

Яндекс

Яндекс

Web Components для практичных людей

Артём Пендюрин, разработчик интерфейсов

На чем ваши компоненты?



- Каждая библиотека требует отдельной реализации
- Реализация для каждой библиотеки требует наличия самой библиотеки

Framework Agnostic



- Для реализации требуется только Web Platform
- Можно использовать в любом окружении

Web Components



Custom Elements



Shadow DOM



HTML Templates



HTML Imports

Стандарты



Custom Elements

Набор API, которые позволяют вам определять пользовательские элементы и их поведение, которые затем могут использоваться в вашем интерфейсе.

```
class CustomElement extends HTMLElement {  
  constructor() {  
    super();  
    //...  
  }  
}  
  
customElements.define('custom-element', CustomElement);
```

Lifecycle Callbacks

```
class CustomElement extends HTMLElement {  
  static get observedAttributes() { return ['name', 'value']; }  
  connectedCallback() {} // Вызывается когда попадает в документ  
  disconnectedCallback() {} // Вызывается когда удаляется из документа  
  adoptedCallback() {} // Вызывается когда переносится в другой документ  
  attributeChangedCallback(name, oldValue, newValue) {  
    } // Вызывается при изменении атрибута  
}  
  
customElements.define('custom-element', CustomElement);
```


Расширение стандартных элементов

При расширении стандартного HTML элемента, нужно передать название элемента при регистрации в CustomElementRegistry.

```
class CustomButton extends HTMLButtonElement {  
    ...  
}
```

```
customElements.define('custom-button', CustomButton, {extends: 'button'});
```

Использование в вёрстке

- Новый custom element можно использовать как обычный тег
- Custom element, расширяющий стандартный элемент нужно объявлять используя атрибут «is»

```
<custom-element></custom-element>
```

```
<button is="custom-button"></button>
```

```
document.createElement( 'custom-element' );
```

```
document.createElement( 'button', {is: 'custom-button'} );
```

Shadow DOM

Набор API, которые позволяют прикреплять инкапсулированное DOM-дерево к элементу, отображающееся отдельно от основного документа, и контролировать связанную с ним функциональность.

```
class CustomElement extends HTMLElement {  
  constructor() {  
    super();  
    const shadow = this.attachShadow({mode: 'open'});  
    shadow.appendChild(document.createElement('p'));  
  }  
}
```

HTML Templates

Элементы `<template>` и `<slot>` позволяющие создать шаблон разметки, который не отображается на странице и может быть использован в качестве структурной основы элемента.

```
<template>
  <style>
    :host {background-color: crimson;}
    p {font-weight: bold;}
  </style>
  <p><slot name="text">NO TEXT</slot></p>
</template>
```

Использование <template>











```
class BoldText extends HTMLElement {  
  constructor() {  
    const shadowRoot = this.attachShadow({mode: 'open'});  
    const template = document.querySelector('template');  
    const content = document.importNode(template.content, true);  
  
    shadowRoot.appendChild(content);  
  }  
}  
  
customElements.define('bold-text', BoldText);  
  
<bold-text></bold-text>  
  
<bold-text><span slot="text">Текст</span></bold-text>
```

HTML Imports

Механизм загрузки веб компонентов.

```
<html>
  <head>
    <link rel="import" href="/components/custom-component.html">
  </head>
  <custom-component></custom-component>
</html>
```

Поддержка

| | |    |  |  |  |
|---|-----------------|---|---|---|---|
|  | Custom Elements | ✓ | ✓ | • | • |
|  | Shadow DOM | ✓ | ✓ | • | • |
|  | HTML Templates | ✓ | ✓ | ✓ | ✓ |
|  | HTML Imports | ✓ | ✗ | ✗ | • |

Что не так с HTML Imports?

Сложности с управлением зависимостями

- Глобальная область видимости
- Модули html файлы
- Повторные запросы

CDN

Можно попробовать использовать один CDN сервер?

- Exact URL matches
- Нельзя собрать бандл

Плоская структура

А что если жестко зафиксировать структуру директорий?

- Это не гарантирует уникальность путей
- Придётся использовать bower



Bower

A package manager for the web

...psst! While Bower is maintained, we recommend using [Yarn](#) and [Webpack](#) for front-end projects [read how to migrate!](#)

Sponsors ([become one](#)):



Web sites are made of lots of things — frameworks, libraries, assets, and utilities. Bower manages all these things for you.

