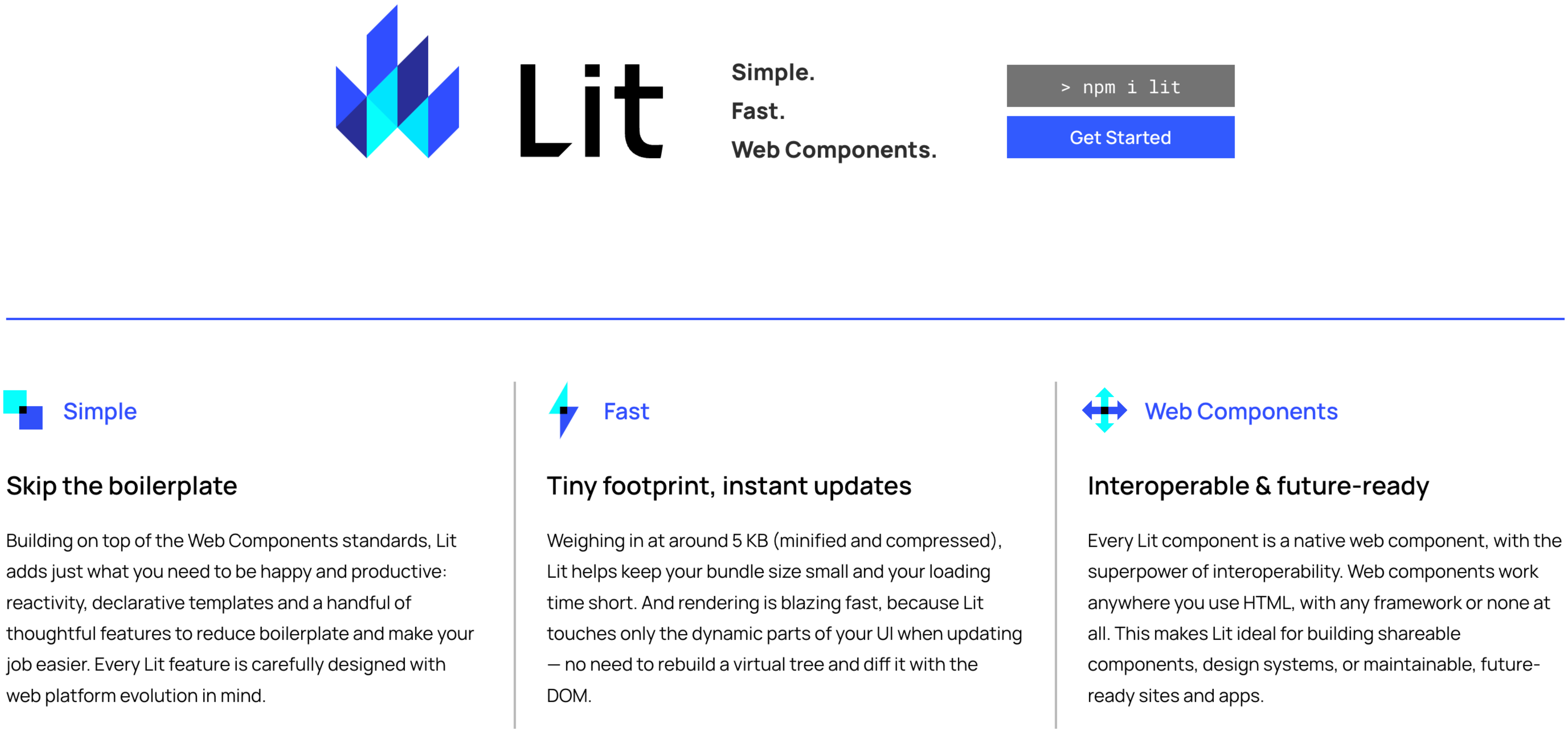
lit-html

An efficient, expressive, extensible HTML templating library for JavaScript.

● TypeScript ☆ 7.1k 427

关于Lit

https://lit.dev/



Simple.
Fast.
Web Components.

```
> npm i lit
```

Simple

Skip the boilerplate

Building on top of the Web Components standards, Lit adds just what you need to be happy and productive: reactivity, declarative templates and a handful of thoughtful features to reduce boilerplate and make your job easier. Every Lit feature is carefully designed with web platform evolution in mind.

Fast

Tiny footprint, instant updates

Weighing in at around 5 KB (minified and compressed), Lit helps keep your bundle size small and your loading time short. And rendering is blazing fast, because Lit touches only the dynamic parts of your UI when updating — no need to rebuild a virtual tree and diff it with the DOM.

