無料電子ブック

学習

Web Component

Free unaffiliated eBook created from
Stack Overflow contributors.

#web-component

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: web-component

It is an unofficial and free Web Component ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Web Component.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: Webコンポーネントのい

このセクションでは、*Web*コンポーネントのと、が*Web*コンポーネントをするについてをします。

Webコンポーネントは、のWebブラウザにされたのしいWebテクノロジであり、HTML、JavaScript、CSSののけをりてなWebをするためにされます。

*Web Compoments*というにまれるトピックはのとおりです。

- カスタム
- HTMLテンプレート
- DOM
- HTMLのインポート

これらのはであり、にまたは々にすることができます。

バージョン

コンポーネント		のリリース
HTMLテンプレート	W3C HTML5	2014-10-28
カスタム	W3Cワーキングドラフトまたは WHATWG HTMLとDOMの	20161013
DOM	W3Cワーキングドラフトまたは WHATWG HTMLとDOMの	2017-01-16
HTMLのインポート	W3Cワーキングドラフト	2016-02-25

Examples

ネイティブ

`<template>`は、のすべてのブラウザでされています。

- クロム、
- エッジ、

- Firefox、
- オペラ、
- サファリ、
- ...

カスタムエレメント`customElements.define()`、Shadow DOM `attachShadow()`、HTML Imports `<link rel="import">`は、ChromeとOperaのバージョンでされています。

ポリフィル

のブラウザでは、polyfillライブラリをできます。

- カスタムの WebReflectionまたはWebcomponents.orgから、
- シャドウDOMの Webcomponents.org 、
- テンプレート Neovov 、
- HTMLのインポートのために Webcomponents.orgから

HTMLテンプレート - Hello World

`<template>`をして、コードでできるHTMLテンプレートをします。

```
<template id="Template1">
    Hello, World !
<template>

<div id="Target1"></div>

<script>
   Target1.appendChild( Template1.content.cloneNode( true ) )
</script>
```

`#Target1` divにテンプレートのがされます。

カスタム - Hello World

"Hello、World"とされる`<hello-world>`というのしいHTMLタグをします。

```
<script>
//define a class extending HTMLElement
class HelloWorld extends HTMLElement {
    connectedCallback () {
      this.innerHTML = 'Hello, World!'
    }
}

//register the new custom element
customElements.define( 'hello-world', HelloWorld )
</script>

<!-- make use the custom element -->
<hello-world></hello-world>
```

シャドーDOM - Hello World

"Hello、World"とされる`div` Shadow DOMをします。そののののわりに。

```
<div id="Div1">intial content</div>

<script>
   var shadow = Div1.attachShadow( { mode: 'open' } )
   shadow.innerHTML = "Hello, World!"
</script>
```

HTMLのインポート - Hello World

「Hello、World」でdivをするHTMLファイルをインポートします。メインのDOMツリーのわりに。

インポートされたファイル*hello.html*

```
<script>
   var div = document.createElement( 'div' )
   div.innerHTML = 'Hello, World!'
   document.body.appendChild( div )
</script>
```

メインファイル*index.html*

```
<html>
  <link rel="import" href="hello.html">
```

Hello Worldの

このでは、Custom Element、Template、Shadow DOMおよびHTML Importをみわせて、「Hello、World」をします。 HTMLの。

ファイル`hello-world.html`

```
<!-- 1. Define the template -->
<template>
   Hello, World!
</template>

<script>
  var template = document.currentScript.ownerDocument.querySelector( 'template' )

  //2. Define the custom element

  customElements.define( 'hello-world', class extends HTMLElement
  {
      constructor()
      {
          //3. Create a Shadow DOM
```

```
          var sh = this.attachShadow( { mode: 'open' } )
          sh.appendChild( document.importNode( template.content, true ) )
      }
  } )
</script>
```

メインファイル`index.html`

```
<html>
<head>
    <!-- 4. Import the HTML component -->
    <link rel="import" href="hello-world.html">
</head>
<body>
    <hello-world></hello-world>
</body>
</html>
```

オンラインでWebコンポーネントのいをむ https://riptutorial.com/ja/web-component/topic/8239/webコンポーネントのい

