



React + БЭМ: Гибкая архитектура дизайн системы

И проблемы выбора...



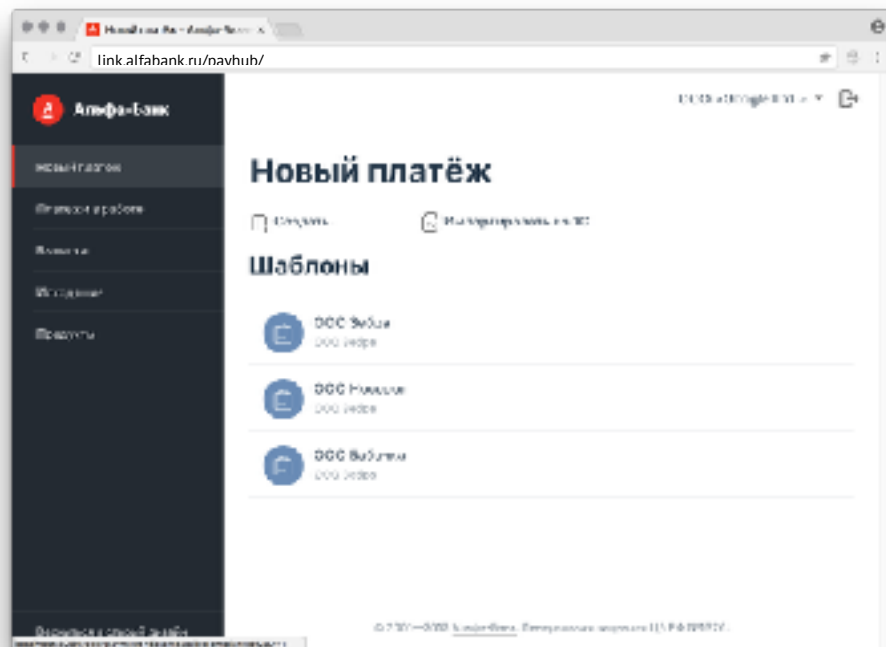
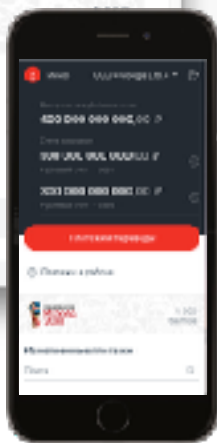
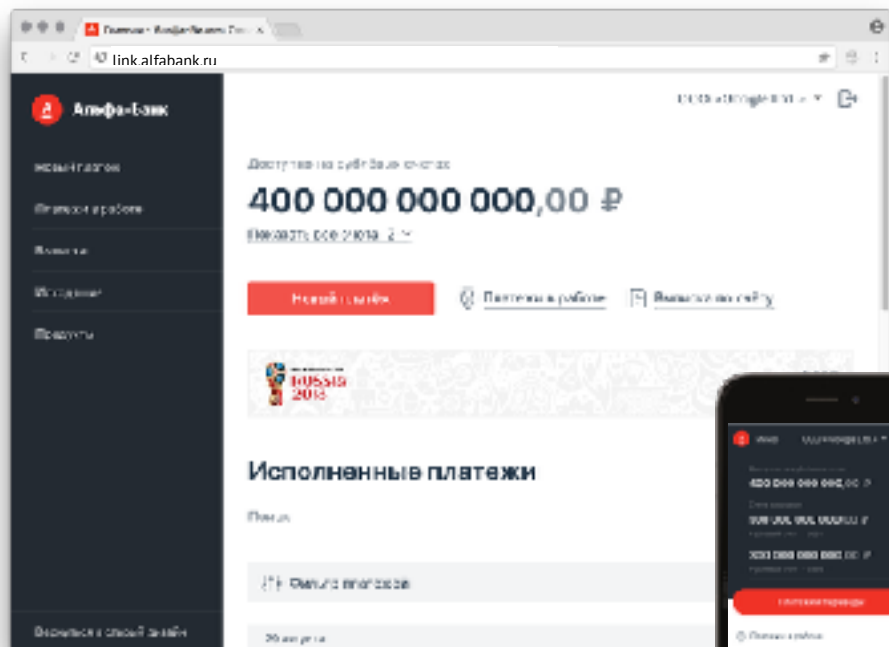
React **БЭМ.Методология **БЭМ.Стек****



