







bem-react: практикум

Владимир Гриненко, руководитель службы общих компонентов
Москва, 23 ноября 2018

Теория



БЭМ

- › Любой интерфейс может быть представлен блоками
- › Блоки могут содержать элементы
- › Блоки и элементы могут иметь модификаторы

БЛОК



Элементы





air-gun.ru







БЛОК



Элементы



Модификаторы



cat_theme_red



cat_state_wet

cat_wet





4GIFs
.com



БЭМ не только про CSS

БЭМ в коде

- › Блоки, элементы и модификаторы можно выразить именами CSS классов
- › Схема именования может быть любой

Классическая схема именования

- › block
- › block-with-long-name
- › block_boolean-modifier
- › block_modifier-name_modifier-value

- › block__element
- › block__element-name_boolean-modifier
- › block__element-name_modifier-name_modifier-value

Еще одна популярная схема

- › block
- › block--modifier
- › block__element
- › block__element--modifier

Новая схема именования

- › Block
- › BlockWithLongName
- › Block_booleanModifier
- › Block_modifierName_modifierValue

- › Block-Element
- › Block-ElementName_booleanModifier
- › Block-ElementName_modifierName_modifierValue



Схема не важна

ru.bem.info/methodology/naming-convention

**CSS
IS
AWESOME**



БЭМ в CSS

- › Обеспечивает скоупинг в CSS
- › Предоставляет правила и структуру
- › Избавляет от вложенности
- › Делает возможным повторное использование верстки
- › Делает HTML и CSS самодокументированным

БЭМ в CSS

- › Используем селекторы классов
- › Не используем:

*

tag

#id

- › Избегаем вложенных селекторов



Миксы

Миксы



Миксы

```
<ul class="Nav">
  <li class="Nav-Item">
    <a class="Link Nav-ItemLink"></a>
  </li>
</ul>
```