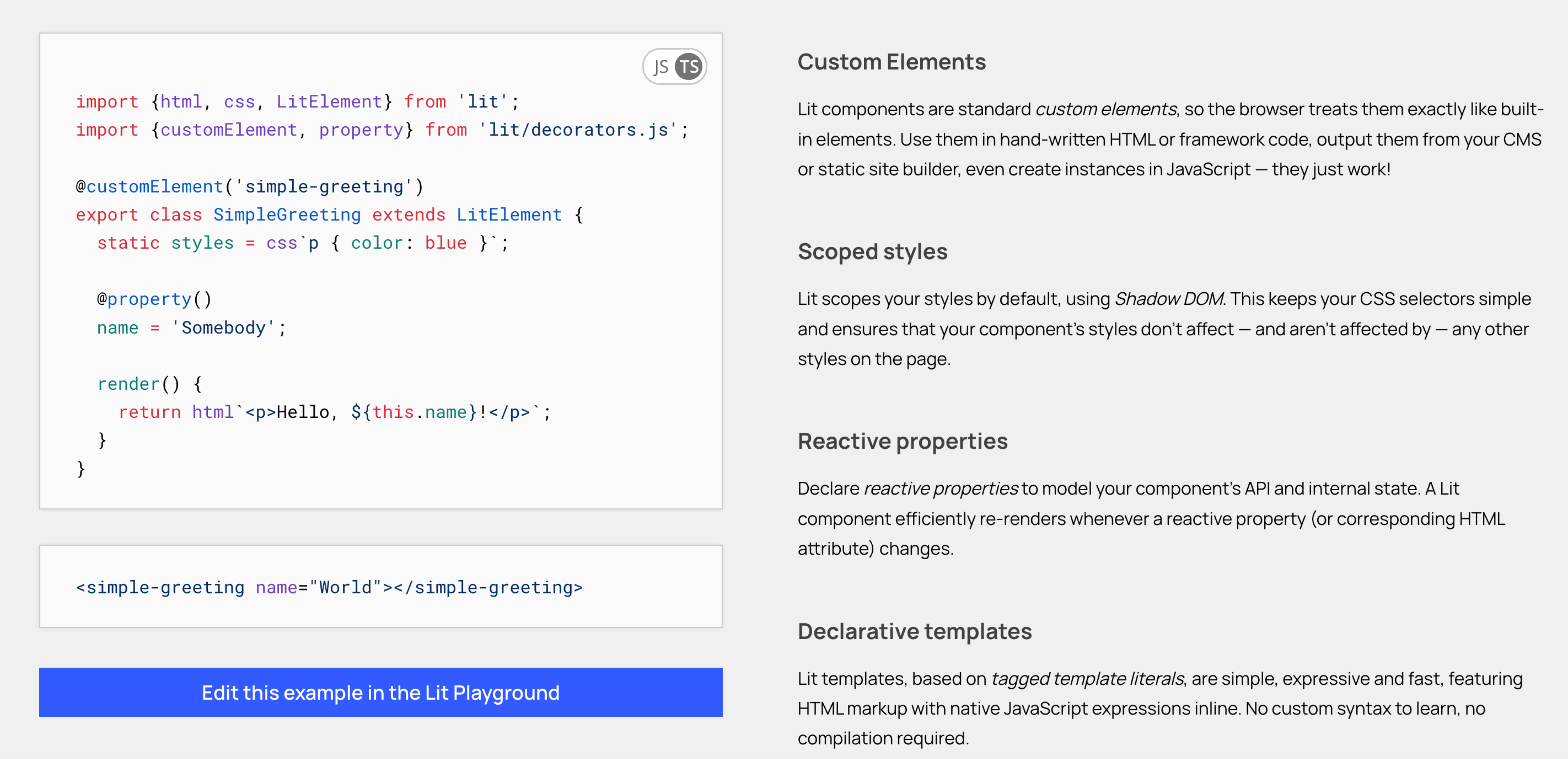
Web Components

Interoperable & future-ready

Every Lit component is a native web component, with the superpower of interoperability. Web components work anywhere you use HTML, with any framework or none at all. This makes Lit ideal for building shareable components, design systems, or maintainable, future-ready sites and apps.

关于Lit

https://lit.dev/



```
import {html, css, LitElement} from 'lit';
import {customElement, property} from 'lit/decorators.js';

@customElement('simple-greeting')
export class SimpleGreeting extends LitElement {
  static styles = css`p { color: blue }`;

  @property()
  name = 'Somebody';

  render() {
    return html`<p>Hello, ${this.name}!</p>`;
  }
}
```

```
<simple-greeting name="World"></simple-greeting>
```

Custom Elements

Lit components are standard *custom elements*, so the browser treats them exactly like built-in elements. Use them in hand-written HTML or framework code, output them from your CMS or static site builder, even create instances in JavaScript — they just work!

Scoped styles

Lit scopes your styles by default, using *Shadow DOM*. This keeps your CSS selectors simple and ensures that your component's styles don't affect — and aren't affected by — any other styles on the page.

Reactive properties

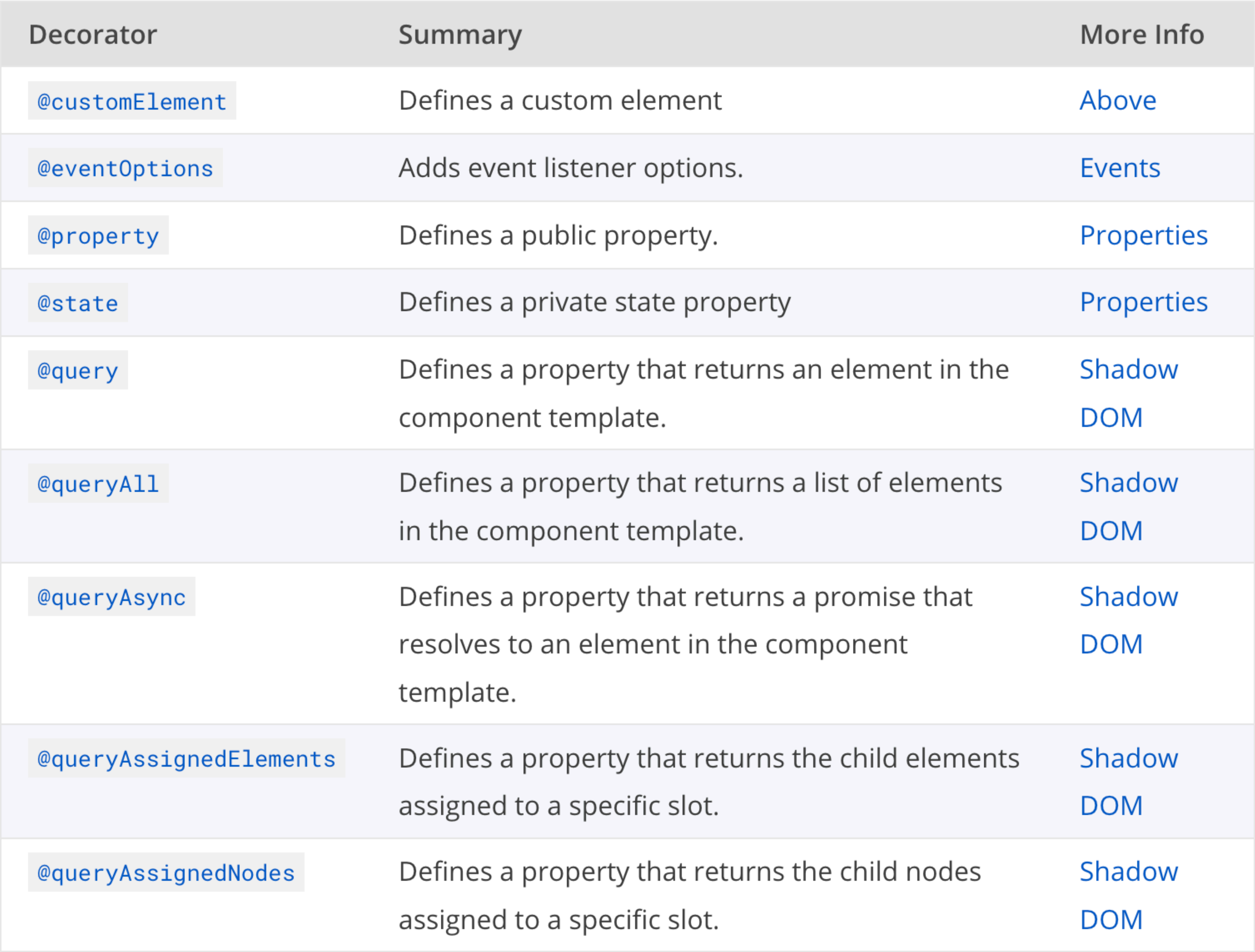
Declare *reactive properties* to model your component's API and internal state. A Lit component efficiently re-renders whenever a reactive property (or corresponding HTML attribute) changes.