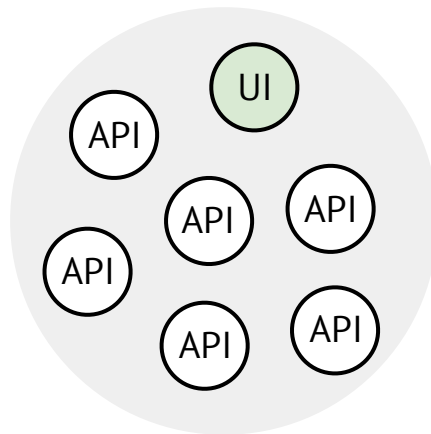
alfalab

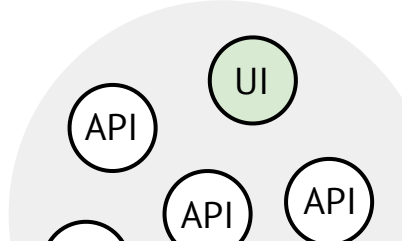
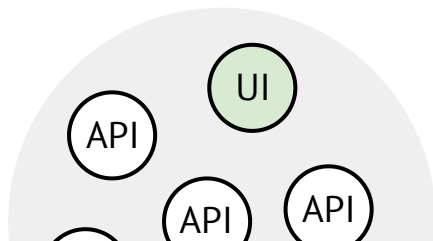
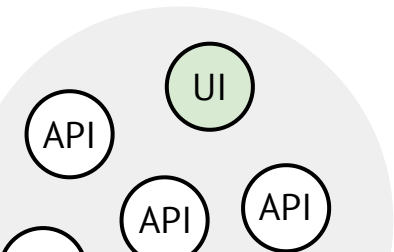
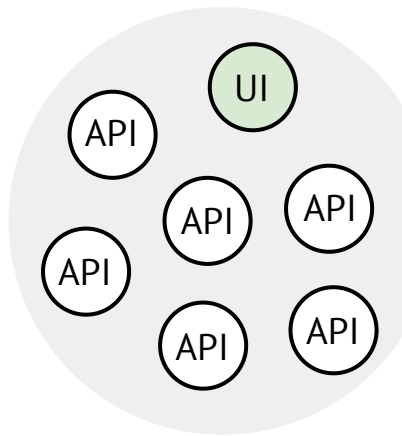
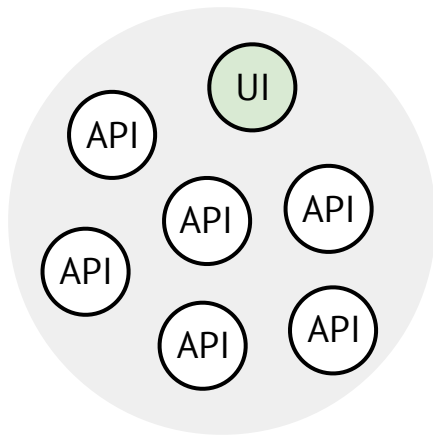
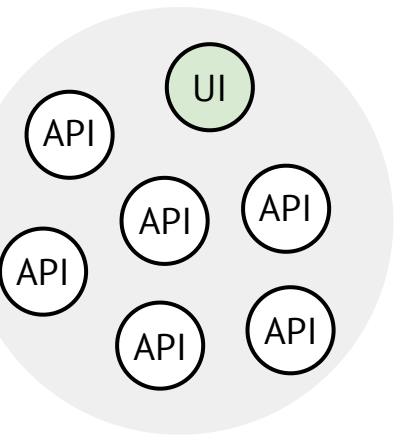
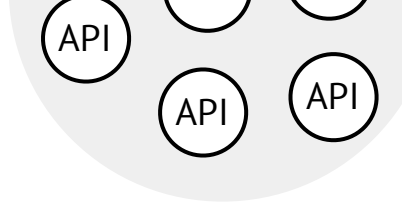
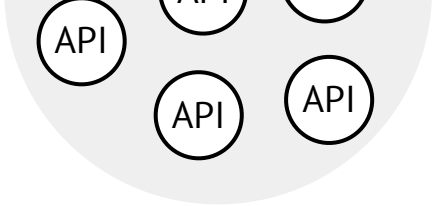
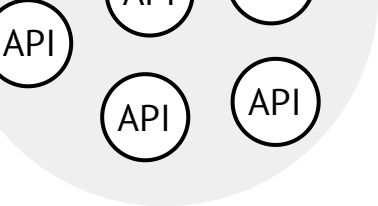