Единые термины

- › CSS
- › JS
- › HTML
- › Тесты
- › Документация
- › и так далее

Единые термины

blocks/

header/

header.**specs**/

header.**css**

header.**js**

header.**tmpl**

header.**svg**

header.**md**

Уровни для компонентов



уровни

project/

 node_modules/some-lib/**library.blocks/**

 button/

 button.css

 button.js

common.blocks/

 header/

 logo/

 button/

 button.css

уровни

project/

 node_modules/some-lib/library.blocks/

button/

 button.css

 button.js

 common.blocks/

 header/

 logo/

button/

 button.css

Уровни помогают

- › Разделить проект по платформам

Уровни для разных платформ



Уровни для разных платформ



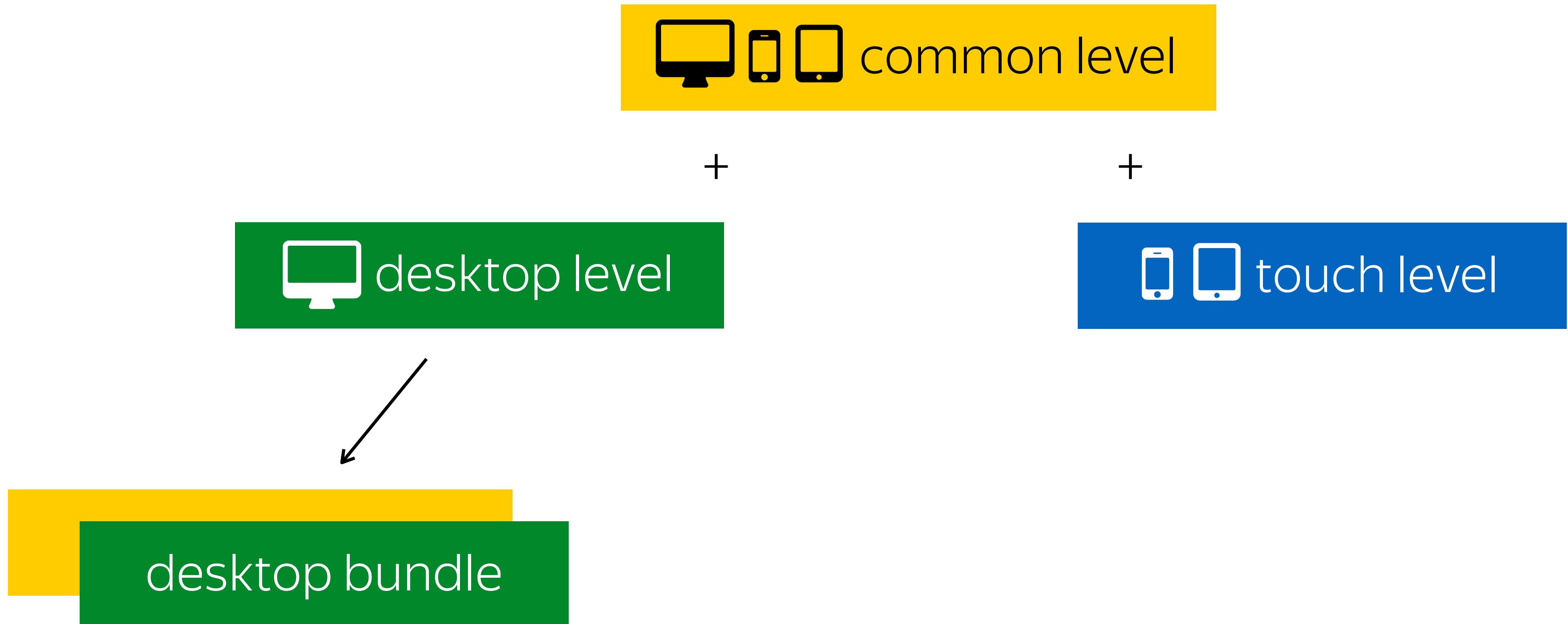
+



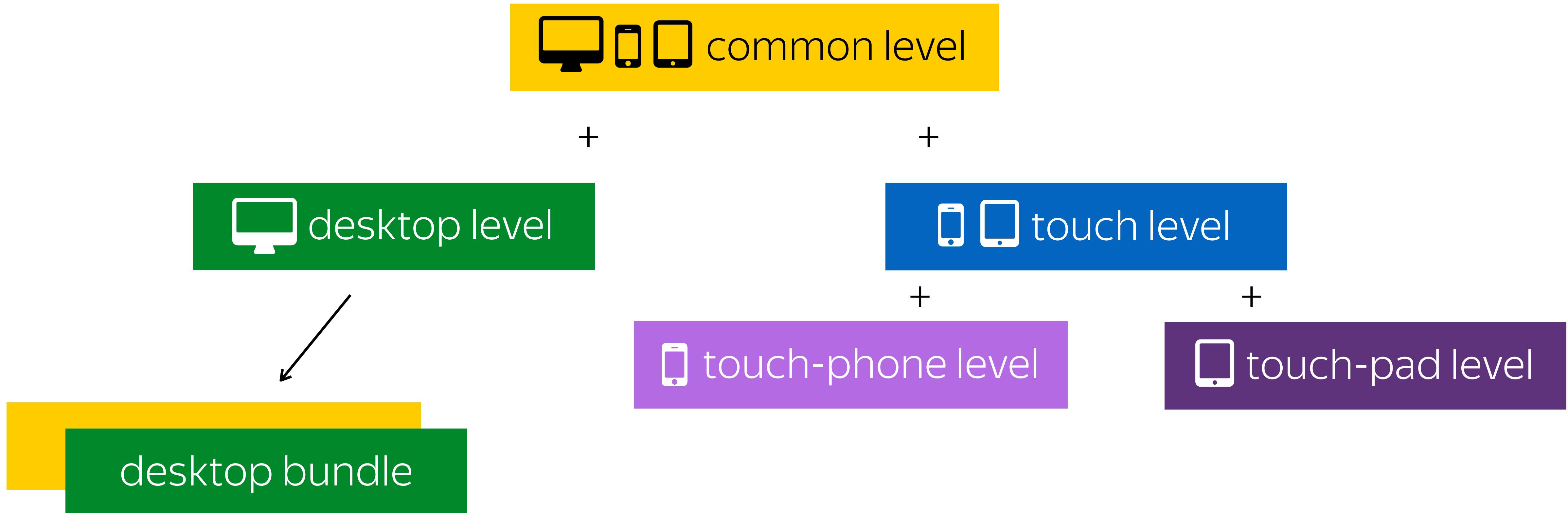
+



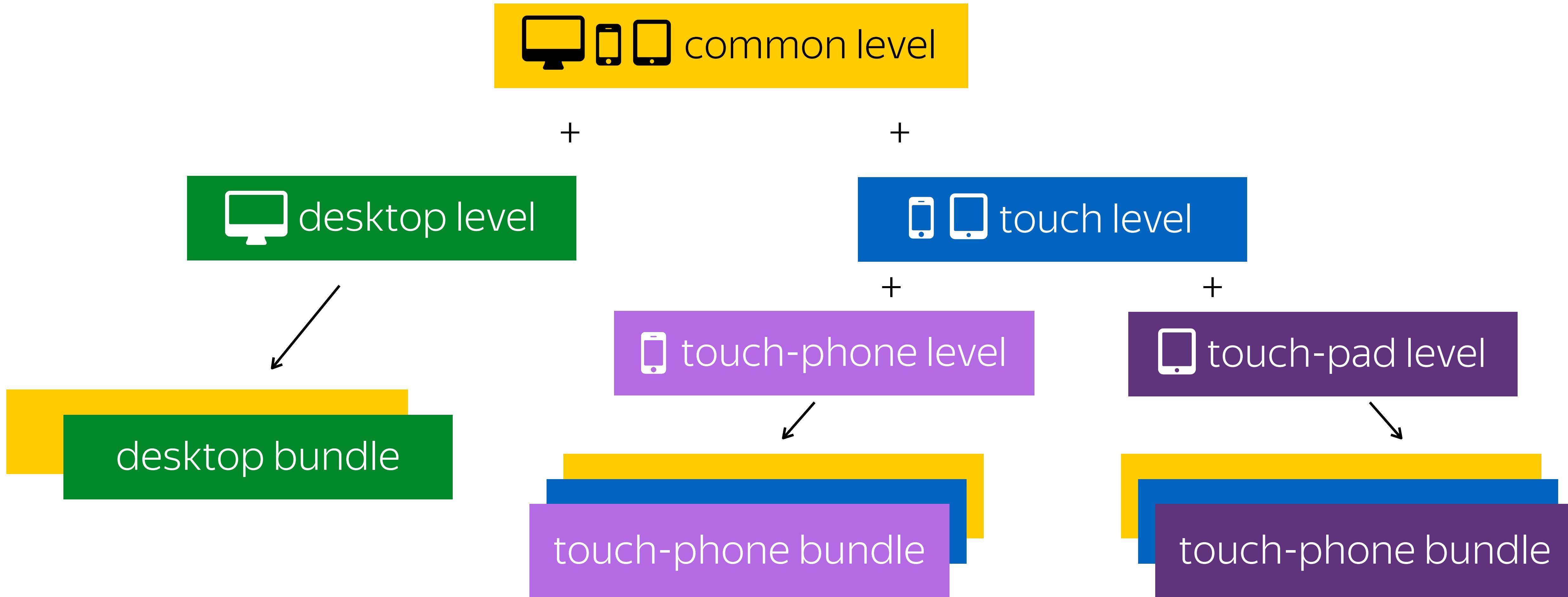
Уровни для разных платформ



Уровни для разных платформ



Уровни для разных платформ



Уровни помогают

- › Разделить проект по платформам
- › Легко обновлять библиотеки

Легко обновлять библиотеки

library level

+

project level



bundle

Легко обновлять библиотеки

library level

my-prj/node_modules/mylib(blocks

+

project level

my-prj(blocks

bundle

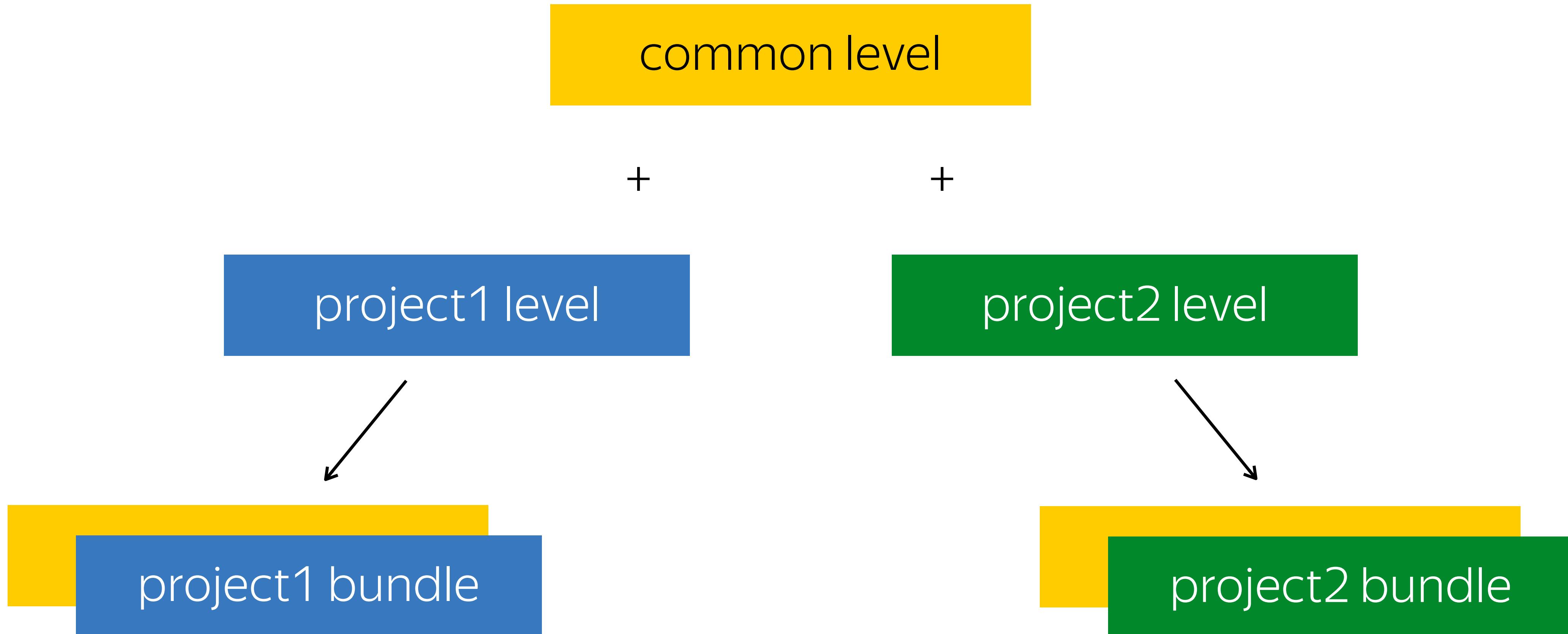
CSS

```
@import "node_modules/mylib(blocks/button.css";  
@import "blocks/button.css";
```

