

Веб-компоненты на твоём любимом фреймворке

MinskJS Meetup #12

Глеб Казачинский

Обо мне

- Глеб
 - 5+ лет занимаюсь вебom
 - Заканчивал ФПМИ
 - Занимаюсь триатлоном
-
- Веду блог по решению задач с собеседований



Зачем этот доклад?

Проблема кросскомандной разработки



Способы решения проблемы

- iframe
- Module Federation
- Веб-компоненты

Характеристики

- Автономная сборка и деплой
- Производительность
- Изоляция стилей
- Управление стилями
- Взаимодействие
- SEO
- Кроссплатформенность
- Разработка на популярном стеке
- Сложность внедрения

iframe

HTML Demo: <iframe>

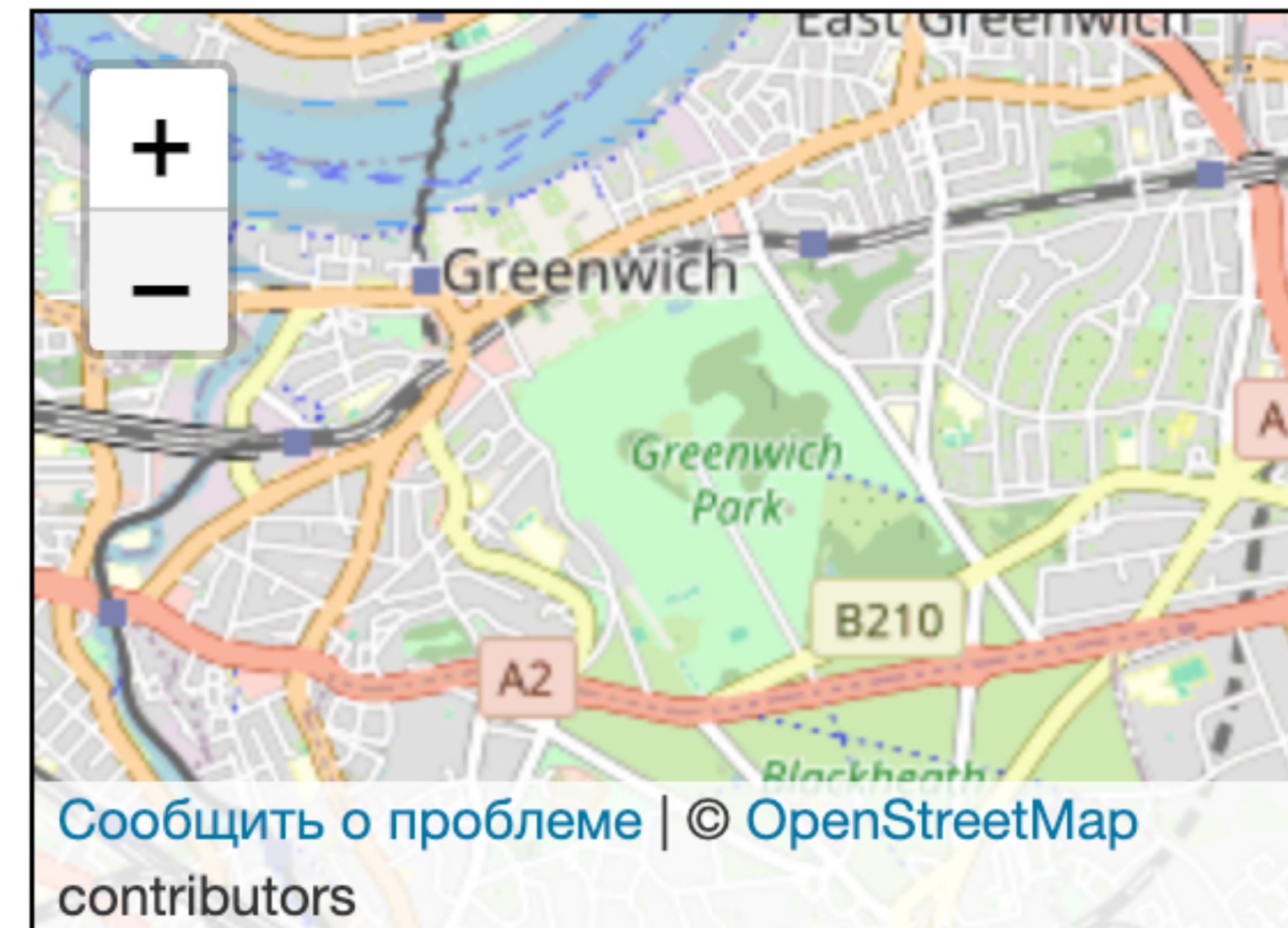
RESET

HTML

CSS

OUTPUT

```
1 <iframe
2   id="inlineFrameExample"
3   title="Inline Frame Example"
4   width="300"
5   height="200"
6
7   src="https://www.openstreetmap.org/export/embed.html?
8   bbox=-0.004017949104309083%2C51.47612752641776%2C0.00030577182769775396%2C51.478569861898606&layer=mapnik">
9 </iframe>
```



Source: [MDN](#)

