

ReactJS Girls Conf 2019

Automate Your React Workflow

By: Monica Powell

@waterproofheart



Hi, I'm Monica!

I'm a Full Stack Engineer at Meetup & organizer of the React Ladies Group in NYC

I've written about git and other tech adventures in publications such as FreeCodeCamp, Hacker Noon and Code Like A Girl.



Follow Along

- Slides:
 - <http://aboutmonica.com/automationgeneration>
- Code:
 - <https://github.com/M0nica/generate-react-components>
 - <https://github.com/M0nica/generate-kawaii-components>

Overview

1) Introduction

**2) React App
Architecture**

**3) What is
Scaffolding?**

**4) Scaffolding
Options**

5) DIY Scaffolding

What does a typical React app look like?

- UI components are the building blocks of React apps
- Multiple components require test, CSS, translation, etc. files locally scoped to each component

```
common/  
  Avatar.jsx  
  Avatar.css  
  APIUtils.jsx  
  APIUtils.test.jsx  
feed/  
  index.jsx  
  Feed.jsx  
  Feed.css  
  FeedStory.jsx  
  FeedStory.test.jsx  
  FeedAPI.jsx  
profile/  
  index.jsx  
  Profile.jsx  
  ProfileHeader.jsx  
  ProfileHeader.css  
  ProfileAPI.jsx
```

Example modified from: <https://reactjs.org/docs/>

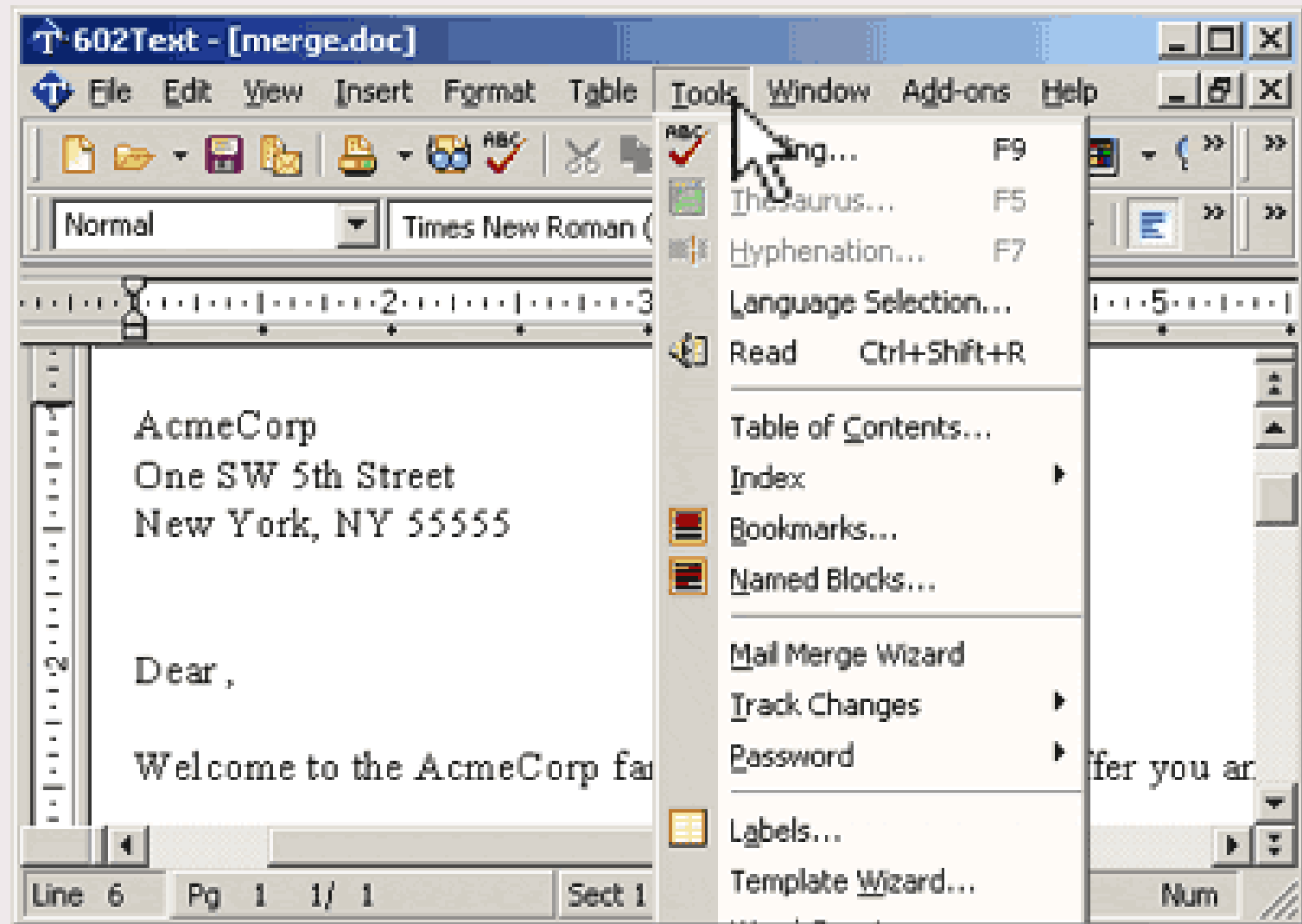
Adding components can be repetitive...