Declarative templates

Lit templates, based on *tagged template literals*, are simple, expressive and fast, featuring HTML markup with native JavaScript expressions inline. No custom syntax to learn, no compilation required.

关于Lit

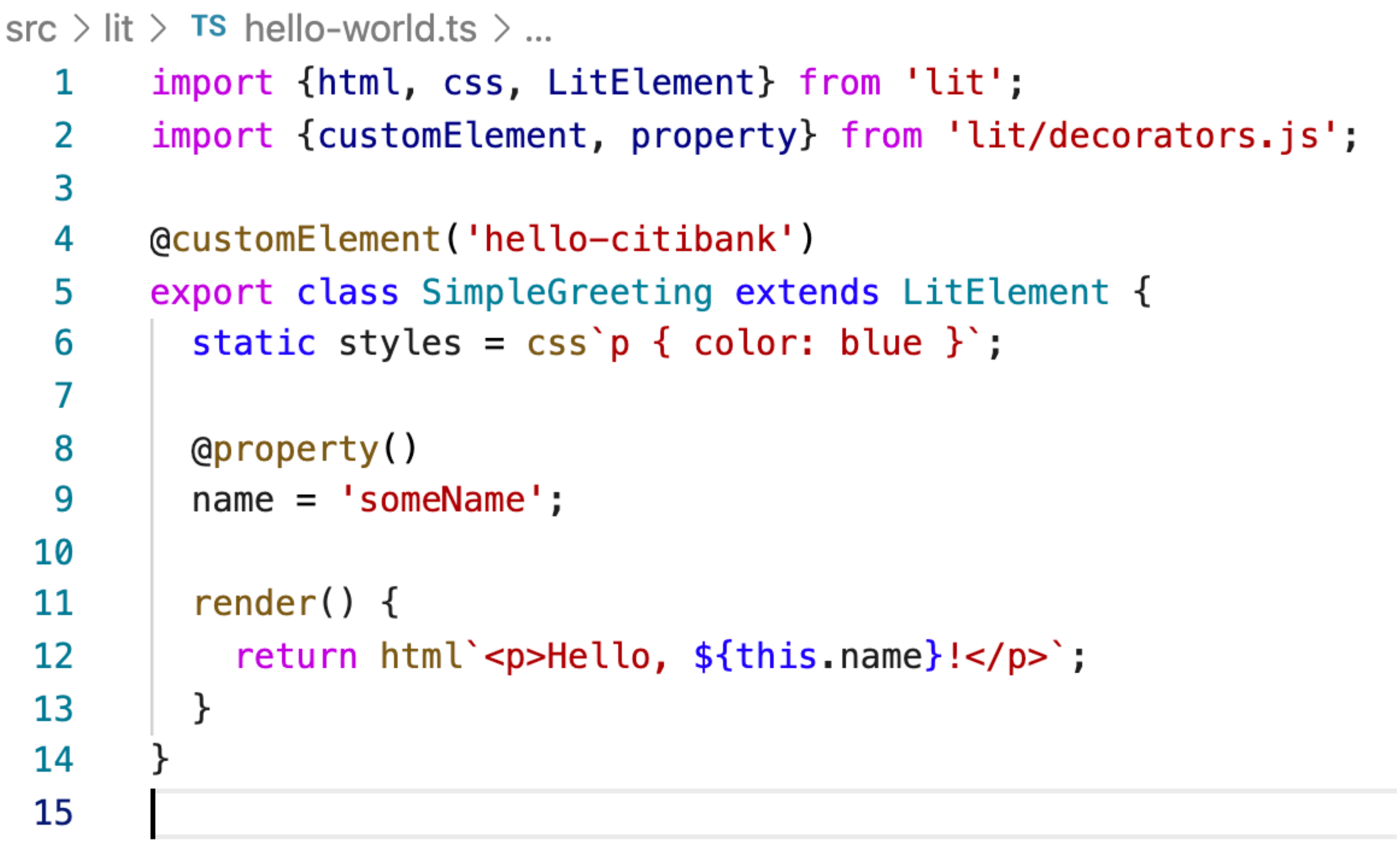
https://lit.dev/docs/components/decorators/



Decorator	Summary	More Info
`@customElement`	Defines a custom element	Above
`@eventOptions`	Adds event listener options.	Events
`@property`	Defines a public property.	Properties
`@state`	Defines a private state property	Properties
`@query`	Defines a property that returns an element in the component template.	Shadow DOM
`@queryAll`	Defines a property that returns a list of elements in the component template.	Shadow DOM
`@queryAsync`	Defines a property that returns a promise that resolves to an element in the component template.	Shadow DOM
`@queryAssignedElements`	Defines a property that returns the child elements assigned to a specific slot.	Shadow DOM
`@queryAssignedNodes`	Defines a property that returns the child nodes assigned to a specific slot.	Shadow DOM

