Как создать мультиплатформенную дизайн-систему на React

React

design system

Илья Лесик lessmess

Software Engineer & Co-founder

lessmess

 github.com/lessmess-dev

 @lessmessdev

 dev.to/lessmess

lessmess.agency

tver.io

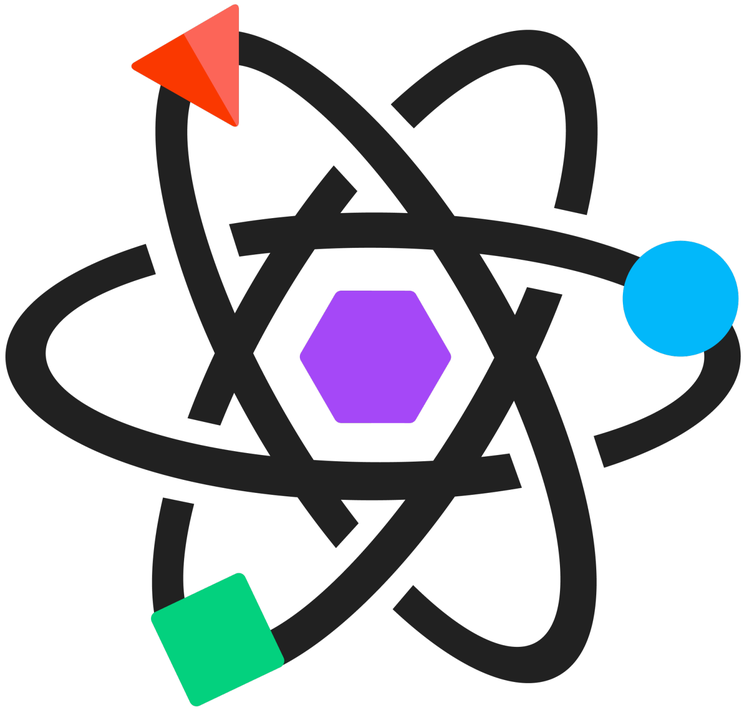
Community Leader

_tverio

TverIO

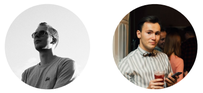
React Figma

PO & Main contributor



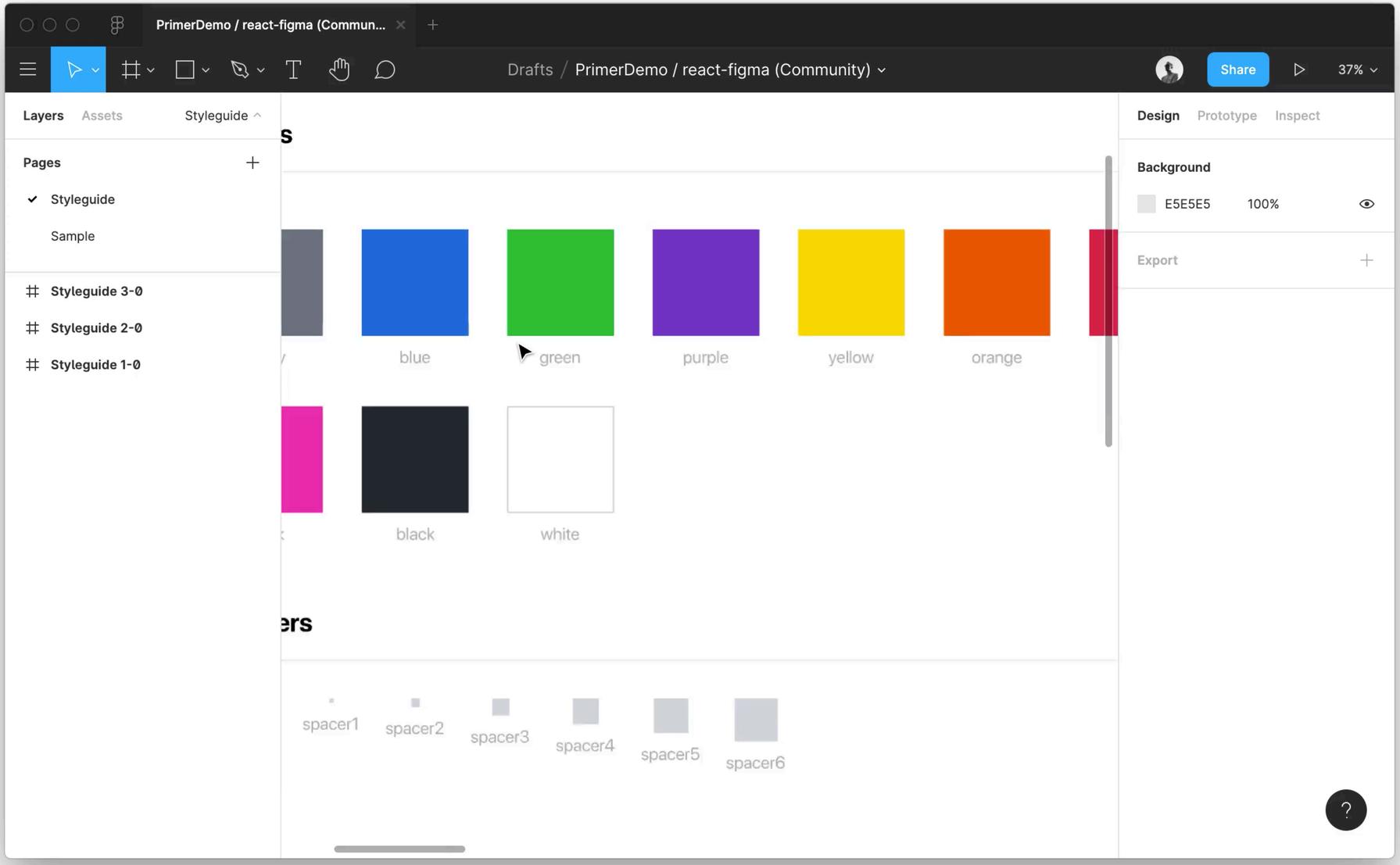
github.com/react-figma

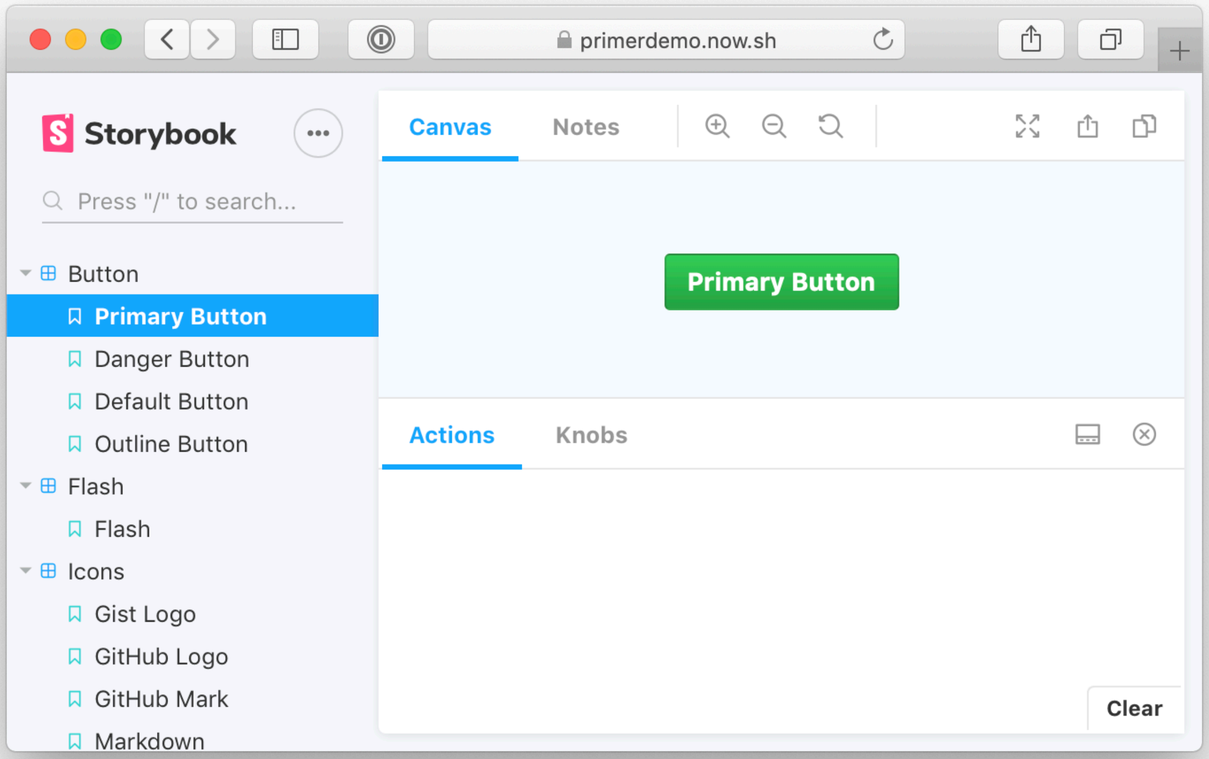
react-figma.dev



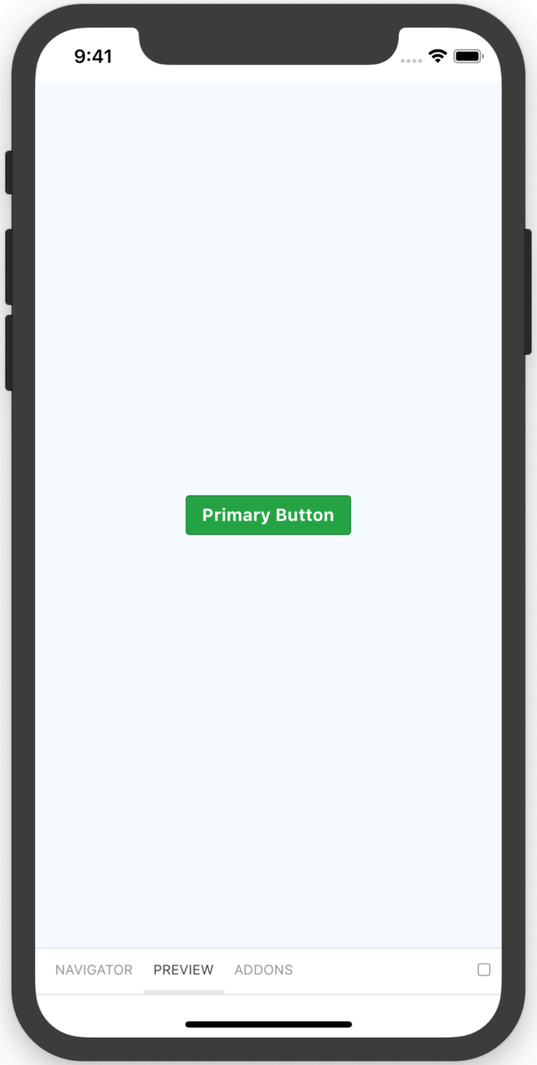
TL;DR

TL;DR


PrimerDemo / react-figma (Commun...
Drafts / PrimerDemo / react-figma (Community)
Share
37%
Layers
Assets
Styleguide
Pages
Styleguide
Sample
Styleguide 3-0
Styleguide 2-0
Styleguide 1-0
Design
Prototype
Inspect
Background
E5E5E5
100%
Export
blue
green
purple
yellow
orange
black
white
spacer1
spacer2
spacer3
spacer4
spacer5
spacer6


primerdemo.now.sh
Storybook
Press "/" to search...
Button
Primary Button
Danger Button
Default Button
Outline Button
Flash
Flash
Icons
Gist Logo
GitHub Logo
GitHub Mark
Markdown
Canvas
Notes
Primary Button
Actions
Knobs
Clear

Веб


9:41
Primary Button
NAVIGATOR
PREVIEW
ADDONS

Мобайл

Проблема

Написали кучу кода на React под веб

Понадобилось мобильное приложение 😱

Проблема

Сделали кучу экранов и компонентов на дизайне

↓

Трудно обеспечить **консистентность** 😱

План доклада

- Кроссплатформенность
- Дизайн-системы
- Пример
- Процесс + Рекомендации
- Будущие планы

Инструменты

- Figma
- React
  - React Native / react-native-web
  - React Figma
- Storybook

Кроссплатформенность


JS
LIGHTYEAR
JS EVERYWERE
imgflip.com

Можем использовать **JS-**код

для хранения элементов **дизайн-системы**

JS-code

Design System

```
const designSystem = {
    fontSizes: [
        12, 14, 16, 20, 24, 32
    ],
    colors: {
        blue: '#07c',
        green:'#0fa',
    }
}
```

Неплохо, но хотим переиспользовать элементы более высокого уровня

high level

Кросс-платформенный React

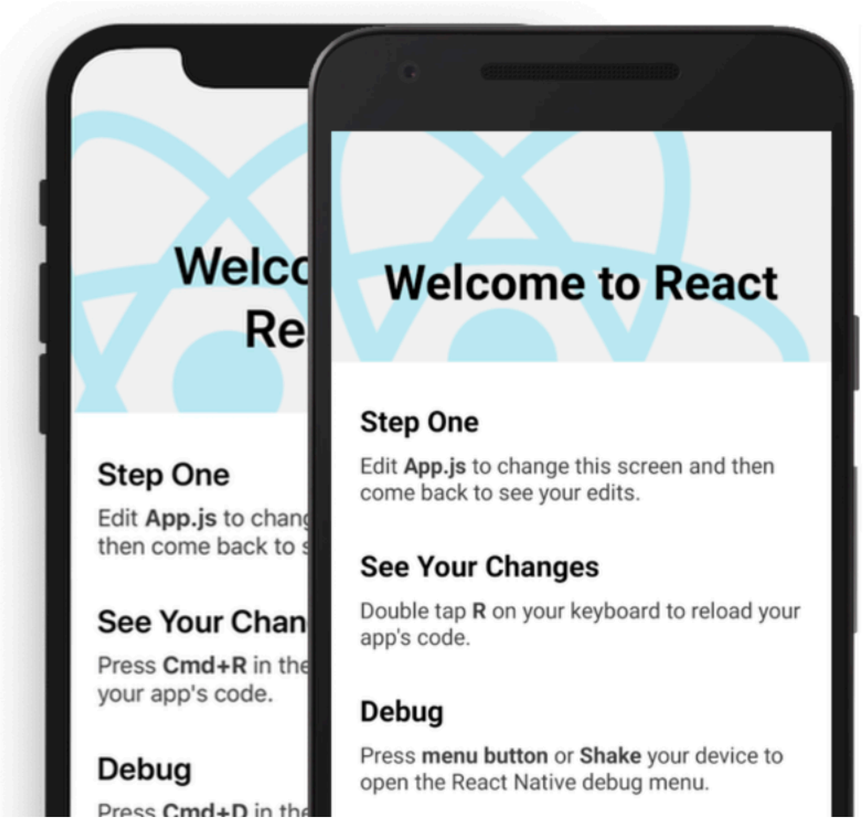
Веб 

```
const Component = () => {
  return <div>Hello</div>
}
```

React DOM

Все остальное 

- react-native

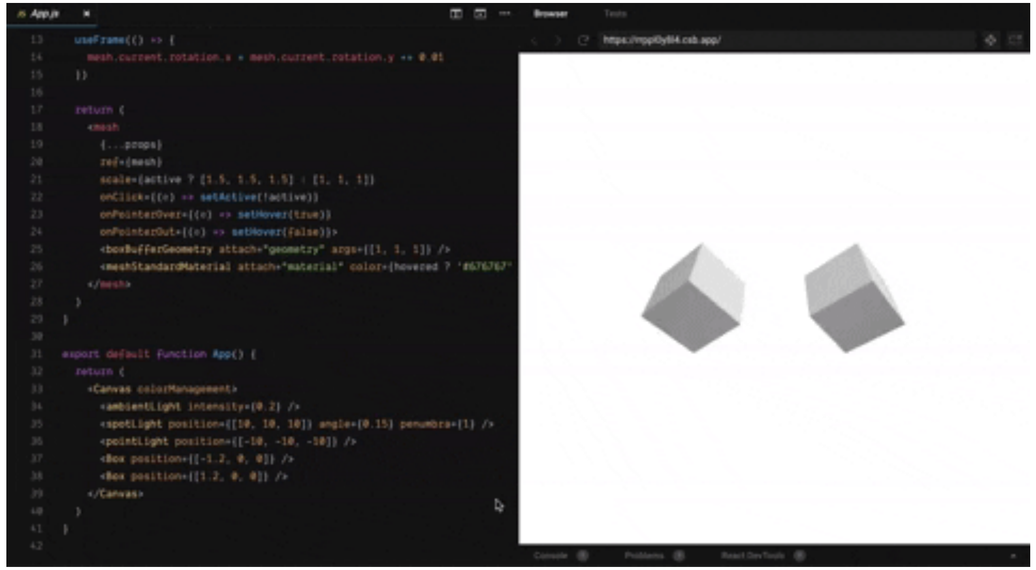


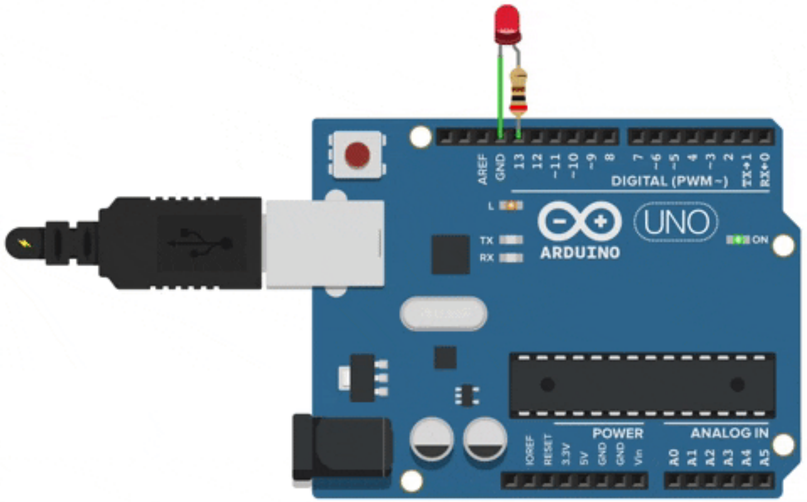
Create native apps for Android and iOS using React

React Native combines the best parts of native development with React, a best-in-class JavaScript library for building user interfaces.

Use a little—or a lot. You can use React Native today in your existing Android and iOS projects or you can create a whole new app from scratch.

- react-tree-fiber
- react-360
- Ink
- react-hardware





- react-sketchapp
- react-figma
- react-xd
- …

README.md

React Sketch.app

render React components to Sketch; tailor-made for design systems

Quick-start 🏃

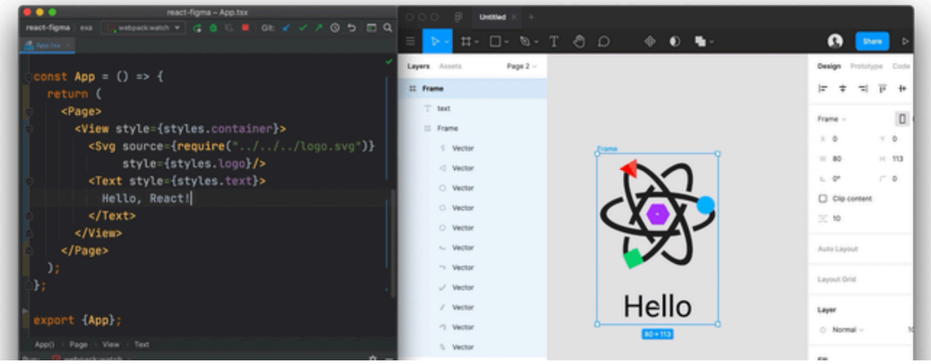
README.md

React Figma

all contributors 7 | npm v0.1.17 | circleci passing | 💬 join the community on spectrum

A React renderer for Figma. Use React components as a source for your designs.

- 🛰 Compatible with react-native, react-sketchapp, react-primitives API.
- 🦄 Flexible layouts support with Yoga Layout.
- ♻️ Hydration and HMR support.
- ⚙️ Built on Figma Plugin API.
- 🚫 **Is not a code generator**.



Почему React хорош для кросс-платформы?

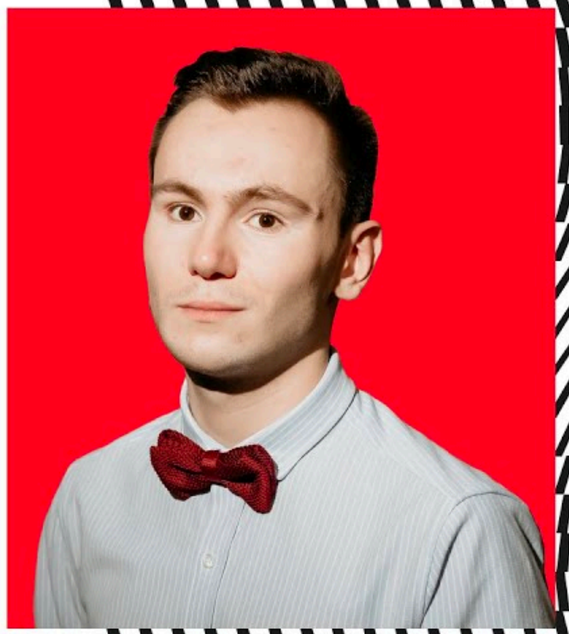
Правильные абстракции!

react-reconciler

```
const Reconciler = require('react-reconciler');

const HostConfig = {
  // You'll need to implement some methods here.
  // See below for more information and examples.
};

const MyRenderer = Reconciler(HostConfig);

const RendererPublicAPI = {
  render(element, container, callback) {
    // Call MyRenderer.updateContainer() to schedule changes on the roots.
    // See ReactDOM, React Native, or React ART for practical examples.
  }
};

module.exports = RendererPublicAPI;
```

React Reconciler: как написать собственный рендерер

Ярослав Лосев



JavaScript JavaScript JavaScript JavaScript JavaScript JavaScript JavaScript JavaScript

tver \o/

React Reconciler: Как написать собственный рендерер

RU / День 1 / 10:45 / Зал 3

В избранное

React — удивительно универсальная технология, прочно вошедшая в нашу жизнь. Применение React уже давно вышло за пределы веба, его используют в огромном множестве задач: от разработки мобильных приложений и до генерации макетов. Интересно ли вам, как такое стало возможно?