Keeping track of all these packages and making sure they are up to date (or set to the specific versions you need) is tricky. Bower to the rescue!

Bower can manage components that contain HTML, CSS, JavaScript, fonts or even image files. Bower

Feature detection

А если загружать только то, что необходимо?

- Динамически загружаемые зависимости
- Непредсказуемые версии

ECMAScript modules

- Решают ту же проблему
- Стандарт ECMAScript 2015 (6th Edition, ECMA-262)

Полифилы

Подключаются при необходимости

```
if (!(
    'registerElement' in document &&
    'import' in document.createElement('link') &&
    'content' in document.createElement('template')
)) {
    const script = document.createElement('script');
    script.src = 'webcomponents-lite.min.js';
    document.body.appendChild(script);
}
```

Полифил Shadow DOM

- Есть «полная» и «лёгкая» версии
- Используется wrapper
- Ограниченная инкапсуляция CSS

Полифил Custom Elements

- Инициализируется асинхронно
- Использует MutationObserver

```
document.body.style.opacity = 0;

window.addEventListener( 'WebComponentsReady', ( ) => {
    document.body.style.opacity = 1;
});
```


Библиотеки



X-Tag



- Polymer
- X-Tag
- Slim.js

И что с этим делать?



Попробуем написать компонент

```
<template>
  <style>
    blockquote {font-size: 150%;}
    blockquote:before, blockquote:after {
      display: block; font-size: 150%; color: darkgray;
    }
    blockquote:before {content: '«'}
    blockquote:after {content: '»'}
  </style>
  <blockquote><slot name="text">empty</slot></blockquote>
</template>
```

```
const template = document.querySelector( 'template' )
```

```
class CustomQuote extends HTMLElement {
```

```
  constructor() {
```

```
    super( );
```

```
    const shadow = this.attachShadow({mode: 'open'});
```

```
    const content = document.importNode(template.content, true);
```

```
    shadow.appendChild(content);
```

```
  }
```

```
}
```

```
customElements.define( 'custom-quote', CustomQuote );
```

<custom-quote></custom-quote>

<<

empty

>>

<https://jsfiddle.net/ErBlack/5vze862v/1/>

The screenshot shows the 'Elements' panel of a web browser's developer tools. The DOM tree is expanded to show the structure of a custom element, `<custom-quote>`. The element is currently selected, and its shadow root is visible. The shadow root contains a `<style>` element, a `<blockquote>` element, and a `<slot name='text'>` element. The `<slot>` element contains the text 'empty'. The `<blockquote>` element contains the `<slot>` element. The `<custom-quote>` element is highlighted in the DOM tree. The breadcrumb at the bottom shows the path: `html > body > custom-quote`. The 'Styles' panel is also visible at the bottom.

```
<html>
  <head>...</head>
  <body>
    ... <custom-quote> == $0
      #shadow-root (open)
        <style>...</style>
        <blockquote>
          ::before
          <slot name="text">
            "empty"
            ↳ #text reveal
          </slot>
          ::after
        </blockquote>
      </custom-quote>
    </body>
  </html>
```

html body custom-quote

Styles Event Listeners DOM Breakpoints Properties

Filter :hov .cls +

```
<custom-quote><p slot="text">Some people would rather die, than think. In fact, they do.</p></custom-quote>
```

<<

Some people would
rather die, than think.
In fact, they do.

>>

<https://jsfiddle.net/ErBlack/5vze862v/3/>

The screenshot shows the 'Elements' panel of a web browser's developer tools. The DOM tree is expanded to show the following structure:

- `<html>`
 - `<head>...</head>`
 - `<body>`
 - `<custom-quote>`
 - `#shadow-root (open)`
 - `<style>...</style>`
 - `<blockquote>`
 - `::before`
 - `<slot name="text">`
 - `"empty"`
 - `<p>` (This element is highlighted with an orange background)
 - `</slot>`
 - `::after`
 - `</blockquote>`
 - `<p slot="text">Some people would rather die, than think. In fact, they do.</p>`
 - `</custom-quote>`
 - `</body>`
 - `</html>`

Below the DOM tree, a breadcrumb trail shows the path: `html > body > custom-quote > p`, with `p` being the selected element. At the bottom, the 'Styles' panel is visible, showing a filter input and a list of styles including `:hov` and `.cls`.

```
p {  
    margin: 0 0 0 1em;  
}
```

```
p:first-letter {  
    font-size: 125%;  
    color: crimson;  
}
```



Some people would rather die, than think. In fact, they do.