Hello world

VS Code 演示



```
import {html, css, LitElement} from 'lit';
import {customElement, property} from 'lit/decorators.js';

@customElement('hello-citibank')
export class SimpleGreeting extends LitElement {
  static styles = css`p { color: blue }`;

  @property()
  name = 'someName';

  render() {
    return html`<p>Hello, ${this.name}!</p>`;
  }
}

```

样式与事件的写法

[VS Code 演示]

模板字符串处理函数

function Tag(strings, personExp，ageExp,…)

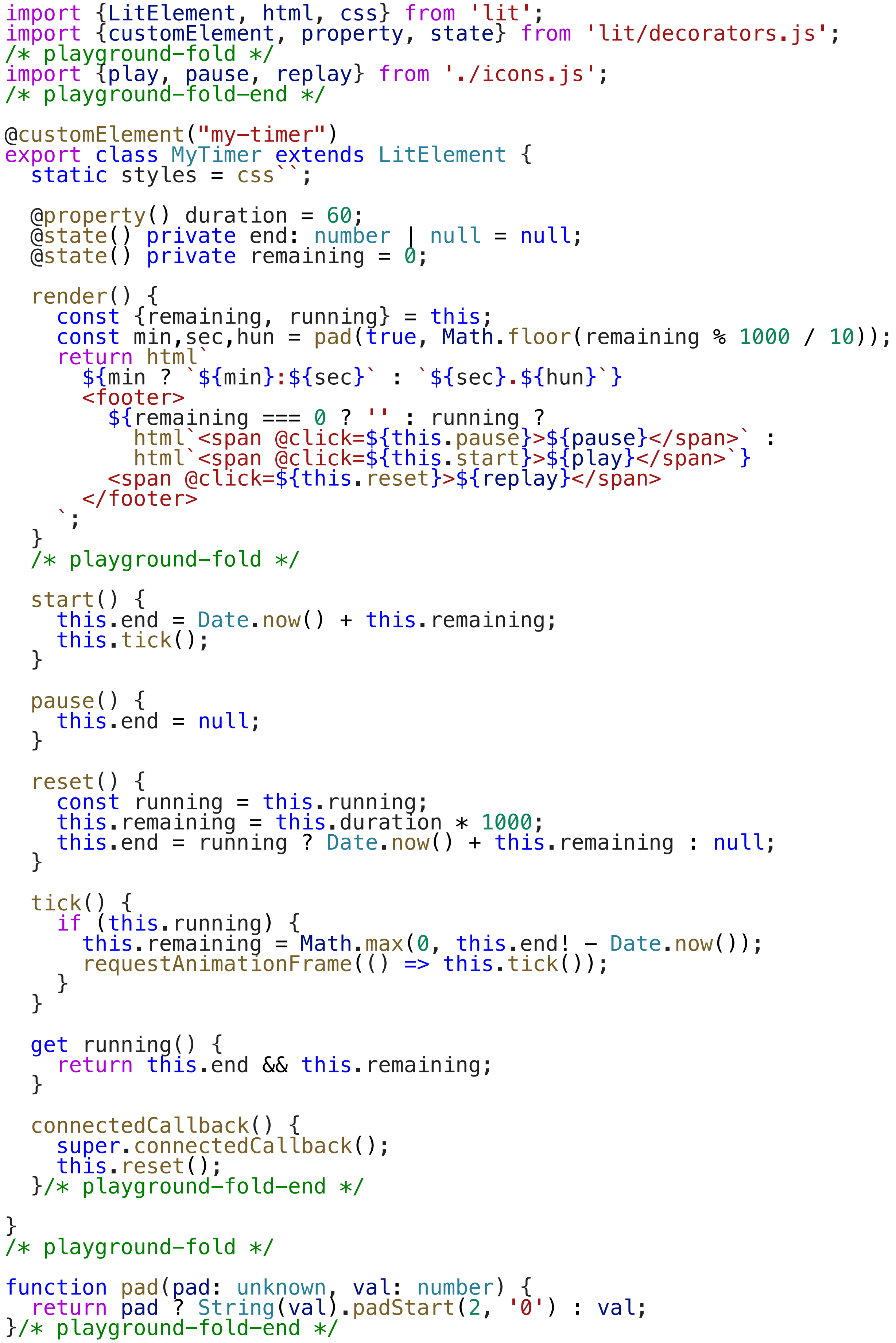
使用可构造样式表：

```
<style id="sourcestyle">
  h1 {
    color: green
  }
</style>

<my-element></my-element>

<script>
  customElements.define('my-element', class extends HTMLElement {
    constructor() {
      const sheet = new CSSStyleSheet();
      sheet.replace(document.getElementById("sourcestyle").innerHTML);

      super() // returns this/element scope
        .attachShadow({ mode: 'open' }) // both sets and returns
this.shadowRoot
        .innerHTML = `<h1>Hello World!</h1>`;

      this.shadowRoot.adoptedStyleSheets = [sheet];
    }
  })
</script>
```