В ходе доклада мы разберем, что такое React Reconciler и как с его помощью создаются рендеры, напишем мини-версию привычного React DOM и увидим, как можно связать React с более экзотичными средами на примере отрисовки React-компонентов в Figma.

Доклад окажется полезным практикующим разработчикам и прольет свет на внутреннее устройство архитектуры React и React DOM.

Спикеры

Ярослав Лосев

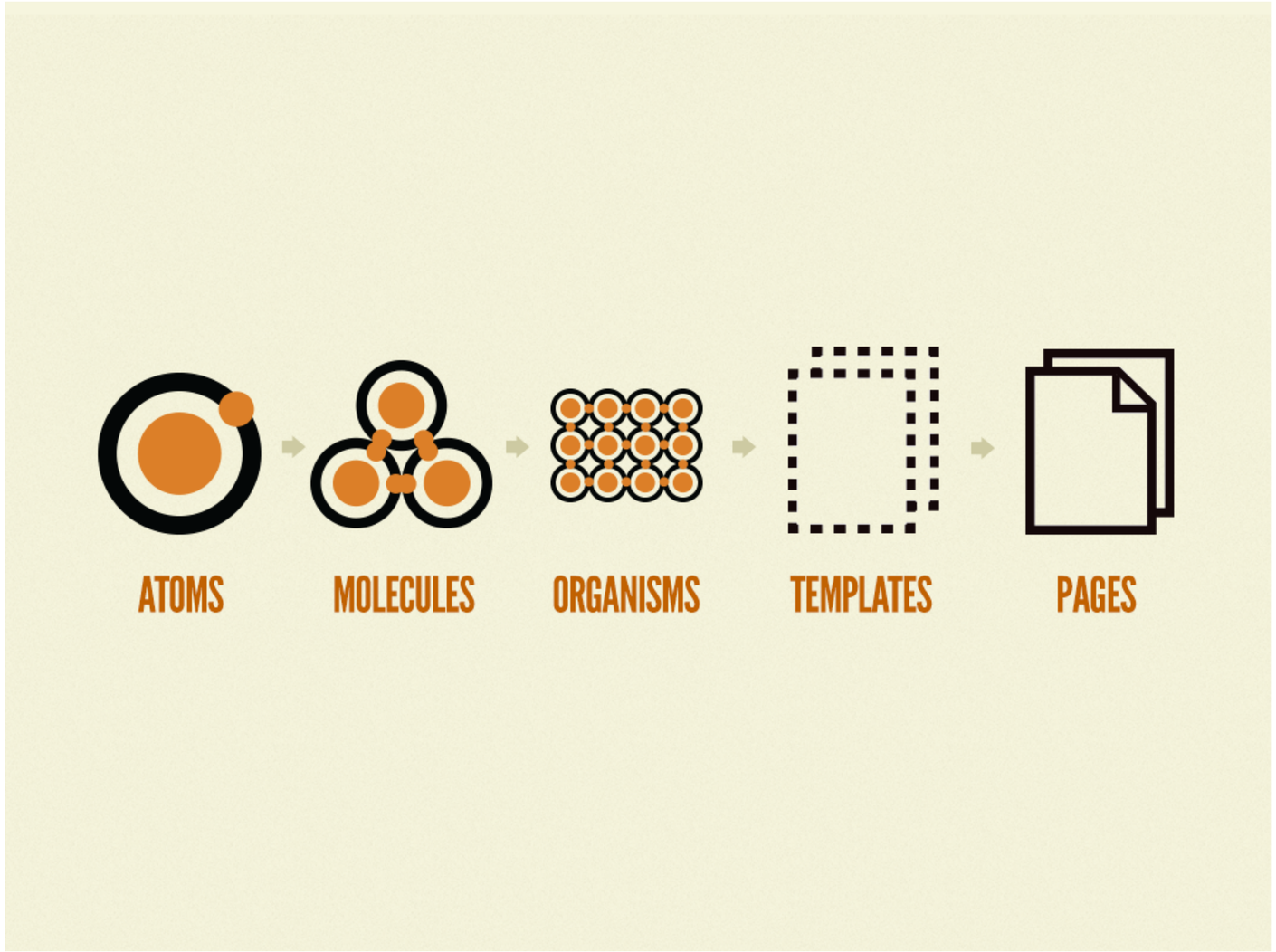
Фронтенд-инженер. Контрибьютор проекта React Figma. Занимается разработкой еще со школы, а свой первый UI написал на Delphi. В настоящее время без ума от React.

Кроссплатформенность возможна за счет:

- **JavaScript** (почти везде есть интерпретатор)
- **React** (react-reconciler)

Дизайн-системы

Design System



Atomic Design

by Brad Frost

От простого к сложному

Стайл-гайд + Визуальные элементы

Стайл-гайд 🎨

Стайл-гайд — это набор базовых значений

- Цвета
- Типографика
- Размеры/отступы

typography

HEADING 1

HEADING 2

HEADING 3

HEADING 4

Lead paragraph. We decided to skip this tournament due to objective obstacles that prevent us from gathering players at boot camp.

Body copy. The coronavirus has caused teams to limit travel and play from home, meaning that if certain players don't live near the team's training facility or even within the same region, it makes competing in high-level tournaments difficult. All of Gambit's players live within the CIS region, but it appears that the organization wants the team to practice more before competing in another event.

Body active style

UPPERCASE

UPPERCASE SMALL

Body small. 2013–2020 Gambit Gaming

colors

spacing

gambit.gg - стайл-гайд

UI-кит 🧰

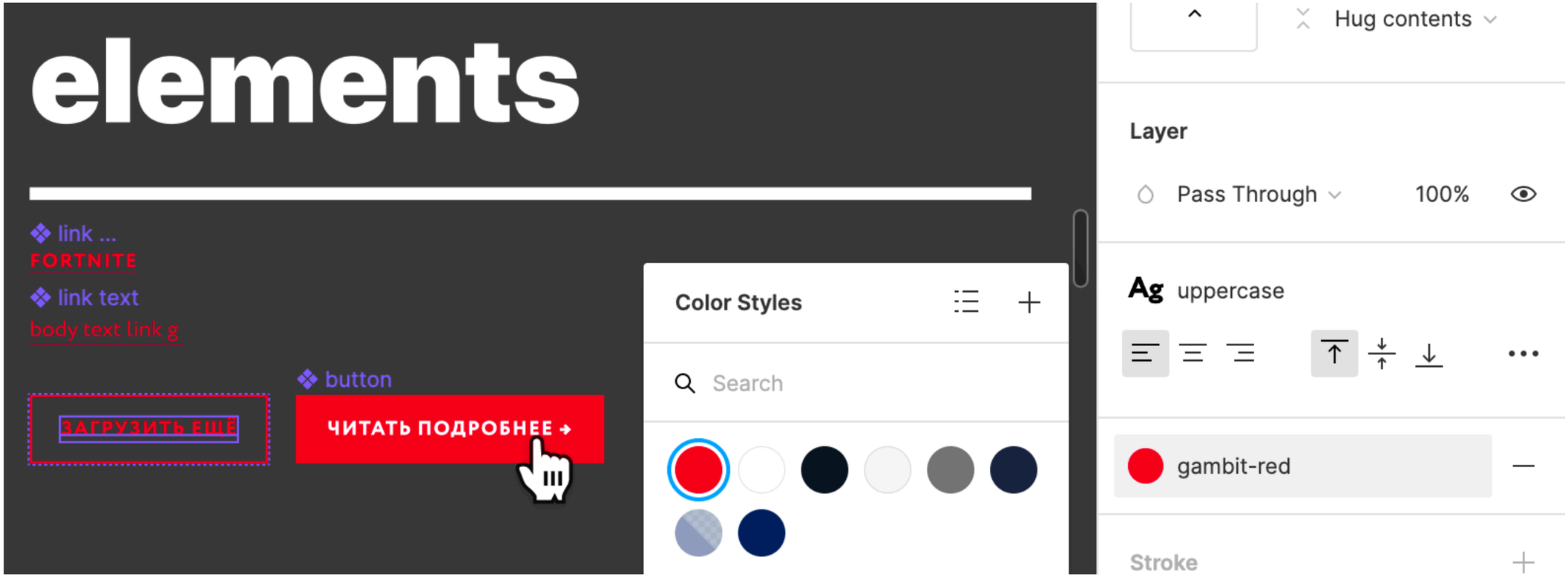
UI-kit

UI-кит — это набор визуальных элементов

- Иконки
- Кнопки
- Инпуты
- Тултипы
- …

UI-kit

UI-кит **использует** стайл-гайд

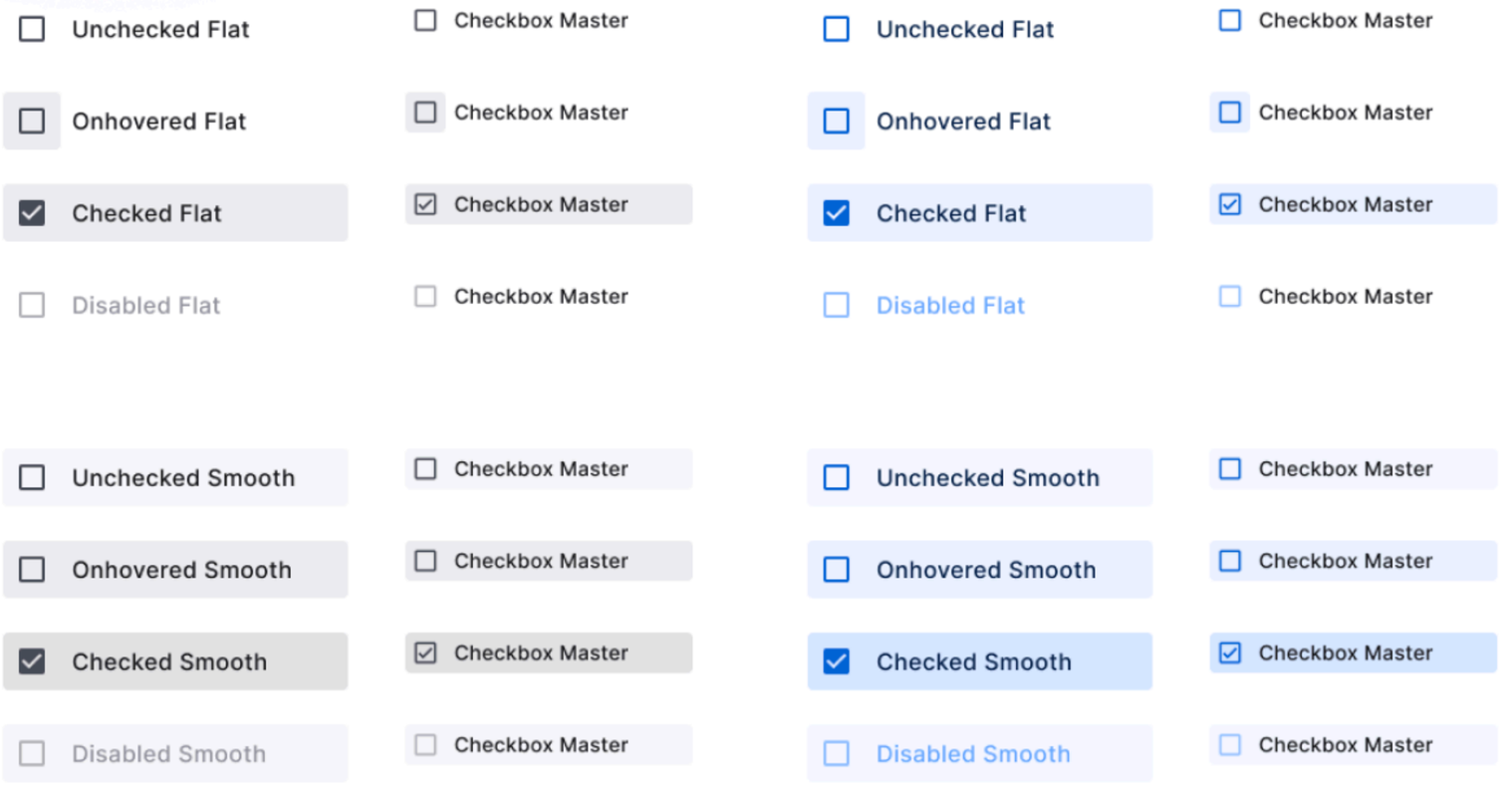


gambit.gg - UI-kit

UI-кит содержит разные **состояния** элементов

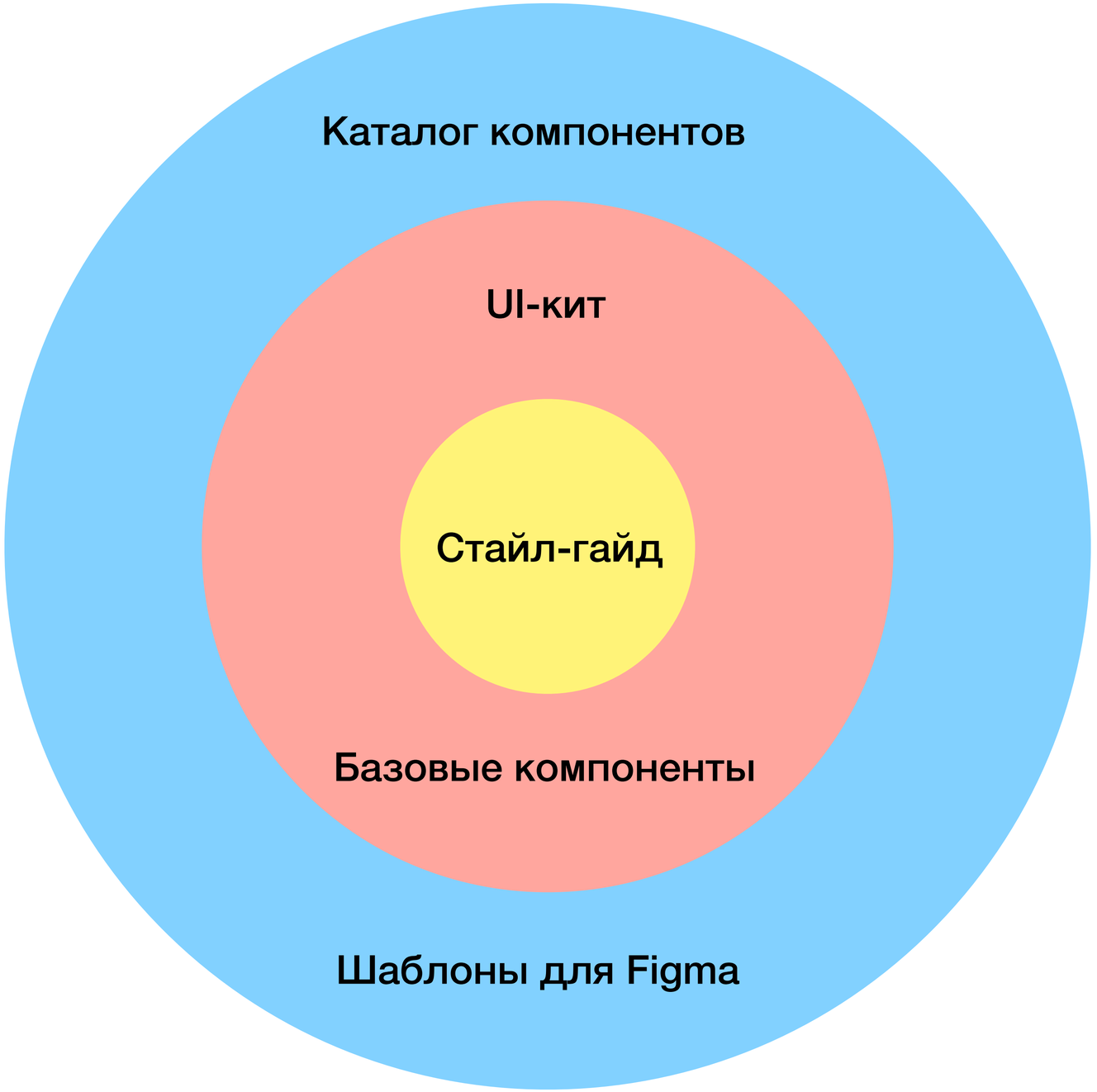
Setproduct Design System 2.0 - UI kit - Buttons

Checkboxes



Setproduct Design System 2.0 - UI kit - Checkboxes

Картина целиком


Каталог компонентов
UI-кит
Стайл-гайд
Базовые компоненты
Шаблоны для Figma

Все это может быть в **коде**!

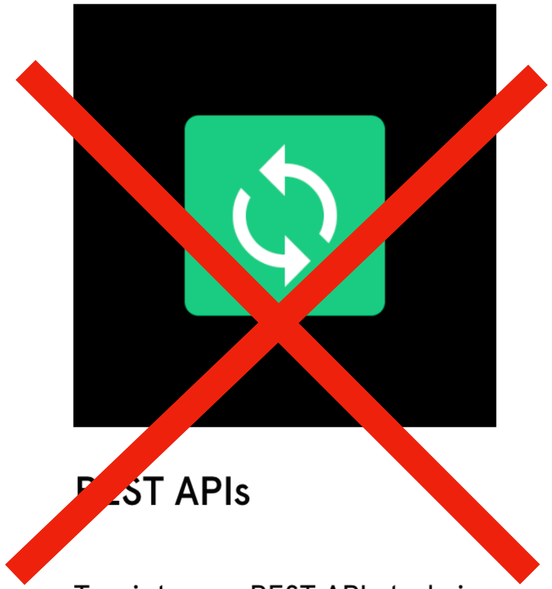
По аналогии с Infrastructure as Code (IaC)

Design System as Code

Code

Но как отобразить дизайн-систему в **Figma**?

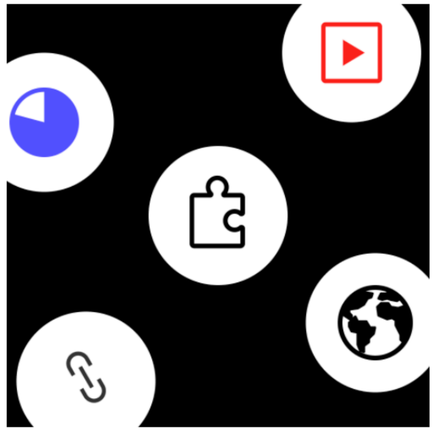
Только
на **чтение**



REST APIs

Tap into our REST APIs to bring Figma into your external tools and products.

Go to docs

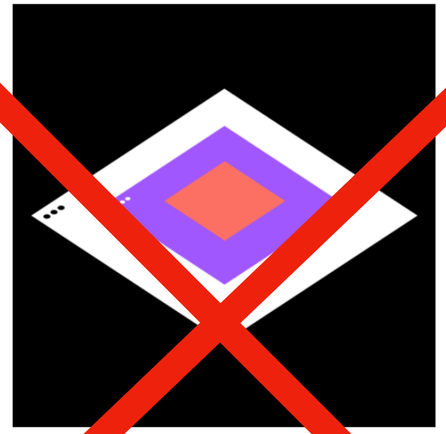


Plugins

Automate tasks and improve your workflow. Publish plugins for your team or to the world.

See plugins

Встраивание
в **iframe**

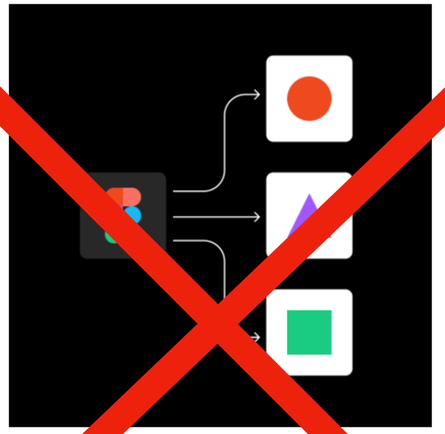


Embeds

Embed live Figma designs and prototypes wherever you need them.