iframe

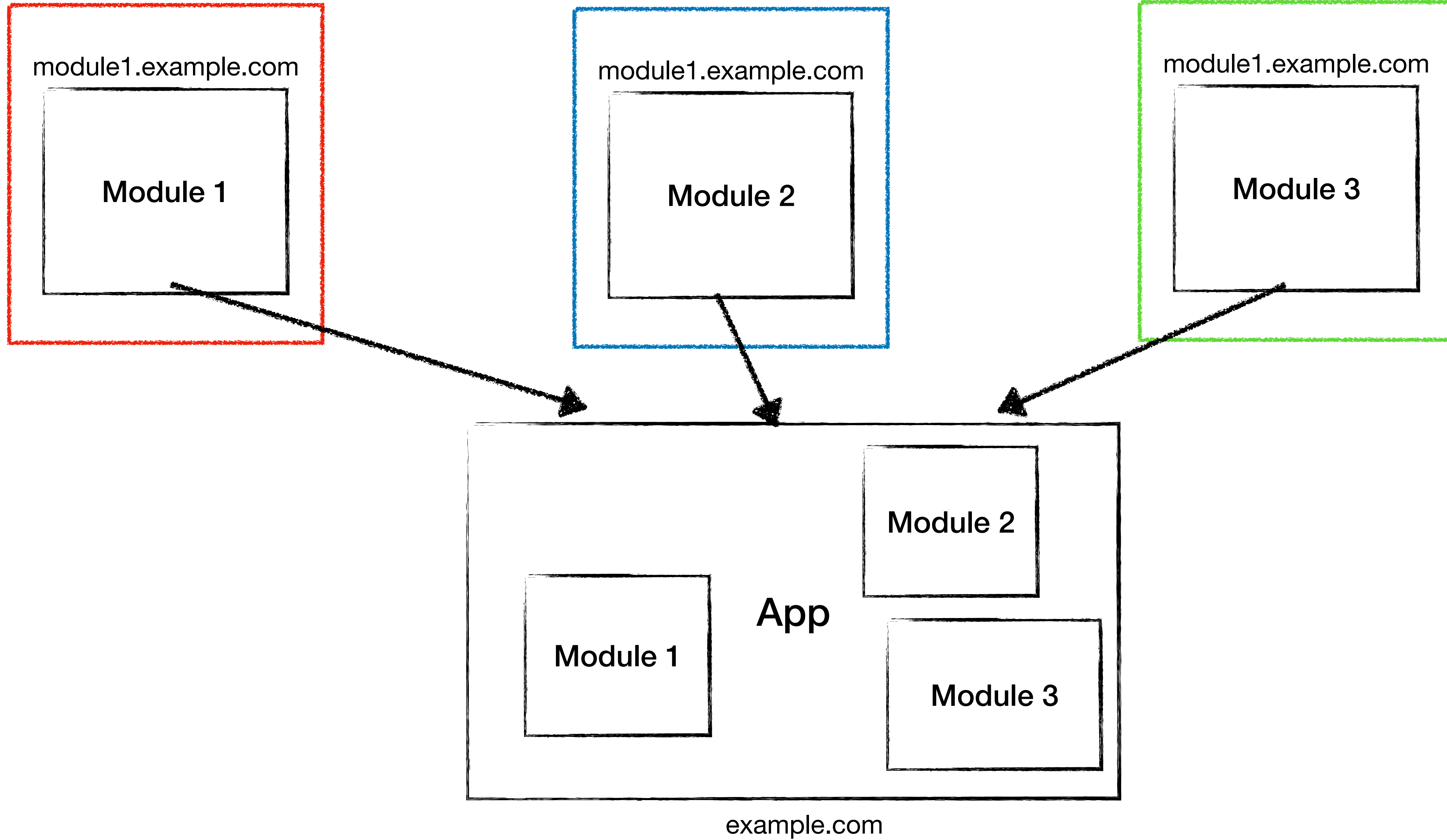
Характеристика	Оценка
Автономная сборка и деплой	Хорошо
Производительность	Плохо
Изоляция стилей	Хорошо
Управление стилями	Плохо
Взаимодействие	Плохо
SEO	Средне
Кроссплатформенность	Хорошо
Разработка на популярном стеке	Хорошо
Сложность внедрения	Хорошо

Module Federation

Team A

Team B

Team C



Module Federation

Характеристика	Оценка
Автономная сборка и деплой	Хорошо
Производительность	Хорошо
Изоляция стилей	Плохо
Управление стилями	Хорошо
Взаимодействие	Хорошо
SEO	Хорошо
Кроссплатформенность	Средне
Разработка на популярном стеке	Хорошо
Сложность внедрения	Плохо

Веб- КОМПОНЕНТЫ

Веб-компоненты



```
▼ <my-angular-chat ng-version="19.1.8" messages="[{"id":"1","message":"I like RxJs","sender":"Angular"}, {"id":"2","message":"I like JSX","sender":"React"}]" style="--primary-color: #e41b41; --secondary-color: #fdb018;"> == $0
  ▼ #shadow-root (open)
    ▶ <style>...</style>
    ▼ <div class="chat-container">
      <h2>Angular чат</h2>
      ▶ <div class="messages-container">...</div> flex
      ▼ <div class="input-container"> flex
        <input type="text" placeholder="Введите сообщение..." class="ng-untouched ng-pristine ng-valid">
        <button>Отправить</button>
      </div>
    </div>
  </my-angular-chat>
```

MinskJS Meetup #12

Online, 19 марта 2025 в 19:00

Да кто такие эти ваши
веб-компоненты?!



Алексей Назаренко
Независимый разработчик

Веб-компоненты

Характеристика	Оценка
Автономная сборка и деплой	Хорошо
Производительность	Хорошо
Изоляция стилей	Хорошо
Управление стилями	Средне
Взаимодействие	Средне
SEO	Хорошо
Кроссплатформенность	Хорошо
Разработка на популярном стеке	Хорошо
Сложность внедрения	Хорошо

Сравнение

Характеристика	iframe	Module Federation	Веб-компоненты
Автономная сборка и деплой	Хорошо	Хорошо	Хорошо
Производительность	Плохо	Хорошо	Хорошо
Изоляция стилей	Хорошо	Плохо	Хорошо
Управление стилями	Плохо	Хорошо	Средне
Взаимодействие	Плохо	Хорошо	Средне
SEO	Средне	Хорошо	Хорошо
Кроссплатформенность	Хорошо	Средне	Хорошо
Разработка на популярном стеке	Хорошо	Хорошо	Хорошо
Сложность внедрения	Хорошо	Плохо	Хорошо

Как разрабатывать веб-компоненты?

- Vanilla JS (Custom Elements API)
- Библиотеки (Lit, Stencil)
- Фреймворки (Angular, React и др.)



Жизненный цикл веб-компонента (Custom Elements API)

Имя	Вызывается, когда
constructor	Элемент создан
connectedCallback	Элемент вставлен в DOM
disconnectedCallback	Элемент удален из DOM
attributeChangedCallback	Изменен наблюдаемый атрибут



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

```
  static get observedAttributes() {  
    return ['name'];  
  }
```

```
  get name() {  
    return this.getAttribute('name');  
  }
```

```
  set name(value) {  
    this.setAttribute('name', value);  
  }
```

```
  connectedCallback() {  
    this.div = document.createElement('div');  
    this.text = document.createTextNode(this.name ?? '');  
    this.div.appendChild(this.text);  
    this.appendChild(this.div);  
  }
```

```
  //...
```



```
// ...
```