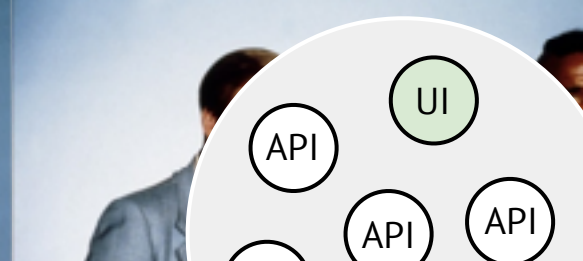
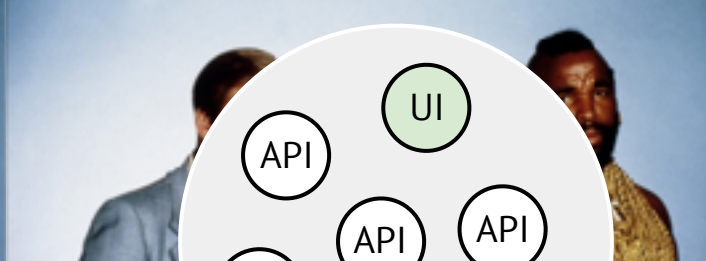
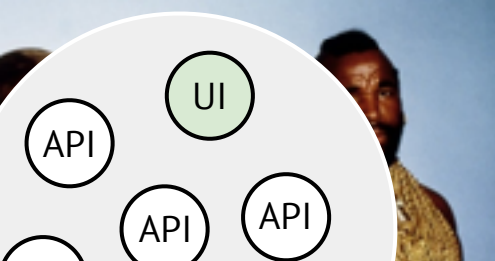
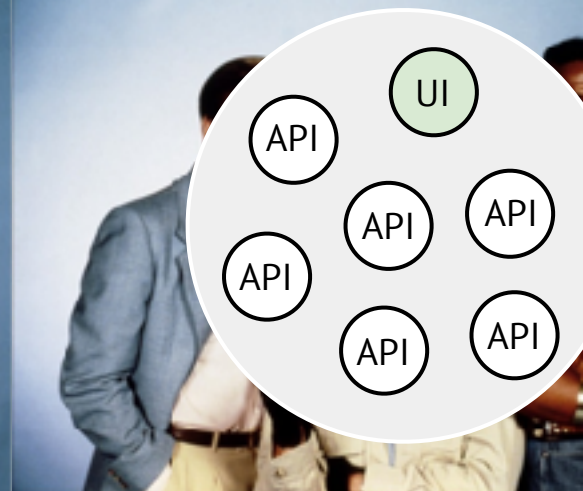
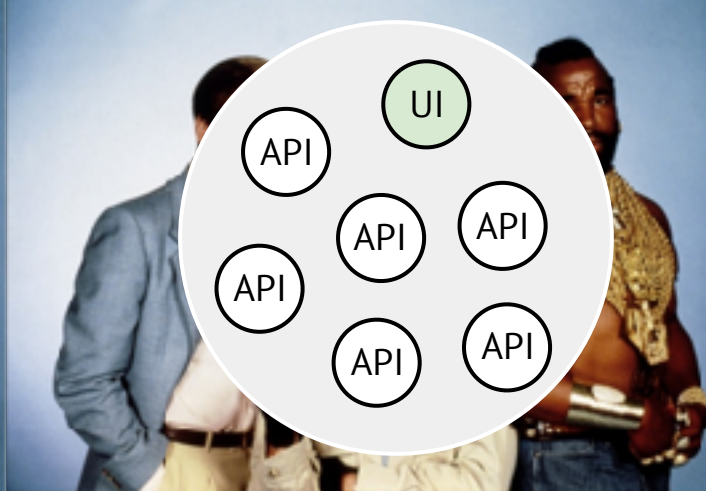
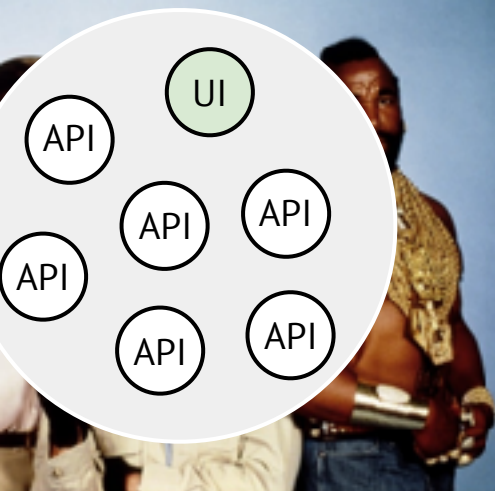
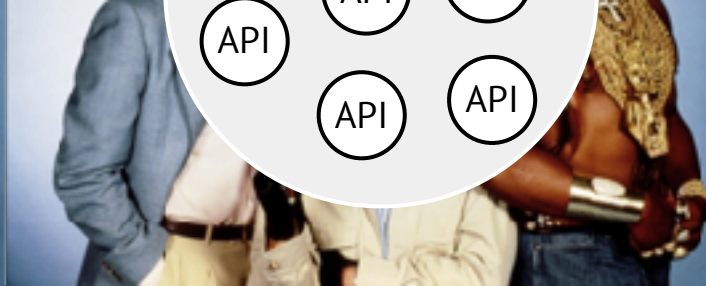
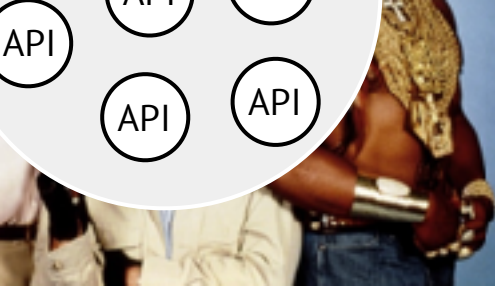




Микросервисы









Эй, а кто тут главный?