Add embeds

Обертки над
REST API и
Эмбедами



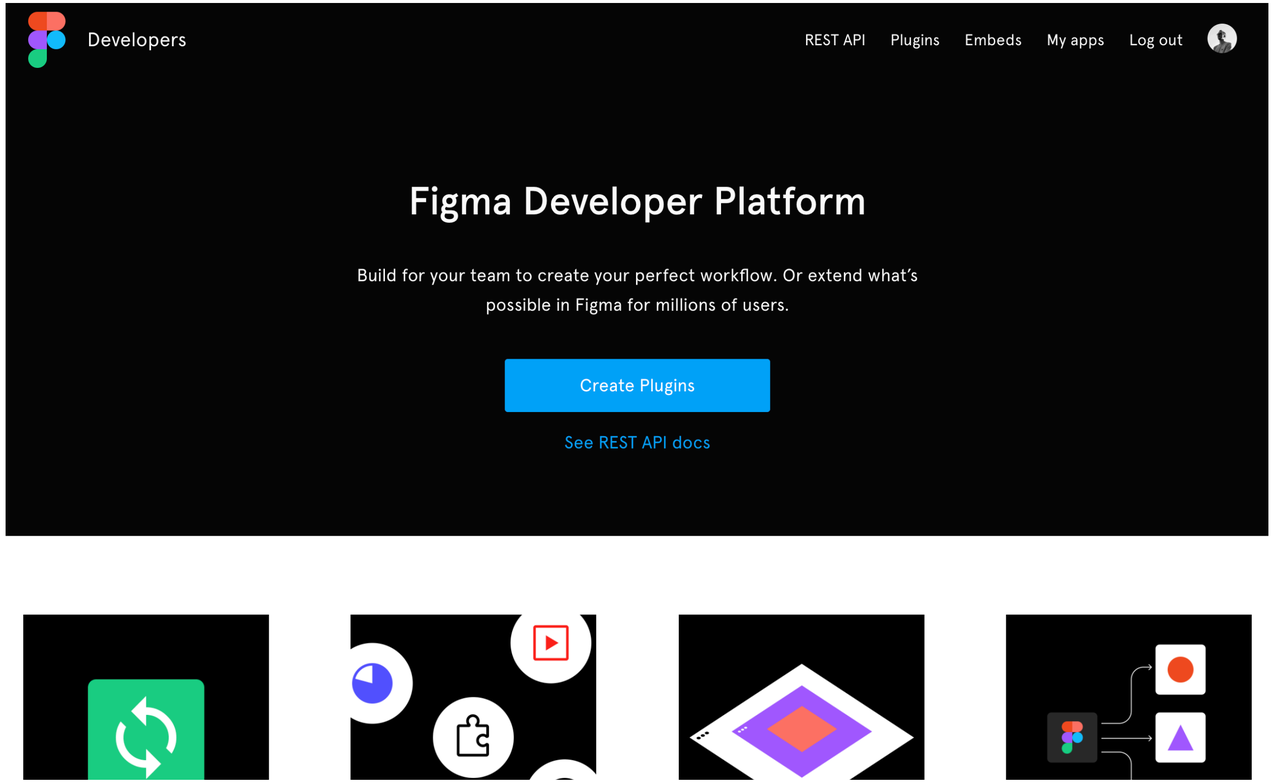
Integrations

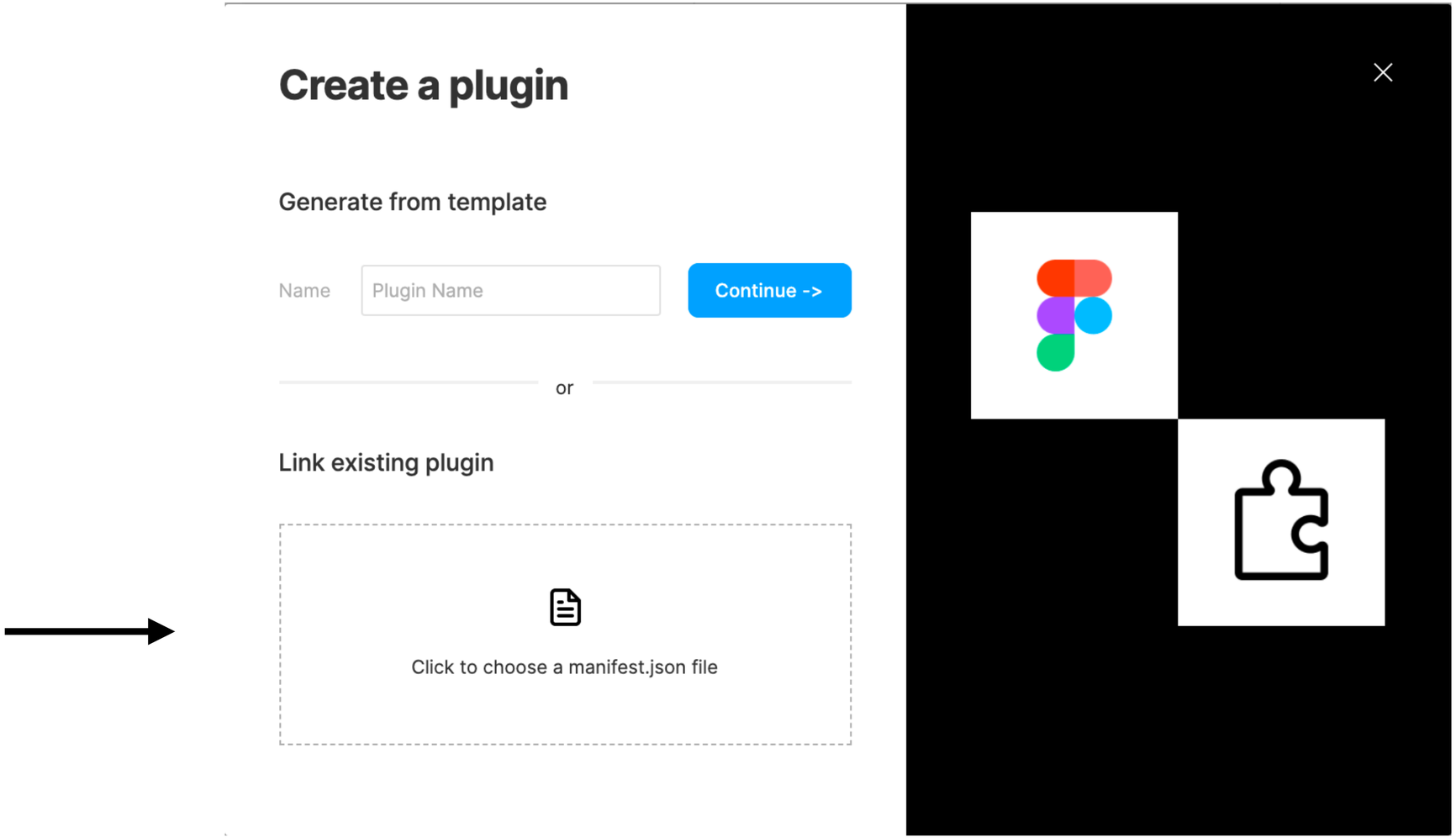
Connect Figma with other tools in your design workflow. Collaborate better as a team.

View integrations

Источник: figma.com/developers

Figma Plugins API




Create a plugin
Generate from template
Name
Plugin Name
Continue ->
or
Link existing plugin
Click to choose a manifest.json file

manifest.json

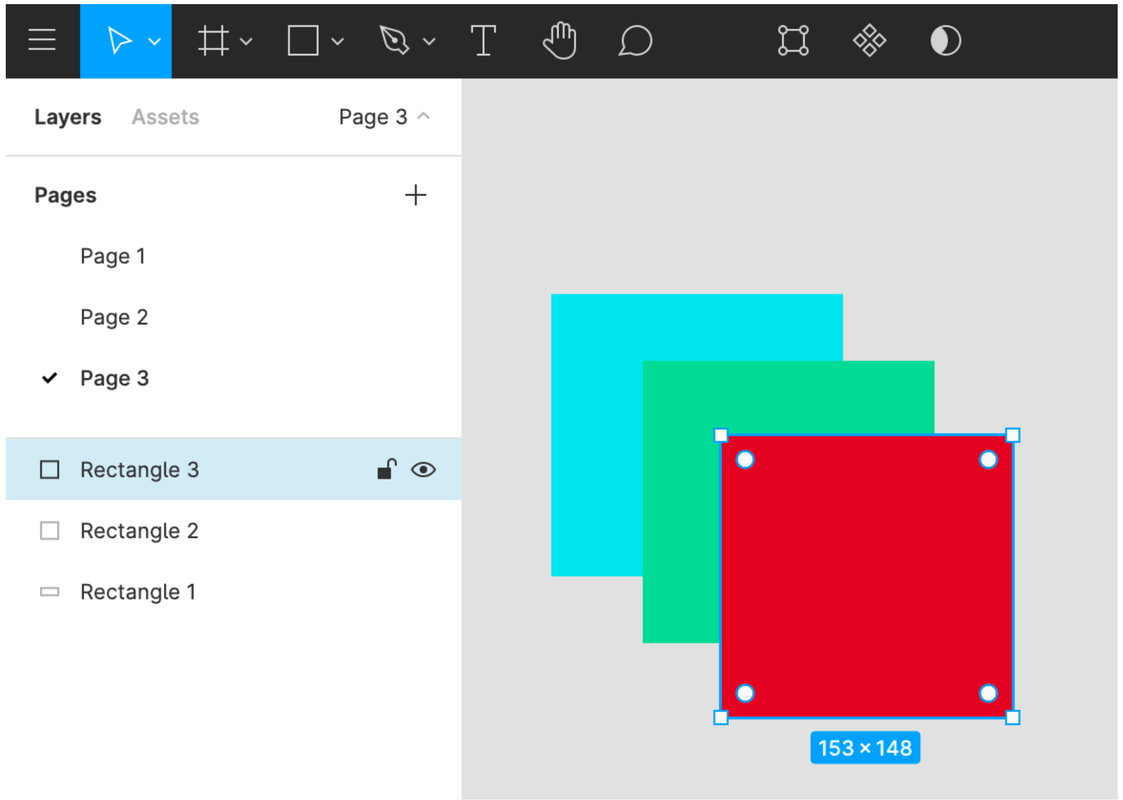
```
{
   "name": “Our design-system",
   "id": “…”,
   "api": "1.0.0",
   "main": "code.js",
   "ui": "ui.html"
}
```

Две точки входа

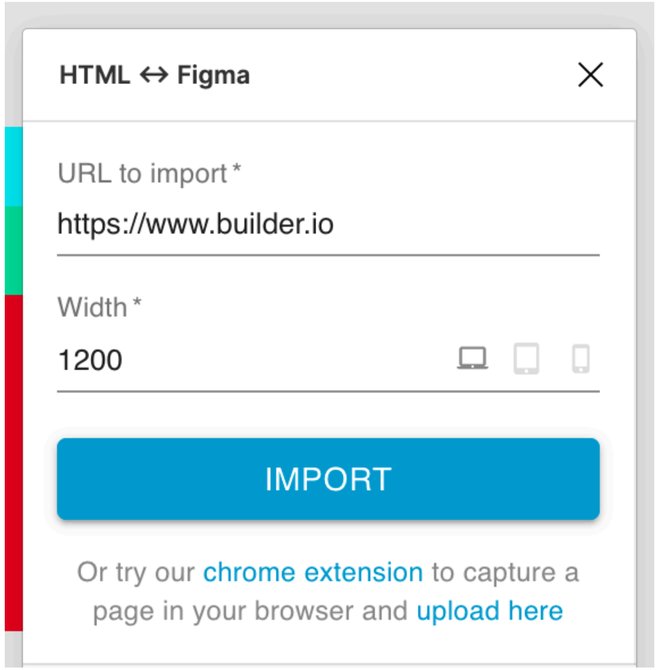
Сообщения

main ⟷ **ui**

Управление содержимым документа



Интерфейс плагина



В main-треде доступно **Figma Plugins API**

Код можно писать на ES6

Figma Plugins API

```
const page = figma.createPage();
const rect = figma.createRectangle();
rect.x = 10;
rect.y = 10;
rect.resize(50, 50);
page.appendChild(rect);
```

Код можно бандлить — **Webpack**

И использовать **react-figma** **вместо чистого API**

Тот же самый код, только на **react-figma**:

```
<Page>
    <Rectangle x={10} y={10} width={50} height={50} />
</Page>
```

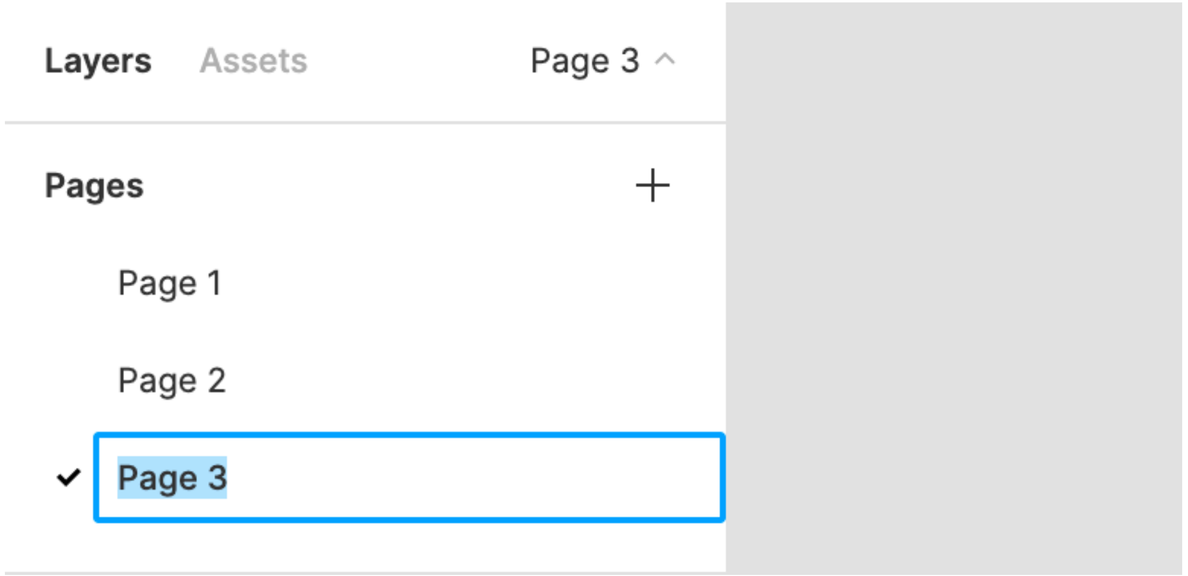
Преимущества **react-figma** перед Figma Plugins API

- Декларативный подход
- Все фичи React: хуки, стейт
- **Кроссплатформенность** + Yoga Layout
- DX: Гидратация, HMR, React DevTools
- Интегрируется с множеством других инструментов

Примитивы react-figma

<Page />

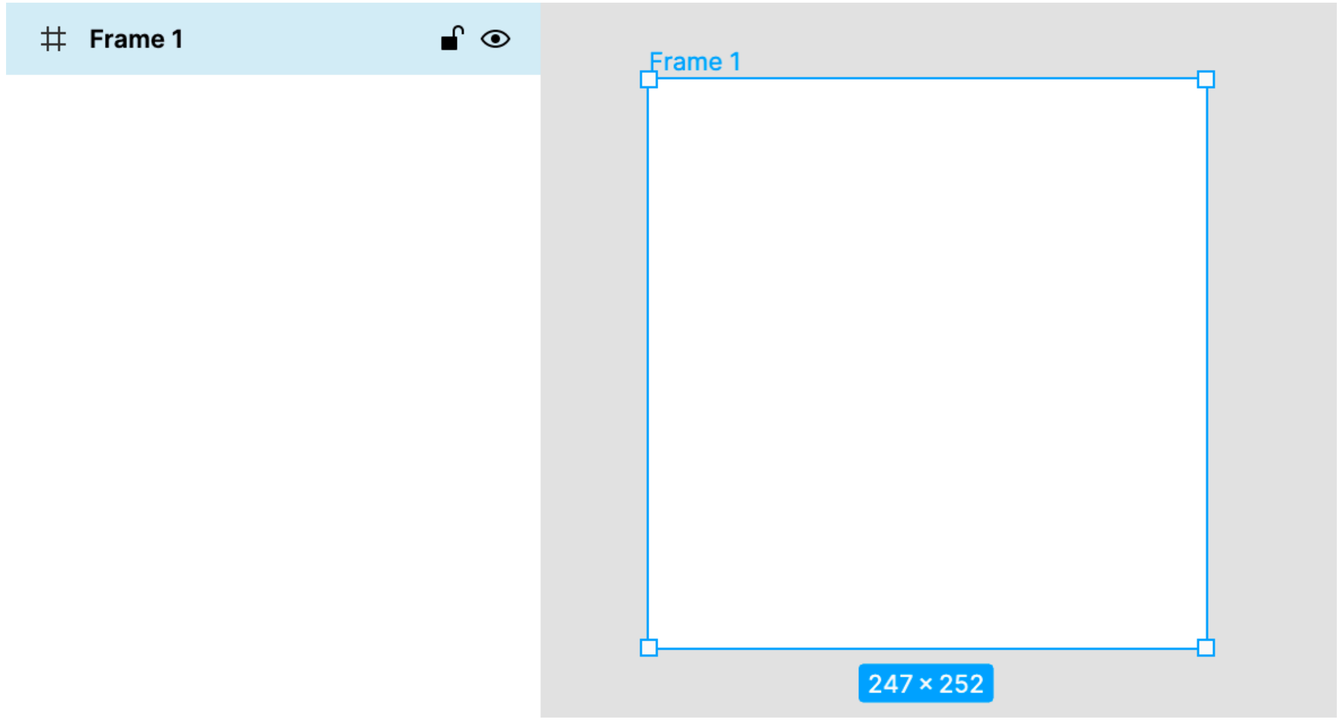
figma.createPage()



<Frame />

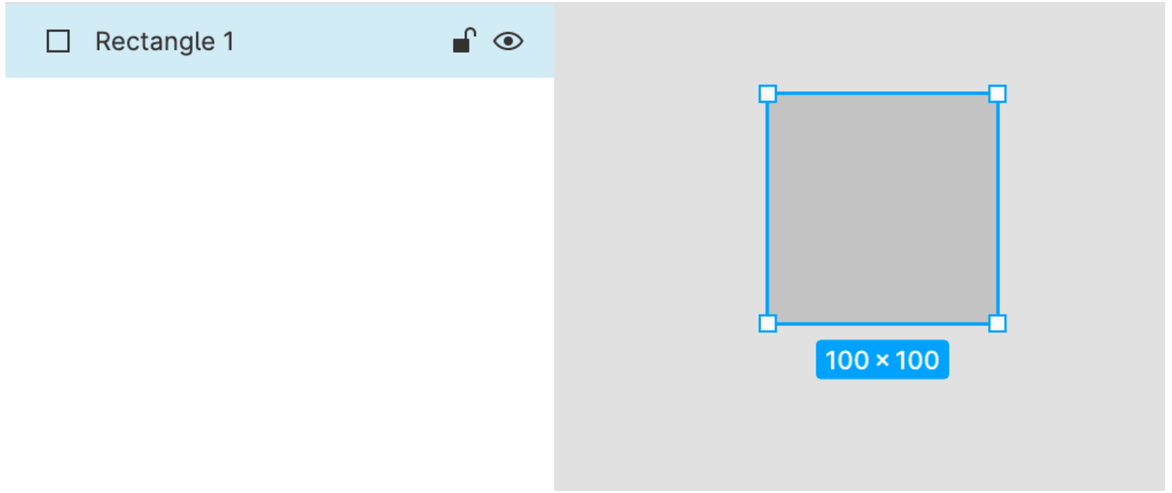
figma.createFrame()

Основной блок Figma



<Rectangle />

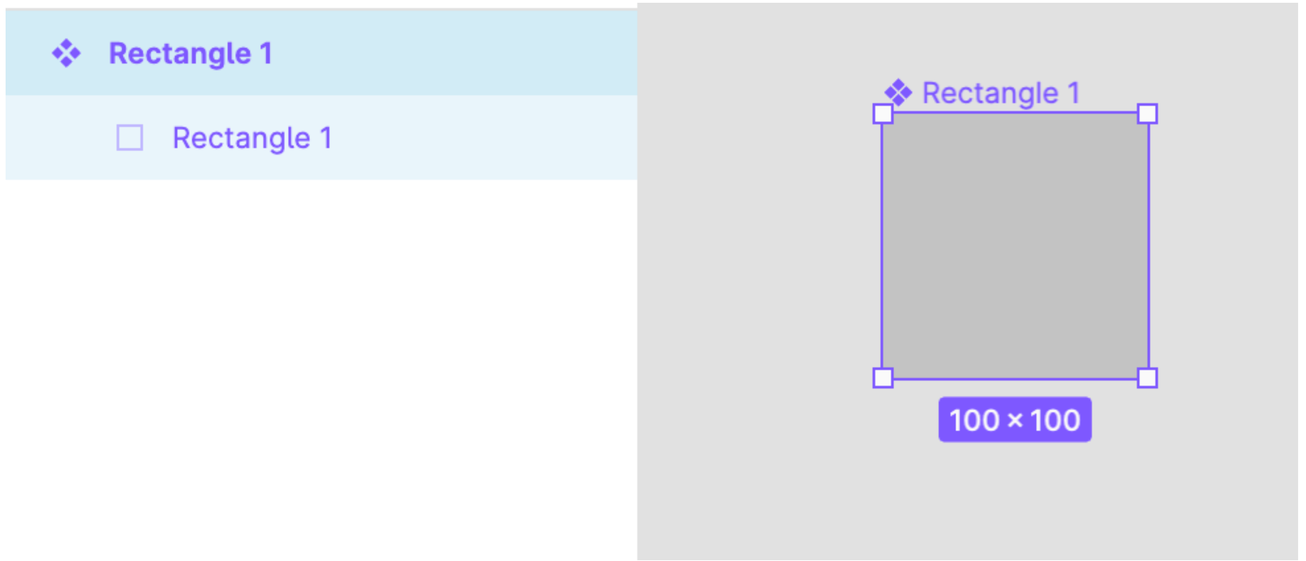
figma.createRectangle()



<Component>

...