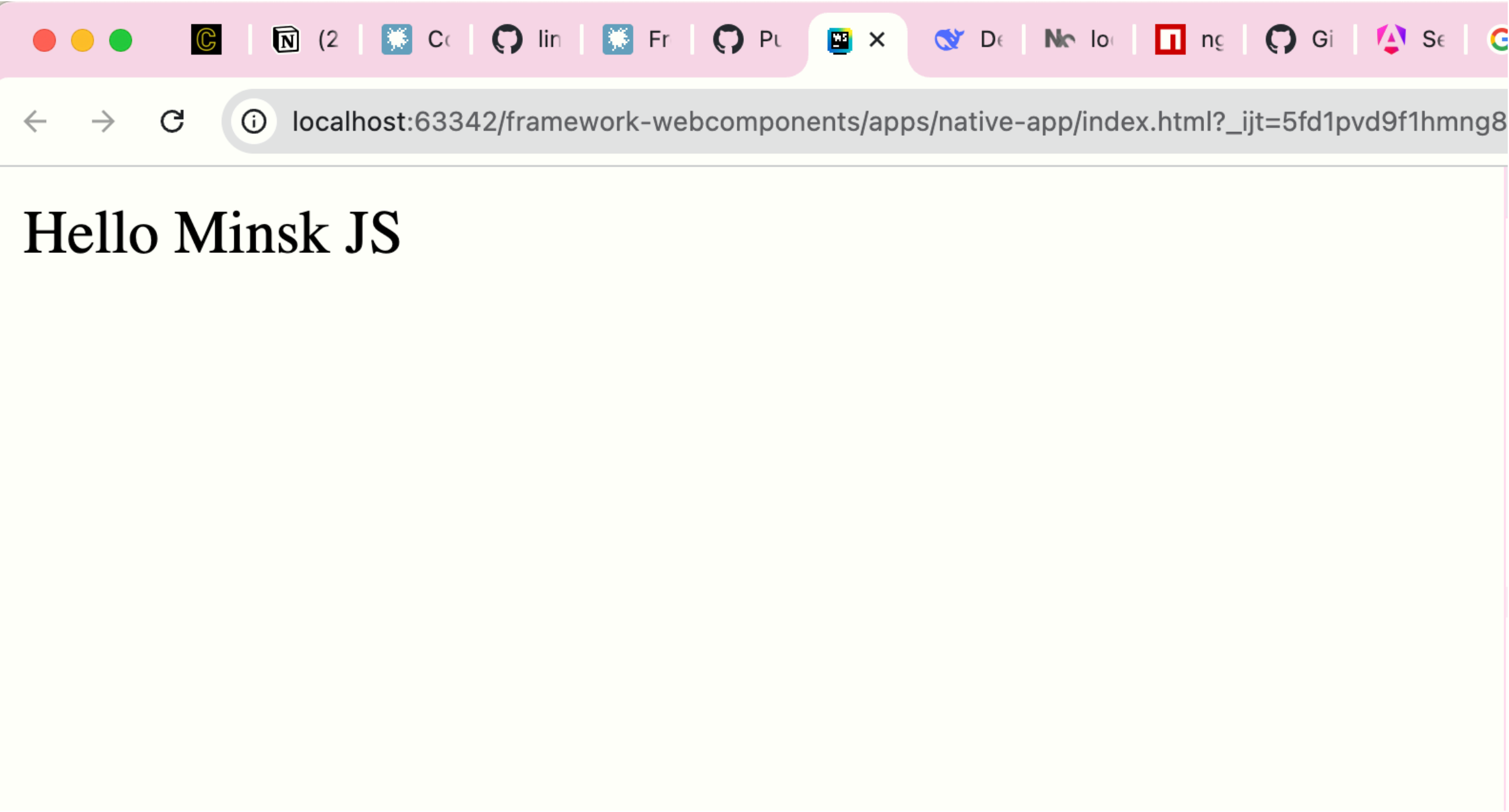
```
disconnectedCallback() {  
  this.removeChild(this.div);  
}
```

```
attributeChangedCallback(attr, oldValue, newValue) {  
  if (attr === 'name' && this.text) {  
    this.text.textContent = newValue;  
  }  
}  
}
```

```
customElements.define(  
  'my-hello-world-custom-elements-api',  
  HelloWorldCustomElementsApi  
);
```



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Native App</title>
  </head>
  <body>
    <my-hello-world-custom-elements-api name="Hello Minsk JS"></my-hello-world-custom-elements-api>
    <script type="module" src="hello-world-custom-elements-api.js"></script>
  </body>
</html>
```



Как разрабатывать веб-компоненты?

- Vanilla JS (Custom Elements API)
- Библиотеки (Lit, Stencil)
- Фреймворки (Angular, React и др.)



Как написать веб-компонент на Angular?

Hello world веб-компонент на Angular

I am Gleb Say hello

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Native App</title>
  </head>
  <body>
    <my-hello-world name="Gleb"></my-hello-world>
    <script type="module" src="angular-hello-world.js"></script>
  </body>
</html>
```



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

```
@Component({  
  template: `<span>I am {{ name ?? 'stranger' }}</span>  
    <button (click)="onClick()" type="button">Say hello</button>`,  
  styles: `  
    span {  
      color: var(--text-color, red);  
    }  
  `,  
  standalone: true,  
})
```

```
export class HelloComponent {  
  @Input() name?: string;  
  @Output() change = new EventEmitter();
```

```
  onClick() {  
    this.change.emit(`${this.name ?? 'stranger'} says hello`);  
  }  
}
```



```
class HelloComponentClass extends HTMLElement {  
  constructor() {  
    super();  
  }  
  
  static get observedAttributes() {  
  }  
  
  connectedCallback() {  
  }  
  
  disconnectedCallback() {  
  }  
  
  attributeChangedCallback(attrName, oldVal, newVal) {  
  }  
}
```

CustomElementClass

constructor

connectedCallback

disconnectedCallback

attributeChangedCallback



AngularCustomElementsBridge

prepare

initComponent

destroyComponent

setInput



```
import { AngularCustomElementsBridge } from './angular-elements-bridge';
```

```
export class CustomElementsWrapper extends HTMLElement {  
    static bridge: AngularCustomElementsBridge;
```

```
    constructor() {  
        super();  
        CustomElementsWrapper.bridge.prepare();  
    }
```

```
    static get observedAttributes() {  
        return AngularCustomElementsBridge.attributes;  
    }
```

```
    connectedCallback() {  
        CustomElementsWrapper.bridge.initComponent(this);  
    }
```

```
    disconnectedCallback() {  
        CustomElementsWrapper.bridge.destroyComponent();  
    }
```

```
    attributeChangedCallback(attrName: string, oldVal: any, newVal: string) {  
        CustomElementsWrapper.bridge.setInput(attrName, newVal);  
    }
```

```
}
```

AngularCustomElementsBridge

prepare

initComponent

destroyComponent

setInput





```
// CustomElement.constructor -> Bridge.prepare  
prepare() {  
    this.componentFactory.inputs.forEach((input) =>  
        AngularCustomElementsBridge.attributes.push(input.templateName)  
    );  
}
```

```

// CustomElement.connectedCallback -> Bridge.intiComponent
initComponent(element: HTMLElement) {
  const componentInjector = Injector.create([], this.injector);

  this.componentRef = this.componentFactory.create(
    componentInjector,
    null as any,
    element
  );

  this.componentFactory.inputs.forEach(
    (prop) =>
      (this.componentRef!.instance[prop.propName] =
        this.initialInputValues[prop.propName])
  );

  const eventEmitters = this.componentFactory.outputs.map(
    ({ propName, templateName }) => {
      const emitter = (this.componentRef!.instance as any)[
        propName
      ] as EventEmitter<any>;
      return emitter.pipe(
        map((value: any) => ({ name: templateName, value })))
    }
  );
}
// ...