Уровни помогают

- › Разделить проект по платформам
- › Легко обновлять библиотеки
- › Повторно использовать блоки на разных проектах

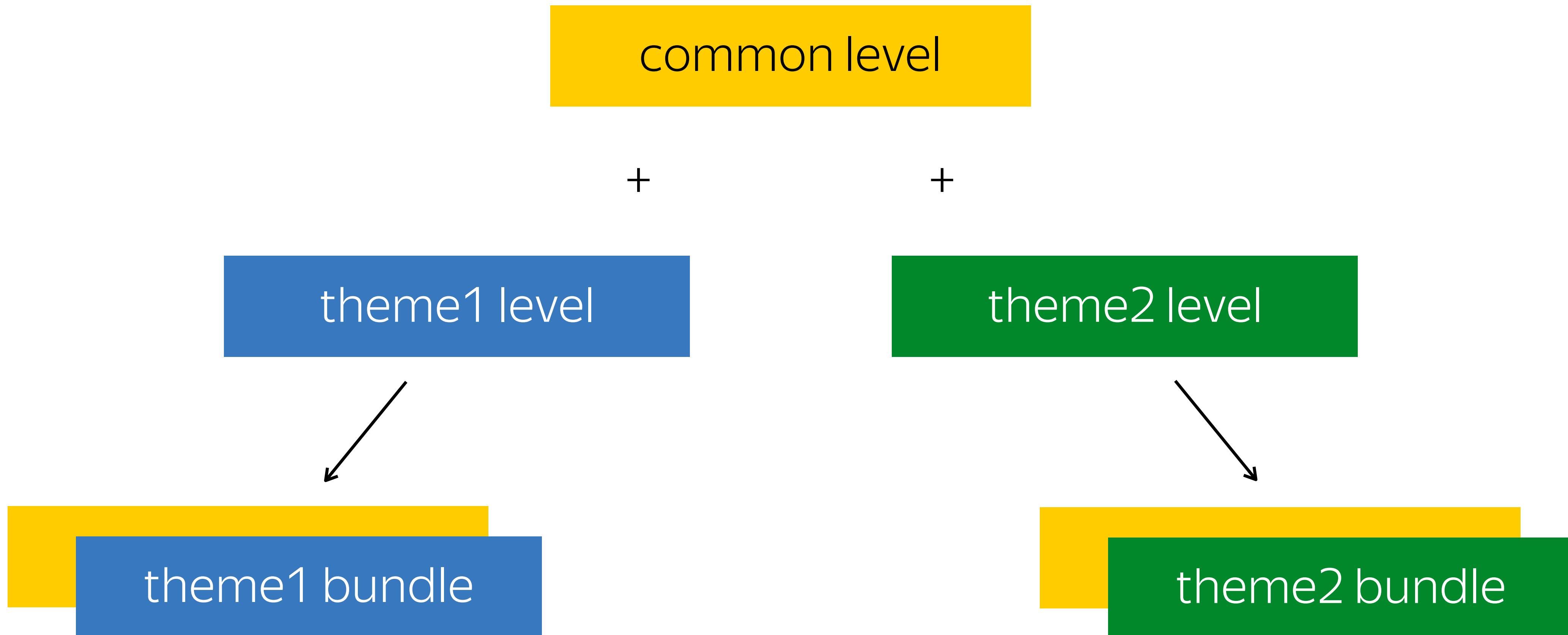
Повторно использовать блоки на проектах



Уровни помогают

- › Разделить проект по платформам
- › Легко обновлять библиотеки
- › Повторно использовать блоки на разных проектах
- › Поддерживать темы оформления