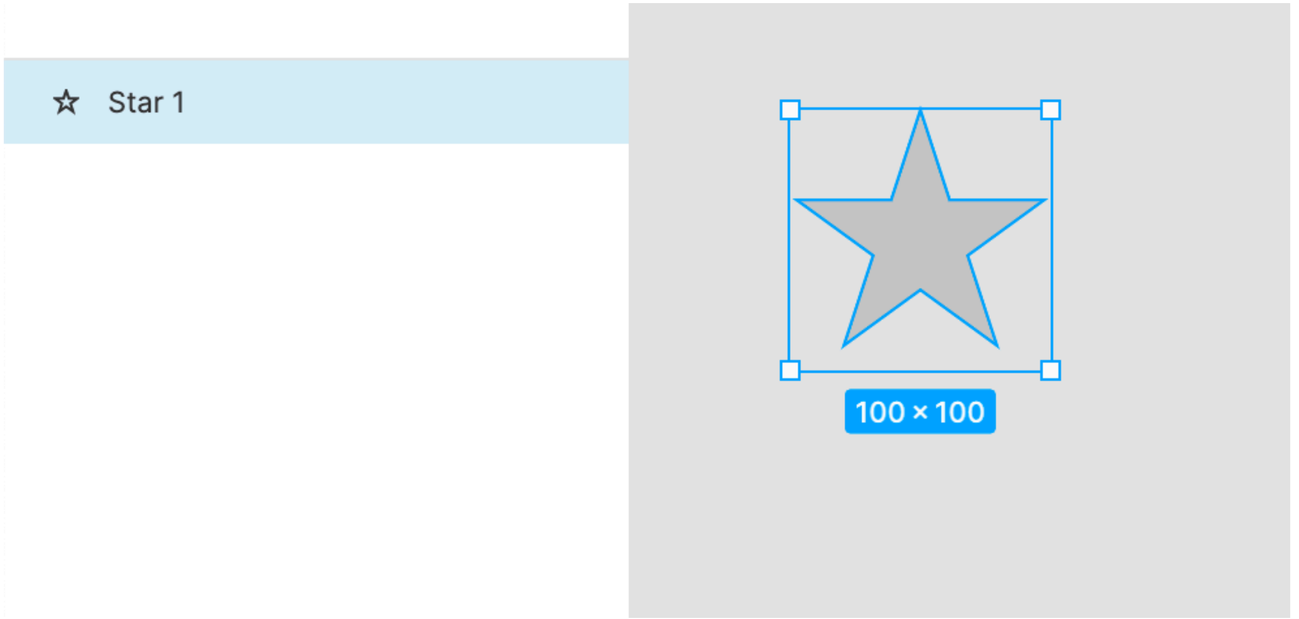
</Component>



figma.createComponent()

<Star />

figma.createStar()

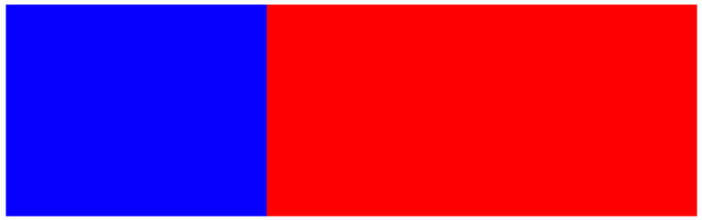


UI примитивы

Совместимы с react-native / react-native-web / …

View — фундаментальный компонент для построения UI

```
import React from "react";
import { View } from "react-native";

const ViewBoxesWithColor = () => {
    return (
        <View
            style={{
                flexDirection: "row",
                height: 100,
                padding: 20
            }}
        >
            <View style={{ backgroundColor: "blue", flex: 0.3 }} />
            <View style={{ backgroundColor: "red", flex: 0.5 }} />
        </View>
    );
};

export default ViewBoxesWithColor;
```



Text — отображение текста

```
import React from "react";
import { Text } from "react-native";

const SomeText = () => {
    return (
        <Text>
            Hello, World!
        </Text>
    );
};

export default SomeText;
```

Hello, World!

StyleSheet — абстракция, *похожая* на CSS StyleSheets

```
const App = () => (
    <View style={styles.container}>
        <Text style={styles.title}>React Native</Text>
    </View>
);

const styles = StyleSheet.create({
    container: {
        flex: 1,
        padding: 24,
        backgroundColor: "#eaeaea"
    },
    title: {
        marginTop: 16,
        paddingVertical: 8,
        borderWidth: 4,
        borderColor: "#20232a",
        borderRadius: 6,
        backgroundColor: "#61dafb",
        color: "#20232a",
        textAlign: "center",
        fontSize: 30,
        fontWeight: "bold"
    }
});
```

Почему не **Веб**-based?

Хотим писать `<div className="alert" />`

Проблемы:

Обработка CSS

- CSS ::after, ::before
- CSS ::hover, ::focus
- CSS Grid, etc.

Поддерживается styled-components 💅

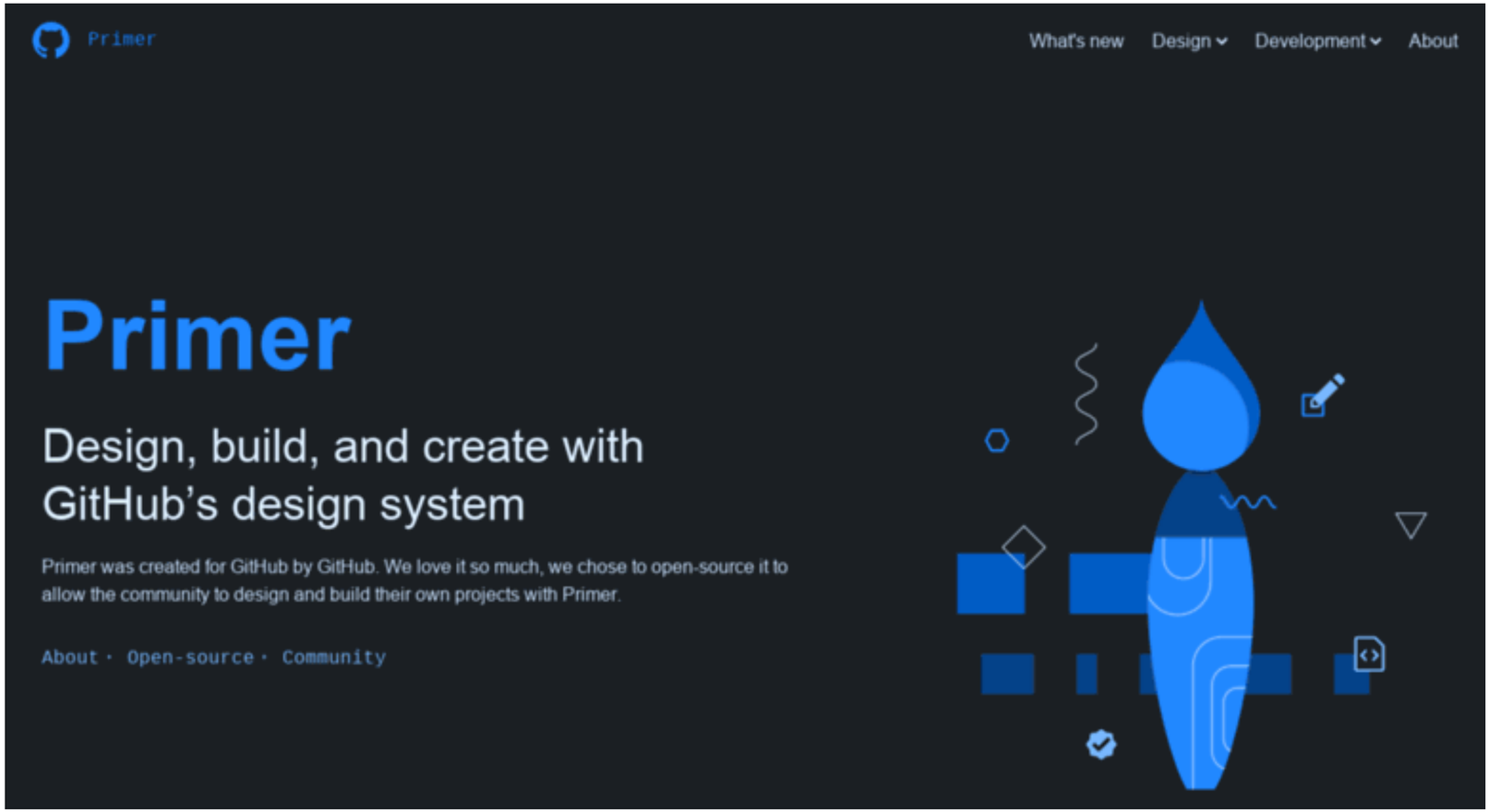
```
const SwatchTile = styled.View`
    height: 250px;
    width: 250px;
    border-radius: 4px;
    margin: 4px;
    background-color: ${props => props.hex};
    justify-content: center;
    align-items: center;
    ${props => (props.hex === '#ffffff' ? `border-width: 2px;\nborder-color: black;\nborder-style: solid;` : '')}
`;
```

github.com/react-figma/react-figma/blob/master/examples/styled-components/

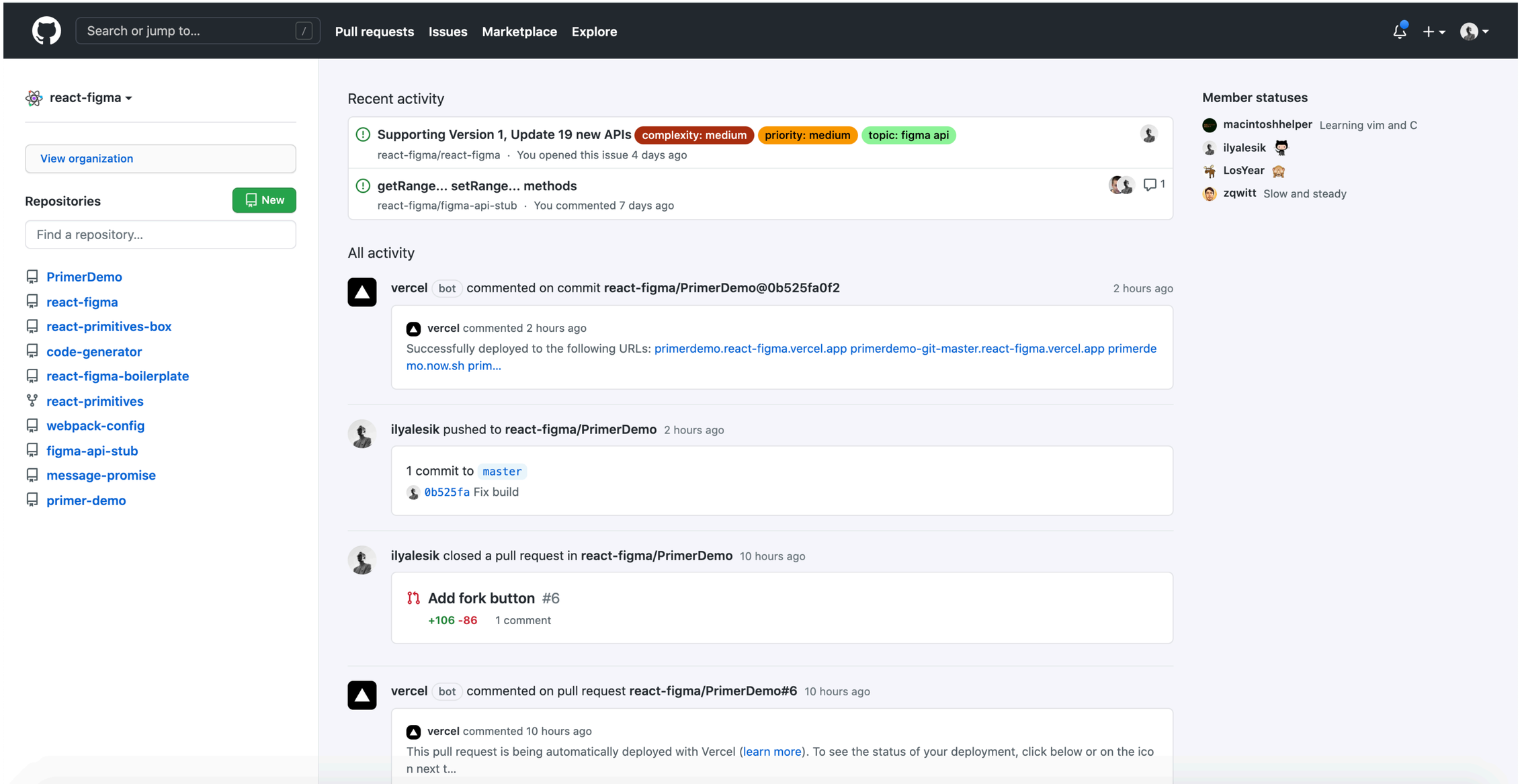
- Стайлгайд — JS-конфиг
- Базовые элементы — React (в react-native стиле)
- Шаблоны для Figma и каталог - тоже в коде!

Пример?

Primer!


Primer
What's new
Design
Development
About
Primer
Design, build, and create with GitHub's design system
Primer was created for GitHub by GitHub. We love it so much, we chose to open-source it to allow the community to design and build their own projects with Primer.
About · Open-source · Community

Интерфейс GitHub — на Primer



В Primer есть:

- UI-kit под Веб
- Шаблоны Sketch/Figma
- Octicons

Нет:

- UI-kit на **react-native**
- Консистентности*

Сделаем кросс-платфороменный вариант

Нет задачи повторить полностью

Создадим react-native проект

\+ react-figma-boilerplate 

После генерации

```
├── android - gradle project for React Native app
├── assets - React Native app assets
├── ios - iOS project for React Native app
├── src
│   ├── components
│   ├── App.tsx - React Figma app
│   ├── code.tsx - entty point for Figma plugin Main-thead
│   ├── ui.html - entty point for Figma plugin UI-thead
│   └── ui.tsx
├── app.json - react-native config file
├── babel.config.js - babel config for react-native
├── figma.d.ts - figma plugin typings
├── figma.webpack.confgi.js - Webpack config for react-figma
├── manifest.json - Figma plugin manifest
├── metro.config.js - config for Metro bundler
├── package.json
├── tsconfig.json
└── yarn.lock
```

Цвета 🎨

Colors

Color system

Stable

View source

Color palette

Gray | Blue | Green | Purple | Yellow | Orange | Red | Pink | Black | White

```
export const colors = {
    gray: "#6a737d",
    blue: "#0366d6",
    green: "#28a745",
    purple: "#6f42c1",
    yellow: "#ffd33d",
    orange: "#f66a0a",
    red: "#d73a49",
    pink: "#ea4aaa",
    black: "#24292e",
    white: "#ffffff"
};
```

Отобразим **цвета** в Figma

Стили

```
import * as React from "react";
import {StyleSheet, Text, View} from "react-figma";

const styles = StyleSheet.create({
    container: {
        alignItems: "center"
    },
    rect: {
        width: 100,
        height: 100
    },
    text: {
        marginTop: 10
    }
});
```

Компонент

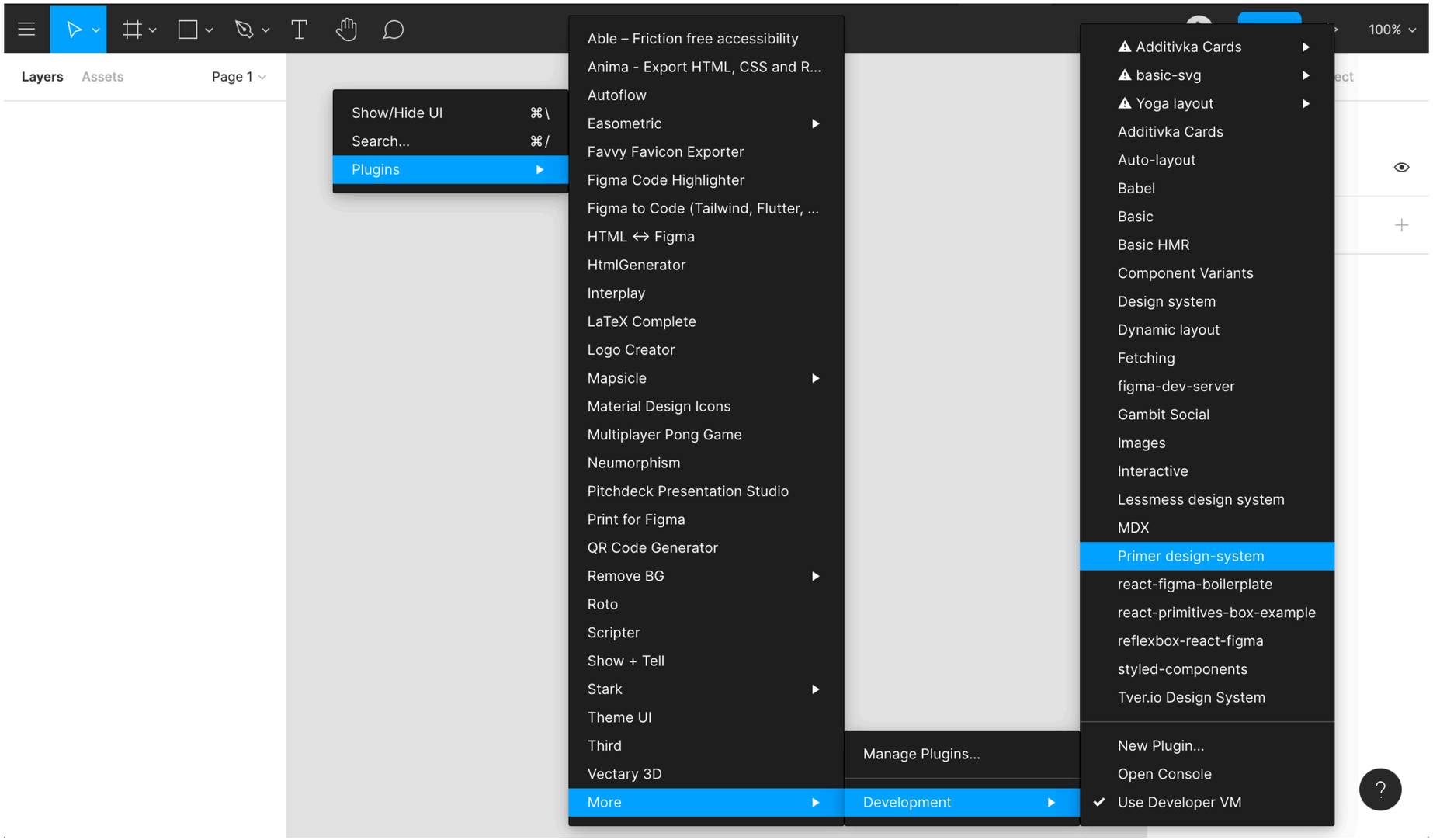
```
…

export const StyleguideColor = (props) => {
    const {color, withBorder, name} = props;
    return <View name="Color container" style={[styles.container, props.style]}>
        <View style={[
            styles.rect,
            {backgroundColor: color},
            withBorder && {borderColor: "#c8c8c8", borderWidth: 1}
        ]} />
    </View>
};
```