```



```
//...
```

```
this.outputEvents = merge(...eventEmitters);
```

```
this.ngElementEventsSubscription =  
this.outputEvents.subscribe((e) => {  
    const customEvent = document.createEvent('CustomEvent');  
    customEvent.initCustomEvent(e.name, false, false, e.value);  
    element.dispatchEvent(customEvent);  
});
```

```
this.applicationRef = this.injector.get(ApplicationRef);
```

```
this.applicationRef.attachView(this.componentRef.hostView);
```

```
}
```



```
// CustomElement.disconnectCallback -> Bridge.destroyComponent
```

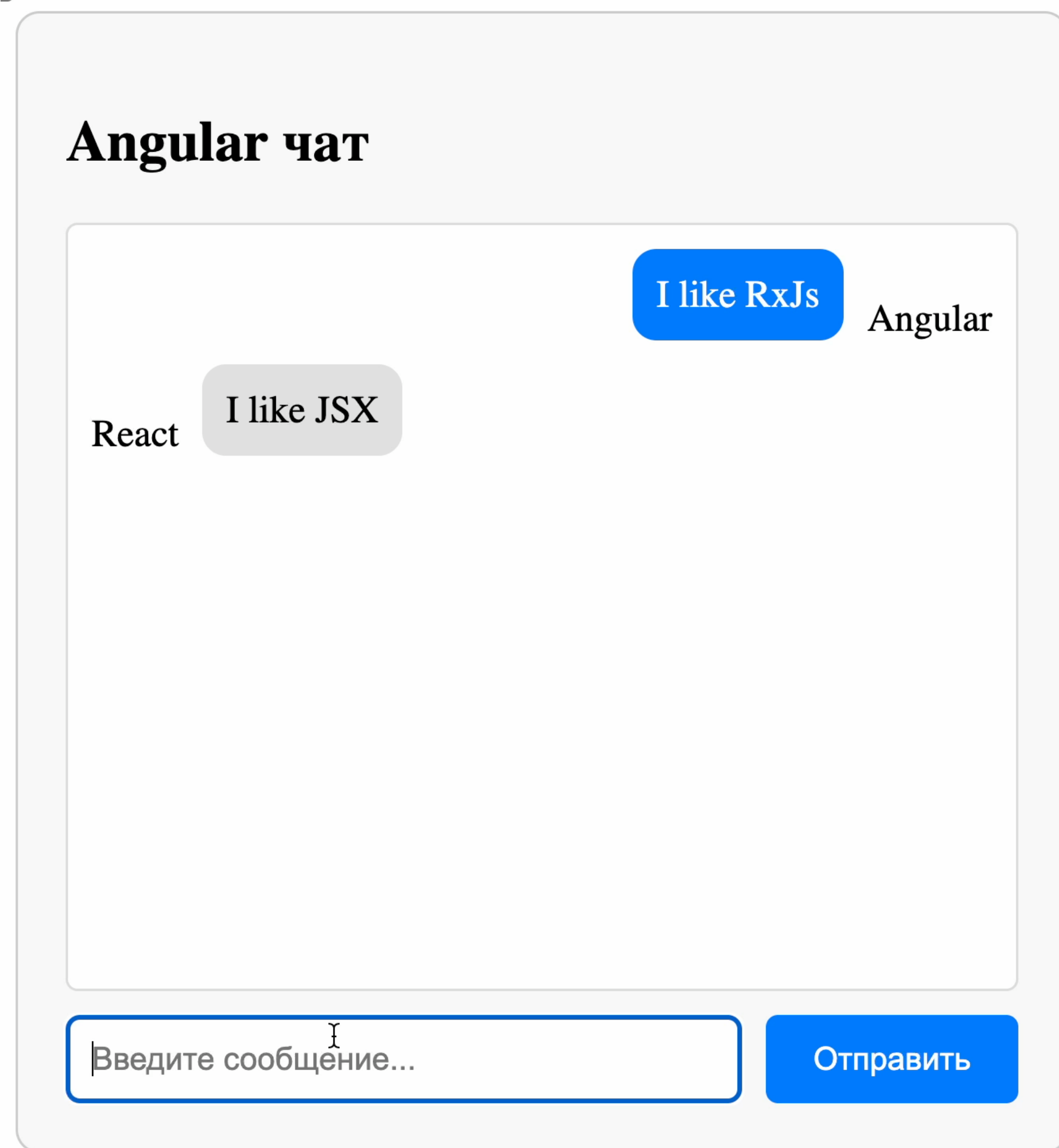
```
destroyComponent() {  
  this.componentRef!.destroy();  
  this.ngElementEventsSubscription!.unsubscribe();  
}
```

```
getInput(propName: string) {  
  return this.componentRef && this.componentRef.instance[propName];  
}
```

```
// CustomElement.attributeChangedCallback -> Bridge.setInput
```

```
setInput(propName: string, value: string) {  
  if (!this.componentRef) {  
    this.initialInputValues[propName] = value;  
    return;  
  }  
  if (this.componentRef.instance[propName] === value) {  
    return;  
  }  
  this.componentRef.instance[propName] = value;  
  this.componentRef.changeDetectorRef.detectChanges();  
}
```