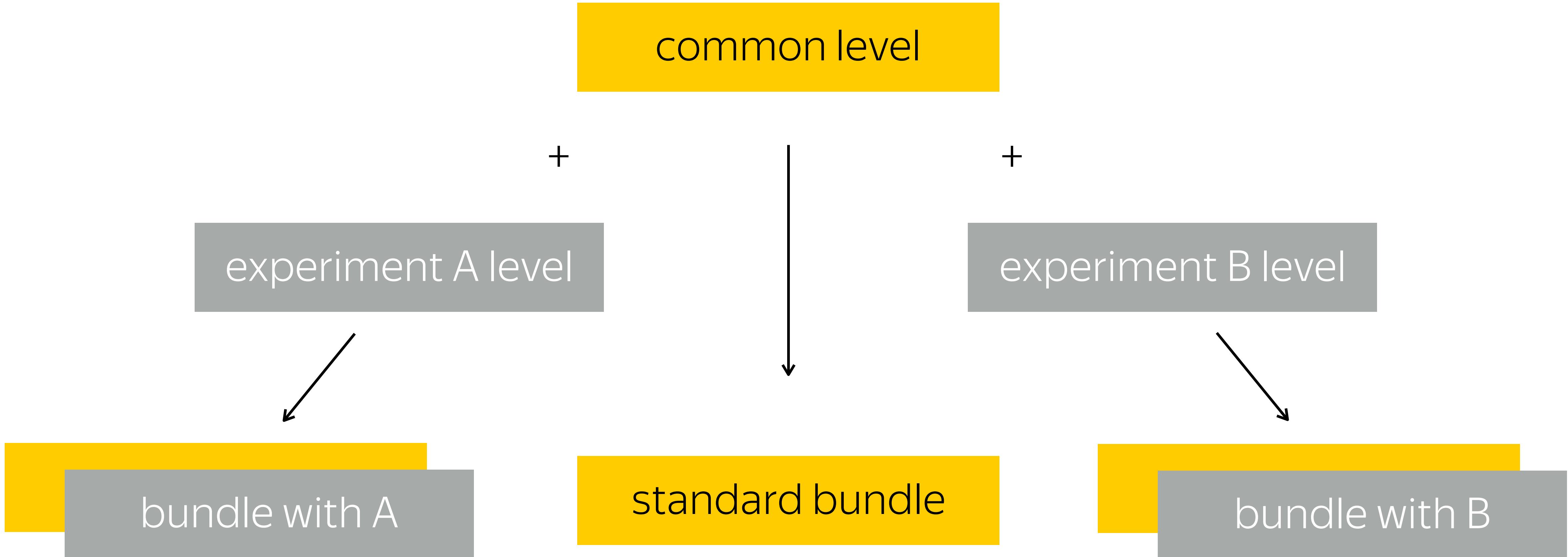
Темы оформления



Уровни помогают

- › Разделить проект по платформам
- › Легко обновлять библиотеки
- › Повторно использовать блоки на разных проектах
- › Поддерживать темы оформления
- › Проводить дешевые эксперименты

Эксперименты



Эксперименты

prj/

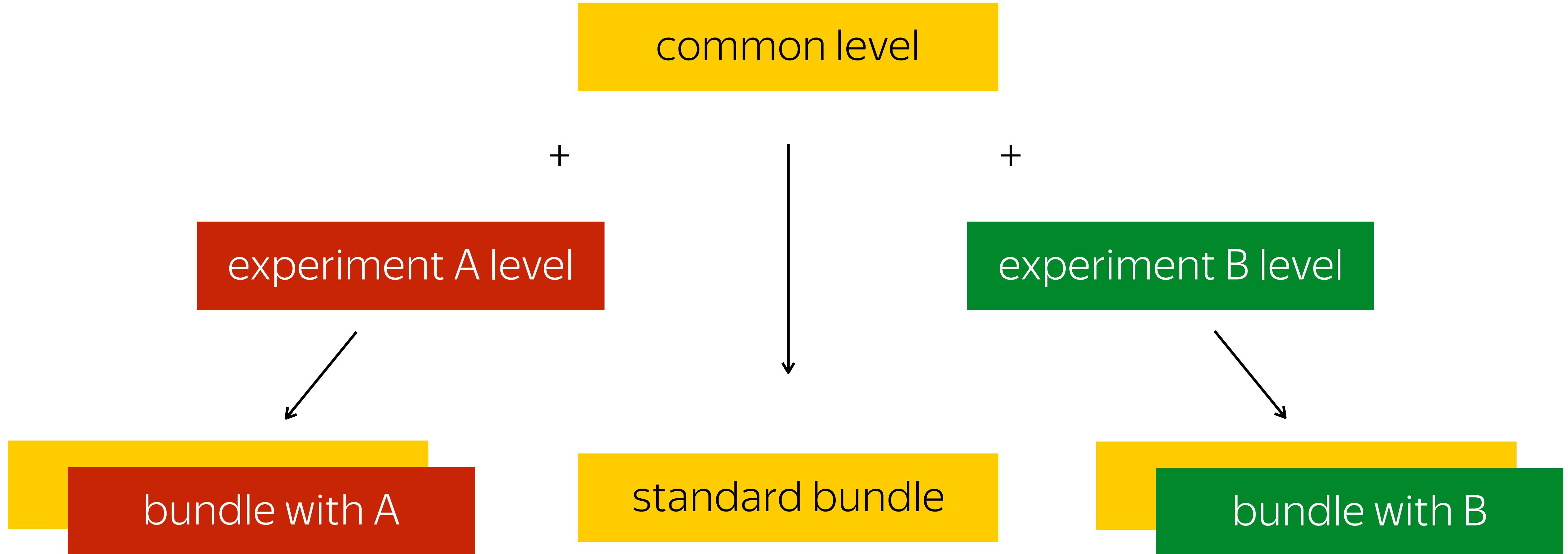
common.blocks/

experiments/

A.blocks/

B.blocks/

Эксперименты



Эксперименты

common level

+

experiment B level

↓

bundle with B

config.js

```
const levels = [
  'path/to/common.blocks',
  'experiments/A.blocks',
  'experiments/B.blocks'
];
```