2: Webコンポーネントのテスト

き

スタイル、テンプレート、コンポーネントクラスをってコンポーネントをテストしたいときにすべきこと。

Examples

WebpackとJest

JestはReactアプリケーションをむすべてのJavaScriptコードをテストするためにFacebookによってされています。 Jestののの1つは、された「ゼロ」のをすることです。エンジニアにすぐにできるツールがされると、よりくのテストがされ、としてよりしたなコードベースがされることがわかりました。

GitHubのweb-components-webpack-es6-boilerplateとしてながあります。

はとNodeJSのには、テストしますjsdomを 。のプロセスはです。プロジェクトがのようになっているとして、 Webpackのにってみましょう。

```
-src
    --client
    --server
-webpack
    --config.js
package.json
```

`server render`ロジックをのものからする`server render`されたなディレクトリ。 **Webpack** `config.js`ファイルにはのモジュールがまれます

```
  resolve: {
    modules: ["node_modules"],
    alias: {
      client: path.join(__dirname, "../src/client"),
      server: path.join(__dirname, "../src/server")
    },
    extensions: [".js", ".json", ".scss"]
  },
```

Webpackをするように**Jest**をすることができます。

```
module.exports = {
  setupTestFrameworkScriptFile: "<rootDir>/bin/jest.js",
  mapCoverage: true,
  moduleFileExtensions: ["js", "scss", "html"],
  moduleDirectories: ["node_modules"],
```

```
  moduleNameMapper: {
    "src/(.*)$": "<rootDir>/src/$1"
  },
  transform: {
    "^.+\\.(js|html|scss)$": "<rootDir>/bin/preprocessor.js"
  },
  testMatch: ["<rootDir>/test/**/?(*.)(spec|test).js"],
  testPathIgnorePatterns: ["<rootDir>/(node_modules|bin|build)"]
};
```

どこでこのをするがありますか

`jest` keyののの`package.json`ファイルでうか、こののようにプロジェクトルートの`jest.config.js`ファイルをします。

したいのは、たちの`html`ファイルがしくインポートされるようにすることです。つまり、 `js`ファイルのファイルをしようとすると、 `babel-jest`だけをするとエラーがスローされるため、カスタム`preprocessor`でエスケープしてしまうことになります。

ここでは、のなことはある`setupTestFrameworkScriptFile`にまれたスクリプト`custom elements`にpolyfillsを`jsdom`。 `preprocessor.js`いはのとおりです

```
const babelJest = require("babel-jest");

const STYLE_URLS_REGEX = /styles:\s*\[\s*((?:'|").*\s*(?:'|")).*\s*.*\]/g;
const ESCAPE_TEMPLATE_REGEX = /(\${|\`)/g;

module.exports.process = (src, path, config) => {
  if (path.endsWith(".html")) {
    src = src.replace(ESCAPE_TEMPLATE_REGEX, "\\$1");
    src = "module.exports=`" + src + "`;";
  }
  src = src.replace(STYLE_URLS_REGEX, "styles: []");

  return babelJest.process(src, path, config);
};
```

このスクリプトがうことはですスタイルファイルのを、 `require('template.html')`でインポートするときに、テンプレートをテストするがない/にして、エスケープするがないようにします。に、コンテンツをバベルトランスにします。

にうべきなことは、 `web components polyfills`をめることです。デフォルトでは、 `jsdom`はまだそれらをサポートしていません。これをうには、 `setupTestFrameworkScriptFile`をこのでするだけです`jest.js`にはのが`jest.js`れています

```
require("document-register-element/pony")(window);
```

このようにして、 `jsdom web components` **API**にアクセスできます。

すべてをしたら、のようなにするがあります

```
-bin
    --jest.js
    --preprocessor.js
-src
    --client
    --server
-webpack
    --config.js
-test
package.json
jest.config.js
```

`test`ディレクトリでテストをけ、コマンド `yarn run jest --no-cache --config $(node jest.config.js)` て`test`をし`yarn run jest --no-cache --config $(node jest.config.js)`。

オンラインでWebコンポーネントのテストをむ https://riptutorial.com/ja/web-component/topic/10057/webコンポーネントのテスト

クレジット

S. No		Contributors
1	Webコンポーネントのい	Community, Mike, Supersharp
2	Webコンポーネントのテスト	Vardius