Более сложный веб-компонент на Angular

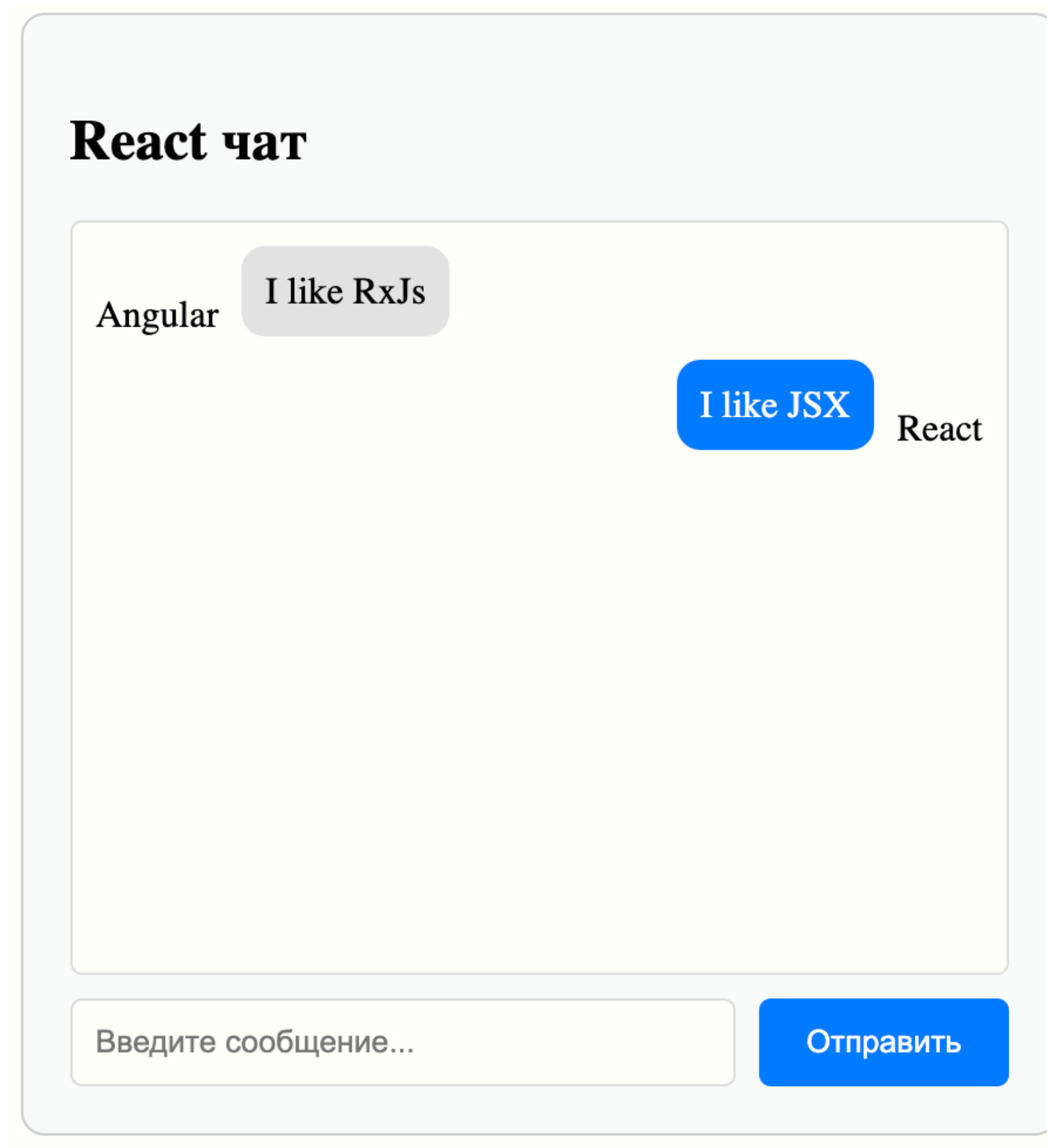


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Native App</title>
  </head>
  <body>
    <my-angular-chat></my-angular-chat>

    <script type="module" src="angular-chat.js"></script>
  </body>
</html>
```

Как написать веб-компонент на React?

Сделаем чат веб-компонент на React



CustomElementClass

constructor

connectedCallback

disconnectedCallback

attributeChangedCallback



ReactCustomElementsBridge

prepare

initComponent

destroyComponent

setInput



```
import { ReactCustomElementsBridge } from './custom-elements-bridge';

export class CustomElementsWrapper extends HTMLElement {
    static bridge: ReactCustomElementsBridge<any>;

    constructor() {
        super();

        CustomElementsWrapper.bridge.prepare(this);
    }

    static get observedAttributes() {
        return ReactCustomElementsBridge.attributes;
    }

    connectedCallback() {
        CustomElementsWrapper.bridge.initComponent();
    }

    disconnectedCallback() {
        CustomElementsWrapper.bridge.destroyComponent();
    }

    attributeChangedCallback(attrName: string, oldVal: any, newVal: string) {
        CustomElementsWrapper.bridge.setInput(attrName, newVal);
    }
}
```

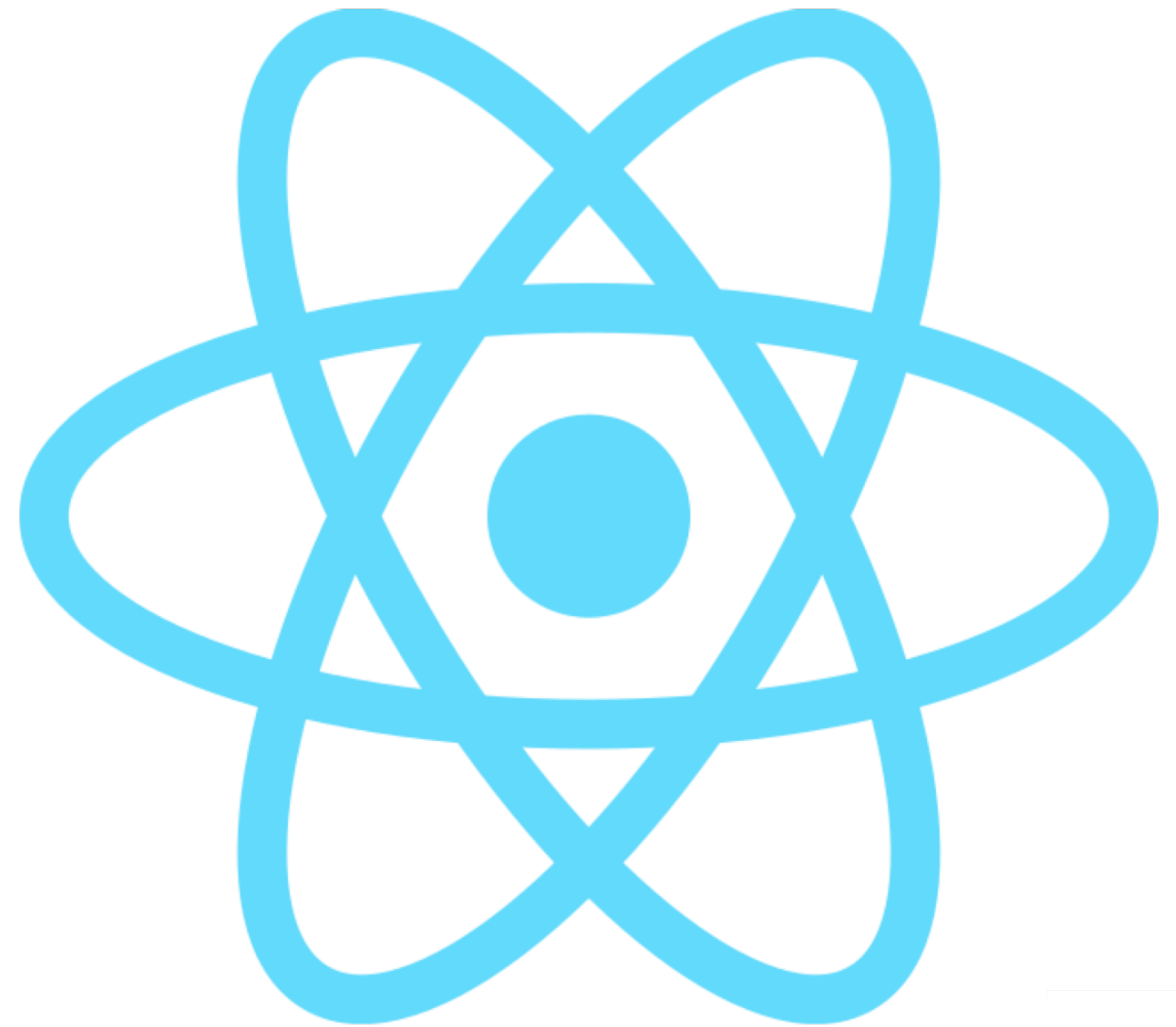
ReactCustomElementsBridge

prepare

initComponent

destroyComponent

setInput