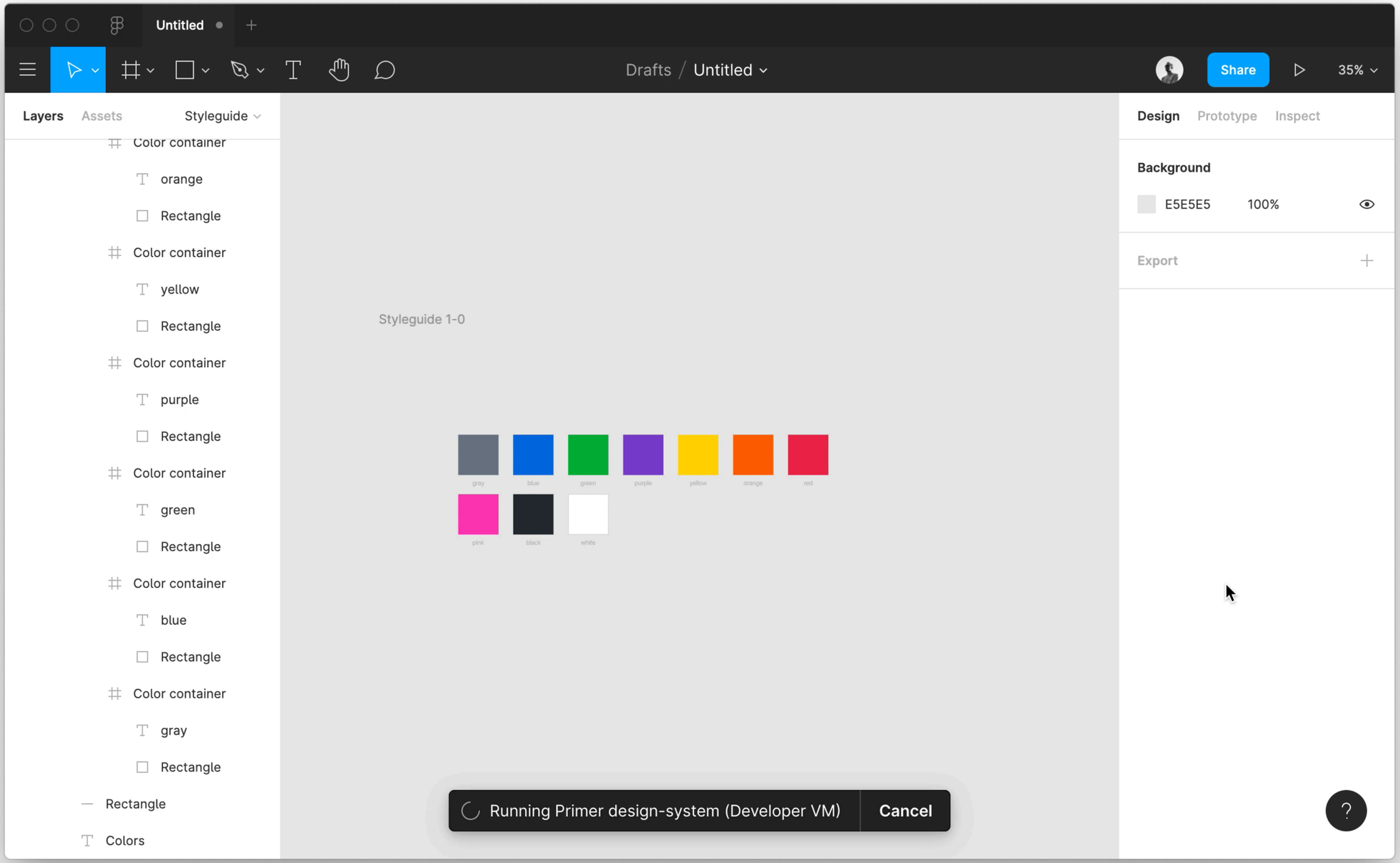
Собираем цвета на одном фрейме

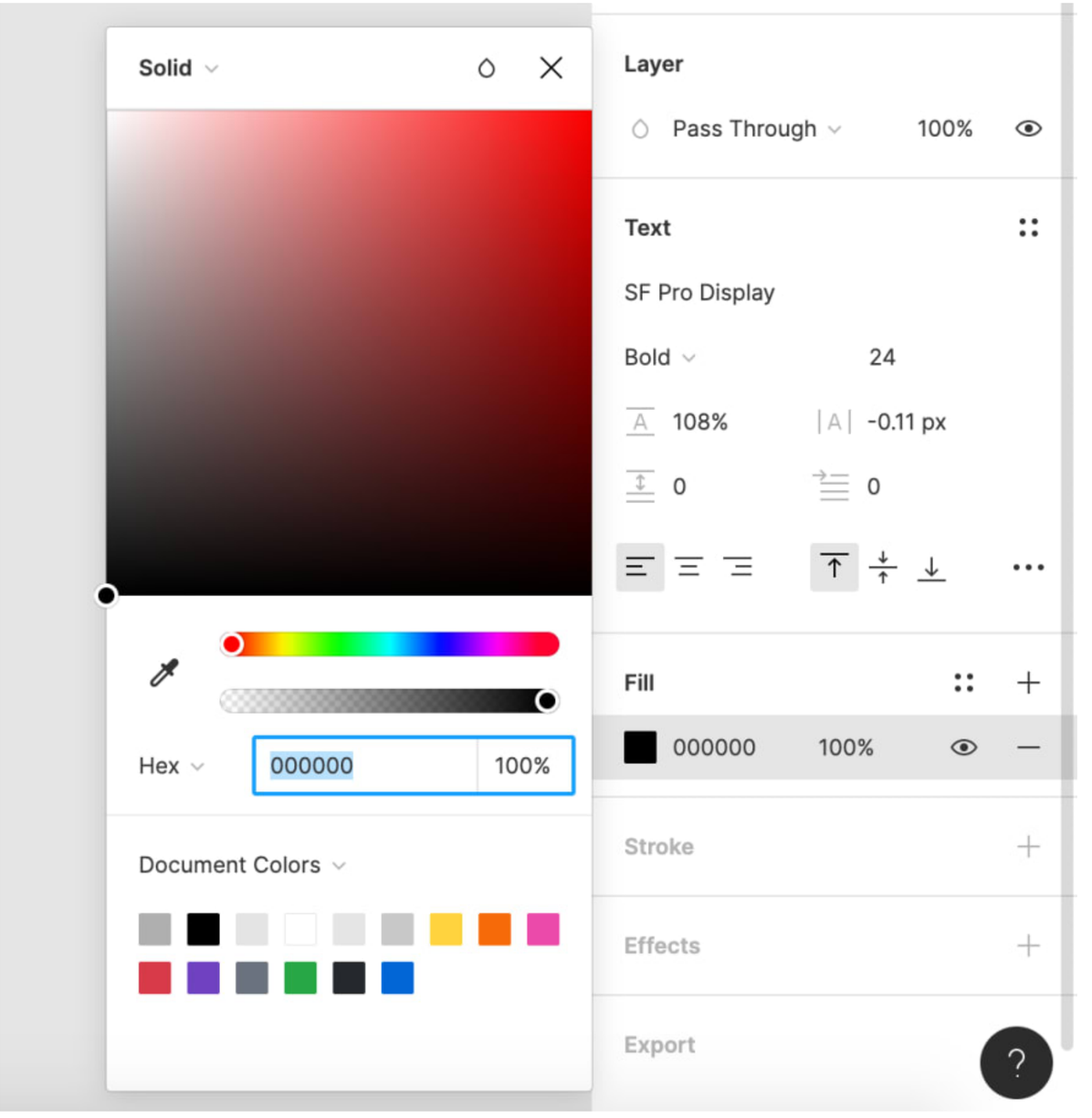
```
import * as React from "react";
import {StyleSheet, Frame, View, Text} from "react-figma";
import {colors} from "../../tokens/colors";
import {StyleguideColor} from "../../components/styleguide-color/StyleguideColor";

export const Styleguide10 = (props) => {
    return <Frame name="Styleguide 1-0" style={[styles.frame, props.style]}>
        <View>
            <View style={styles.colorsContainer}>
                <StyleguideColor name="gray"   color={colors.gray} style={styles.colorWrapper}/>
                <StyleguideColor name="blue"   color={colors.blue} style={styles.colorWrapper}/>
                <StyleguideColor name="green"  color={colors.green} style={styles.colorWrapper}/>
                <StyleguideColor name="purple" color={colors.purple} style={styles.colorWrapper}/>
                <StyleguideColor name="yellow" color={colors.yellow} style={styles.colorWrapper}/>
                <StyleguideColor name="orange" color={colors.orange} style={styles.colorWrapper}/>
                <StyleguideColor name="red"    color={colors.red} style={styles.colorWrapper}/>
                <StyleguideColor name="pink"   color={colors.pink} style={styles.colorWrapper}/>
                <StyleguideColor name="black"  color={colors.black} style={styles.colorWrapper}/>
                <StyleguideColor name="white"  color={colors.white} style={styles.colorWrapper}/>
            </View>
        </View>
    </Frame>
}
```

После того как добавим Figma плагин

Выбор из меню плагинов




Untitled
Drafts / Untitled
Share
35%
Layers
Assets
Styleguide
Color container
orange
Rectangle
Color container
yellow
Rectangle
Color container
purple
Rectangle
Color container
green
Rectangle
Color container
blue
Rectangle
Color container
gray
Rectangle
Rectangle
Colors
Styleguide 1-0
gray
blue
green
purple
yellow
orange
red
pink
black
white
Running Primer design-system (Developer VM)
Cancel
Design
Prototype
Inspect
Background
E5E5E5
100%
Export

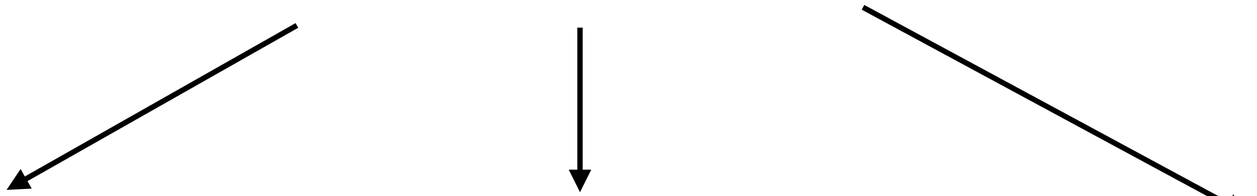

Solid
Hex
000000
100%
Document Colors
Layer
Pass Through
100%
Text
SF Pro Display
Bold
24
108%
-0.11 px
0
0
Fill
000000
100%
Stroke
Effects
Export

Gray

$gray	#6a737d
$gray-000	#fafbfc
$gray-100	#f6f8fa
$gray-200	#e1e4e8
$gray-300	#d1d5da
$gray-400	#959da5
$gray-500	#6a737d
$gray-600	#586069
$gray-700	#444d56
$gray-800	#2f363d
$gray-900	#24292e

```
export const colors = {
    // Gray
    gray: "#6a737d",
    gray000: "#6a737d",
    gray100: "#f6f8fa",
    gray200: "#e1e4e8",
    gray300: "#d1d5da",
    gray400: "#959da5",
    gray500: "#6a737d",
    gray600: "#586069",
    gray700: "#444d56",
    gray800: "#2f363d",
    gray900: "#24292e",

    ...
}
```

Стайл-гайд



Figma

react-figma

Web

CSS-in-JS

react-native-web

Mobile

react-native

Код становится **single source of truth** для дизайн-системы

Octicons 

octicons


Gist
GitHub

Создадим компонент **<Icon />** для иконки

</Icon>

iOS + Android

```
import * as React from "react";
import { SvgXml } from 'react-native-svg';

export const Icon = (props) => {
    const {src: source, height = 16, width, ratio = 1, ...otherProps} = props;

    return <SvgXml xml={source} height={height} width={width || Math.round(height * ratio)} {...otherProps} />;
};
```

Web

```
import * as React from "react";

export const Icon = (props) => {
    const {src, height = 16, width, ratio = 1, ...otherProps} = props;
    return <img src={src} style={{height, width: width || ratio * height}} {...otherProps} />
};
```

Figma

```
import * as React from "react";
import {Svg} from "react-figma";

export const Icon = (props) => {
    const {src: source, height = 16, width, ratio = 1, ...otherProps} = props;
    return <Svg source={source} height={height} width={width || Math.round(height * ratio)} {...otherProps} />
};
```

figma.createNodeFromSvg(svg: string)

Создадим компонент под каждую иконку

- icons
  - broadcast.svg
  - browser.svg
  - circuit-board.svg
  - code.svg
  - database.svg
  - dependent.svg
  - diff.svg
  - diff-added.svg
  - diff-ignored.svg
  - diff-modified.svg
  - diff-removed.svg
  - diff-renamed.svg
  - eye.svg
  - eye-closed.svg
  - file.svg
  - file-binary.svg
  - file-code.svg
  - file-directory.svg
  - file-media.svg
  - file-pdf.svg
  - file-submodule.svg
  - file-symlink-directory.svg
  - file-symlink-file.svg
  - file-zip.svg

icons/logo-github.svg

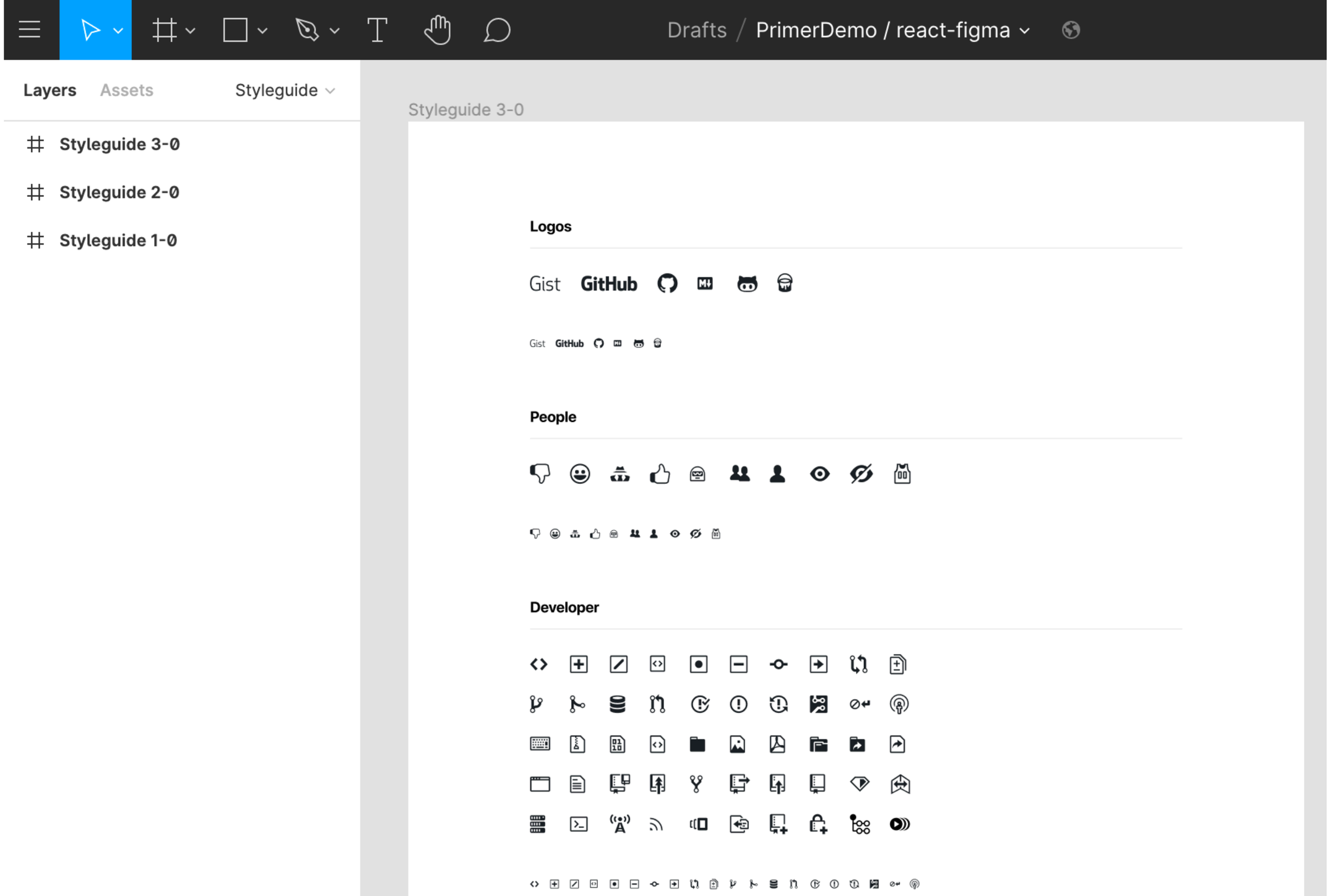
```
import * as React from "react";
import icon from "./icons/logo-github.svg"
import {Icon} from "../../wrappers/icon/Icon";🤔

export const GitHubLogo = (props) => {
    return <Icon ratio={45.0/16} src={icon} {...props} />
};
```

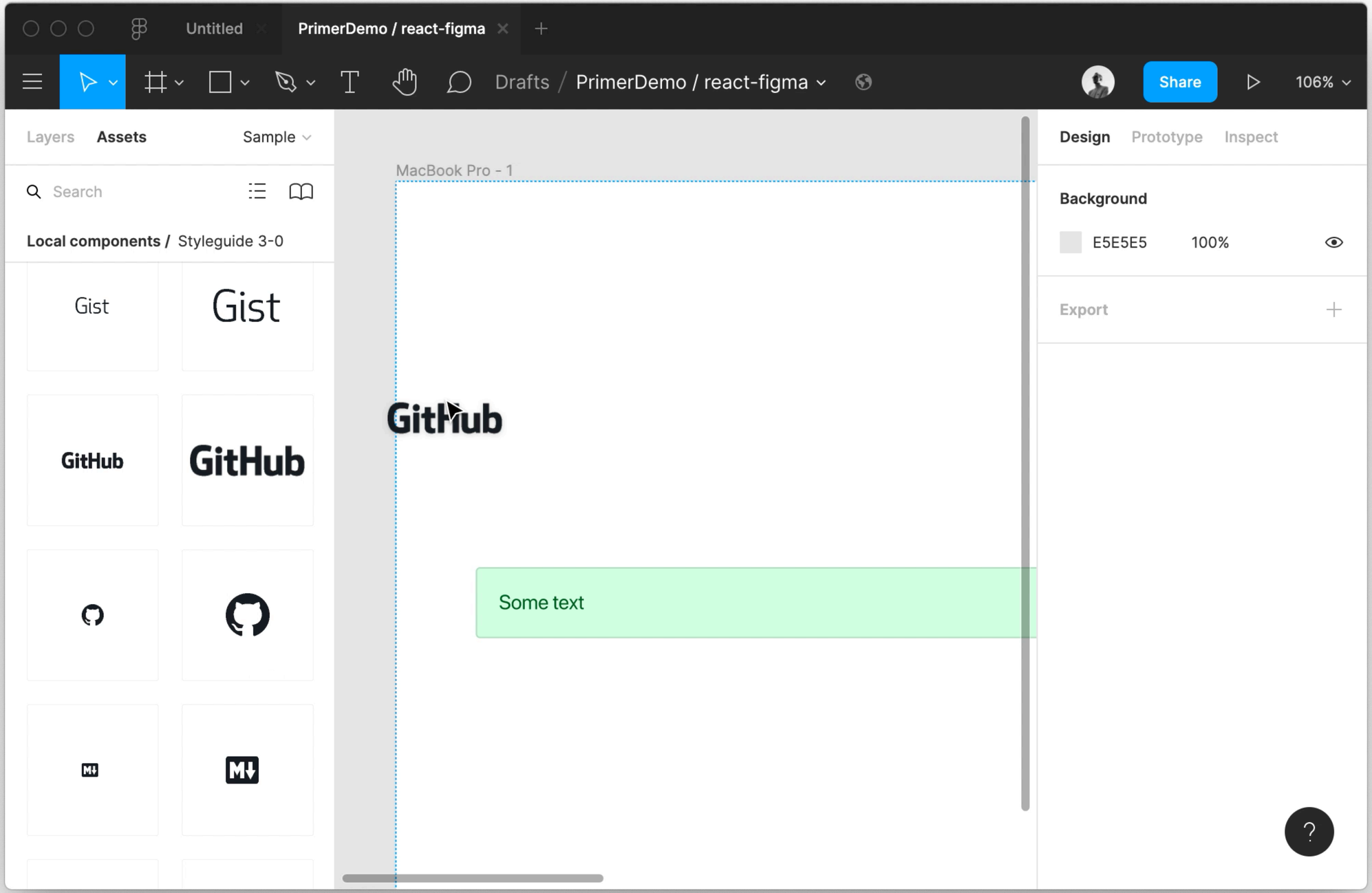
Покажем все иконки на фрейме

Icons

Frame

```
<View style={{marginTop: 69}}>
    <StyleguideSeparatorWrapper>
        <StyleguideLabel text="Logos" />
    </StyleguideSeparatorWrapper>
    <View style={styles.iconsLine}>
        <Component name="logo-gist-5" style={styles.iconComponent5}>
            <GistLogo height={spacer5} />
        </Component>
        <Component name="logo-github-5" style={styles.iconComponent5}>
            <GitHubLogo height={spacer5} />
        </Component>
        <Component name="logo-github-mark-5" style={styles.iconComponent5}>
            <GitHubMark height={spacer5} />
        </Component>
        <Component name="logo-markdown-5" style={styles.iconComponent5}>
            <Markdown height={spacer5} />
        </Component>
        <Component name="logo-octoface-5" style={styles.iconComponent5}>
            <Octoface height={spacer5} />
        </Component>
        <Component name="logo-paintcan-5" style={styles.iconComponent5}>
            <Paintcan height={spacer5} />
        </Component>
    </View>
    <View style={styles.iconsLine}>
        <Component name="logo-gist-3" style={styles.iconComponent3}>
            <GistLogo height={spacer3} />
        </Component>
        <Component name="logo-github-3" style={styles.iconComponent3}>
            <GitHubLogo height={spacer3} />
        </Component>
```


Drafts / PrimerDemo / react-figma
Layers
Assets
Styleguide
Styleguide 3-0
Styleguide 2-0
Styleguide 1-0
Styleguide 3-0
Logos
Gist
GitHub
People
Developer


Untitled
PrimerDemo / react-figma
Drafts / PrimerDemo / react-figma
Share
106%
Layers
Assets
Sample
Search
Local components / Styleguide 3-0
Gist
Gist
GitHub
GitHub
MacBook Pro - 1
GitHub
Some text
Design
Prototype
Inspect
Background
E5E5E5
100%
Export

Что-то более сложное?

Кнопка!

Button

Button


```
<button class="btn" type="button">Button button</button>
<a class="btn" href="#url" role="button">Link button</a>
```

You can find them in two sizes: the default `.btn` and the smaller `.btn-sm`.


```
<button class="btn" type="button">Button</button>
<button class="btn btn-sm" type="button">Small button</button>
```

Primary button

Primary buttons are green and are used to indicate the *primary* action on a page. When you need your buttons to stand out, use `.btn.btn-primary`. You can use it with both button sizes—just add `.btn-primary`.


```
<button class="btn btn-primary" type="button">Primary button</button>
<button class="btn btn-sm btn-primary" type="button">Small primary button</button>
```

Danger button

Danger buttons are red. They help reiterate that the intended action is important or potentially dangerous (e.g., deleting a repo or transferring ownership). Similar to the primary buttons, just add `.btn-danger`.

Danger button Small danger button

```
<button class="btn btn-danger" type="button">Danger button</button>
<button class="btn btn-sm btn-danger" type="button">Small danger button</button>
```

Удобнее всего начать писать код на **react-native**

API **react-native** поддерживается на остальных платформах

Но не наоборот!

