

Rapid SPA

Single Page Application

— just simplified

Thanks



Consulting Engineer, Chief Architect – Front-End Engineering CoE @Unisys

Disclaimer:


The statements I make here are mine alone and do not necessarily reflect the views of Unisys.

Author – SPA.js

who I am

Quick Intro

spa.js.org

 SPA JS

HomeGet StartedBlog

Single Page Application - Just Simplified


Liberate your development
Write less, Do more.

2.67.0


[Get Started](#)[Download](#)[GitHub](#)

POWERFUL yet SIMPLE


SPA JS is built on the popular paradigm "KISS" (Keep It Simple, Stupid) that emphasizes on keeping a framework simple and not complicating it. Ideally a powerful framework should do all the heavy lifting and must make it easy for the developers to focus on the product features and not the framework. The SPA JS framework is powerful yet simple to learn, easy to adapt and easy to scale. A component developed once can easily be deployed in any other SPA application without complex configurations. Give it a try, you will love it.




Write Less,
Deliver Faster



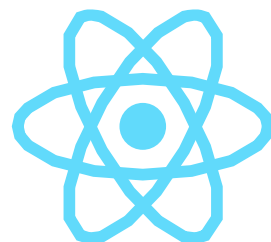
All Platforms,
Cross Browsers



Component Based,
Reusable

Copyright © 2005 - 2019 SPA JS v2.67.0 | MIT | 

Why not ?



Knockout.

ember



Thomas "Kernel Panic" Fuchs

@thomasfuchs



Follow



~15 years trying to make everyone separate HTML, JS & CSS. And then suddenly everything went south and we're writing code like this.

```
1 export default class TodoList extends Component {
2   state = { todos: [], text: '' };
3   setText = e => {
4     this.setState({ text: e.target.value });
5   };
6   addTodo = () => {
7     let { todos, text } = this.state;
8     todos = todos.concat({ text });
9     this.setState({ todos, text: '' });
10  };
11  render({ }, { todos, text }) {
12    return (
13      <form onSubmit={this.addTodo} action="javascript:">
14        <input value={text} onChange={this.setText} />
15        <button type="submit">Add</button>
16        <ul>
17          { todos.map( todo => (
18            <li>{todo.text}</li>
19          )) }
20        </ul>
21      </form>
22    );
23  }
24 }
```