Эксперименты

prj /

common.blocks /

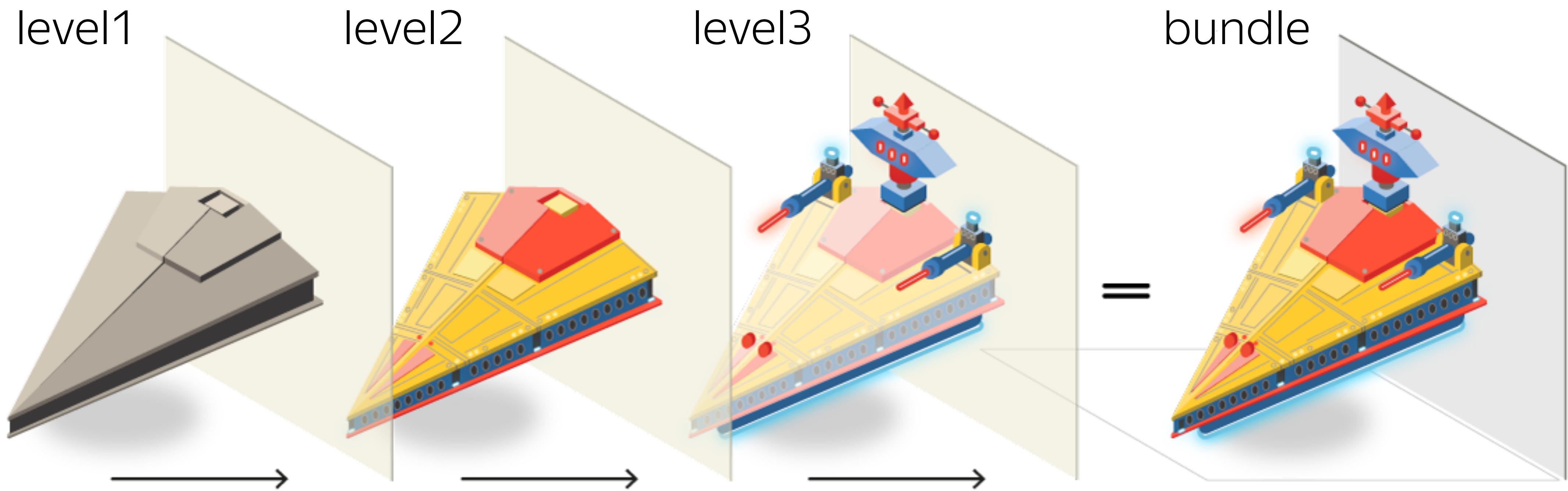
experiments /

~~A.blocks /~~

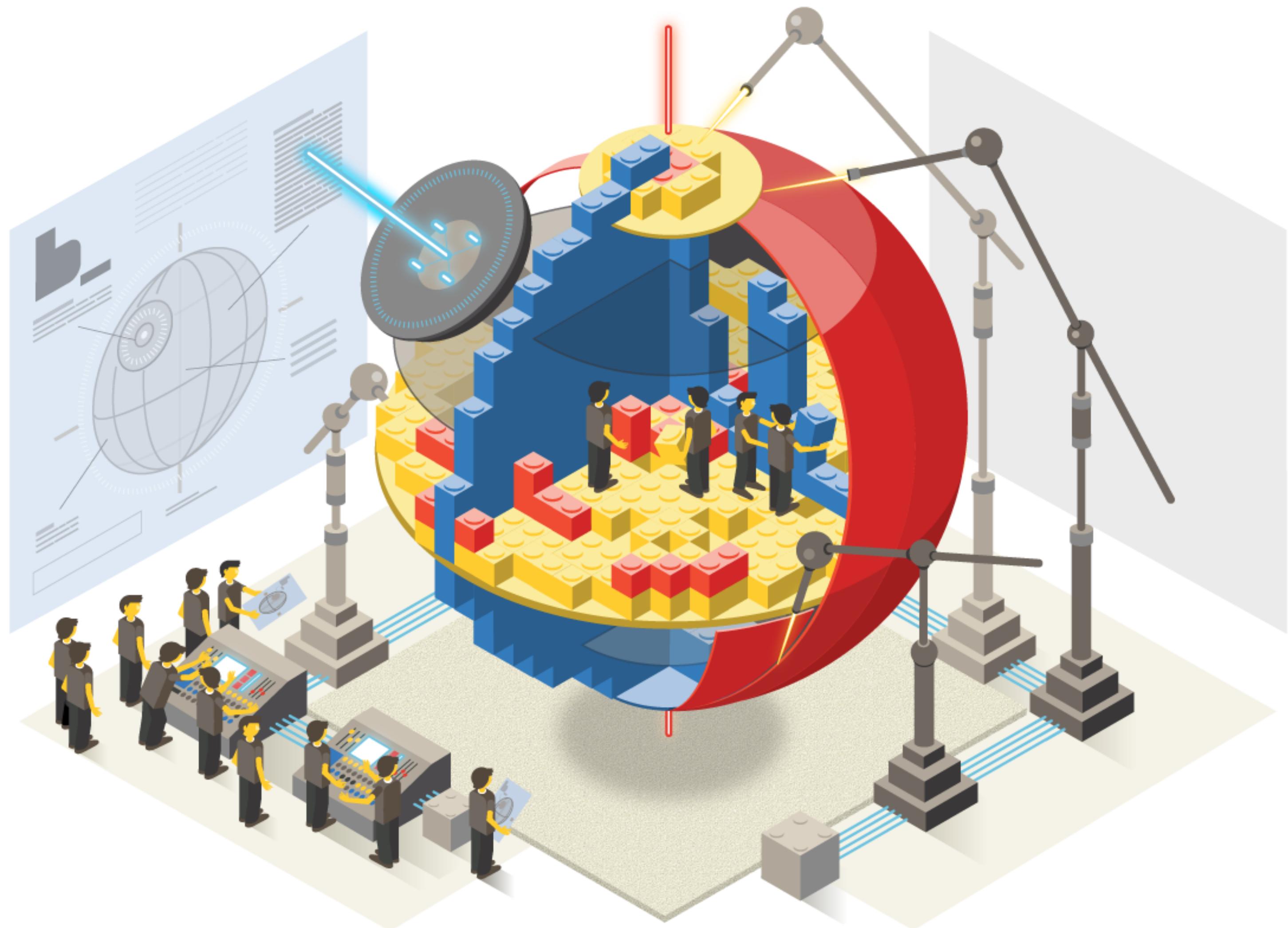
B.blocks /

Уровни: все сразу

- › Разделить проект по платформам
- › Легко обновлять библиотеки
- › Повторно использовать блоки на разных проектах
- › Поддерживать темы оформления
- › Проводить дешевые эксперименты







Практика



Вы должны уметь

- › React (и функциональное программирование)
- › (желательно) Typescript
- › БЭМ

bem-react

- › Генерация имен классов по БЭМ
- › Динамические модификаторы
- › Реализация переопределений через DI

Три источника, три составные части

- › classname
- › core
- › di

classname

```
npm i @bem-react/classname
```

classname

```
import { cn } from '@bem-react/classname';

const cat = cn('Cat');

cat(); // Cat
cat({ size: 'm' }); // Cat Cat_size_m
cat('Tail'); // Cat-Tail
cat('Tail', { length: 'small' }); // Cat-Tail Cat-Tail_length_small

const dogPaw = cn('Dog', 'Paw');

dogPaw(); // Dog-Paw

// Dog-Paw Dog-Paw_color_black Dog-Paw_exists
dogPaw({ color: 'black', exists: true });
```

classname

```
function WelcomeShri(props) {
  return (
    <div className="WilcomeShri">
      <h1 className="WilcomeShri-Title">Привет, ШРИ!</h1>
      <div className={
        'WilcomeShri-Description' + (props.isFull ?
          'WilcomeShri-Description_visible' :
          '')}>
        Даешь БЭМ в ваш React!
      </div>
    </div>
  );
}
```

classname

```
import { cn } from '@bem-react/classname';

const cnWelcomeShri = cn('WelcomeShri');

function WelcomeShri(props) {
  return (
    <div className={cnWelcomeShri()}>
      <h1 className={cnWelcomeShri('Title')}>Привет, ШРИ!</h1>