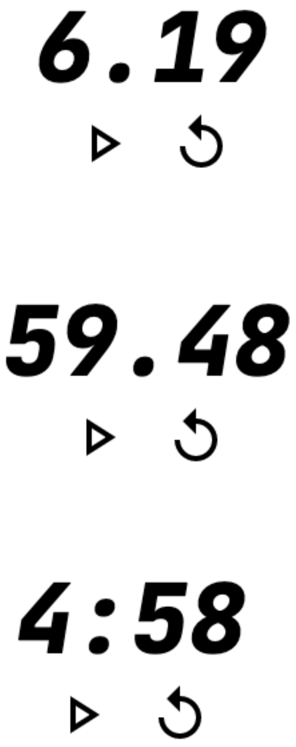
自定义对话框的实现

VS Code 演示

更多Demos举例



```
import {LitElement, html, css} from 'lit';
import {customElement, property, state} from 'lit/decorators.js';
/* playground-fold */
import {play, pause, replay} from './icons.js';
/* playground-fold-end */

@customElement("my-timer")
export class MyTimer extends LitElement {
  static styles = css``;

  @property() duration = 60;
  @state() private end: number | null = null;
  @state() private remaining = 0;

  render() {
    const {remaining, running} = this;
    const min,sec,hun = pad(true, Math.floor(remaining % 1000 / 10));
    return html`
      ${min ? `${min}:${sec}` : `${sec}.${hun}`}
      <footer>
        ${remaining === 0 ? '' : running ?
          html`<span @click=${this.pause}>${pause}</span>` :
          html`<span @click=${this.start}>${play}</span>`}
        <span @click=${this.reset}>${replay}</span>
      </footer>
    `;
  }
  /* playground-fold */

  start() {
    this.end = Date.now() + this.remaining;
    this.tick();
  }

  pause() {
    this.end = null;
  }

  reset() {
    const running = this.running;
    this.remaining = this.duration * 1000;
    this.end = running ? Date.now() + this.remaining : null;
  }

  tick() {
    if (this.running) {
      this.remaining = Math.max(0, this.end! - Date.now());
      requestAnimationFrame(() => this.tick());
    }
  }

  get running() {
    return this.end && this.remaining;
  }

  connectedCallback() {
    super.connectedCallback();
    this.reset();
  }/* playground-fold-end */

}
/* playground-fold */

function pad(pad: unknown, val: number) {
  return pad ? String(val).padStart(2, '0') : val;
}/* playground-fold-end */
```