8:30 AM - 19 Dec 2016

○ <https://twitter.com/thomasfuchs/status/810885087214637057>

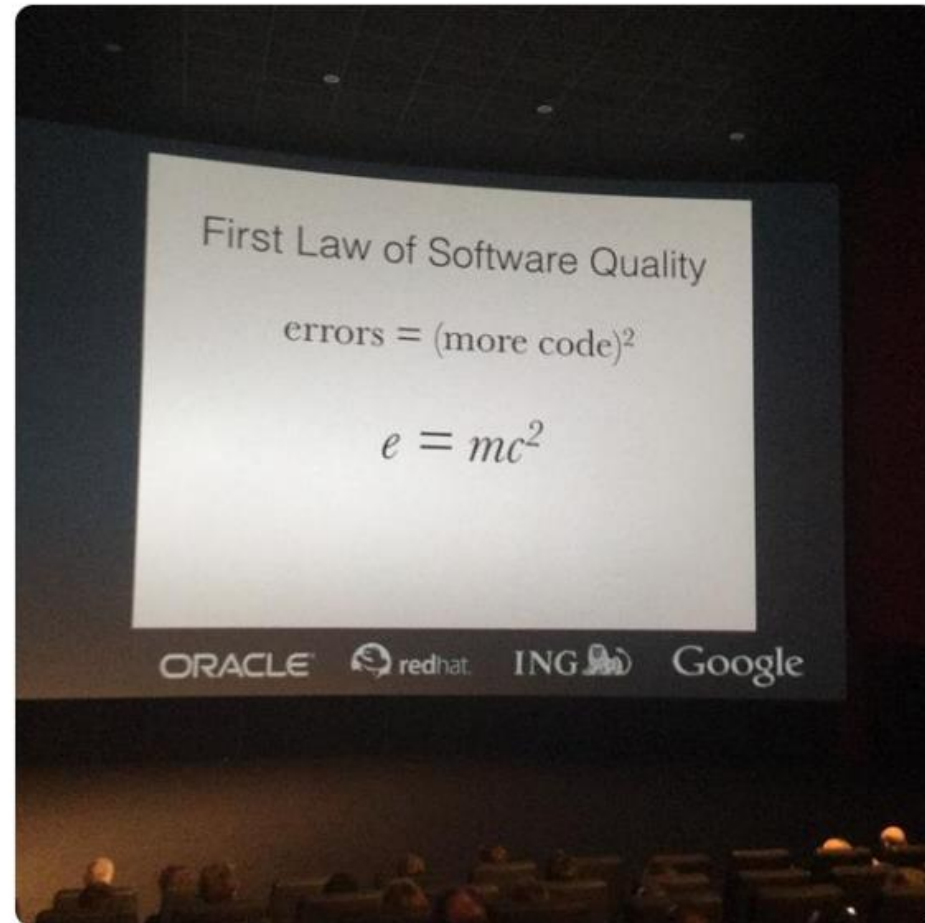
Mike Richards

@MMRichards

Follow



First law of software quality: errors = (more code)², or $e = mc^2$



2:26 PM - 25 May 2015 from [Decatur, GA](#)

○ <https://twitter.com/thomasfuchs/status/810885087214637057>



The way you get programmer productivity is not by increasing the lines of code per programmer per day. That doesn't work. **The way you get programmer productivity is by eliminating lines of code you have to write.** Right? The line of code that's the *fastest to write*, that *never breaks*, that *doesn't need maintenance*, **is the line you never had to write** – right? So, **the goal here is to eliminate 80% of the code that you have to write for your app. That's the goal.** And so, along the way, if we can provide WYS –this and WYS-that and visual this and visual that... well that's fine. But **the high order bit is to eliminate 80% of the code.** When you drag a line into the interface builder you're eliminating a line of code in one form. But that only goes so far. Maybe it can go further. I've seen a lot of demos that try to take it all the way back into **the algorithmic part of the code base and none of them have ever been any good.**

Get Started

Let's get started with building Single Page Application. Building a SPA begins with creating a simple folder structure inorder to keep the files organized.

SPA Folder Structure

Create the following base folder structure under the application context.

Example:

AppContext: `myApp`

Web Access: `http://host:port/ myApp`

`myApp` folder structure:

```
└─ MyApp
  └─ index.html
    └─ app
      └─ components
      └─ js
      └─ css
      └─ language
    └─ xlib
      └─ JQuery
      └─ handlebars
      └─ spa
```

The `index.html` is the starting page of the application. Depending on the deployment technology stack the file/extension can be `index.htm` or `index.php` or `index.cgi` or `index.pl` or `index.cshtml` or `default.html`

The default starting page must be configured in the respective web server.

```
app/components – for the web page components
app/css – for the css files
app/js – for common javascript files
app/language – for i18n language files
xlib – for 3rd party library/framework files. It's highly recommended to create sub folder for each of those libraries/frameworks.
```

In some cases (bower or npm or yarn package downloader), the 3rd party libraries could be organized in other folders like `bower_components` or `node_modules`.

The relevant files from these folders (.js and/or .css) will be referred in application start page (`index.html`)

Editor Extension

Install [Visual Studio Code Extension](#) for code snippets in HTML and Javascript files.

Using CLI

Unlike other frameworks, Single Page Applications can be built using SPA.js without a CLI tool. However, if you prefer to create App and Component Structure quickly with some basic code snippets, use `spa-cli`.

[Collapse](#) | [Expand](#)[Get Started](#)[Downloading SPA JS](#)[Application Starting Page](#)[SPA Component](#) >[SPA API \(The Model\)](#) >[SPA i18n](#) >[SPA Forms](#) >[SPA Routes](#) >[SPA Events](#) >[SPA Defaults](#) >[SPA Utils](#) >[SPA Extend](#) >[SPA Examples and Videos](#) v[Examples](#)[Videos](#)[Feedback / Contact](#)[Collapse](#) | [Expand](#)

Feedback / Contact

[< Previous](#)[Next >](#)

What do you think of SPA.js? Feel free to write your comments.

Name *

Email *

Subject *

Message *