      <div className={cnWelcomeShri('Description', { visible:
props.isFull })}>
        Даешь БЭМ в ваш React!
      </div>
    </div>
  );
}
```

classname — МИКСЫ

```
import { cn } from '@bem-react/classname';

const cat = cn('Cat');

cat(null, ['Dog', 'Fox']); // Cat Dog Fox
cat({ size: 'm' }, ['Dog', 'Fox']); // Cat Cat_size_m Dog Fox
cat('Tail', ['Dog', 'Fox']); // Cat-Tail Dog Fox

// Cat-Tail Cat-Tail_length_small Dog Fox
cat('Tail', { length: 'small' }, ['Dog', 'Fox']);
```

core

```
npm i @bem-react/core
```



4GIFs
.com

core

```
const cnWelcomeShri = cn('WelcomeShri');

function WelcomeShri(props: any) {
  return (
    <div className={props.className}>
      <h1 className={cnWelcomeShri('Title')}>Привет, ШРИ!</h1>
      <div className={cnWelcomeShri('Description', { visible:
props.isFull })}>
        Даешь БЭМ в ваш React!
      </div>
    </div>
  );
}

export const App = withBemMod(cnWelcomeShri(), { isFull: true })
(WelcomeShri)
```

di

```
npm i @bem-react/di
```

di — кладем в регистр

```
import { App as AppCommon } from './App';
import { cn } from '@bem-react/classname';
import { Registry, withRegistry } from '@bem-react/di';
import { WelcomeShri } from '../WelcomeShri/WelcomeShri@desktop';

const cnApp = cn('App');
const cnWelcomeShri = cn('WelcomeShri');

const registry = new Registry({ id: cnApp() });

registry.set(cnWelcomeShri(), WelcomeShri);

export const App = withRegistry(registry)(AppCommon);
```

di — вынимаем из реестра

```
<RegistryConsumer>
  {registries => {
    const registry = registries[cnApp()];
    const WelcomeShri = registry.get<IWelcomeShriProps>(cnWelcomeShri());

    return <WelcomeShri/>;
  }}
</RegistryConsumer>
```