Ладно, руководитель-то мой где?





Ну хоть, кто-нибудь мне скажет, что делать?





Короче, БЭМ или React? Я запутался...



Придумал! Составлю список с критериями!

БЭМ.*Стек*

React

БЭМ.*Стек*

Придумали в Яндексе

React

БЭМ.Стек

Придумали в Яндексе

React

Придумали в Facebook

БЭМ.Стек

Придумали в Яндексе

Уровни переопределения... ммм...

React

Придумали в Facebook

БЭМ.Стек

Придумали в Яндексе

Уровни переопределения... ммм...

React

Придумали в Facebook

VirtualDOM! Вроде быстрый!

БЭМ.Стек

Придумали в Яндексе

Уровни переопределения... ммм...

Можно рендерить на сервере...

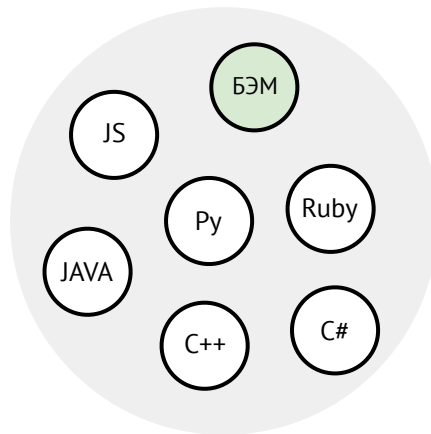
React

Придумали в Facebook

VirtualDOM! Вроде быстрый!

Angular 2 вышел...

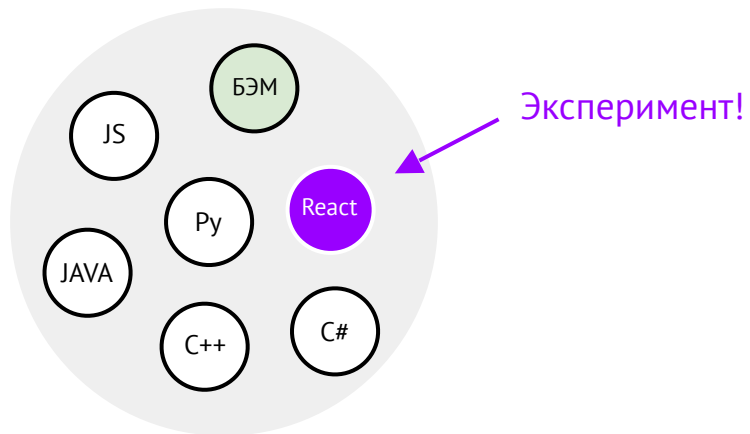
Микросервисы



Микросервисная архитектура

позволяет выбирать технологии через механизм свободного рынка

Микросервисы





БЭМ vs React



React

Почему **БЭМ**.Стек проиграл?

Почему **БЭМ**.Стек проиграл?

bem-core

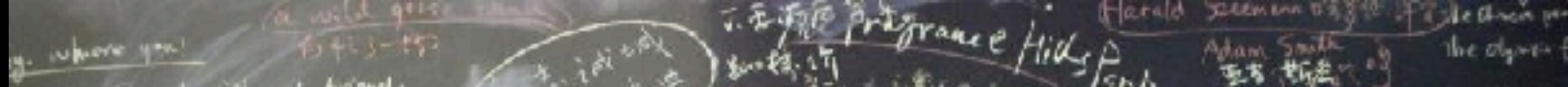


enb, ymodules...