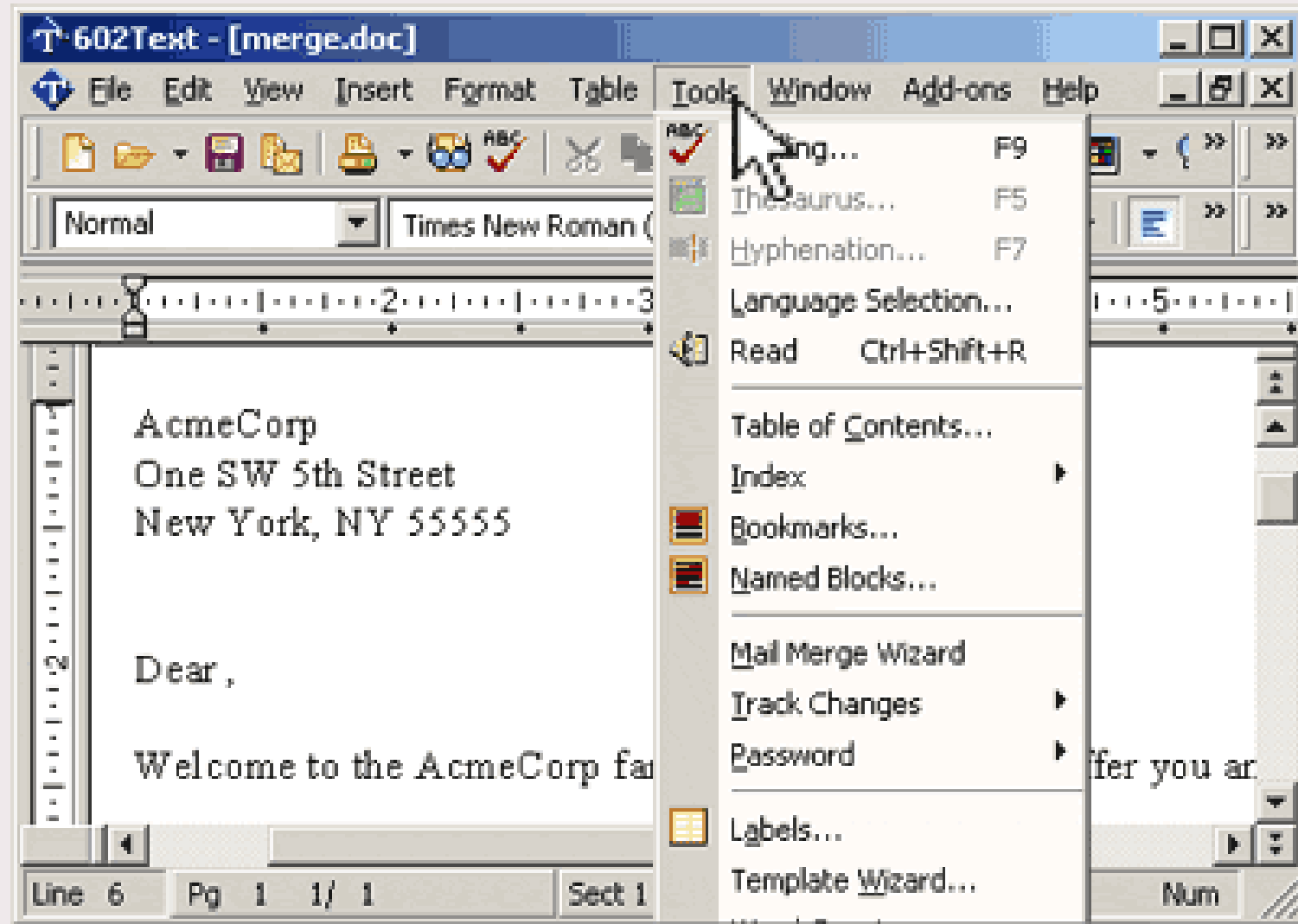


What if you want consistency across components **without** having to copy + paste + then manually change code between files?

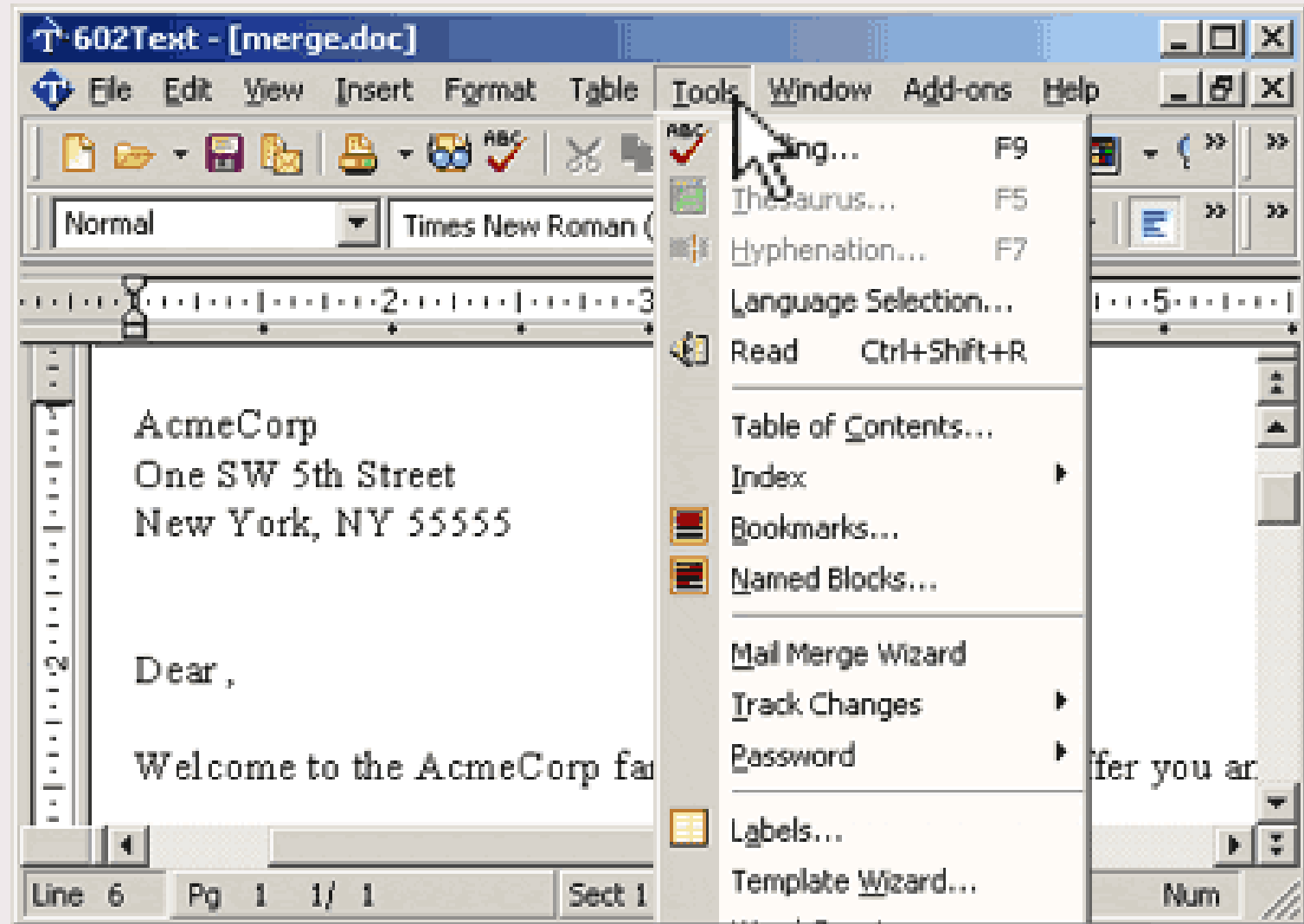
The answer is mail merge



The answer is ~~mail merge~~



The answer is templates



Is it worth the time?

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

		HOW OFTEN YOU DO THE TASK					
		50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
HOW MUCH TIME YOU SHAVE OFF	1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
	5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
	30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
	1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
	5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
	30 MINUTES		6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
	1 HOUR		10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
	6 HOURS				2 MONTHS	2 WEEKS	1 DAY
1 DAY					8 WEEKS	5 DAYS	

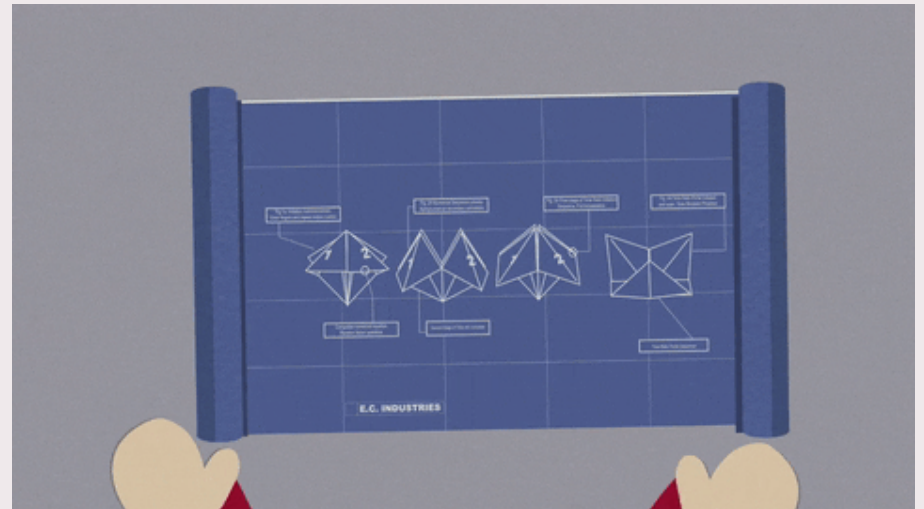
Scaffolding

- Scaffolding software is the process of generating starter-code based on templates.
- The starter-code is comprised of boilerplate code that is shared between all code components that share the same conventions.
- Minimal Viable Component != ready for production.

Scaffolding

Some benefits of scaffolding are:

- Reduce time to generate new projects or components.
- less error-prone than copy & pasting & editing
- encourage consistency and implementation of design patterns



Scaffolding Can Improve Developer Experience

Having an **efficient** way to **consistently** implement standards minimizes certain questions about conventions and helps engineers have a more **enjoyable** onboarding experience when working with a new codebase.

Scaffolding Can Answer FAQs



// Where can I find the translations for this component?

// What's the current source of truth for our code standards?

// Does every component go in the same folder?

// How are components named?

// What style should JavaScript components be written in?

Scaffolding Tools

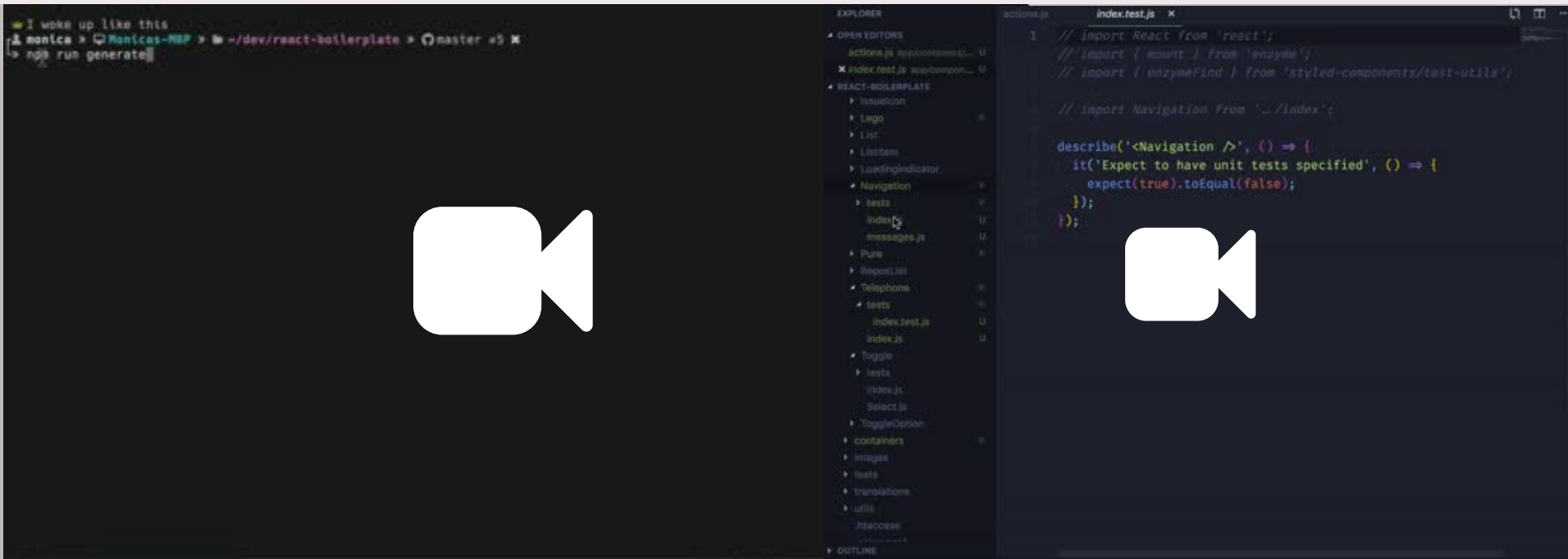


React-Boilerplate



- [React-boilerplate](#) reduces overhead for developing a React app optimized for **production**
- **One-size-fits-all** approach can unnecessarily increase complexity and dependencies
- Includes CLI tool to scaffold React components, containers, routes, selectors and sagas₂

React-Boilerplate



The screenshot displays a code editor interface with a dark theme. On the left, a terminal window shows the command `ng generate` being executed. The main editor area is split into two panes. The left pane shows the Explorer view with a file tree for a project named 'react-boilerplate'. The tree includes folders for 'Navigation', 'Pure', 'Telephone', 'Toggle', and 'containers', among others. The 'Navigation' folder is expanded, showing 'index.test.js' and 'messages.js'. The right pane shows the code editor for 'index.test.js', containing the following code:

```
1 // import React from 'react';  
  // import { mount } from 'enzyme';  
  // import { enzymeFind } from 'styled-components/test-utils';  
  
  // import Navigation from '../index';  
  
  describe('<Navigation />', () => {  
    it('Expect to have unit tests specified', () => {  
      expect(true).toEqual(false);  
    });  
  });
```



Separate Templates Based on Type of Component

```
1 import React, { Component } from 'react';
2
3 export class MyComponent extends Component {
4   render() {
5     return (
6       <div></div>
7     );
8   }
9 }
```

Class

```
1 const MyComponent = (props) => {
2   return (
3     <div></div>
4   );
5 }
```

Functional

Ignite



- [React Ignite](#) is similar to React-boilerplate but for React Native
- Includes various customizable tools to jump start the creation of React Native applications
- *"Ignite speeds up your app development by over two weeks on average."*

Should developer use existing React scaffolding tools?

The beauty of an effective scaffolding tool is that these tools are designed to be **highly-customizable** to fit your needs without having to compromise on your standards or best practices.

Code Generators

More lightweight scaffolding solutions than
React Boilerplate and React Ignite



generator nod npm v0.4.0 build passing coverage 100%

*Tool for generating React components by replicating your own.
It's intended to work no matter how your file structure is.*



Hygen

The scalable code generator that saves you time.

PLOP anything!

Getting Started

Plop on Github

- github.com/diegohaz/generact
- github.com/jondot/hygen
- github.com/amwmedia/plop

PlopJS

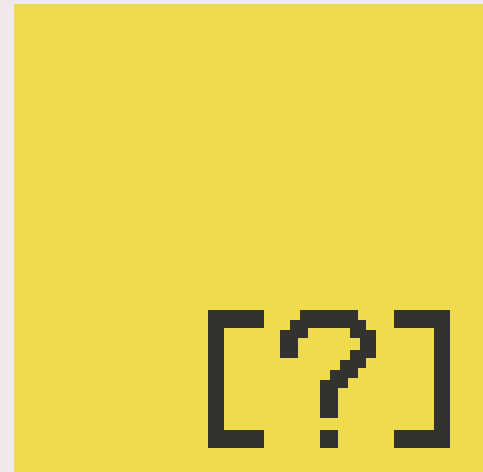
- PlopJS is a microgenerator, and is used under the hood in React-Boilerplate
- It generates files based on user input via a Command Line Interface (CLI).
- Minimal setup required
- plopjs.com

PlopJS

handlebars



+



- Handlebars templating language
- (similar to Mustache)

- Inquirer.js is a a collection of common interactive command line user interfaces.

Let's generate Kawaii Components

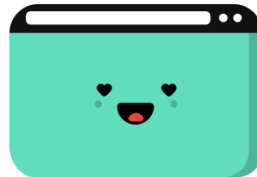
React Kawaii



React Kawaii is a library of cute SVG illustrations (react components). Ideal if you want to give some cuteness and personality to your react application.

npm v0.12.0

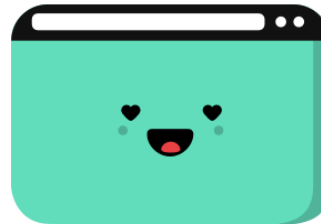
Example



```
<Browser size="200" mood="lovestruck" color="#61DDBC" />
```

Structure of Kawaii Components

```
1 import { Backpack } from 'react-kawaii';  
2  
3 const Example = () => <Backpack size={ 300 } mood="lovestruck" color="#EC4067" />;
```



```
<Browser size="200" mood="lovestruck" color="#61DDBC" />
```

<https://github.com/miukimiu/react-kawaii>

@waterproofheart

Getting started with Plop

Now what do we want to do with the user
input we collected?

```
1 /**
2  *
3  * {{properCase mood}}{{properCase kawaiiComponent}}
4  * Description: This component is a {{mood}} {{kawaiiComponent}}.
5  */
6
7 import React from 'react';
8 import { {{kawaiiComponent}} } from 'react-kawaii'
9
10 function {{properCase mood}}{{properCase kawaiiComponent}}() {
11   return (
12     <{{kawaiiComponent}} size={{size}} mood="{{mood}}" color="{{color}}" />
13
14   );
15 }
16
17 {{properCase mood}}{{properCase kawaiiComponent}}.propTypes = {};
18
19 export default {{properCase mood}}{{properCase kawaiiComponent}};
```

Getting started with Plop

Now what do we want to do with the user
input we collected?

```
1 /**
2  *
3  * {{properCase mood}}{{properCase kawaiiComponent}}
4  * Description: This component is a {{mood}} {{kawaiiComponent}}.
5  */
6
7 import React from 'react';
8 import { {{kawaiiComponent}} } from 'react-kawaii'
9
10 function {{properCase mood}}{{properCase kawaiiComponent}}() {
11   return (
12     <{{kawaiiComponent}} size={{size}} mood="{{mood}}" color="{{color}}" />
13
14   );
15 }
16
17 {{properCase mood}}{{properCase kawaiiComponent}}.propTypes = {};
18
19 export default {{properCase mood}}{{properCase kawaiiComponent}};
```

Getting started with Plop

Plop is essentially a node module that accesses the plop API through a plop object that is passed in as a parameter.

```
1 module.exports = function (plop) {  
2   // create your generators here  
3   plop.setGenerator('basics', {  
4     description: 'this is a skeleton plopfile',  
5     prompts: [], // array of inquirer prompts  
6     actions: [] // array of actions  
7   });  
8 };
```

Getting started with Plop

Plop is essentially a node module that accesses the plop API through a plop object that is passed in as a parameter.

```
1 module.exports = function (plop) {  
2   // create your generators here  
3   plop.setGenerator('basics', {  
4     description: 'this is a skeleton plopfile',  
5     prompts: [], // array of inquirer prompts  
6     actions: [] // array of actions  
7   });  
8 };
```

Which component should we import?

Let's add out first prompt to select which Kawaii component we want to import

```
1 {
2   type: "list",
3   name: "kawaiiComponent",
4   message: "Which Kawaii component would you like to generate?:",
5   choices: [
6     "Backpack",
7     "Browser",
8     "Cat",
9     "CreditCard",
10    "File",
11    "Ghost",
12    "IceCream",
13    "Mug",
14    "Planet",
15    "SpeechBubble"
16  ]
17 },
```

Which component should we import?

Let's add out first prompt to select which Kawaii component we want to import

```
1 {
2   type: "list",
3   name: "kawaiiComponent",
4   message: "Which Kawaii component would you like to generate?:",
5   choices: [
6     "Backpack",
7     "Browser",
8     "Cat",
9     "CreditCard",
10    "File",
11    "Ghost",
12    "IceCream",
13    "Mug",
14    "Planet",
15    "SpeechBubble"
16  ]
17 },
```


What color should component be?

Let's add out first prompt to select which color we want our component to be

```
1  {
2    type: "list",
3    name: "color",
4    message: "What color should the component be:",
5    choices: ["eggplant", "mint", "bubblegum pink", "purple", "lime"],
6    filter: colour =>
7      ({
8        "eggplant": "#2E294E",
9        "mint": "#1B998B",
10       "bubblegum pink": "#EC4067",
11       "purple": "#9A39AB",
12       "lime": "#C5D86D"
13     }[colour])
14   },
```

What color should component be?

Let's add out first prompt to select which color we want our component to be

```
1  {
2    type: "list",
3    name: "color",
4    message: "What color should the component be:",
5    choices: ["eggplant", "mint", "bubblegum pink", "purple", "lime"],
6    filter: colour =>
7      ({
8        "eggplant": "#2E294E",
9        "mint": "#1B998B",
10       "bubblegum pink": "#EC4067",
11       "purple": "#9A39AB",
12       "lime": "#C5D86D"
13     }[colour])
14  },
```

What size should our component be?

Let's create a prompt for what size our component should be

```
1  {
2      type: "list",
3      name: "size",
4      message: "What size should the component be:",
5      choices: ["extra small", "small", "medium", "large", "extra large"],
6      filter: val =>
7          ({
8              "extra small": 100,
9              "small": 200,
10             "medium": 300,
11             "large": 500,
12             "extra large": 800
13         }[val])
14 }
```

What size should our component be?

Let's create a prompt for what size our component should be

```
1  {
2      type: "list",
3      name: "size",
4      message: "What size should the component be:",
5      choices: ["extra small", "small", "medium", "large", "extra large"],
6      filter: val =>
7          ({
8              "extra small": 100,
9              "small": 200,
10             "medium": 300,
11             "large": 500,
12             "extra large": 800
13         }[val])
14 }
```

What mood should our component have?

Let's add a prompt to select which the mood we want our component to have

```
1  {
2      type: "list",
3      name: "mood",
4      message: "Mood:",
5      choices: [
6          "sad",
7          "shocked",
8          "happy",
9          "blissful",
10         "lovestruck",
11         "excited",
12         "ko"
13     ]
14 }
```

What mood should our component have?

Let's add a prompt to select which the mood we want our component to have

```
1  {
2      type: "list",
3      name: "mood",
4      message: "Mood:",
5      choices: [
6          "sad",
7          "shocked",
8          "happy",
9          "blissful",
10         "lovestruck",
11         "excited",
12         "ko"
13     ]
14 }
```

Getting started with Plop

Now what do we want to do with the user input we collected?

```
1 actions: [  
2   {  
3     type: "add",  
4     path: "src/{{properCase mood}}{{properCase kawaiiComponent}}.jsx",  
5     templateFile: "src/plop-templates/kawaii-component-template.hbs"  
6   },  
7   {  
8     type: "add",  
9     path: "src/{{properCase mood}}{{properCase kawaiiComponent}}.test.jsx",  
10    templateFile: "src/plop-templates/test.hbs"  
11  }  
12 ]
```

Getting started with Plop

Now what do we want to do with the user input we collected?

```
1 actions: [  
2   {  
3     type: "add",  
4     path: "src/{{properCase mood}}{{properCase kawaiiComponent}}.jsx",  
5     templateFile: "src/plop-templates/kawaii-component-template.hbs"  
6   },  
7   {  
8     type: "add",  
9     path: "src/{{properCase mood}}{{properCase kawaiiComponent}}.test.jsx",  
10    templateFile: "src/plop-templates/test.hbs"  
11  }  
12 ]
```

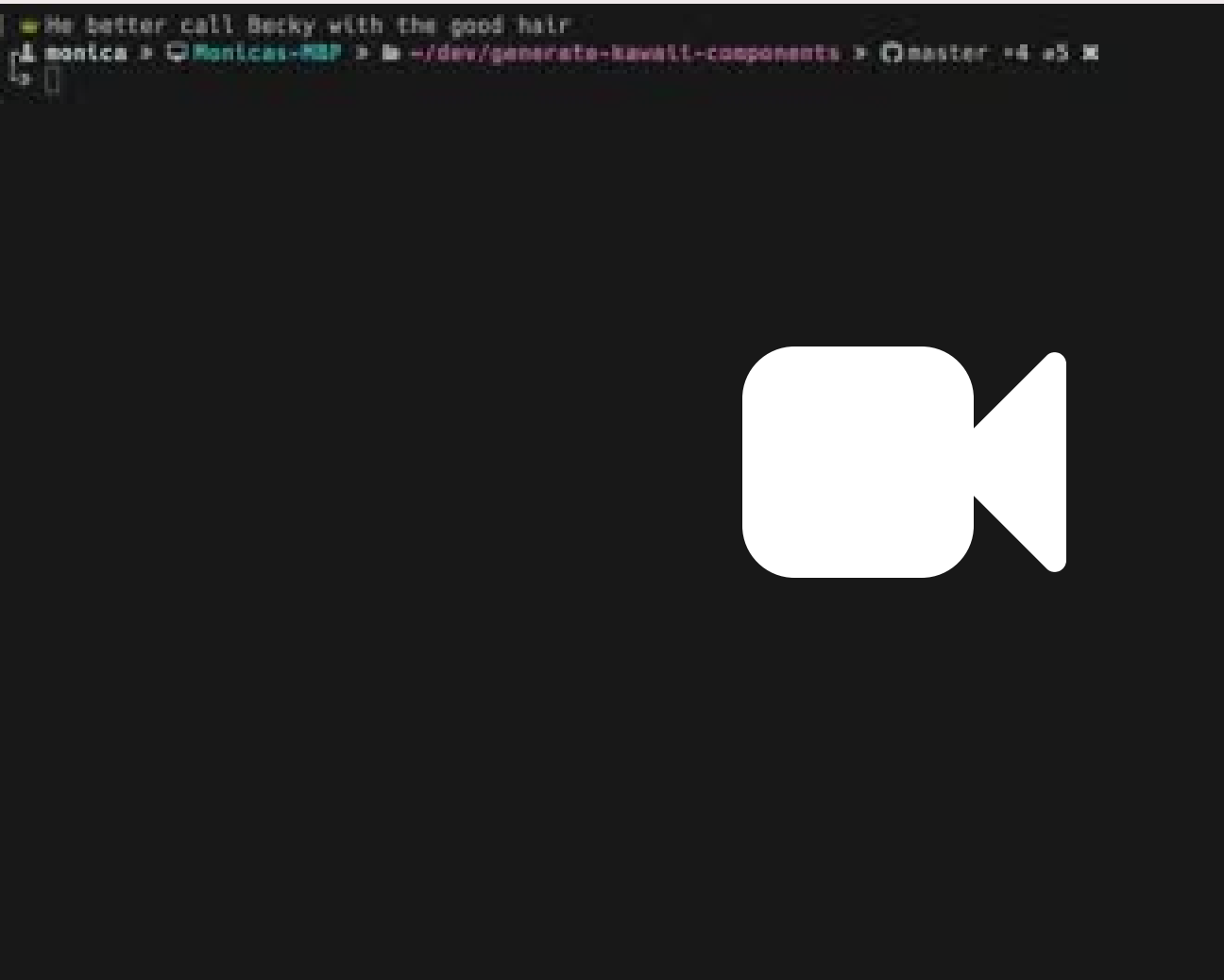

Finished Plop File

```
1 module.exports = function(plop) {
2   plop.setGenerator("Generate Kawaii Component", {
3     description: "Templates from the React Kawaii library ",
4     prompts: [
5       {
6         type: "list",
7         name: "kawaiiComponent",
8         message: "Which Kawaii component would you like to generate?:",
9         choices: [
10          "Backpack",
11          "Browser",
12          "Cat",
13          "CreditCard",...
14        ]
15      }, // truncated for example...
16    ],
17    actions: [
18      {
19        type: "add",
20        path: "src/{{properCase mood}}{{properCase kawaiiComponent}}.jsx",
21        templateFile: "src/plop-templates/kawaii-component-template.hbs"
22      },
23      {
24        type: "add",
25        path: "src/{{properCase mood}}{{properCase kawaiiComponent}}.test.jsx",
26        templateFile: "src/plop-templates/test.hbs"
27      }
28    ]
29  });
30 };
```

Finished Plop File

```
1 module.exports = function(plop) {
2   plop.setGenerator("Generate Kawaii Component", {
3     description: "Templates from the React Kawaii library ",
4     prompts: [
5       {
6         type: "list",
7         name: "kawaiiComponent",
8         message: "Which Kawaii component would you like to generate?:",
9         choices: [
10          "Backpack",
11          "Browser",
12          "Cat",
13          "CreditCard",...
14        ]
15      }, // truncated for example...
16    ],
17    actions: [
18      {
19        type: "add",
20        path: "src/{{properCase mood}}{{properCase kawaiiComponent}}.jsx",
21        templateFile: "src/plop-templates/kawaii-component-template.hbs"
22      },
23      {
24        type: "add",
25        path: "src/{{properCase mood}}{{properCase kawaiiComponent}}.test.jsx",
26        templateFile: "src/plop-templates/test.hbs"
27      }
28    ]
29  });
30 };
```

Getting started with Plop



@waterproofheart

Getting started with Plop

Now what do we want to do with the user
input we collected?

```
1 /**
2  *
3  * LovestruckBackpack
4  * Description: This component is a lovestruck
5  * Backpack.
6  */
7 import React from 'react';
8 import { Backpack } from 'react-kawaii'
9
10 function LovestruckBackpack() {
11   return (
12     <Backpack size={ 200 } mood="lovestruck"
13     color="#EC4067" />
14   );
15 }
16 LovestruckBackpack.propTypes = {};
17
18 export default LovestruckBackpack;
```



Getting started with Plop

Now what do we want to do with the user
input we collected?

```
1 /**
2  *
3  * LovestruckBackpack
4  * Description: This component is a lovestruck
5  * Backpack.
6  */
7 import React from 'react';
8 import { Backpack } from 'react-kawaii'
9
10 function LovestruckBackpack() {
11   return (
12     <Backpack size={ 200 } mood="lovestruck"
13     color="#EC4067" />
14   );
15 }
16 LovestruckBackpack.propTypes = {};
17
18 export default LovestruckBackpack;
```



Challenge

- Create a CLI tool that you can adapt to at least **two** different use cases.
- You can either use two separate templates *or* one template with conditional logic.



Challenge

Some ideas:

- generate new Gatsby blog posts and pages
- generator components from a component library that you use regularly

Examples:

- <https://github.com/M0nica/generate-react-components>
- <https://github.com/M0nica/generate-kawaii-components>

Thank You!



Twitter: **@waterproofheart**

aboutmonica.com | | **monica.dev**

If you're in NYC come to a React Ladies Meetup:

meetup.com/React-Ladies/