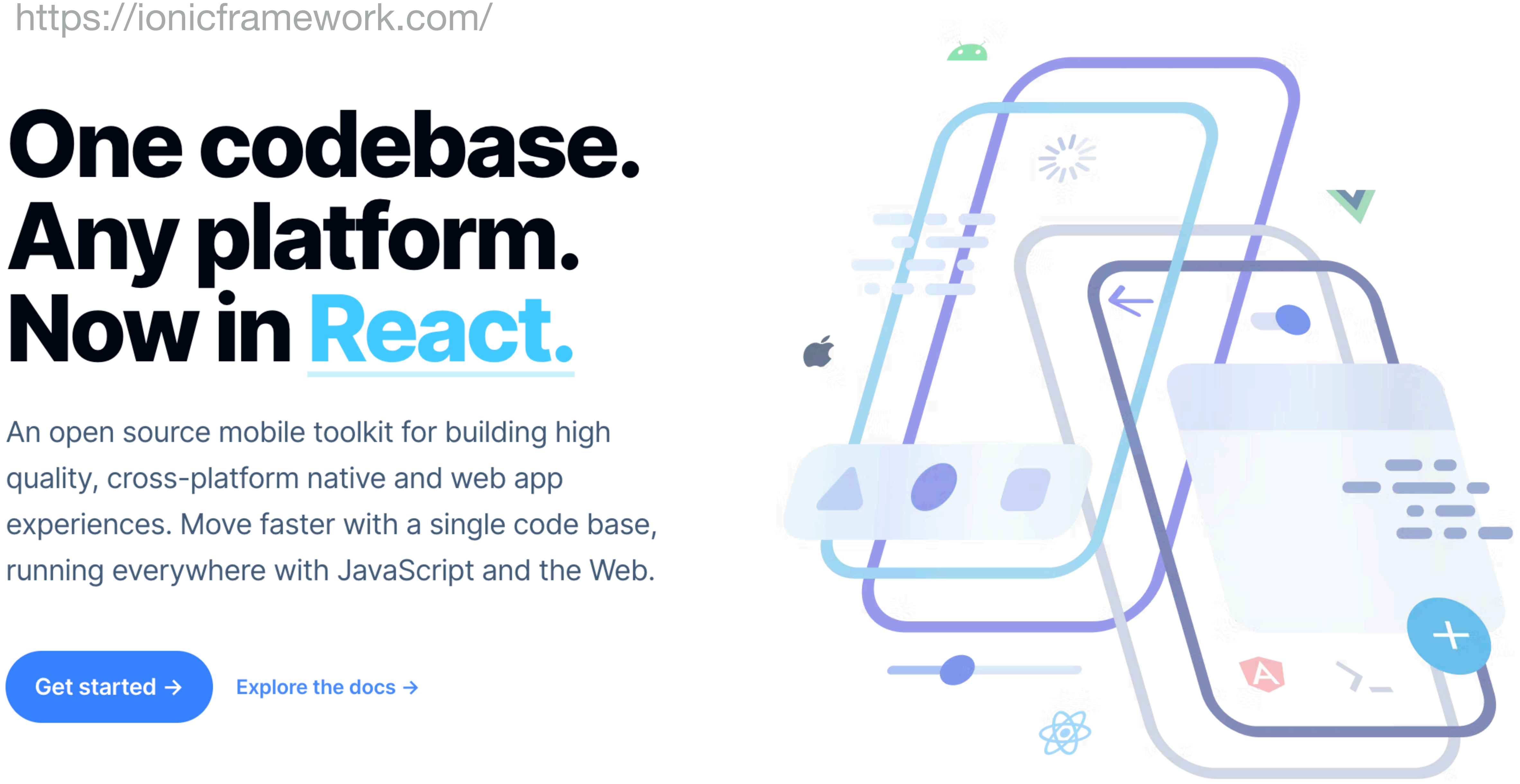
https://lit.dev/docs/

ionic

stencil

关于Ionic

https://ionicframework.com/



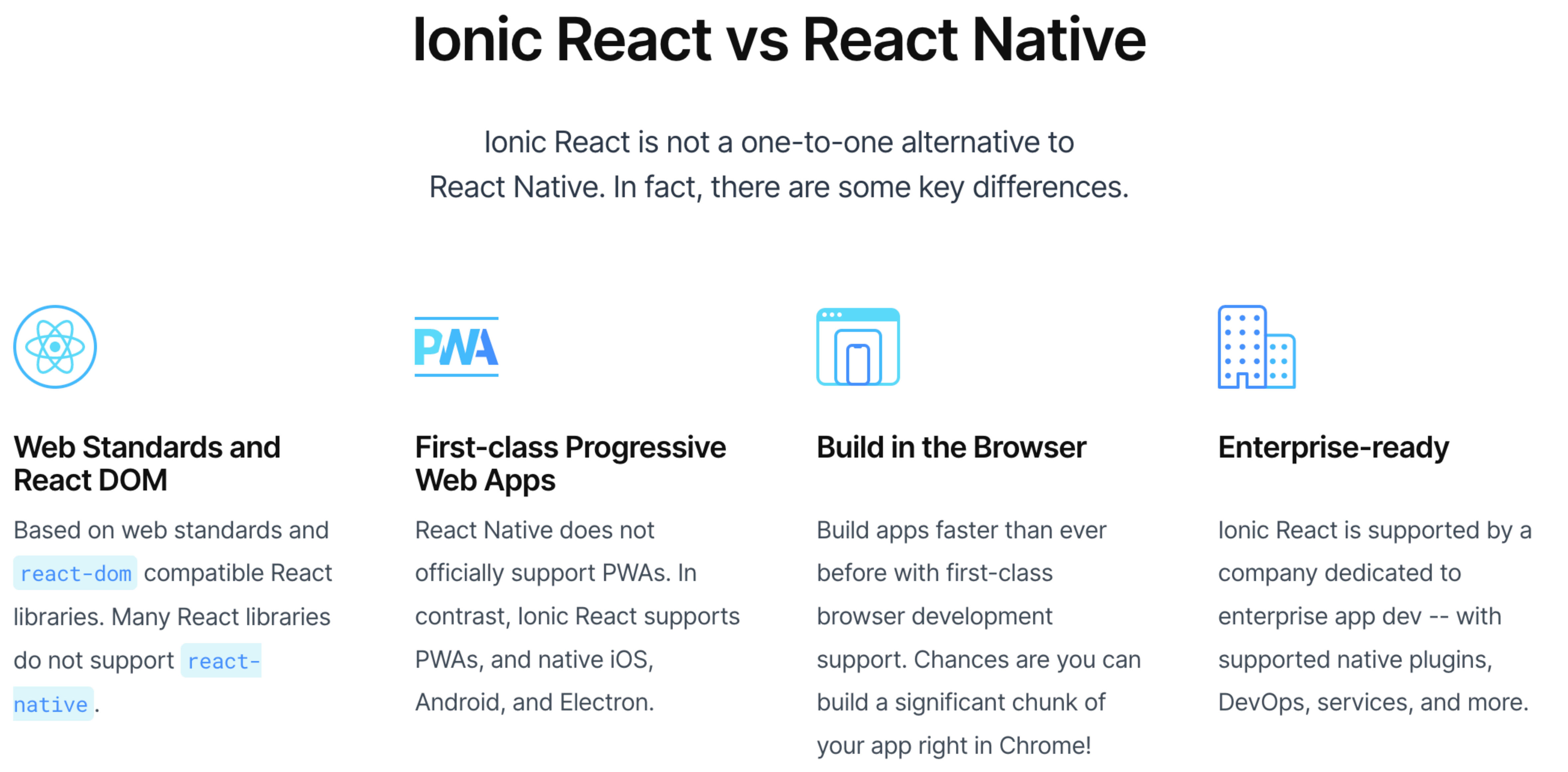
One codebase. Any platform. Now in React.

An open source mobile toolkit for building high quality, cross-platform native and web app experiences. Move faster with a single code base, running everywhere with JavaScript and the Web.

Ionic

Stencil

Ionic是一个开源的移动应用程序开发框架，它可以轻松地使用web技术构建高质量的跨平台的移动应用。可以让我们快速开发移动App、移动端WEB页面、微信公众平台应用，混合app web页面。



Ionic React vs React Native

Ionic React is not a one-to-one alternative to React Native. In fact, there are some key differences.

Web Standards and React DOM

Based on web standards and `react-dom` compatible React libraries. Many React libraries do not support `react-native`.

First-class Progressive Web Apps

React Native does not officially support PWAs. In contrast, Ionic React supports PWAs, and native iOS, Android, and Electron.