```
// CustomElement.constructor -> Bridge.prepare
prepare(customElement: CustomElementsWrapper) {
  this.customElement = customElement;
```

```
    if (this.options.shadow) {
      this.container = this.customElement.attachShadow({
        mode: this.options.shadow,
      }) as unknown as HTMLElement;
    } else {
      this.container = this.customElement;
    }
  }
```

```
    this.props = Object.entries(this.options.props!).reduce(
      (acc, [propName, propFormat]) => {
        const propValue = this.customElement?.getAttribute(propName);
        const formattedPropValue = this.transformPropValue(propValue!);
        return {
          ...acc,
          [propName]: formattedPropValue,
        };
      },
      {}
    );
  }
}
```

```
// CustomElement.connectedCallback -> Bridge.initComponent
initComponent() {
  this.mount();
}

mount() {
  this.root = createRoot(this.container!);

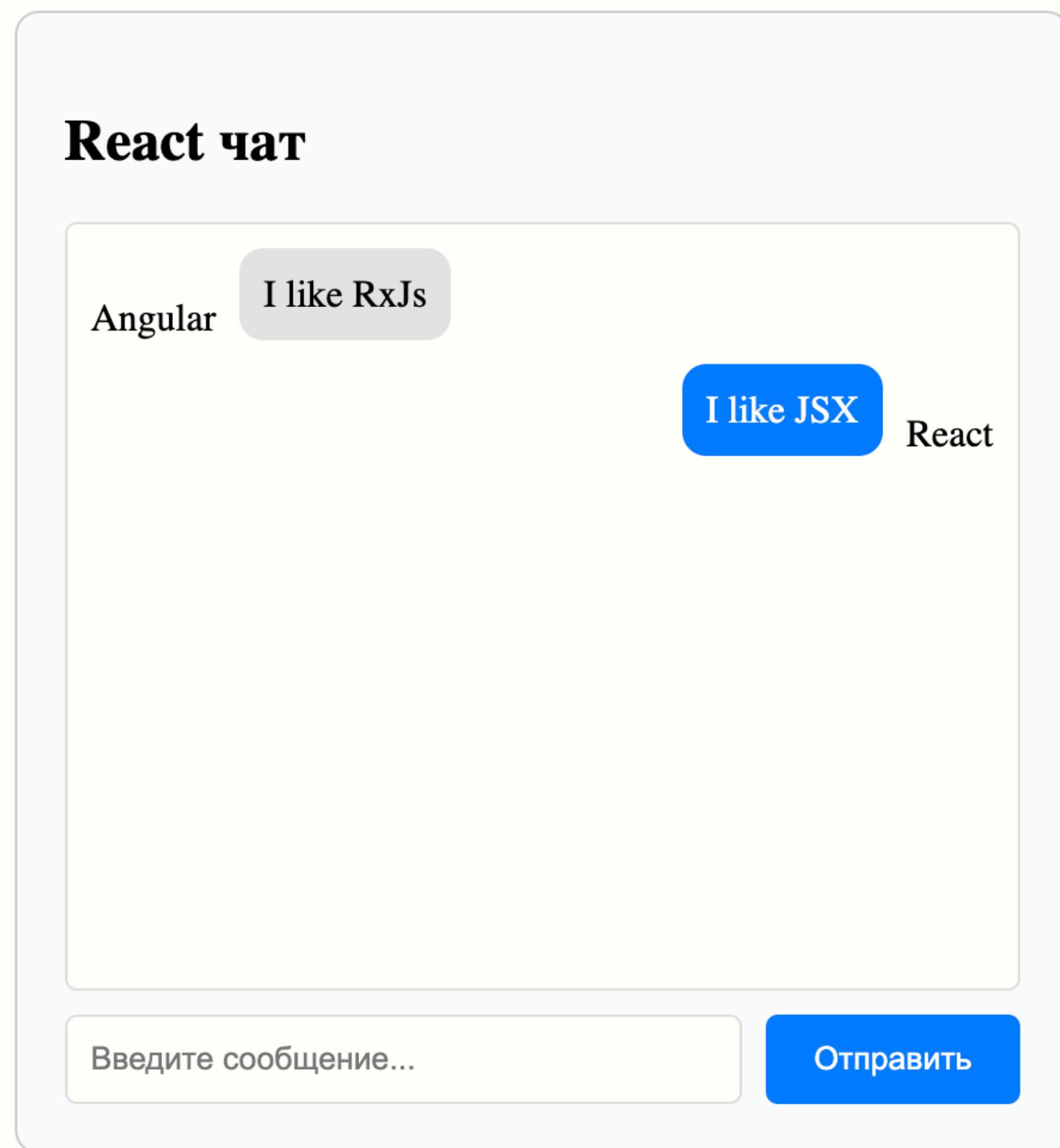
  const element = React.createElement(this.component, this.props);
  this.root.render(element);
}
```

```
// CustomElement.disconnectedCallback -> Bridge.destroyComponent
destroyComponent() {
  this.root.unmount();
}
```

```
// CustomElement.attributeChangedCallback -> Bridge.setInput
setInput(propName: string, value: string) {
  this.props[propName] = this.transformPropValue(value);
  const element = React.createElement(this.component, this.props);
  console.log(this.props);