Базовые стили для кнопки

```
export const commonButtonStyle = {
    container: {
        justifyContent: "center",
        alignItems: "center",
        height: 32,
        borderRadius: 3,
        borderWidth: 1,
        borderColor: "rgba(27,31,35,0.2)",
    },
    text: {
        fontFamily: "SF Pro Text",
        fontWeight: "bold",
        fontSize: typeScale.size5,
        textAlign: "center",
        zIndex: 1,
        marginLeft: 12 + borderSize,
        marginRight: 12 + borderSize
    }
};
```

```
export const commonButtonSmallStyle = StyleSheet.create({
    container: {
        height: 28
    },
    text: {
        fontSize: typeScale.size6,
        lineHeight: 20,
        marginLeft: 10 + borderSize,
        marginRight: 10 + borderSize
    }
});
```

Default Button

Default Button

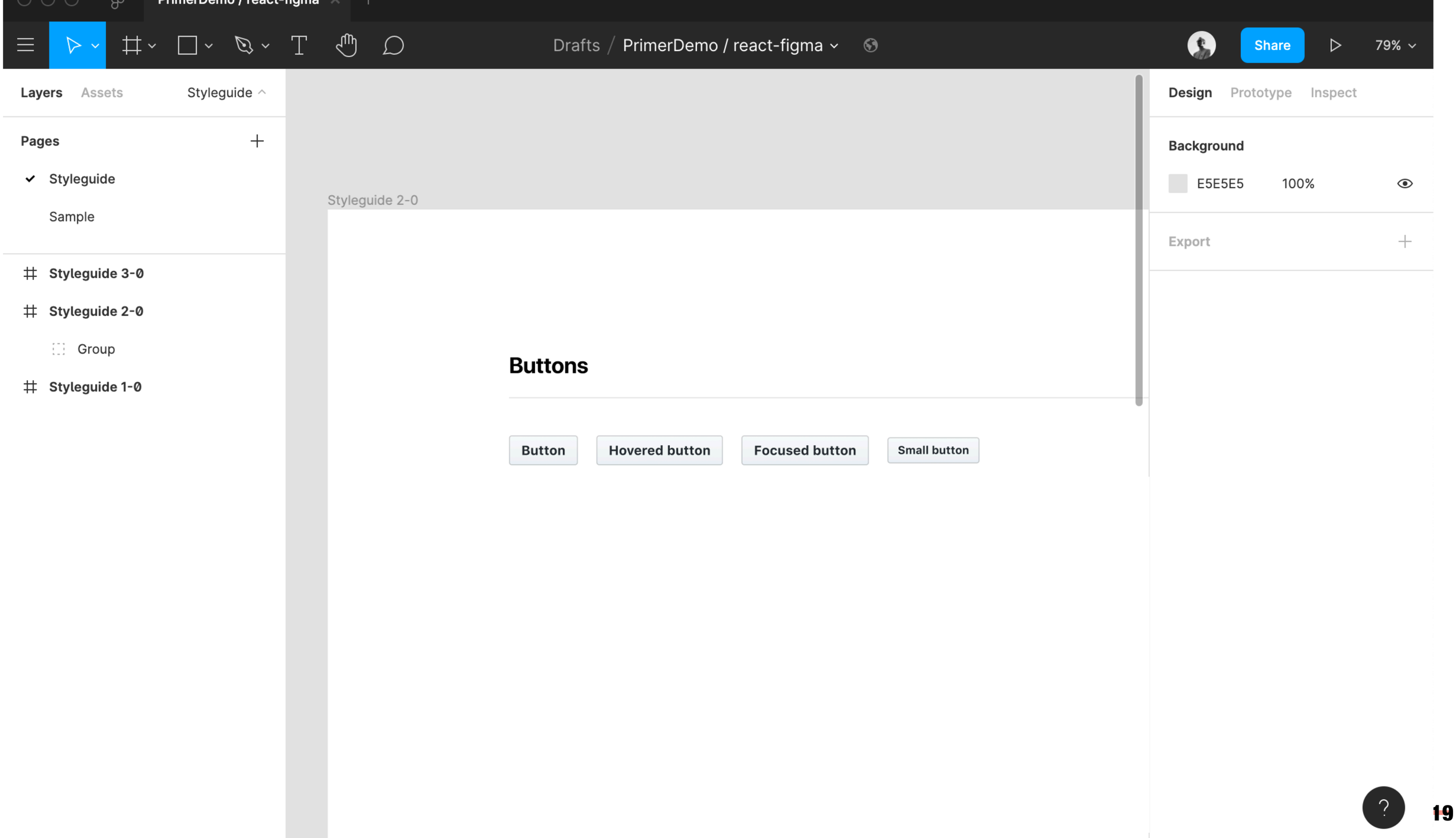
```
const styles = StyleSheet.create({
    container: {
        backgroundColor: colors.gray100,
        backgroundImage: `linear-gradient(-180deg, ${colors.gray100} 0%, ${colors.gray300} 90%)`,
    },
    text: {
       color: colors.gray900,
    }
});
```

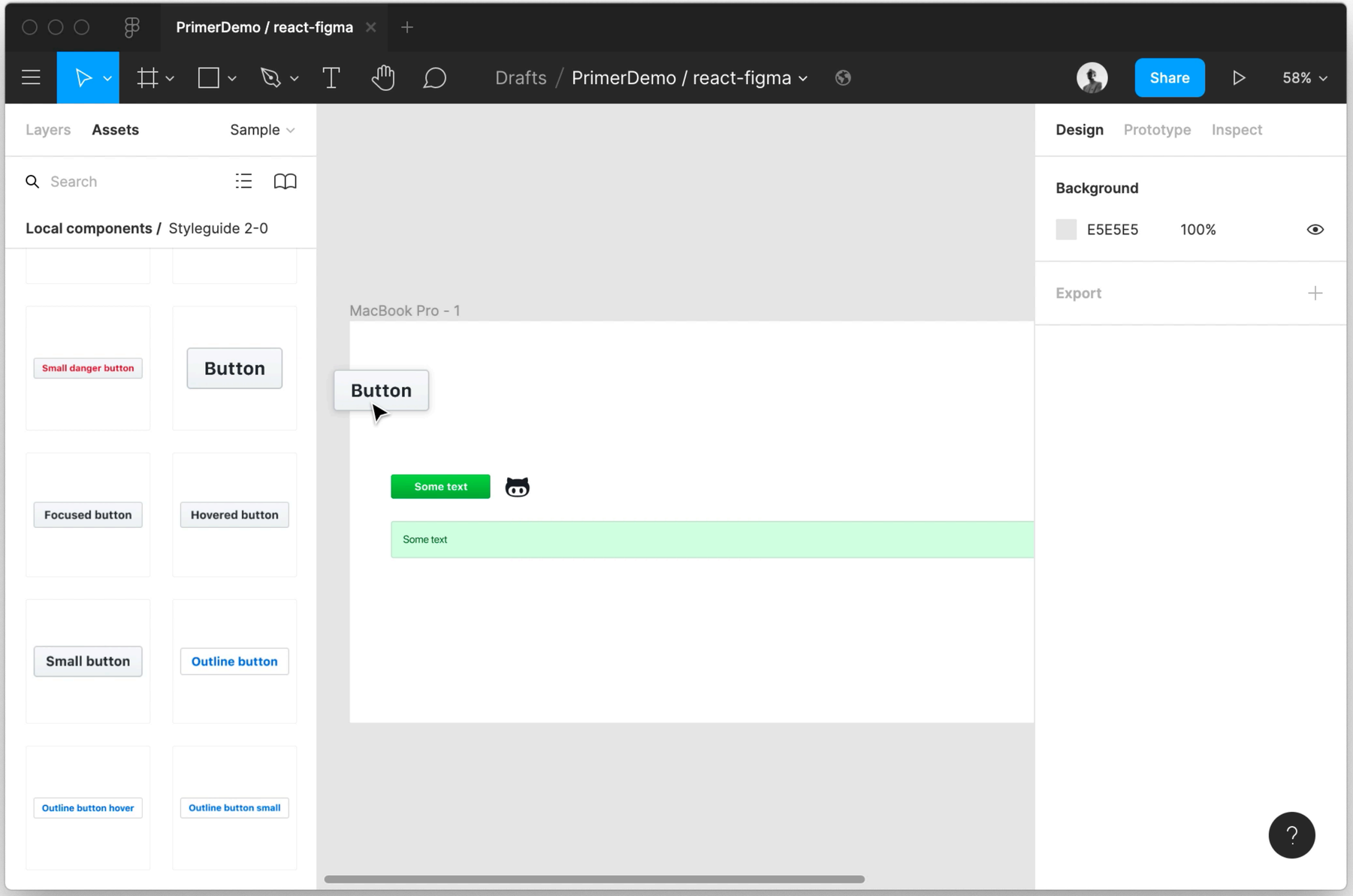
Пользуемся значениями из стайл-гайда

```
export const DefaultButton = (props) => {
    const { style, children, isHover, isFocus, isSmall } = props;
    return (
        <View
            style={[
                commonButtonStyle.container,
                styles.container,
                isHover && hoverStyles.container,
                isFocus && focusStyles.container,
                isSmall && commonButtonSmallStyle.container,
                style,
            ]}
        >
            <Text
                style={[
                    commonButtonStyle.text,
                    styles.text,
                    isSmall && commonButtonSmallStyle.text,
                ]}
            >
                {children}
            </Text>
        </View>
    );
};
```

Покажем разные состояния кнопки на фрейме

```
<View style={styles.buttonsLine}>
    <Component name="button-default">
        <DefaultButton>Button</DefaultButton>
    </Component>
    <Component name="button-default-hover" style={styles.buttonMargin}>
        <DefaultButton isHover >Hovered button</DefaultButton>
    </Component>
    <Component name="button-default-focus" style={styles.buttonMargin}>
        <DefaultButton isFocus >Focused button</DefaultButton>
    </Component>
    <Component name="button-default-small" style={styles.buttonMargin}>
        <DefaultButton isSmall >Small button</DefaultButton>
    </Component>
</View>
```


PrimerDemo / react-figma
Drafts / PrimerDemo / react-figma
Share
79%
Layers
Assets
Styleguide
Pages
Styleguide
Sample
Styleguide 3-0
Styleguide 2-0
Group
Styleguide 1-0
Styleguide 2-0
Buttons
Button
Hovered button
Focused button
Small button
Design
Prototype
Inspect
Background
E5E5E5
100%
Export


PrimerDemo / react-figma
Drafts / PrimerDemo / react-figma
Share
58%
Layers
Assets
Sample
Search
Local components / Styleguide 2-0
Small danger button
Button
Focused button
Hovered button
Small button
Outline button
Outline button hover
Outline button small
MacBook Pro - 1
Button
Some text
Some text
Design
Prototype
Inspect
Background
E5E5E5
100%
Export

Можем использовать ту же самую кнопку в **вебе**!

И в **мобильных приложениях**

Но:

- Нужна обработка нажатия - она **разная** на разных платформах
- Нужно обеспечить доступность (**a11y**)

Для этого сделаем компонент-обертку!

Веб

```
const style = {
    background: "none",
    border: "none",
    padding: 0
};

export default function Button(props: ButtonProps) {
    const {children, onClick} = props;
    const [isHover, hoverHandlers] = useHover();
    const [isFocus, focusHandlers] = useFocus();
    return <button style={style} onClick={onClick} {...hoverHandlers} {...focusHandlers}>
        {children({isHover, isFocus})}
    </button>
}
```

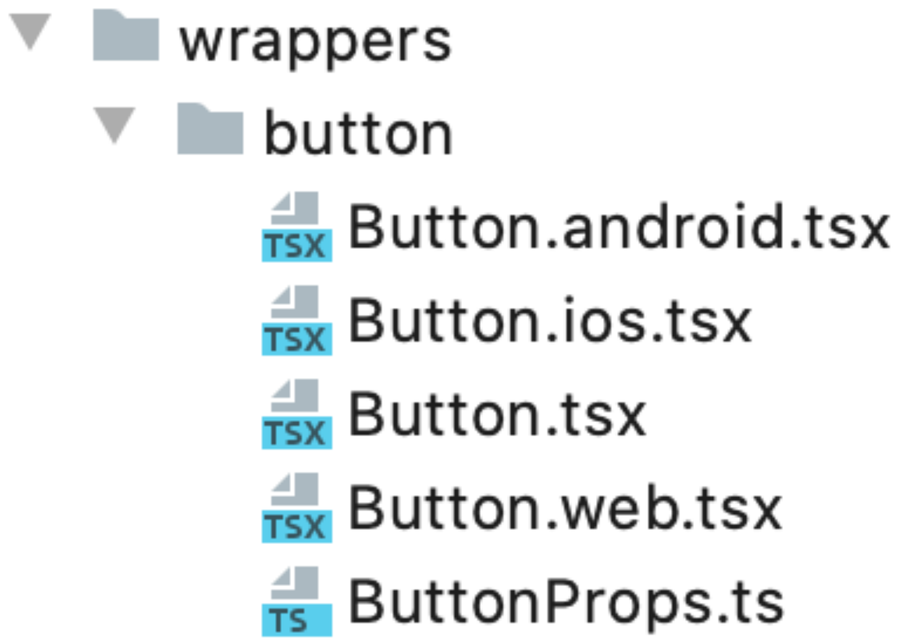
iOS

```
export default function Button(props: ButtonProps) {
    const {children, onClick} = props;
    const [isFocus, focusHandlers] = useFocus();
    return <TouchableHighlight onPress={onClick} {...focusHandlers}>
        {children({isFocus})}
    </TouchableHighlight>;
}
```

Android

```
export default function Button(props: ButtonProps) {
    const {children, onClick} = props;
    const [isFocus, focusHandlers] = useFocus();
    return <TouchableNativeFeedback onPress={onClick} {...focusHandlers}>
        {children({isFocus})}
    </TouchableNativeFeedback>;
}
```

Хотим оборачивать удобно


wrappers
button
Button.android.tsx
Button.ios.tsx
Button.tsx
Button.web.tsx
ButtonProps.ts

iOS и Android

Это работает с коробки!

Для всего остального -

Вебпак-алиасинг!

Веб

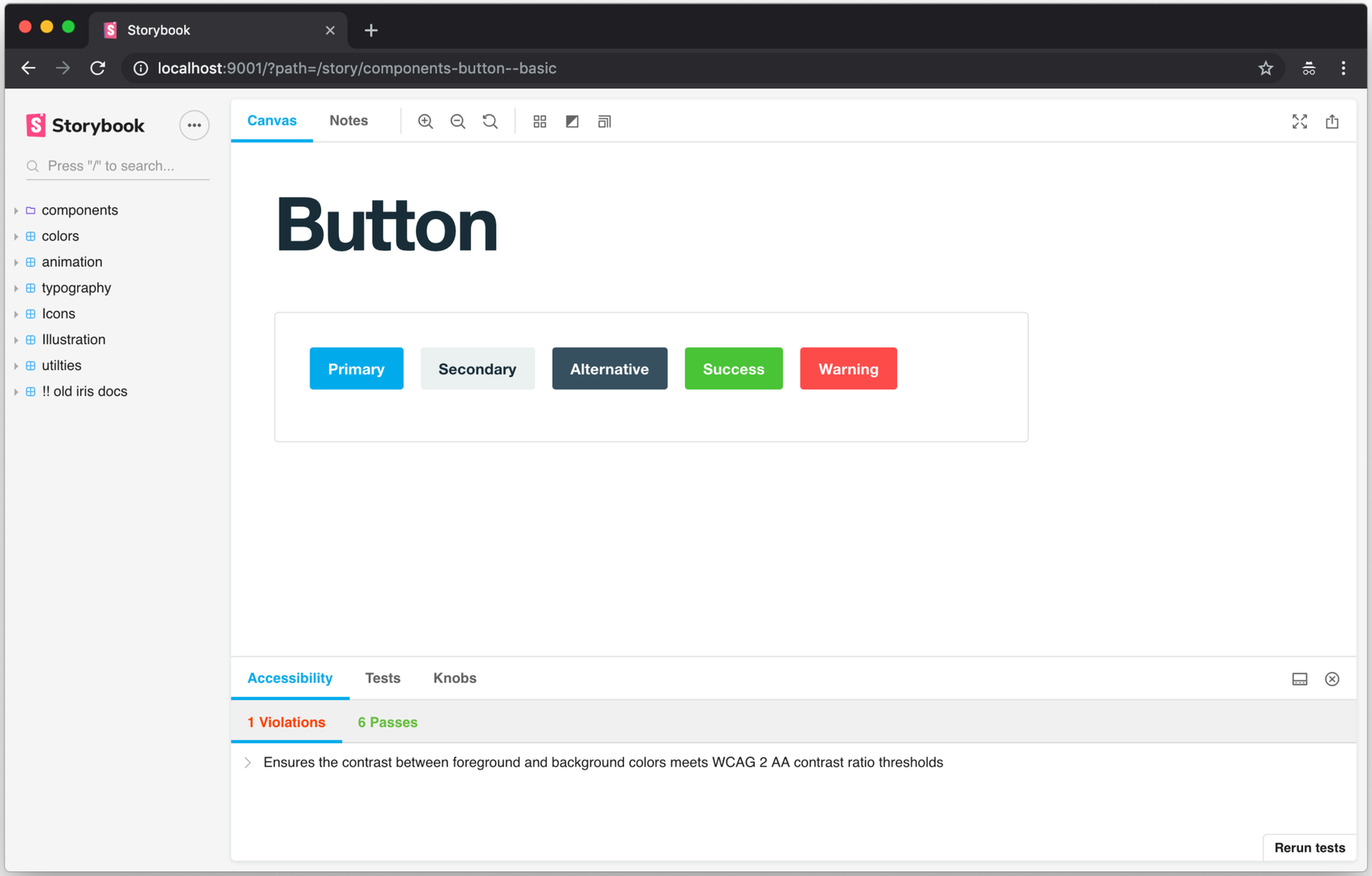
```
module.exports = configure({
    resolve: {
        extensions: ['.web.tsx', ‘.web.ts','.tsx', '.ts', '.jsx', '.js'],
        alias: {
            'react-native$': ‘react-native-web’
        }
    }
});
```

Figma

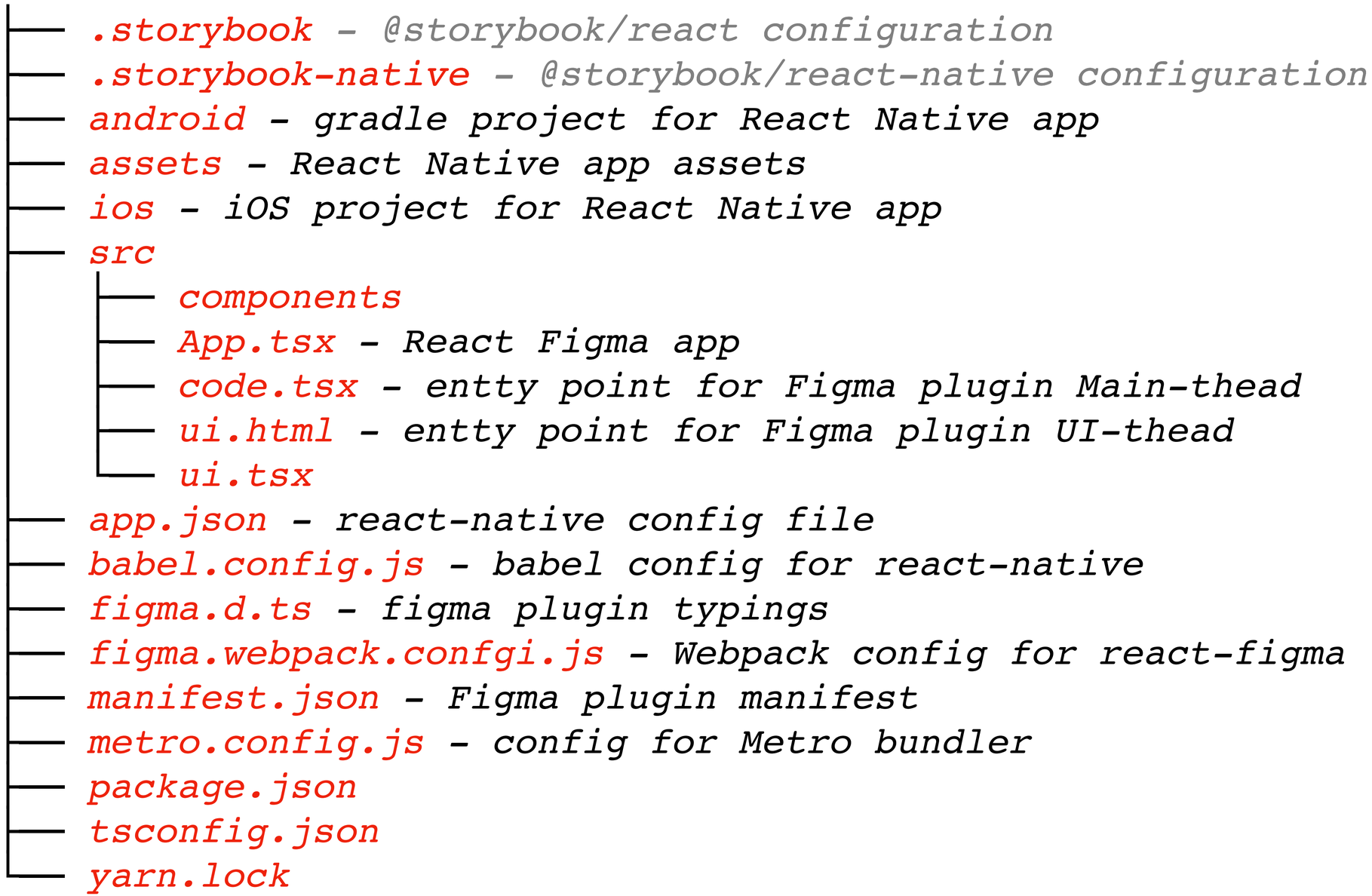
```
module.exports = configure({
    resolve: {
        extensions: ['.figma.tsx', '.figma.ts','.tsx', '.ts', '.jsx', '.js'],
        alias: {
            'react-native$': 'react-figma'
        }
    }
});
```

```tsx
import Button from "../../wrappers/button/Button";

export default (props: { onClick?: () => void }) => (
    <Button onClick={props.onClick}>
        {({ isHover, isFocus }) => (
            <DefaultButton {...props} isFocus={isFocus} isHover={isHover} />
        )}
    </Button>
);
```