Build in the Browser

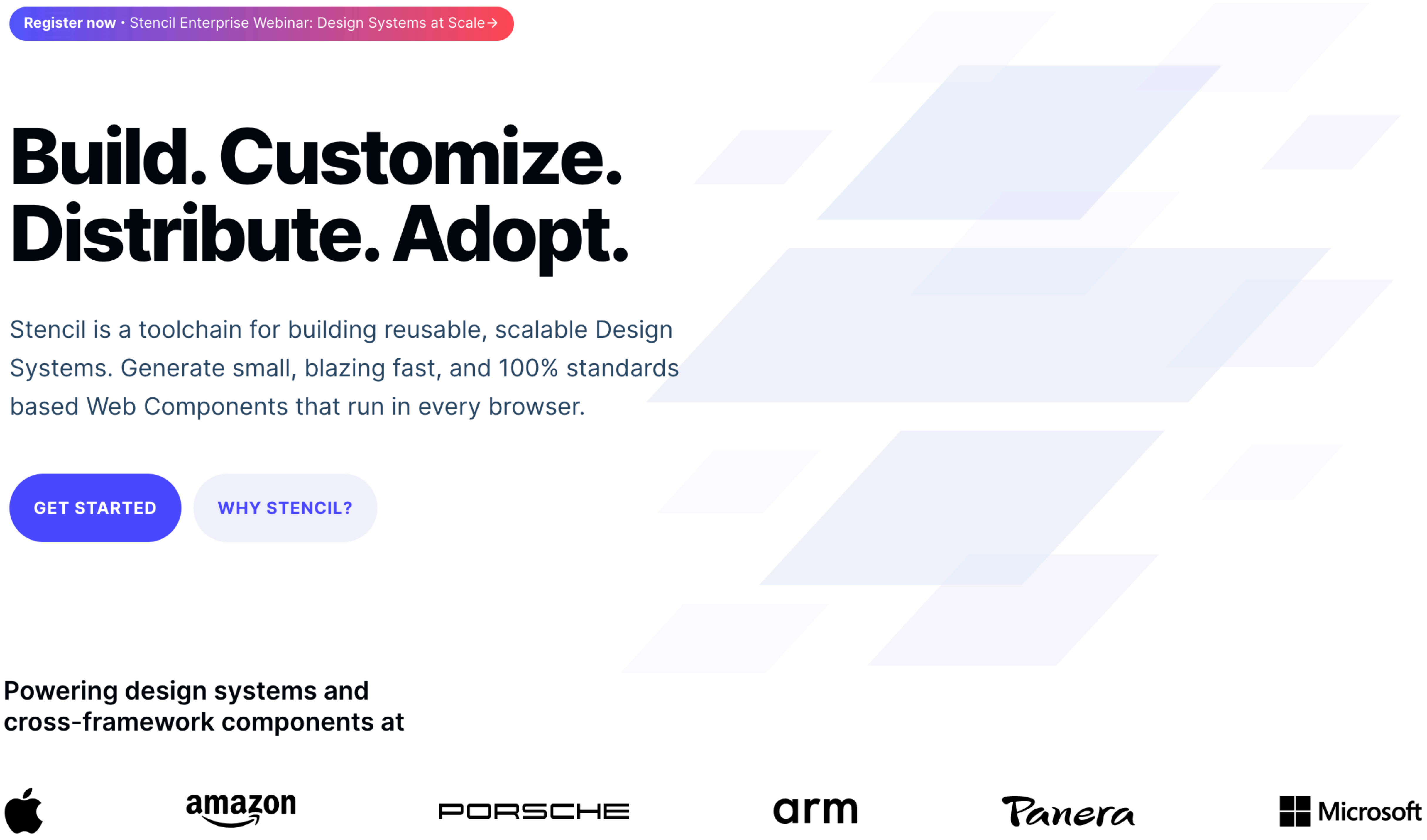
Build apps faster than ever before with first-class browser development support. Chances are you can build a significant chunk of your app right in Chrome!

Enterprise-ready

Ionic React is supported by a company dedicated to enterprise app dev -- with supported native plugins, DevOps, services, and more.

关于Stencil

https://stenciljs.com/



Register now • Stencil Enterprise Webinar: Design Systems at Scale →

Build. Customize. Distribute. Adopt.

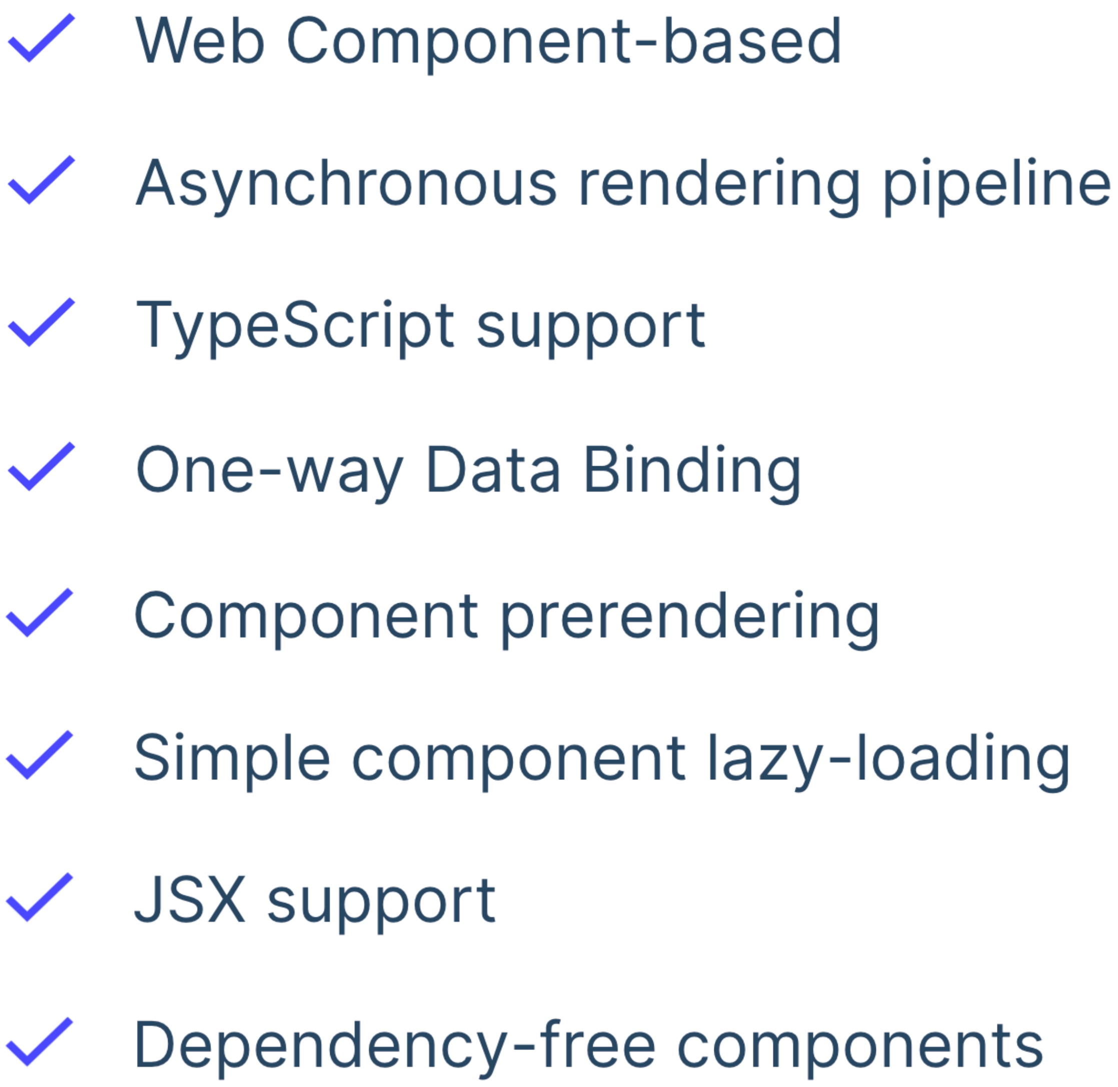
Stencil is a toolchain for building reusable, scalable Design Systems. Generate small, blazing fast, and 100% standards based Web Components that run in every browser.