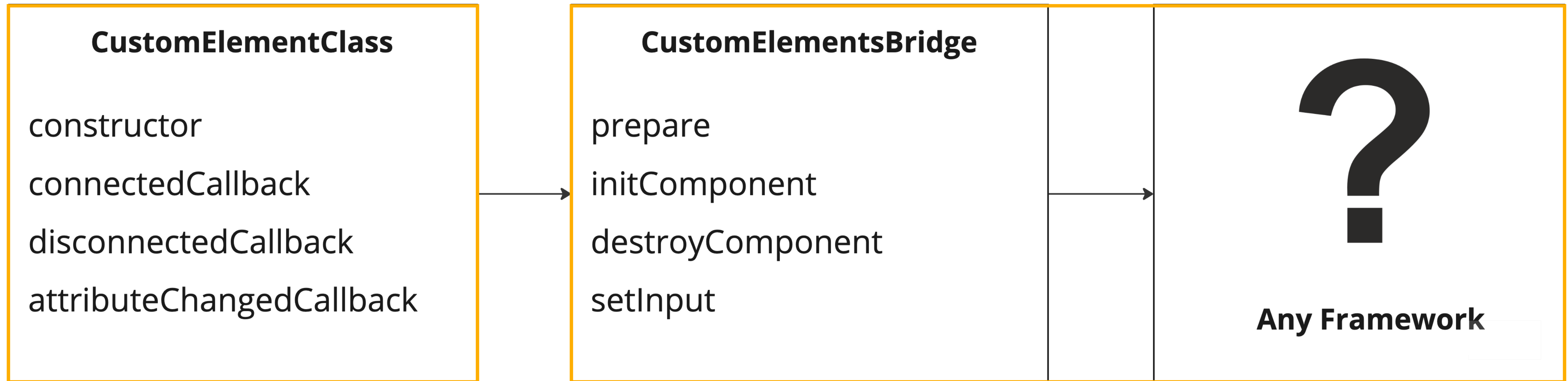
  this.root.render(element);
}
```

Результат работы своего react-to-webcomponent



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Native App</title>
  </head>
  <body>
    <my-react-chat></my-react-chat>

    <script type="module" src="react-chat.js"></script>
  </body>
</html>
```



Готовые решения

- Angular — @angular/elements
- React — @react-to-webcomponent
- Vue — defineCustomElement
- Svelte — compilerOptions: { customElement: true, ... }