Определяем платформу

```
import DeviceDetector from 'device-detector-js';

const ua = new DeviceDetector().parse(navigator.userAgent) || {};
const device = ua.device || { type: 'desktop' };

ReactDOM.render(
  device.type === 'desktop' ? <AppDesktop /> : <AppTouch />,
  document.getElementById('root')
);
```

Инструментарий





Инструментарий

- › bem-tools-create
- › postcss-css-to-bem-css

Что дальше

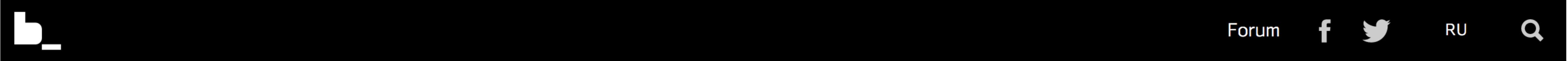




Quick start Key concepts Naming convention CSS JavaScript File structure Redefinition level Block modification Build Declarations
Solved problems History Articles FAQ

The main landing page for BEM methodology. It features a large graphic of blue LEGO blocks on a grid background. Overlaid on the graphic is the word "BEM" in large white letters, with "METHODOLOGY" in a white box and "DOCUMENTATION >" in a yellow box. Below this is a descriptive text block: "BEM methodology was invented at Yandex to develop sites which should be launched fast and supported for a long time. It helps to create extendable and reusable interface components." A page number "79" is in the bottom right corner.

bem.info/forum



DOCUMENTATION

- Libraries

STUDY MATERIALS

- Talks
- Events
- Blog
- Forum**
- Jobs
- Feedback form
- Subscribe

COMMUNITY

- Build with BEM
- Authors
- Acknowledgements

Tags

Forum

 tadalafil 14 June 2017

BEM at FullStack conference community

We are happy to announce the [FullStack conference](#) in London. Are you looking for all the latest JavaScript, Node.js and IoT technologies and discover the latest best practices and ideas? You are welcome to the FullStack conference.

This July BEM will be there. Don't think that it will only be about CSS. BEM will present its FullStack platform.

Now is the time of component web. Web Components are actively developing but still have some limitations such as web browsers support. BEM methodology can be applied to any platform and already has implementations which work in all the browsers. It is created as open source software and is suitable for any programming language or any framework.

Let's talk on what do you really know about BEM Methodology and existing BEM Platform. BEM was originated by Yandex more than 10 years ago. It was approved by huge number of large, small and scalable projects, with different developers and changing support teams.

Harry Roberts was one of the first who told about BEM in English speaking world. But he uses only the CSS part of the BEM methodology and doesn't go further. This July at the FullStack conference in London the BEM inventors **Sergey Berezhnoy** and **Vladimir Grinenko** will present the Unknown parts of BEM.

4 things you may have missed about BEM methodology:

- BEM is not only for CSS

Sort

Recently updated

Labels

- asktheteam
- bem-components
- bem-core
- bem-forum
- bem.info
- bugs
- build
- community
- css
- deploy
- design
- documentation
- html
- jobs
- js
- template engine
- testing

80



t.me/bem_ru

Telegram Edit View Window

Telegram

БЭМ
674 members

Pinned message
🔥 Встречайте! Скринкаст с доклада Антона @awinogradov про новые...



Denis

13:56

Здравствуйте,

Кто-нибудь уже изучал вопрос как custom properties вписываются в концепцию БЭМа? Может есть какие статьи на эту тему?



Roman

14:00



Denis

Здравствуйте, Кто-нибудь уже изучал вопрос как custom...

Прекрасно вписываются. <https://t.me/whitepapertools> полностью построены на этих свойствах.

Telegram

whitepaper
Declarative Design System <https://whitepaper.tools>



Denis

14:10

Спасибо, изучу.

Понятно, вы их используете для создания тем оформления. Получается можно налету поменять тему оформления изменив класс, прикольно)



Sergey Belozyorov

edited 17:16



Denis

Понятно, вы их используете для создания тем оформлени...

Да. Но только при 2-х условиях

1. Нужные браузеры поддерживают это.

2. Дизайнер понимает темы (а не рисует так, что невозможно использовать микс темы).

facebook.github.io/create-react-app

github.com/awinogradov/cra-ssr-boilerplate

youtu.be/pVzIkCidOYg

b_-
KEEP
CALM
AND
BEM



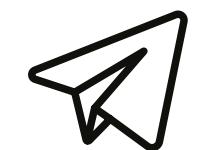
FOX41
WDRB

Спасибо!

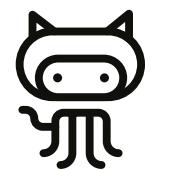
Владимир Гриненко



info@bem.info



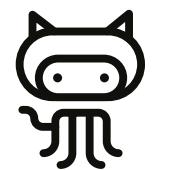
t.me/bem_ru



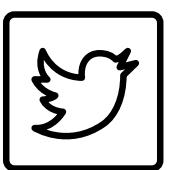
tadatuta



tadatuta



bem



bem_ru