Powering design systems and cross-framework components at

amazon
PORSCHE
arm
Panera
Microsoft

关于Stencil

The magical, reusable web component compiler.



- ✓ Web Component-based
- ✓ Asynchronous rendering pipeline
- ✓ TypeScript support
- ✓ One-way Data Binding
- ✓ Component prerendering
- ✓ Simple component lazy-loading
- ✓ JSX support
- ✓ Dependency-free components

Ionic与Stencil

React 用腻了

Angular 用腻了

jQuery 可能会被笑话

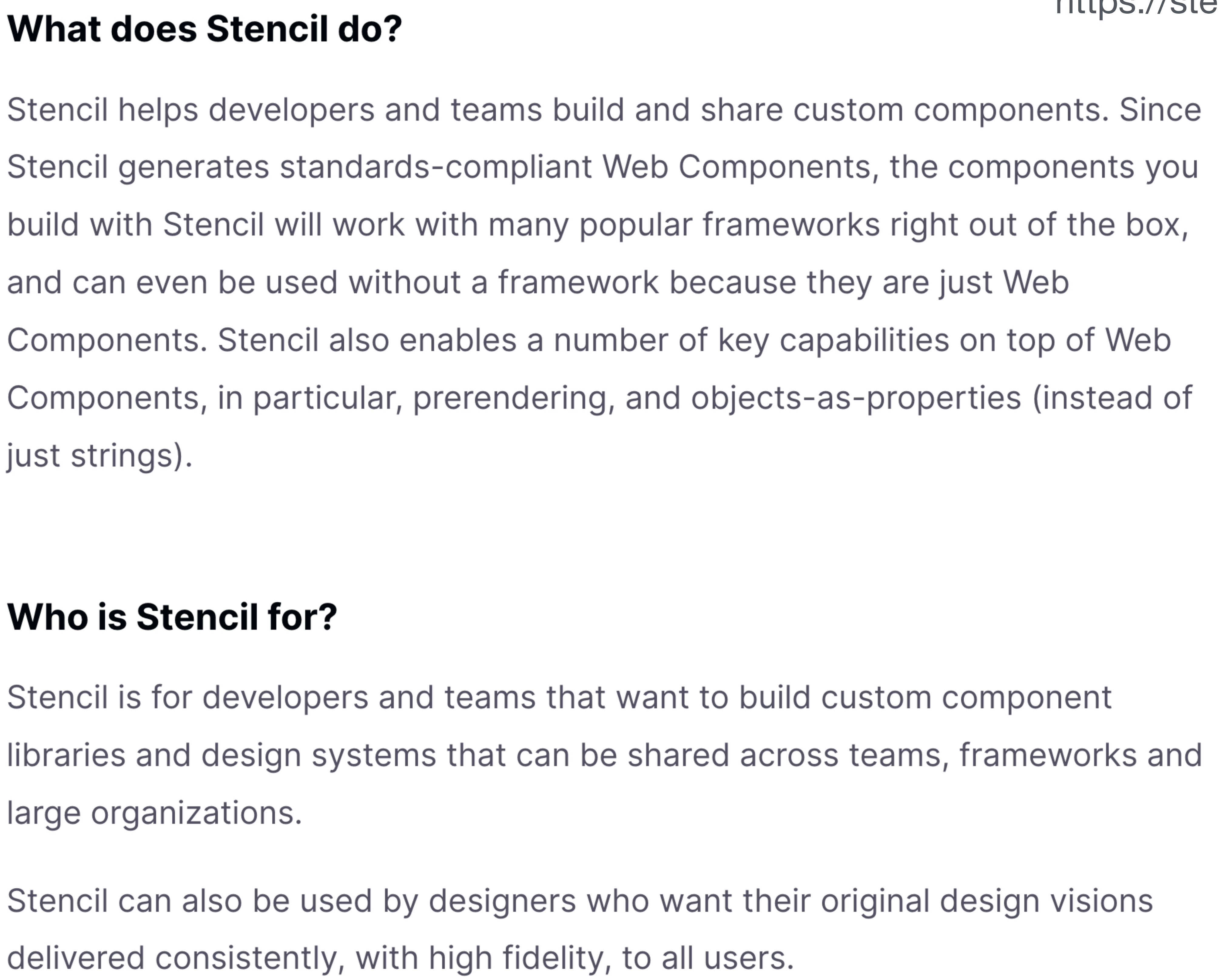
原生 JavaScript 已经不会了

直到我从 Web Components 框架里找到了 Stencil.js…

Ionic4是一次重大的升级，它第一次实现了与框架无关，不再必须使用Angular，你可以使用Vue，React，JQuery或者不使用任何框架来进行Ionic开发，这一切都是因为可以使用Stencil来构建标准Web Components…

基于标准Web组件，它实现了框架无关。以它来生成的组件，你可以直接拿给Angular、Vue、React、JQuery或者干脆不用任何框架来开发使用。

https://stenciljs.com/docs/faq



What does Stencil do?

Stencil helps developers and teams build and share custom components. Since Stencil generates standards-compliant Web Components, the components you build with Stencil will work with many popular frameworks right out of the box, and can even be used without a framework because they are just Web Components. Stencil also enables a number of key capabilities on top of Web Components, in particular, prerendering, and objects-as-properties (instead of just strings).

Who is Stencil for?

Stencil is for developers and teams that want to build custom component libraries and design systems that can be shared across teams, frameworks and large organizations.

Stencil can also be used by designers who want their original design visions delivered consistently, with high fidelity, to all users.

Demo

```
git clone https://github.com/ionic-team/stencil-app-starter my-app

cd my-app

npm install
```

——————————————————————————

```
npm init stencil

选择 app 类型

输入项目名称 app

 cd app

npm install 安装依赖项

npm run start 启动测试
```

内容回顾

内容回顾