Пример

Напишем date-picker

10-12-88

← October 1988 →

| Mo | Tu | We | Th | Fr | Sa | Su |
|----|----|----|----|----|----|----|
| | | | | | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | | | | | | |

Шаблон

```
<style class="main">...</style>
<input type="text" placeholder="MM-DD-YYYY" />
<div class="calendar">
  <div class="header">
    <div>&larr;</div><div class="month"></div><div>&rarr;</div>
  </div>
  <div class="daynames">Mo Tu We Th Fr Sa Su</div>
  <div class="days">
    <div class="grid"></div>
  </div>
</div>
```

Замена HTML imports

Пока про импорты не договорились подключим шаблон через `html-loader`

```
const template = document.createElement('template');  
  
template.innerHTML = require('./template.html');  
  
module.exports = template;
```

```
const template = require('./template');

class DatePicker extends HTMLElement {
  constructor() {
    super();

    const shadow = this.attachShadow({mode: 'open'});
    const content = document.importNode(template.content, true);

    shadow.appendChild(this._initContent(content));
  }
}

customElements.define('date-picker', DatePicker);
```

```
class DatePicker extends HTMLElement {  
    /* ... */  
    _initContent(content) {  
        this._input = content.querySelector('input');  
        this._input.value = this.getAttribute('value');  
  
        this._input.addEventListener('input', this._onInput);  
        this._input.addEventListener('focus', this._onInputFocus);  
        this._input.addEventListener('blur', this._onInputBlur);  
  
        /* ... */  
  
        return content;  
    }  
}
```

Выбранное значение

```
class DatePicker extends HTMLElement {  
  
    get value() {  
        return this._input.value;  
    }  
  
    set value(value) {  
        this._input.value = value;  
    }  
}
```

Обработка изменения атрибута

```
class DatePicker extends HTMLElement {  
    static get observedAttributes() {  
        return ['value'];  
    }  
  
    /* ... */  
  
    attributeChangedCallback(attrName, oldVal, newVal) {  
        if (attrName === 'value') this.value = newVal;  
    }  
}
```

События

```
class DatePicker extends HTMLElement {  
  set value(value) {  
    this._input.value = value;  
    this._handleChange(Date.parse(value));  
  }  
  _handleChange(parsed) {  
    this.dispatchEvent(new CustomEvent('change', {detail: parsed}));  
    this.dispatchEvent(new CustomEvent('input', {  
      bubbles: true,  
      detail: parsed  
    }));  
  }  
}
```

Где посмотреть



<https://github.com/ErBlack/date-picker>

Что получилось?



Пример

```
<date-picker></date-picker>
```

```
<date-picker value="02.28.2018"></date-picker>
```

```
require( '../lib/DatePicker' );
```

```
const datePicker = document.querySelector( 'date-picker' );
```

```
datePicker.addEventListener( 'change', function(e) {
```

```
    //... console.log
```

```
})
```

Пример на React

```
const React = require('react');
const ReactDOM = require('react-dom');
const Component = require('./Component');

ReactDOM.render(
  <Component value="03-17-2018"/>,
  document.getElementById('component')
);
```

```
const React = require('react');
require('../lib/DatePicker');

class Component extends React.Component {
  constructor() {
    super();
    this._onInput = this._onInput.bind(this);
  }
  render() {
    return <date-picker
      value={this.props.value} onInput={this._onInput}
    ></date-picker>;
  }
  _onInput(e) { //... console.log }
}
```

▼ Пример на Vue

```
<div id="app">  
  <date-picker v-bind:value="date" v-on:change="onChange"></date-picker>  
</div>
```

```
const Vue = require('vue');  
require('../lib/DatePicker');
```

```
const app = new Vue({  
  el: '#app',  
  data: {date: '02-25-2018'},  
  methods: {onChange(e) { //... console.log }}  
});
```

В итоге

- Не нужны зависимости
- Не важен стек
- Изолированные стили
- Нужны зависимости
- Всё ещё черновик
- Нет достаточной поддержки

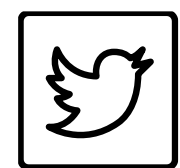
Спасибо за внимание



Вопросы

Артём Пендюрин

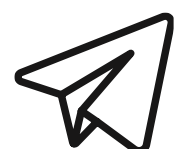
разработчик интерфейсов



<https://twitter.com/ErBlack>



<https://github.com/ErBlack>



<https://t.me/ErBlack>