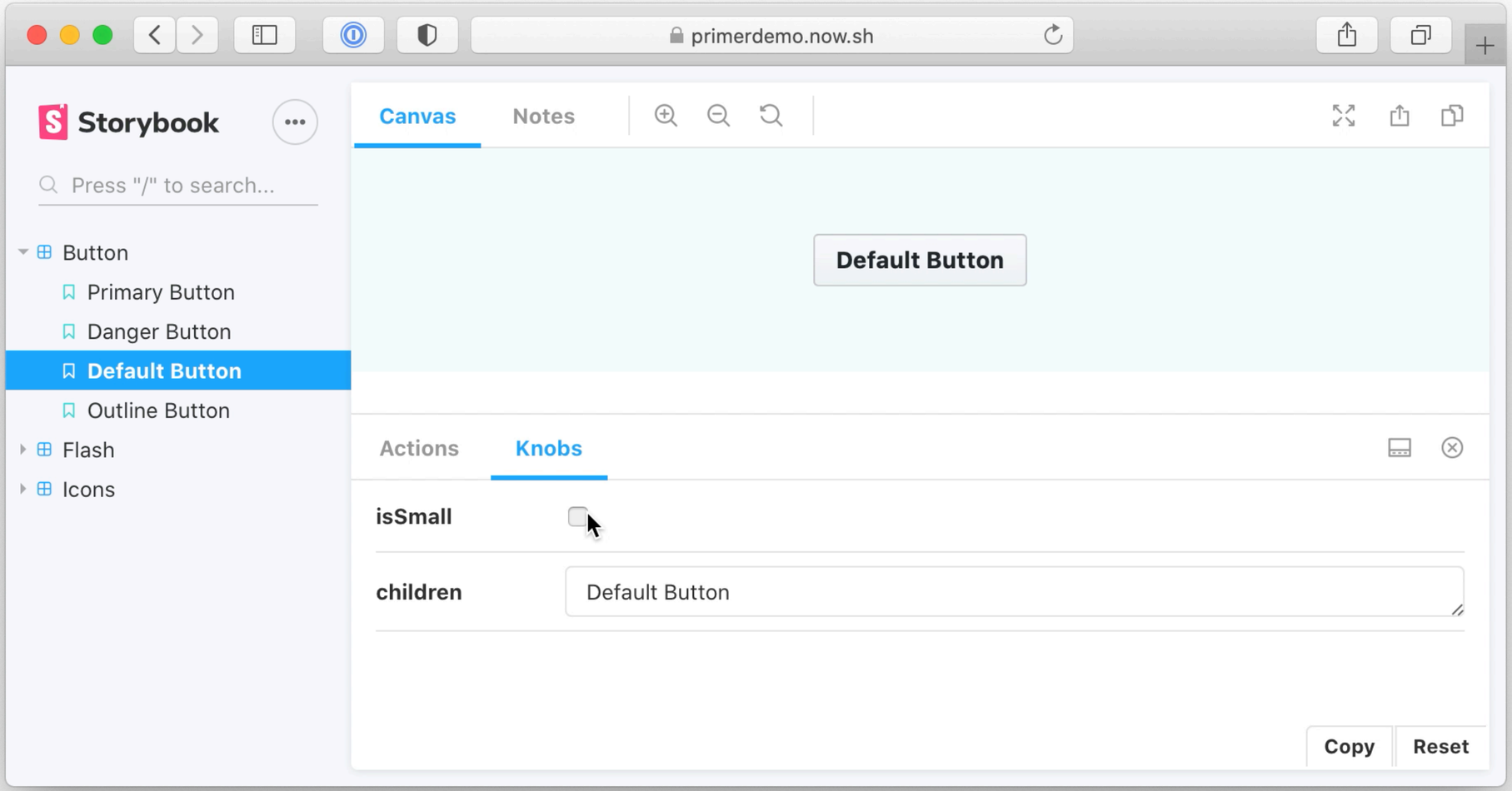
Каталог компонентов

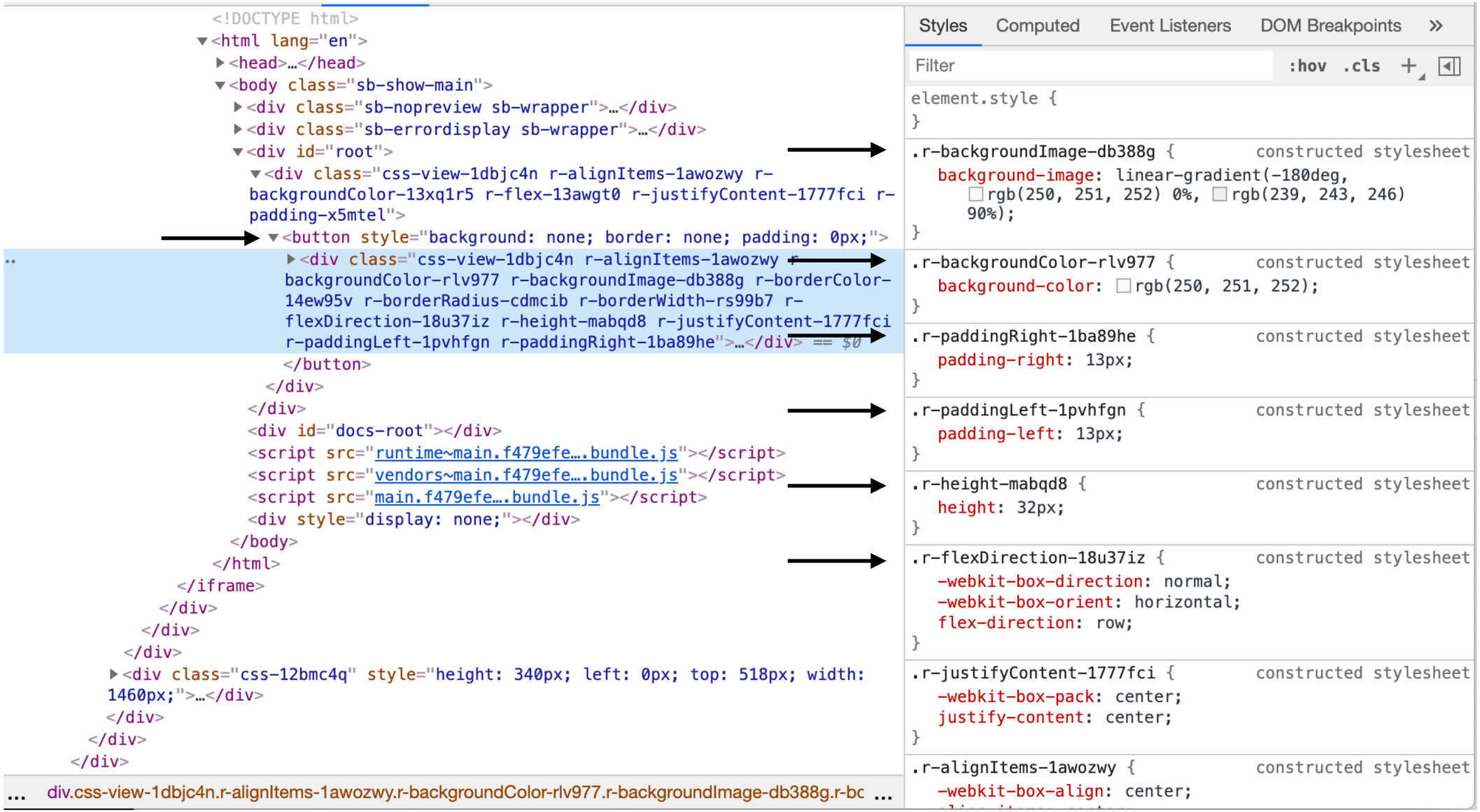

Storybook
localhost:9001/?path=/story/components-button--basic
Storybook
Press "/" to search...
components
colors
animation
typography
Icons
Illustration
utilties
!! old iris docs
Canvas
Notes
Button
Primary
Secondary
Alternative
Success
Warning
Accessibility
Tests
Knobs
1 Violations
6 Passes
Ensures the contrast between foreground and background colors meets WCAG 2 AA contrast ratio thresholds
Rerun tests

- @storybook/react
- @storybook/react-native
- @storybook/addons


.storybook - @storybook/react configuration
.storybook-native - @storybook/react-native configuration
android - gradle project for React Native app
assets - React Native app assets
ios - iOS project for React Native app
src
components
App.tsx - React Figma app
code.tsx - entty point for Figma plugin Main-thead
ui.html - entty point for Figma plugin UI-thead
ui.tsx
app.json - react-native config file
babel.config.js - babel config for react-native
figma.d.ts - figma plugin typings
figma.webpack.confgi.js - Webpack config for react-figma
manifest.json - Figma plugin manifest
metro.config.js - config for Metro bundler
package.json
tsconfig.json
yarn.lock

```
import * as React from 'react';
import {storiesOf} from "@storybook/react-native";
import {boolean, withKnobs, text} from "@storybook/addon-knobs";
import {defaultBackground} from "../storybook-decorators/DefaultBackground";
import { action } from '@storybook/addon-actions';
import DefaultButton from "../default-button/DefaultButton";


storiesOf('Button', module)
    .addDecorator(withKnobs)
    .addDecorator(defaultBackground)
    .add('Default Button', () =>
        <DefaultButton onClick={action('click')}
                       isSmall={boolean("isSmall", false)}>
            {text("children", "Danger Button")}
        </DefaultButton>
    );
```


primerdemo.now.sh
Storybook
Press "/" to search...
Button
Primary Button
Danger Button
Default Button
Outline Button
Flash
Icons
Canvas
Notes
Default Button
Actions
Knobs
isSmall
children
Default Button
Copy
Reset


<!DOCTYPE html>
<html lang="en">
<head>…</head>
<body class="sb-show-main">
…
…

<button style="background: none; border: none; padding: 0px;">
<div class="css-view-1dbjc4n r-alignItems-1awozwy r-backgroundColor-rlv977 r-backgroundImage-db388g r-borderColor-14ew95v r-borderRadius-cdmcib r-borderWidth-rs99b7 r-flexDirection-18u37iz r-height-mabqd8 r-justifyContent-1777fci r-paddingLeft-1pvhfgn r-paddingRight-1ba89he">… == $0
</button>

<script src="runtime~main.f479efe….bundle.js"></script>
<script src="vendors~main.f479efe….bundle.js"></script>
<script src="main.f479efe….bundle.js"></script>

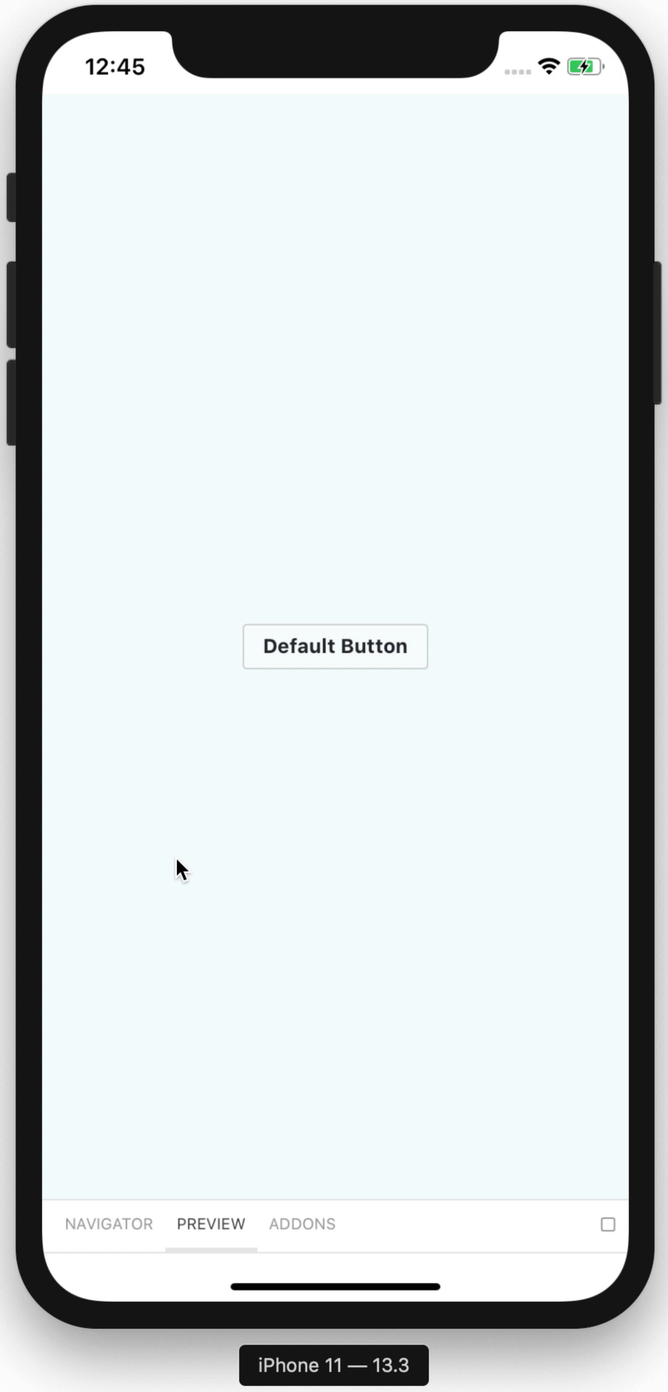
</body>
</html>
</iframe>

…

... div.css-view-1dbjc4n.r-alignItems-1awozwy.r-backgroundColor-rlv977.r-backgroundImage-db388g.r-bc ...
Styles
Computed
Event Listeners
DOM Breakpoints
Filter
:hov .cls
element.style {
}
.r-backgroundImage-db388g { constructed stylesheet
background-image: linear-gradient(-180deg, rgb(250, 251, 252) 0%, rgb(239, 243, 246) 90%);
}
.r-backgroundColor-rlv977 { constructed stylesheet
background-color: rgb(250, 251, 252);
}
.r-paddingRight-1ba89he { constructed stylesheet
padding-right: 13px;
}
.r-paddingLeft-1pvhfgn { constructed stylesheet
padding-left: 13px;
}
.r-height-mabqd8 { constructed stylesheet
height: 32px;
}
.r-flexDirection-18u37iz { constructed stylesheet
-webkit-box-direction: normal;
-webkit-box-orient: horizontal;
flex-direction: row;
}
.r-justifyContent-1777fci { constructed stylesheet
-webkit-box-pack: center;
justify-content: center;
}
.r-alignItems-1awozwy { constructed stylesheet
-webkit-box-align: center;

Тот же самый код будет работать и на **react-native**

iOS-симулятор



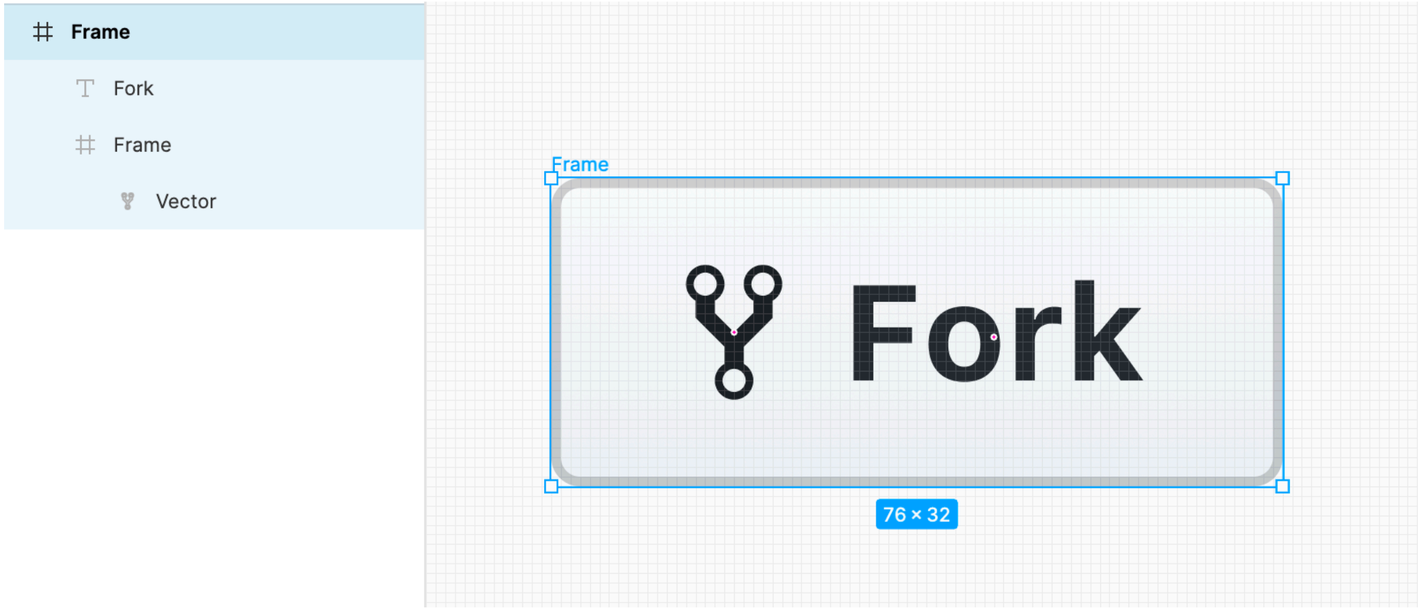
Композиция элементов


Fork

```
export const DefaultButton = (props: IDefaultButton) => {
    const { style, children, isHover, isFocus, isSmall, icon } = props;
    return (
        <View
            style={[
                styles.container,
                isHover && hoverStyles.container,
                isFocus && focusStyles.container,
                isSmall && commonButtonSmallStyle.container,
                style,
            ]}
        >
            {icon}
            <Text style={[styles.text, isSmall && commonButtonSmallStyle.text]}>
                {children}
            </Text>
        </View>
    );
};
```

```
import {DefaultButton} from "./components/default-button/DefaultButton";
import {RepoForked} from "./components/icons/RepoForked";


const App = () => {
    return (
        <Page>
            <DefaultButton icon={<RepoForked style={{marginRight: 6}} />}>
                Fork
            </DefaultButton>
        </Page>
  );
};
```

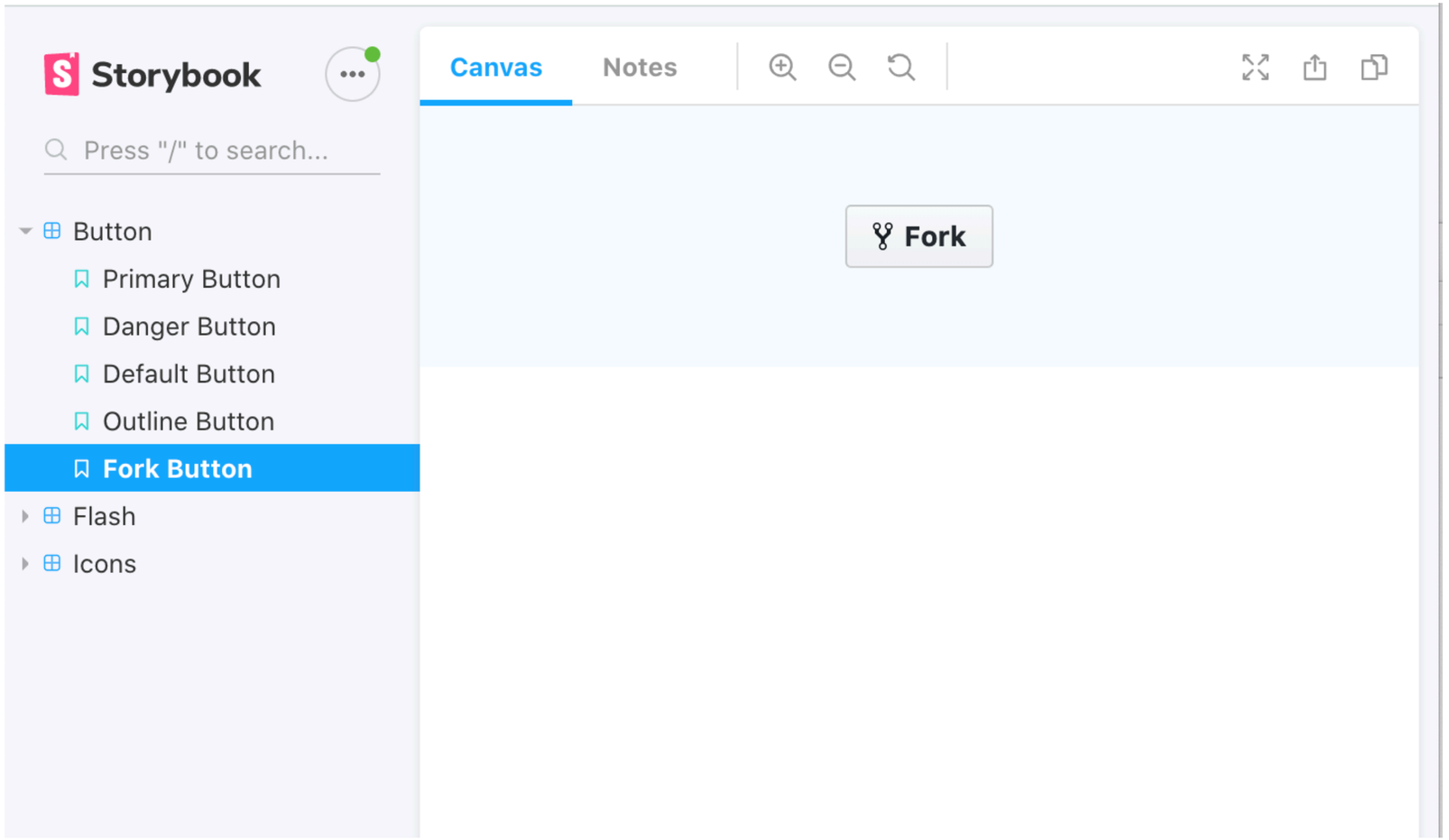
Отобразим в Figma:



Отобразим в Storybook

```
.add('Fork Button', () => <DefaultButton icon={<RepoForked style={{marginRight: 6}} />}>
    Fork
</DefaultButton>);
```

Отобразим в Storybook



Таким образом можно собирать сколь угодно сложные элементы

Полный код примера есть на GitHub:

github.com/react-figma/PrimerDemo

Сделали пример мульти-платформенной дизайн системы

- Стайл-гайд в коде
- UI-кит в коде
- Отображение в Figma
- Каталог компонентов в Storybook

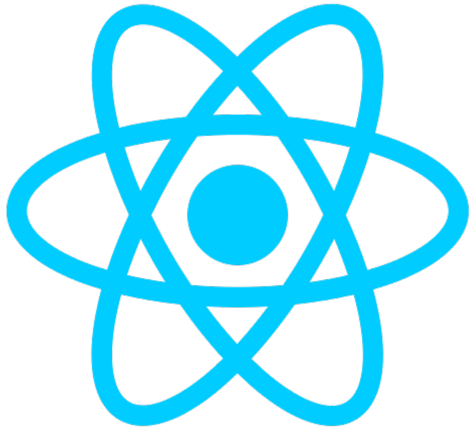
Процесс 

process

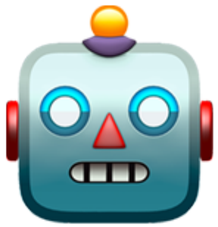
Интуитивный процесс

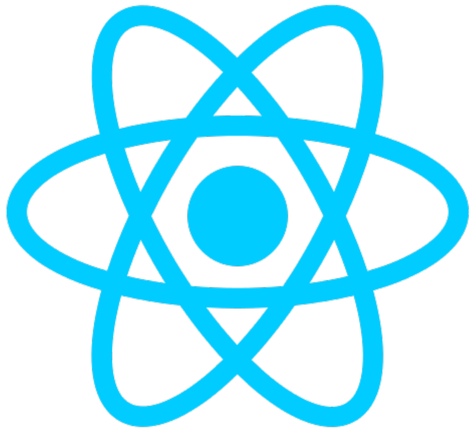
intuitive

process



Хочется **автоматизировать** процесс



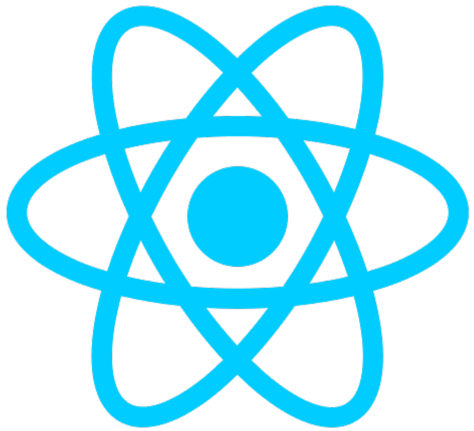


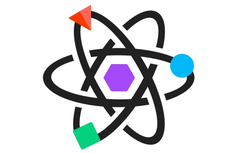
Но у нас другой подход

approach

A React renderer for Figma. Use React components as a source for your designs.