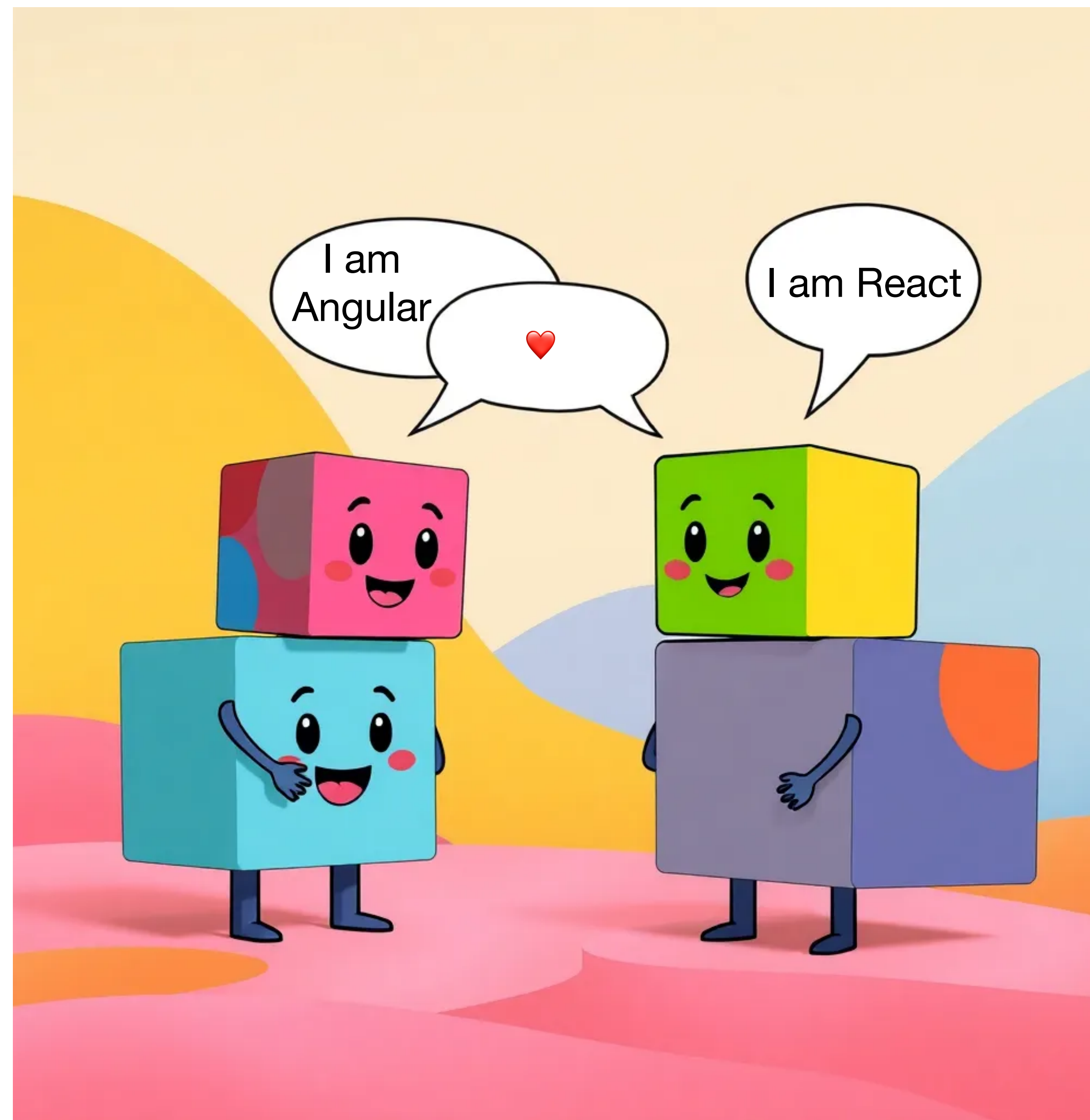
Демо - Чат

Код



Общение компонентов

- Input
- Output



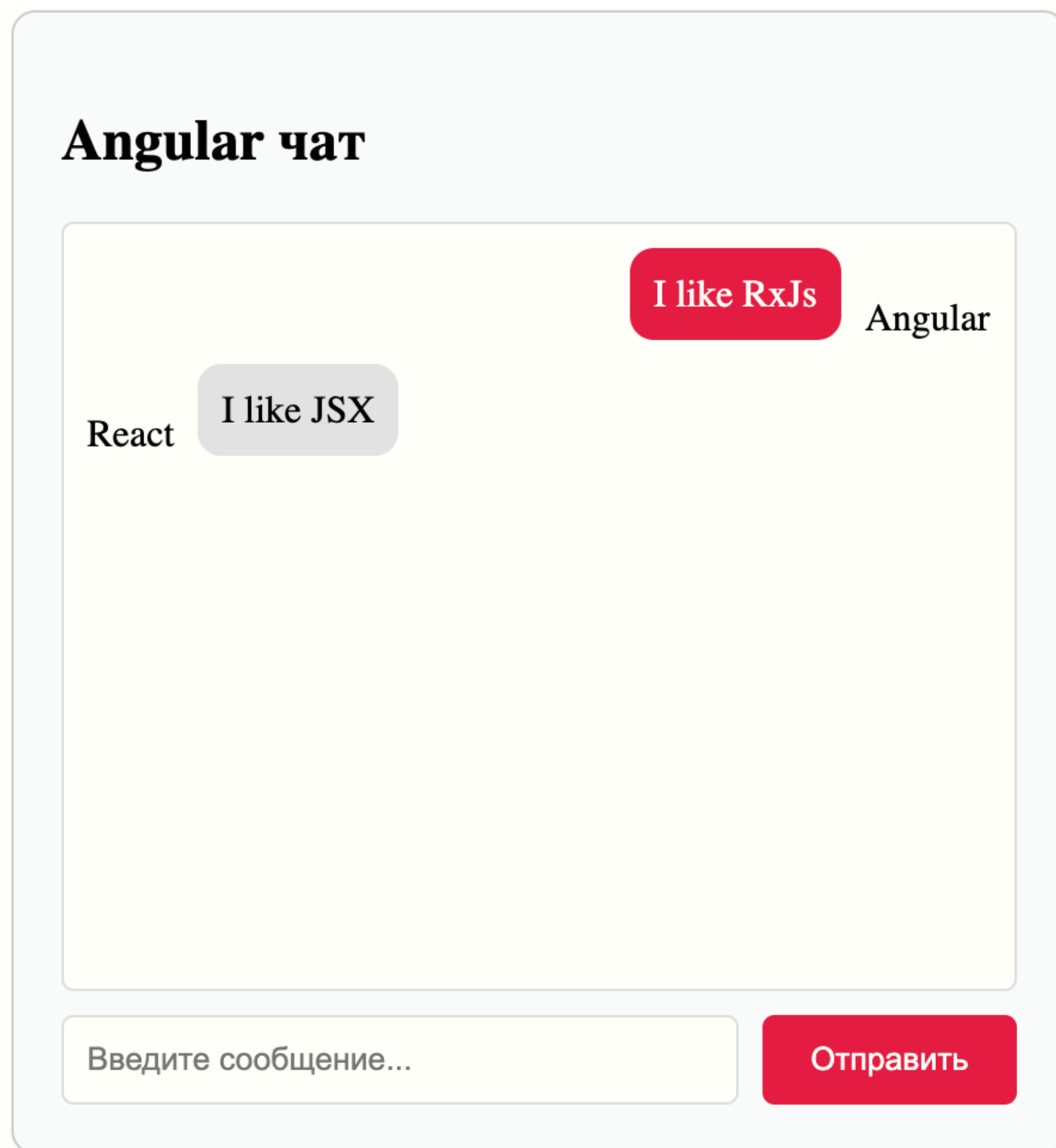
Shadow DOM: on

```
▼ <my-angular-chat-without-lib ng-version="19.1.8" messages="{\"id\":\"1\",\"message\":\"I like RxJs\",\"sender\":\"Angular\"},{\"id\":\"2\",\"message\":\"I like JSX\",\"sender\":\"React\"}]">
  ▼ #shadow-root (open)
    ▶ <style> ... </style>
    ▼ <div class="chat-container">
      <h2>Angular чат</h2>
      ▼ <div class="messages-container"> flex
        ▶ <div class="message-container right"> ... </div> flex
        ▶ <div class="message-container left"> ... </div> flex
        <!-->
      </div>
      ▼ <div class="input-container"> flex == $0
        <input type="text" placeholder="Введите сообщение..."
          class="ng-untouched ng-pristine ng-valid">
        <button>Отправить</button>
      </div>
    </div>
    "Angular"
  </my-angular-chat-without-lib>
```

Shadow DOM: off

```
▼<my-angular-chat ng-version="19.1.8" _ngghost-ng-c981700914
  messages="[{"id":"1","message":"I like RxJs","sender":"Angular"},
  {"id":"2","message":"I like JSX","sender":"React"}]">
  ▼<div _ngcontent-ng-c981700914 class="chat-container">
    <h2 _ngcontent-ng-c981700914>Angular чат</h2>
    ▼<div _ngcontent-ng-c981700914 class="messages-container">
      flex == $0
      ▶<div _ngcontent-ng-c981700914 class="message-container right"
        t">...</div> flex
      ▶<div _ngcontent-ng-c981700914 class="message-container left"
        t">...</div> flex
      <!-->
    </div>
    ▼<div _ngcontent-ng-c981700914 class="input-container"> flex
      <input _ngcontent-ng-c981700914 type="text" placeholder="Вв
      едите сообщение..." class="ng-untouched ng-pristine ng-vali
      d">
      <button _ngcontent-ng-c981700914>Отправить</button>
    </div>
  </div>
</my-angular-chat>
```

Стилизация с Shadow DOM



```
▼ <my-angular-chat ng-version="19.1.8" messages="[{"id":"1","message":"I like RxJs","sender":"Angular"}, {"id":"2","message":"I like JSX","sender":"React"}]" style="--primary-color: #e41b41; --secondary-color: #fdb018;"> == $0
  ▼ #shadow-root (open)
    ▶ <style> ... </style>
    ▼ <div class="chat-container">
      <h2>Angular чат</h2>
      ▶ <div class="messages-container"> ... </div> flex
      ▼ <div class="input-container"> flex
        <input type="text" placeholder="Введите сообщение..." class="ng-untouched ng-pristine ng-valid">
        <button>Отправить</button>
      </div>
    </div>
  </my-angular-chat>
```



```
button {  
  background-color: var(--primary-color, #007bff);  
  
  &:hover {  
    background-color: var(--secondary-color, #0056b3);  
  }  
}
```

Можно ли не грузить зависимости и фреймворк несколько раз?

- External dependencies
- Один файл для нескольких веб-компонентов

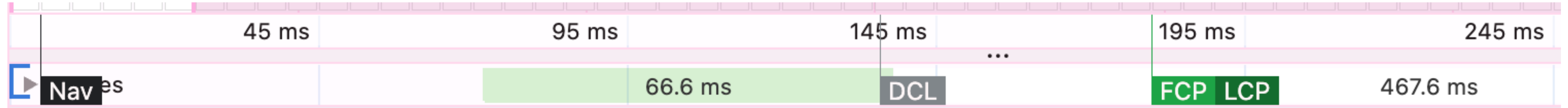
CSS-in-JS

Перформанс

Веб-компонент на фрейморке



Фреймворк



Вывод

Веб-компоненты + фреймворки = Love



Спасибо

Блог:



Код:

