



write these tests now

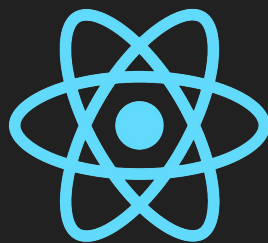
**infobip**

*Vadim Tcaregorodtcev*


Telegram  
@hadoocken

write these tests now







Ah , here we go again.

I - interface segregation

D - dependency inversion

O - open-closed

L - Liskov substitution

S - single responsibility

S O L I D

# Single Responsibility



Hmm, please don't ...



# Open/Closed

To lose weight you don't need to cut your legs off...

© Statham

# Liskov Substitution



```
1 class Animal = {};  
2  
3 class Cat extends Animal = {};  
4  
5 ...  
6  
7 let animal = new Animal();  
8  
9 ...  
10  
11 animal = new Cat(); // correct  
12  
13 ...
```

# Interface Segregation



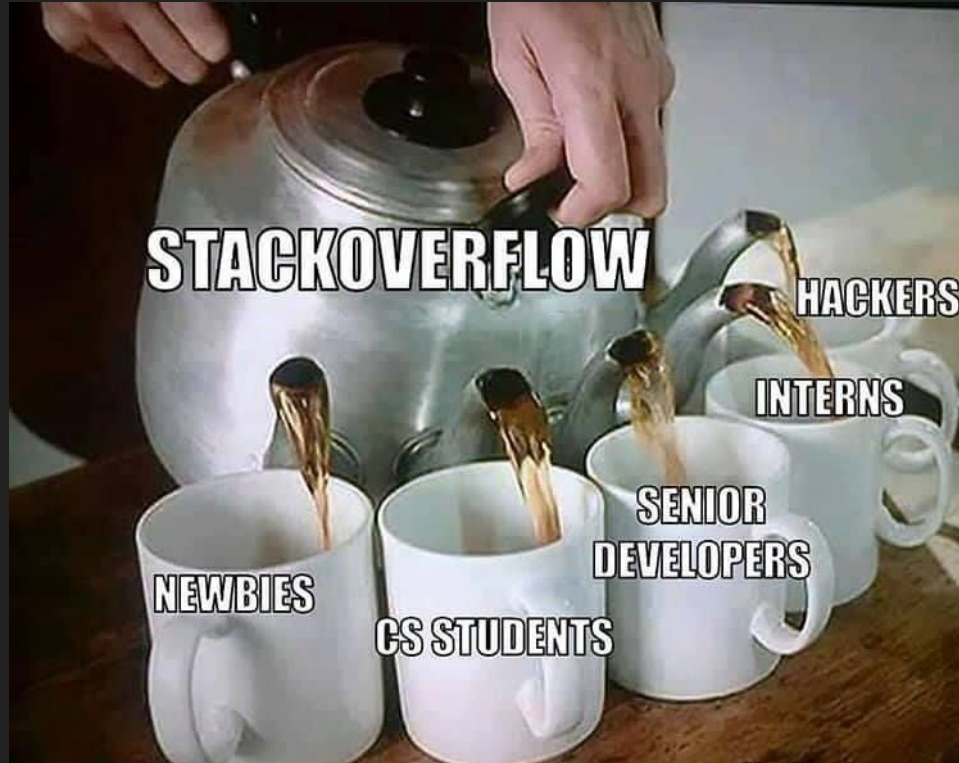
```
1 <ResponsiveGridLayout
2   {...this.props}
3   layouts={this.state.layouts}
4   onBreakpointChange={this.onBreakpointChange}
5   onLayoutChange={this.onLayoutChange}
6   // WidthProvider option
7   measureBeforeMount={false}
8   // I like to have it animate on mount. If you don't, delete `useCSSTransforms` (it's
  default `true`)
9   // and set `measureBeforeMount={true}`.
10  useCSSTransforms={this.state.mounted}
11  compactType={this.state.compactType}
12  preventCollision={!this.state.compactType}
13 >
14   {this.generateDOM( )}
15 </ResponsiveGridLayout>
16
```

# Interface Segregation



```
1 import { Container, Row, Col } from 'react-grid-system';
2
3 <Container>
4   <Row>
5     <Col sm={4}>One of three columns</Col>
6     <Col sm={4}>One of three columns</Col>
7     <Col sm={4}>One of three columns</Col>
8   </Row>
9 </Container>
10
```


# Dependency inversion



# ATOMIC



# Tokens



```
1 :root {  
2   --color-base: banana;  
3  
4   ...  
5  
6   --space-xs: 10px;  
7  
8   ...  
9  
10  --font-size-base: 16px;  
11  
12  ...  
13  
14  --animation-duration: 2s;  
15 }
```

# Atoms



```
1 <button type='submit'>Text</button>
```



```
1 <input type='text' value='text' />
```



# Molecules



```
1 <form>
2   <input type='text' />
3   <button>Search</button>
4 </form>
```

# Organisms



```
1 <header>
2   <aside>
3     <!-- aside molecule -->
4   </aside>
5   <nav>
6     <!-- nav molecule -->
7   </nav>
8 </header>
```

# Templates -> Pages



```
1 <body>
2   <header>
3     <!-- header organism -->
4   </header>
5   <main>
6     <!-- body organism -->
7   </main>
8   <footer>
9     <!-- footer organism -->
10  </footer>
11 </body>
```

S O L I D



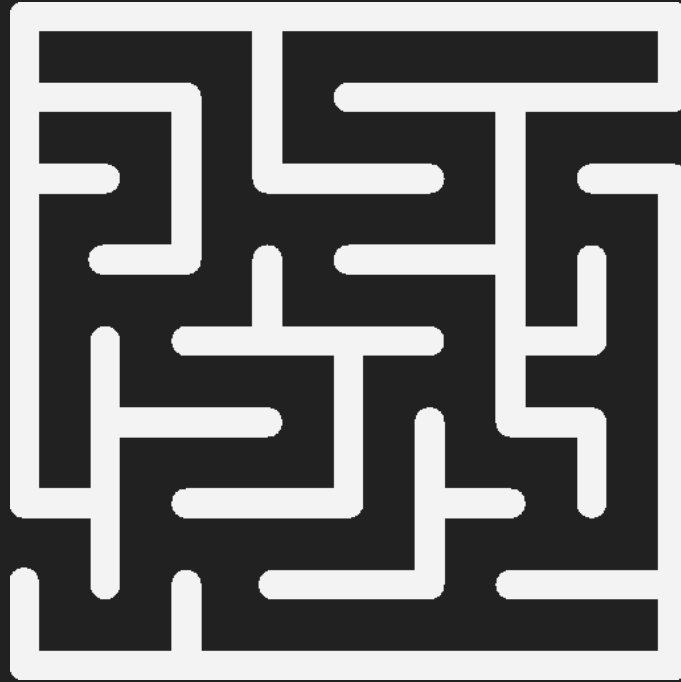
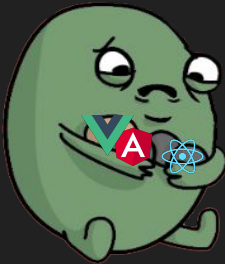
A T O M I C

# Interface

DEV (User)




Component





```
1 import { Component } from 'kit';  
2  
3 ...  
4  
5  
6 <Component />  
7  
8  
9 ...
```

# 1. Define state



```
1 <div>
2   {
3     !items.length &&
4       <p>
5         oh, looks like you miss something to show
6       </p>
7   }
8 </div>
```


Empty -> Loading -> Loaded

Loading next part -> Loading completed



Error



## 2. Define default value



```
1 function makeSome({ fn, prop }) {  
2   let result = someDefaultResult;  
3  
4   if (fn && prop) {  
5     result = fn(prop);  
6   }  
7  
8   return result;  
9 }
```



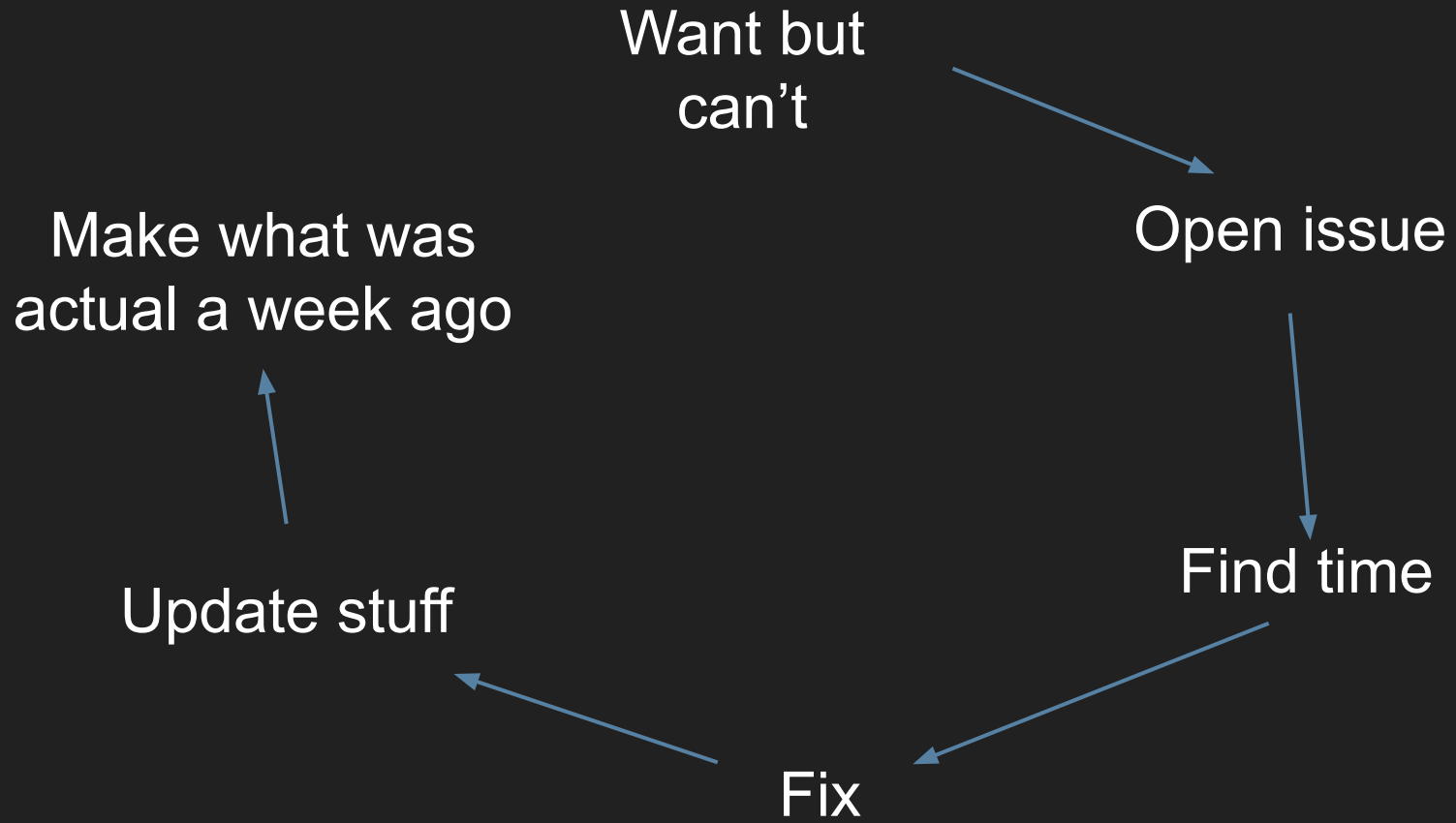
```
1 function makeSome({  
2   fn = () => someDefaultResult,  
3   prop  
4 }) => fn(prop);
```

### 3. Don't reinvent the wheel



```
1 <PasswordInput  
2   changeInputPasswordValue={handleChange}  
3   clickInputPasswordEnterButton={handleEnterClick}  
4 />
```

onChange={...}  
onEnter={...}



## 4. Let them rule




```
1 render() {  
2   const { text, defaultText } = this.props;  
3  
4   return (  
5     <span>{text ? text : defaultText}</span>  
6   );  
7 }
```

<span>

{isUndefined(text) ? defaultText : text}

</span>

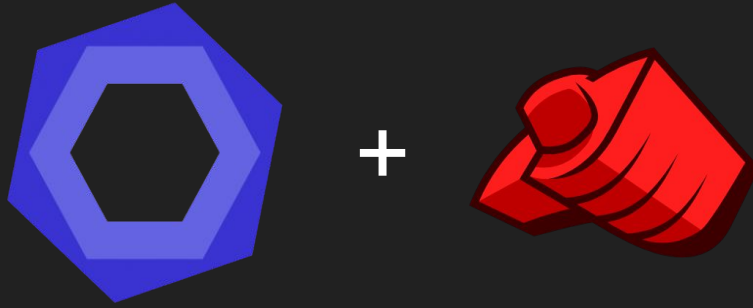
## 5. Keep It Simple, Stupid



```
1 const DEFAULT_MSG = 'ohh, looks like now you are using pure template';
2
3 render() {
4   const { msg = DEFAULT_MSG } = this.props;
5
6   return (
7     <p>
8       { msg }
9     </p>
10  );
11 }
```


1. Define state
2. Define default value
3. Don't reinvent the wheel
4. Let them ( devs ) rule
5. Keep It Simple, Stupid ( KISS )

# Hyper Automation




eslint + lefthook

# Hyper Automatization



```
1 // eslint config
2
3 {
4   ...
5
6   "camelcase": 2,
7
8   ...
9
10  "react/boolean-prop-naming": 2,
11  "react/jsx-no-literals": 2,
12
13  ...
14
15  "jsdoc/check-examples": 2
16
17  // + many others
18  // including self-written rules
19  ...
20 }
```



```
1 // lefthook.yml
2
3 pre-commit:
4   commands:
5     lints:
6       glob: "*.ts,tsx"
7       run: npm run lint:fix && git add .
```



~~1. Define state~~

2. Define default value

3. Don't reinvent the wheel

~~4. Let them ( devs ) rule~~


5. Keep It Simple, Stupid ( KISS )

# SDD aka Storybook Driven Development



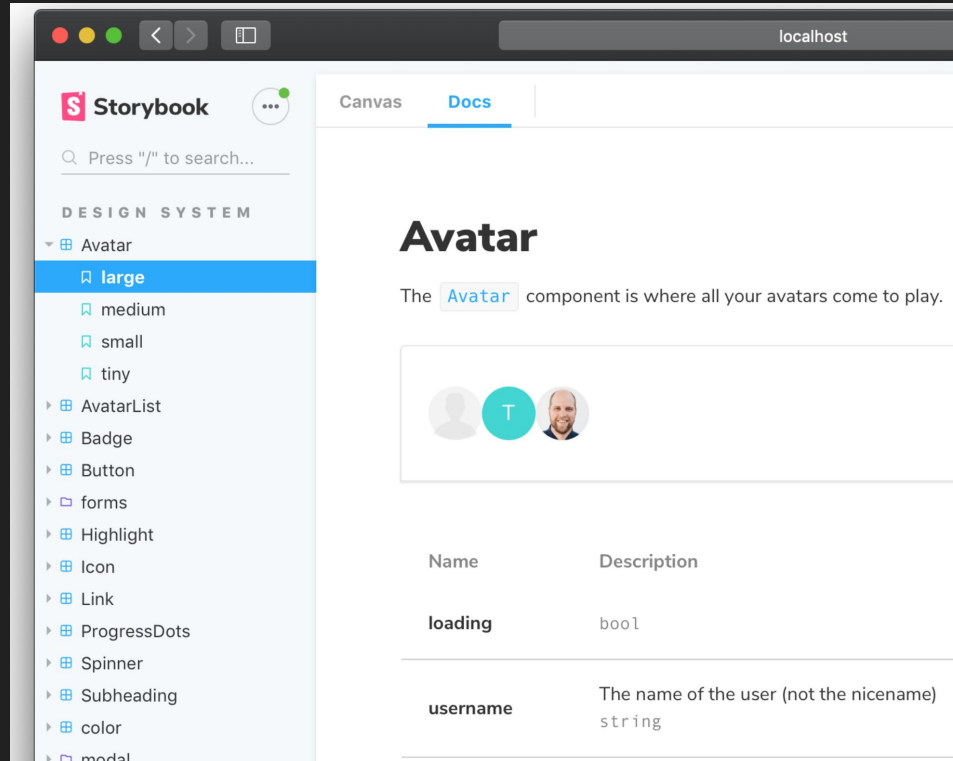
Explorer for your UI

# SDD aka Storybook Driven Development



```
1 export default {  
2   component: Avatar,  
3   title: 'Avatar'  
4 }  
5  
6 export const large = () => <Avatar size="l" />  
7  
8 export const medium = () => <Avatar size="m" />  
9  
10 export const small = () => <Avatar size="s" />  
11  
12 export const tiny = () => <Avatar size="xs" />  
13
```

# SDD aka Storybook Driven Development



1. Define state

2. Define default value

~~3. Don't reinvent the wheel~~

~~4. Let them ( devs ) rule~~

~~5. Keep It Simple, Stupid ( KISS )~~

# Tests and Snapshots




Jest and co.

# Tests and Snapshots

```
1 it('have custom class', () => {
2   const tree = renderer
3     .create(<Button className='aha'>Some text inside button</Button>)
4     .toJSON();
5
6   expect(tree).toMatchSnapshot();
7 });
8
9 it('renders medium', () => {
10  const tree = renderer
11    .create(<Button medium>Some text inside button</Button>)
12    .toJSON();
13
14  expect(tree).toMatchSnapshot();
15 });
16
17 it('renders simple', () => {
18  const tree = renderer
19    .create(<Button simple>Some text inside button</Button>)
20    .toJSON();
21
22  expect(tree).toMatchSnapshot();
23 });
```

# Tests and Snapshots



```
1 import { shallow } from 'enzyme'
2
3 ...
4
5 it('handleEnter', () => {
6   let onSearch = jest.fn()
7   let search = shallow(<Search onSearch={onSearch} />)
8
9   search.instance().handleEnter()
10
11   expect(onSearch).toBeCalledWith(expect.anything())
12 })
13
```



1. Define state
2. Define default value
3. Don't reinvent the wheel
4. Let them ( devs ) rule
5. Keep It Simple, Stupid ( KISS )

write these tests now

thanks.