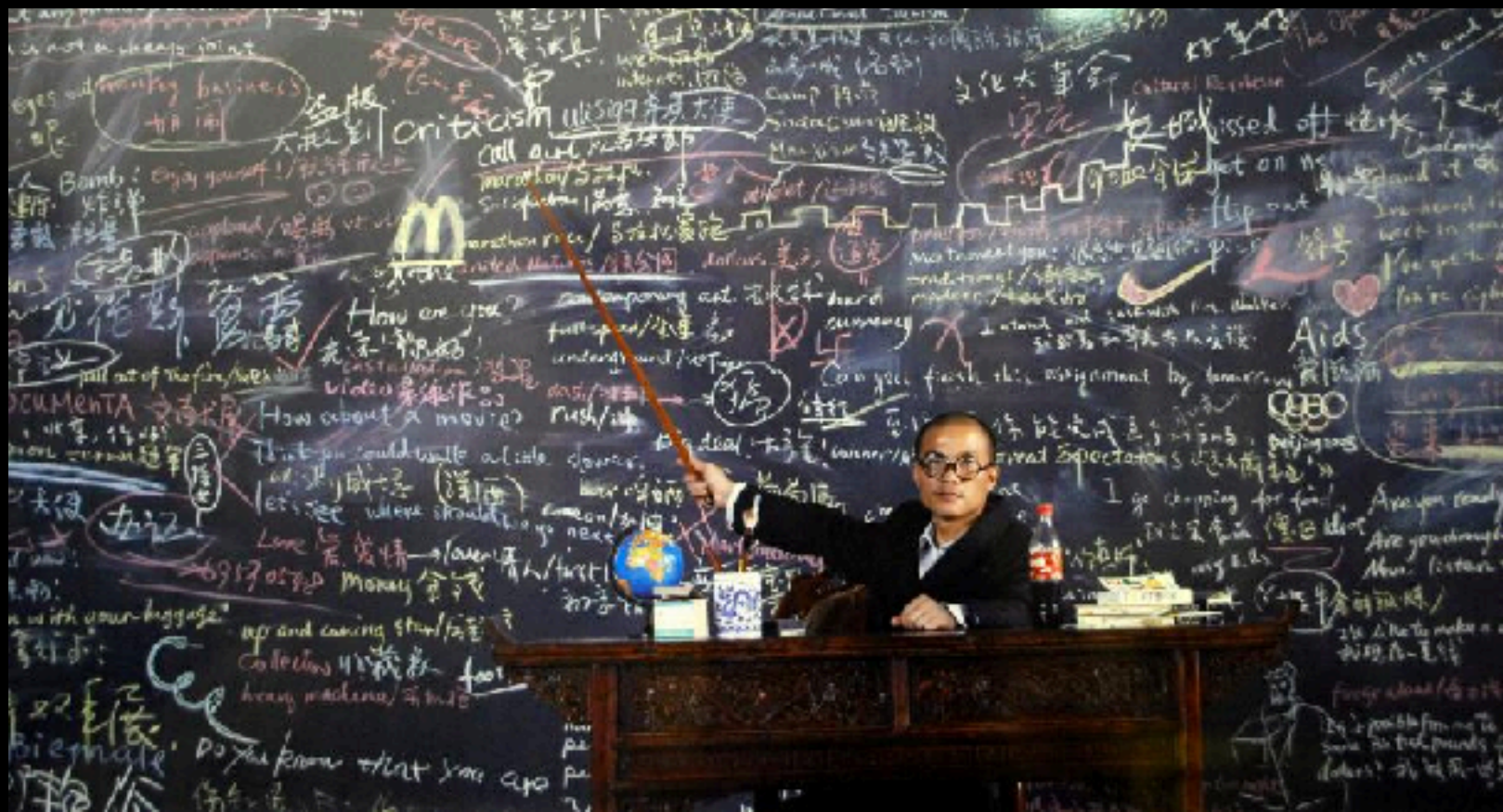
React



webpack, es6...

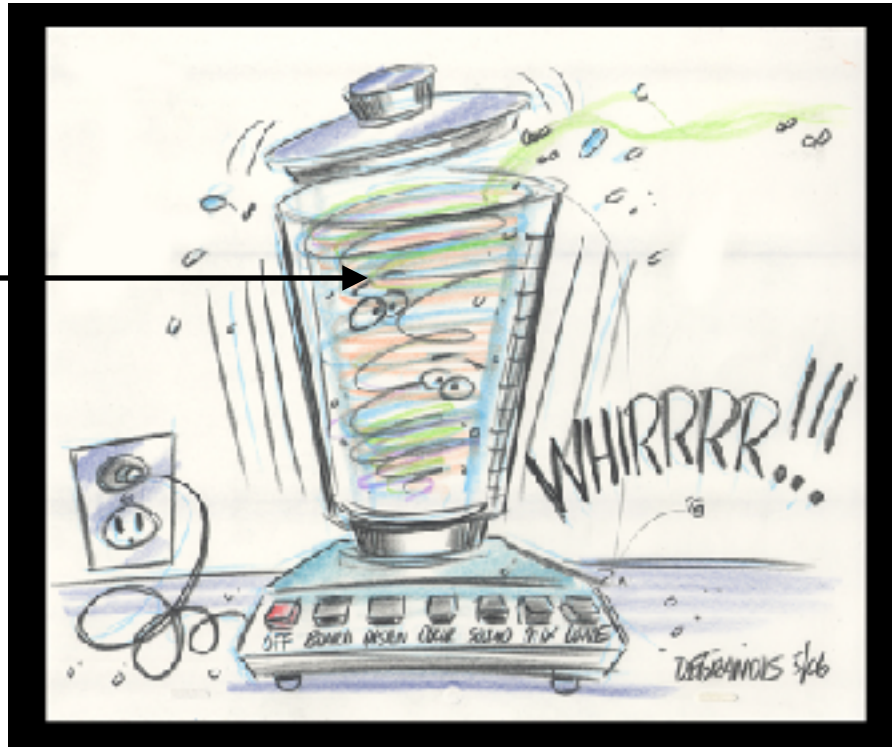


Почему БЭМ.Стек проиграл?



Почему **БЭМ**.Стек проиграл?

Где-то там твои
уровни переопределения



Почему **БЭМ**.Стек проиграл?

Костяк разработчиков **Full Stack Java**:
ES6 и паттерны **React** для них понятнее

Почему **БЭМ**.*Методология* все-таки выжила?

Почему **БЭМ**.Методология все-таки выжила?



Picasso

simple is hard

Martin Scorsese, Filmmaker

Почему **БЭМ**.*Методология* все-таки выжила?

1-ая версия **дизайн системы**
уже была на **БЭМ**. Просто берем **CSS**!

Почему **БЭМ**.*Методология* все-таки выжила?





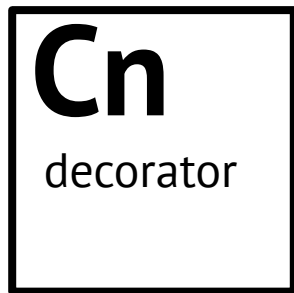
+



React

=

?



github.com/alfa-laboratory/cn-decorator



github.com/alfa-laboratory/arui-feather

Консистентность

Гибкость



Самый простой пример

```
import cn from 'arui-feather/cn';

@cn('input')
class Input extends React.Component {
  render(cn) {
    return <div className={ cn } />;
  }
}
```

```
import cn from 'arui-feather/cn';

@cn('input')
class Input extends React.Component {
  render(cn) {
    return <div className={ cn } />;
  }
}
```

```
import cn from 'arui-feat'
```

```
@cn('input')
```

```
class Input extends React.
```

```
  render(cn) {
```

```
    return <div class=
```

```
  }
```

```
}
```

```
<div class="input"></div>
```



```
@cn('input')
class Input extends React.Component {
  render(cn) {
    return (
      <div className={cn({ disabled: true })}>
        <input className={cn('control')}/>
      </div>
    );
  }
}
```

```
component {
```

```
{ cn({ disabled: true }) }  
sName={ cn('control') }/>>
```

```
<div class="input input_disabled">  
  <input class="input__control" />  
</div>
```

А что, если у меня компонент чуть-чуть отличается?

Проксирование `className``

```
render() {  
  return <Input className="my-class" />;  
}
```

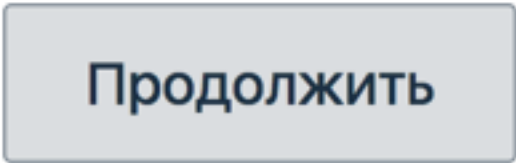
```
<div class="input my-class">  
</div>
```

Как быть, если мой компонент совсем отличается?

Перегрузка имени блока

Продолжить

Предложить



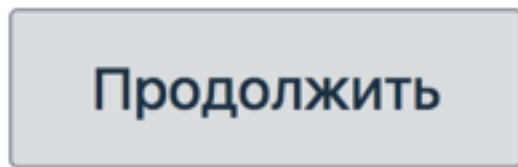
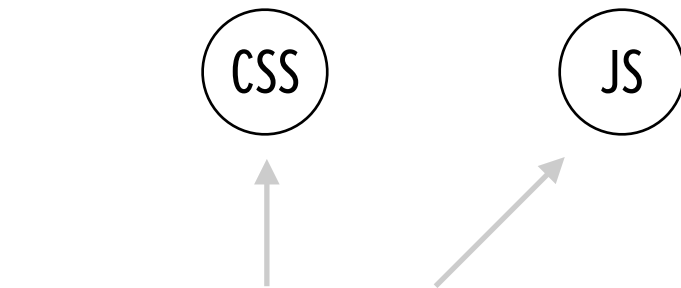
Продолжить

`.button`



Продложить

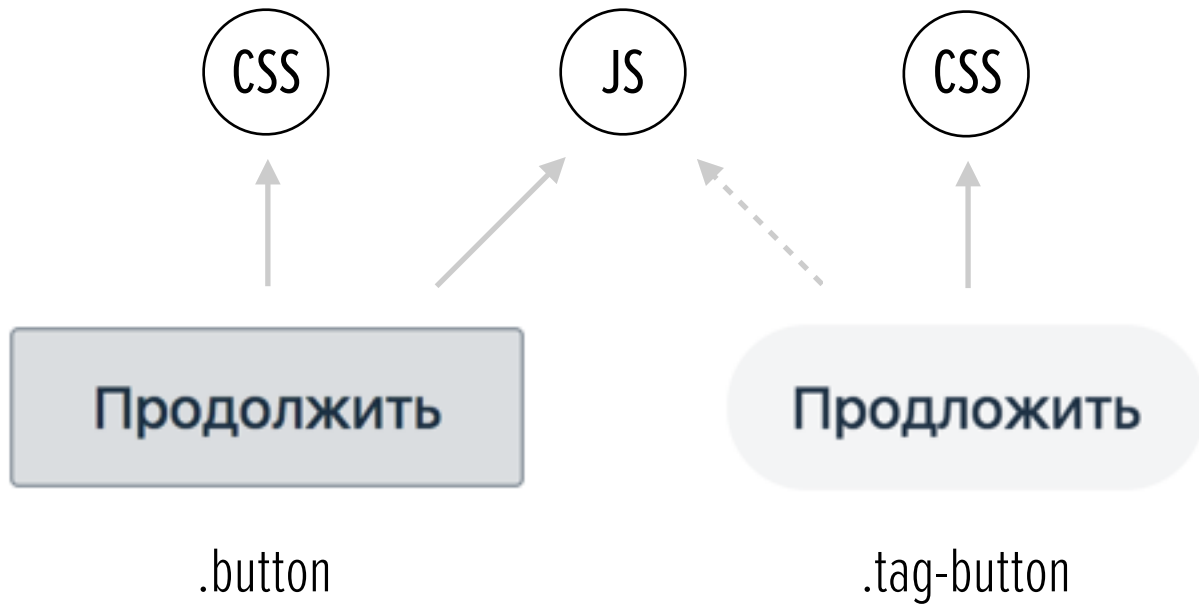
`.tag-button`



.button



.tag-button



```
import cn from 'arui-feather/cn';  
import Input from 'arui-feather/input';  
  
@cn('my-input')  
class MyInput extends Input {};
```

```
import cn from 'arui-feather/  
import Input from 'arui-feathe
```

```
@cn('my-input')
```

```
class MyInput extends Input {
```

```
<div class="my-input"></div>
```

DI компоненты

Vkontakte



- Vkontakte

Facebook

Twitter

Vkontakte ▼

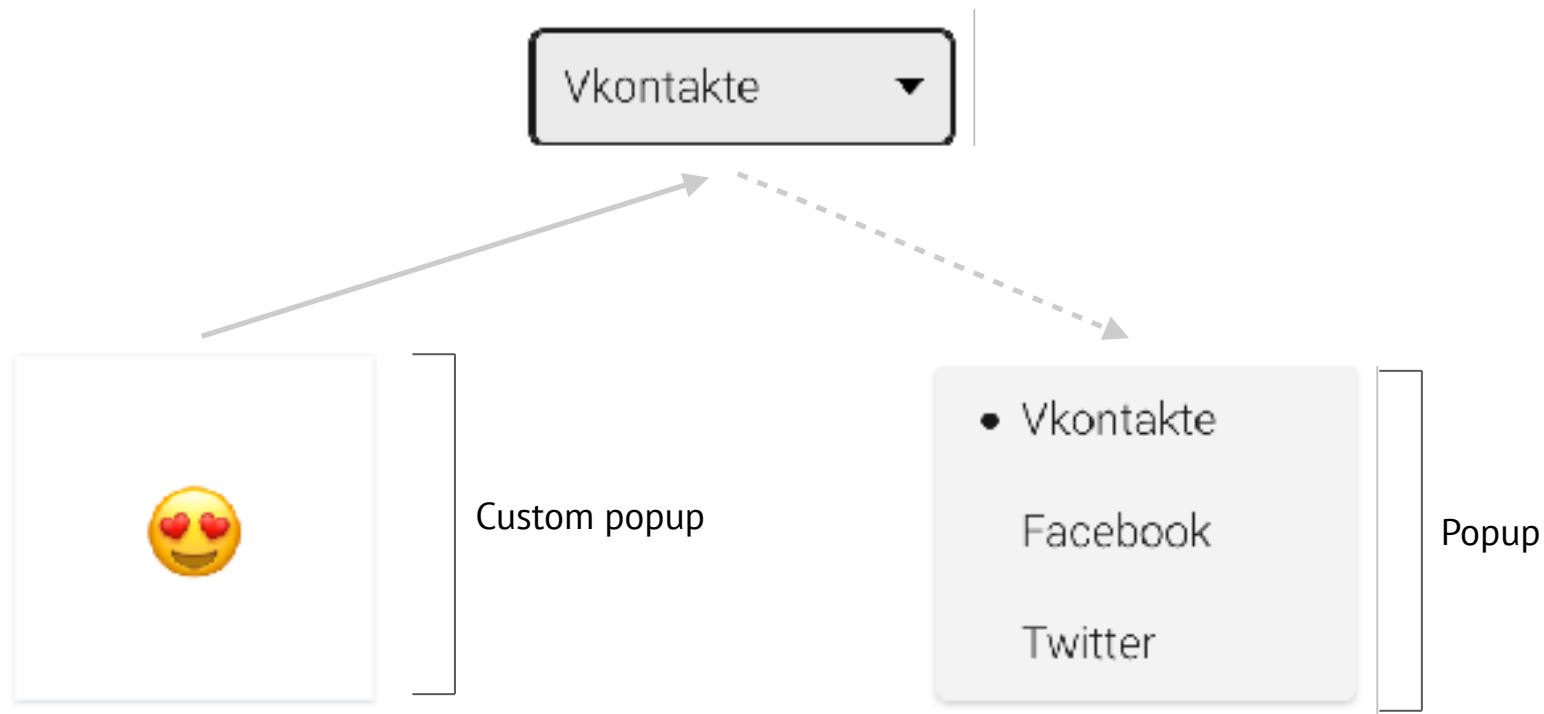
- Vkontakte

Facebook

Twitter

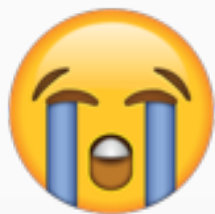
Button

Popup




```
import cn from 'arui-feather/cn';  
import Button from 'arui-feather/button';  
import Popup from 'arui-feather/popup';
```

```
@cn('select')  
class Select extends React.Component {  
  render(cn) {  
    return (  
      <div className={ cn }>  
        <Button />  
        <Popup />  
      </div>  
    );  
  }  
}
```



```
import cn from 'arui-feather/cn';
import Button from 'arui-feather/button';
import Popup from 'arui-feather/popup';

@cn('select', Button, Popup)
class Select extends React.Component {
  render(cn, Button, Popup) {
    return (
      <div className={ cn }>
        <Button />
        <Popup />
      </div>
    );
  }
}
```



```
import cn from 'arui-feather/cn';  
import Select from 'arui-feather/select';  
import MyButton from './my-button';  
import MyPopup from './my-popup';  
  
@cn('my-select', MyButton, MyPopup)  
class MySelect extends Select {};
```



Темы

Facebook ▼

Vkontakte

✓ Facebook

Twitter

Facebook ▼

Vkontakte

✓ Facebook

Twitter

```
render() {  
  return <Input theme="alfa-on-color" />;  
}
```

```
<div class="input input_theme_alfa-on-color">  
</div>
```

```
render() {  
  return (  
    <ThemeProvider theme="alfa-on-color">  
      <Input />  
    </ThemeProvider>  
  );  
}  
  
<div class="input input_theme_alfa-on-color">  
</div>
```

Итого

Итого

React за счет большого комьюнити закрывает
все наши потребности в разработки интерфейсов

Итого

React за счет большого комьюнити закрывает
все наши потребности в разработке интерфейсов

БЭМ.*Методология* проста, гибка, закрывает
все наши потребности в организации верстки

Итого

React за счет большого комьюнити закрывает все наши потребности в разработке интерфейсов

БЭМ. *Методология* проста, гибка, закрывает все наши потребности в организации верстки

cn decorator склеивает все воедино — взбалтывает, но не смешивает!
Композиция, принципы KISS и YAGNI делают дизайн систему конструктором, позволяют легко масштабировать фронтенд.





Так как выбирать технологии?

Так как выбирать технологии?

Как бы вы не выбирали **все исторически сложится**

Так как выбирать технологии?

Как бы вы не выбирали **все исторически сложится**

Поддерживайте **свободный рынок технологий** в компании

Так как выбирать технологии?

Как бы вы не выбирали **все исторически сложится**

Поддерживайте **свободный рынок технологий** в компании

Микросервисная архитектура помогает в этом

Так как выбирать технологии?

Как бы вы не выбирали **все исторически сложится**

Поддерживайте **свободный рынок технологий** в компании

Микросервисная архитектура помогает в этом

Хорошая технология = хороший продукт распространит себя сама

Так как выбирать технологии?

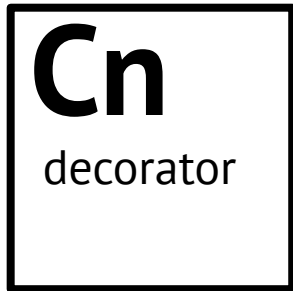
Как бы вы не выбирали **все исторически сложится**

Поддерживайте **свободный рынок технологий** в компании

Микросервисная архитектура помогает в этом

Хорошая технология = хороший продукт распространит себя сама

Экспериментируйте!



 github.com/alfa-laboratory/cn-decorator

 facebook.com/vitaliy.green

 vgalakhov@alfabank.ru