- 🍬 Compatible with react-native, react-sketchapp, react-primitives API.
- 🦄 Flexible layouts support with Yoga Layout.
- ♻️ Hydration and HMR support.
- ⚙️ Built on Figma Plugin API.
- 🚫 **Is not a code generator**.



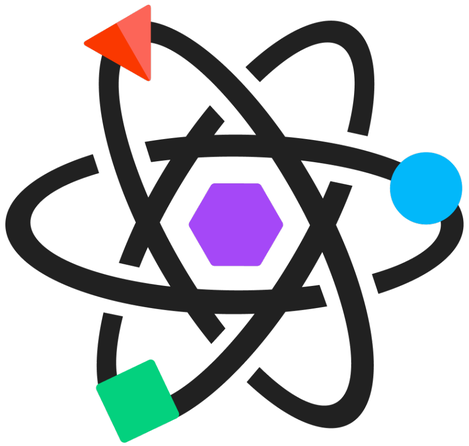


Подход кажется **контринтуитивным**

Но лишь потому, что разработчикам **хочется чуда**

- Генераторы кода пока слишком **не совершенны**
- Лучший код можно написать только **вручную**
- Можно положить дизайн-систему в **VCS**
- Консистентность

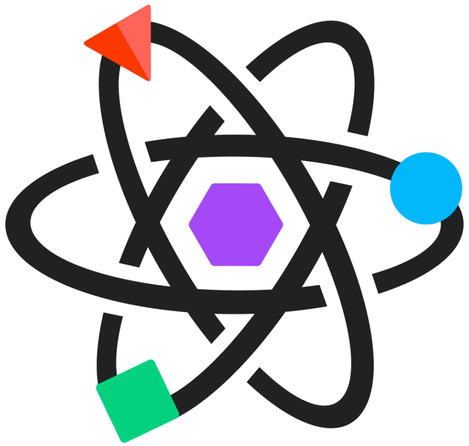
Но подход **усложняется**



А если хотим внести изменения в **дизайн-систему**?

Рисует новый вариант

Вносит изменения



Используют

исправленную

дизайн-систему

При этом возможно совмещение роли **дизайнера** и **разработчика**

Код дизайн-системы понятный

Дизайнеры могут делать в нее MR

Дизайнер в продукте должен быть **технологистом**

В **большой** компании

Дизайн-система 🎨

Продукт 1 Продукт 2 Продукт 3 Продукт 4 …

Свои преимущества:

- Проще управлять процессом внесения изменений в дизайн-систему
- Безопасность: код находится у вас
- Проще запускать новые продукты

Рекомендации

recommendation

Для стайл-гайда лучше использовать

System UI Theme Specification

system-ui.com/theme

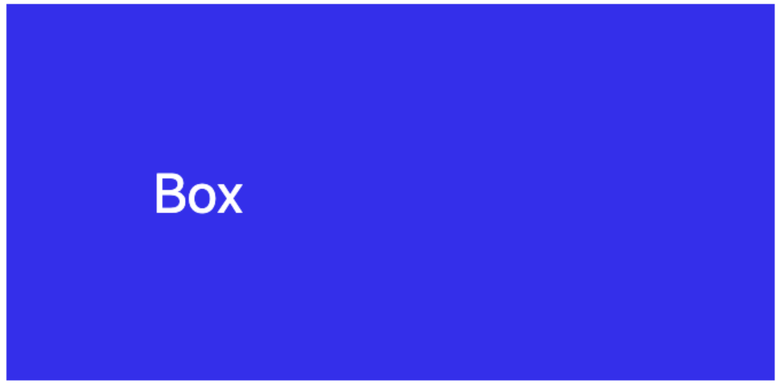
Универсальный формат конфига стайл-гайда

Определяем тему

```
const theme = {
    breakpoints: ['40em', '52em', '64em'],
    fontSizes: [
        12, 14, 16, 20, 24, 32, 48, 64
    ],
    colors: {
        blue: '#07c',
        lightgray: '#f6f6ff'
    },
    space: [
        0, 4, 8, 16, 32, 64, 128, 256
    ],
    fonts: {
        body: 'system-ui, sans-serif',
        heading: 'inherit',
        monospace: 'Menlo, monospace',
    },
    fontWeights: {
        body: 400,
        heading: 700,
        bold: 700,
    },
    buttons: {
        primary: {
            color: 'white',
            bg: 'primary',
        }
    }
}
```

Пользуемся значениями темы

```
<Box
     p={5}
     fontSize={4}
     width={[ 1, 1, 1/2 ]}
     color='white'
     bg='primary'>
     Box
</Box>
```



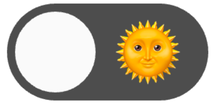
Theme Specification совместим со множеством библиотек

Who's Using this Specification

The following design systems and libraries currently adhere to the specification. If you'd like to add your project to this list, please open a PR.

- GitHub's Primer Components
- Artsy's Palette
- Sprout Social's Seeds
- Modulz Radix
- Priceline Design System
- Styled System
- Theme UI
- Rebass
- Styled System HTML
- xstyled

+ Темизация из коробки



- react-primitives-box — универсальная реализация

React Primitives Box

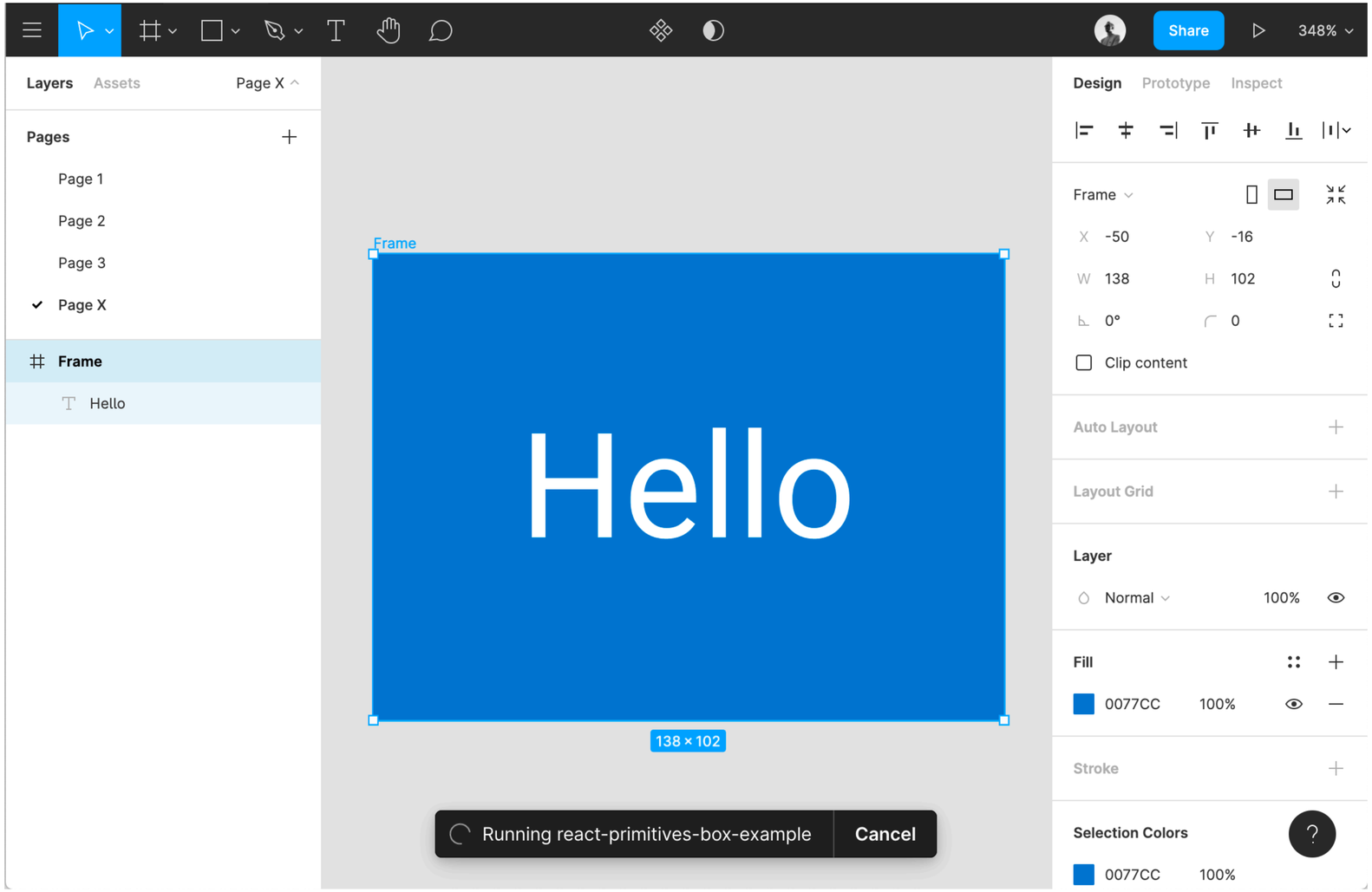
📦 Ergonomic, responsive React layout and grid system built on Styled System, Theme Specification, React Primitives and styled-components.

ci/circleci success | npm v0.0.1 | license MIT | system-ui theme

- Primitive styled components for all your layout needs
- Customize styles inline with the `sx` prop
- Ergonomic responsive array-based values
- Support for component variants
- Styled System props
- Themeable and compatible with the Theme Specification
- Built with Styled System
- Works with Theme UI
- Built with Styled Components

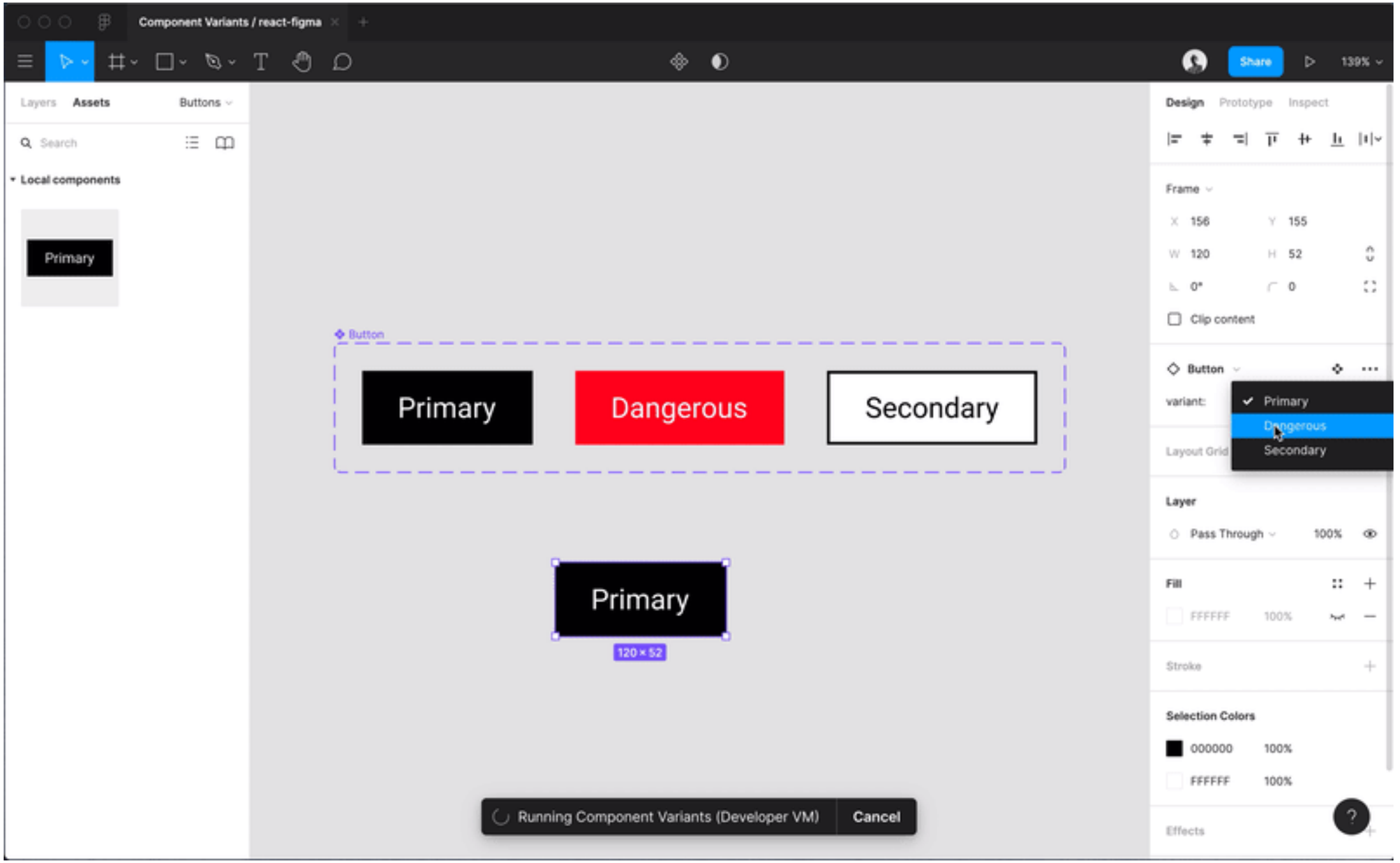
```
const theme = {
  colors: {
    text: "#000",
    background: "#fff",
    primary: "#07c",
  },
  space: [0, 4, 8, 16, 32, 64, 128, 256],
  fonts: {
    roboto: "Roboto",
  },
  fontSizes: [12, 16, 18, 20, 24, 32, 40, 64, 96],
  text: {
    heading1: {
      fontFamily: "roboto",
      fontSize: 5,
      color: "background",
    },
  },
};
```

```
<ThemeProvider theme={theme}>
  <Box
    sx={{
      p: 4,
      bg: "primary",
    }}
  >
    <Text variant={"heading1"}>Hello</Text>
  </Box>
</ThemeProvider>
```


Share
348%
Layers
Assets
Page X
Pages
Page 1
Page 2
Page 3
Page X
Frame
Hello
Frame
Hello
138 × 102
Running react-primitives-box-example
Cancel
Design
Prototype
Inspect
Frame
X -50
Y -16
W 138
H 102
0°
0
Clip content
Auto Layout
Layout Grid
Layer
Normal
100%
Fill
0077CC
100%
Stroke
Selection Colors
0077CC
100%

Component Variants

Разные состояния можно объединять во **варианты**


Component Variants / react-figma
Share
139%
Layers
Assets
Buttons
Search
Local components
Primary
Button
Primary
Dangerous
Secondary
Primary
120 × 52
Running Component Variants (Developer VM)
Cancel
Design
Prototype
Inspect
Frame
X 156
Y 155
W 120
H 52
0°
0
Clip content
Button
variant:
Primary
Dangerous
Secondary
Layout Grid
Layer
Pass Through
100%
Fill
FFFFFF
100%
Stroke
Selection Colors
000000
100%
FFFFFF
100%
Effects

MDX

MDX

MDX

readme.md

MDX: Markdown for the component era 🚀

MDX is an authorable format that lets you seamlessly use JSX in your markdown documents. You can import components, like interactive charts or notifications, and export metadata. This makes writing long-form content with components a blast.

Example

See MDX in action:

```
import { Chart } from '../components/chart'

# Here's a chart

The chart is rendered inside our MDX document.

<Chart />
```

```
# Hello, world!

MDX is an authorable format that lets you seamlessly write JSX in your Markdown documents.

It allows for displaying something unique from your design system. E.g., colors:

<Row>
    <Color value="#6A737D" />
    <Color value="#0366D6" />
    <Color value="#28A745" />
    <Color value="#6F42C1" />
    <Color value="#FFD33D" />
    <Color value="#F66A0A" />
    <Color value="#D73A49" />
</Row>
```

В **Figma:**

Frame

Hello, world!

MDX is an authorable format that lets you seamlessly write JSX in your Markdown documents.
It allows for displaying something unique from your design system. E.g., colors:

Дальнейшие планы

Набор универсальных компонентов

Для стандартных вещей

Button

SVG

...

components

Адаптер react-figma для Storybook

```
storiesOf('Button', module)
    .addDecorator(withKnobs)
    .addDecorator(defaultBackground)
    .add('Default Button', () =>
        <DefaultButton onClick={action('click')}
                       isSmall={boolean("isSmall", false)}>
            {text("children", "Danger Button")}
        </DefaultButton>
    );
```

Добавить фрейм

```
storiesOf('Button', module)
    .addDecorator(withKnobs)
    .addDecorator(defaultBackground)
    .add('Default Button', () =>
        <DefaultButton onClick={action('click')}
                       isSmall={boolean("isSmall", false)}>
            {text("children", "Danger Button")}
        </DefaultButton>
    );
```

Добавить в UI плагина контролы

Респонсив

Все тянется за счет Yoga, но

В react-native есть **Dimensions API**

Dimensions

> `useWindowDimensions` is the preferred API for React components. Unlike `Dimensions`, it updates as the window's dimensions update. This works nicely with the React paradigm.

```
import { Dimensions } from 'react-native';
```

You can get the application window's width and height using the following code:

```
const windowWidth = Dimensions.get('window').width;
const windowHeight = Dimensions.get('window').height;
```

> Although dimensions are available immediately, they may change (e.g due to device rotation, foldable devices etc) so any rendering logic or styles that depend on these constants should try to call this function on every render, rather than caching the value (for example, using inline styles rather than setting a value in a `StyleSheet`).

If you are targeting foldable devices or devices which can change the screen size or app window size, you can use the event listener available in the Dimensions module as shown in the below example.

Implement Dimensions API #321

Open **ilyalesik** opened this issue on 30 Jun · 1 comment

ilyalesik commented on 30 Jun — Member

React Native Dimensions

ilyalesik added help wanted good first issue priority: medium complexity: medium topic: primitives support labels on 30 Jun

Кодогенерация 🤖

code generation

github.com/react-figma/code-generator

README.md

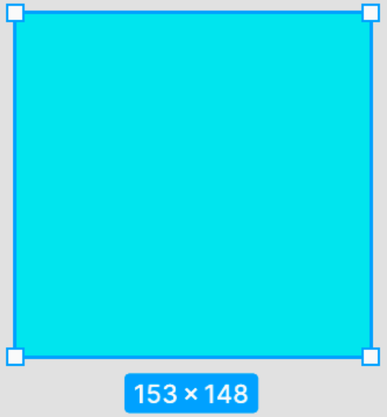
[WIP] Figma Code Generator Plugin

Pipeline

- Serialize Figma node tree to plain JS-object snapshot
- Transform plain JS-object snapshot to AST (via @babel/types)
- Transform AST to code (via @babel/generator)
- Format code (via Prettier)
- Show code with syntax highlighting (via Monaco Editor)

Layers Assets Page 3

Rectangle 1



Babel (Developer VM)

```
1  import * as React from "react";
2  import { View } from "react-figma";
3  export const Component = () => {
4    return <View width={153} height={148} />;
5  };
6
```

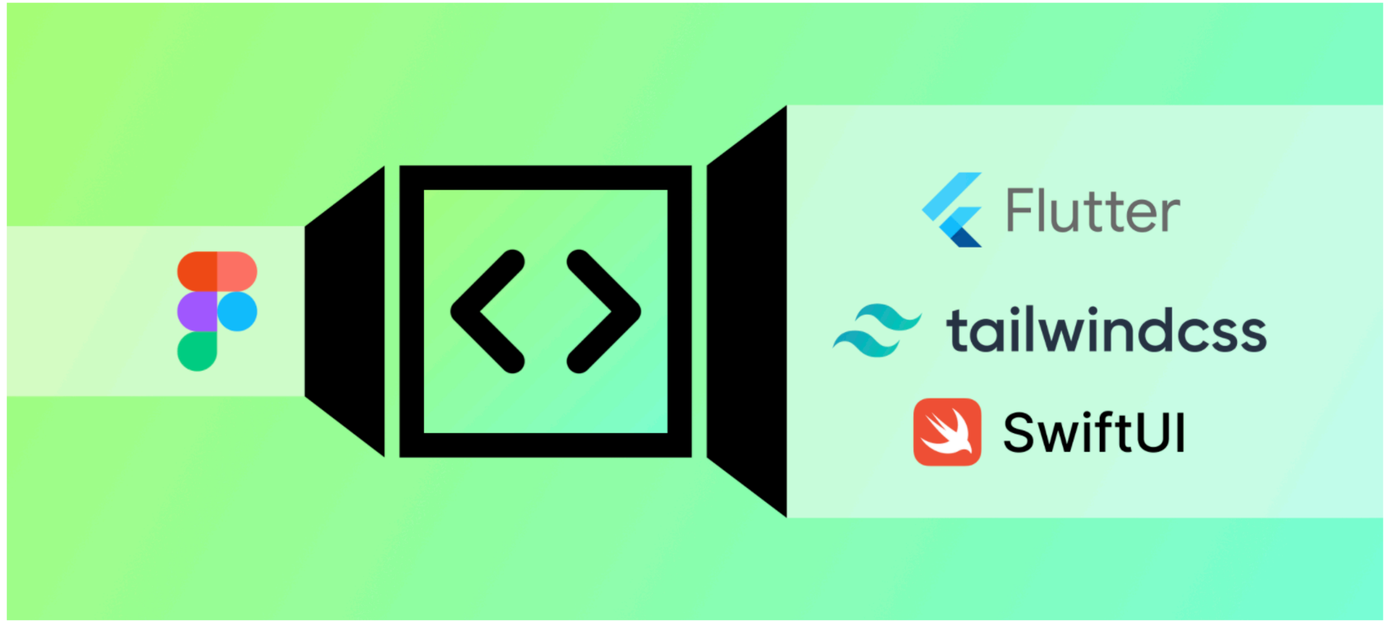
Design Proto

Throu

EE

Export

README.md

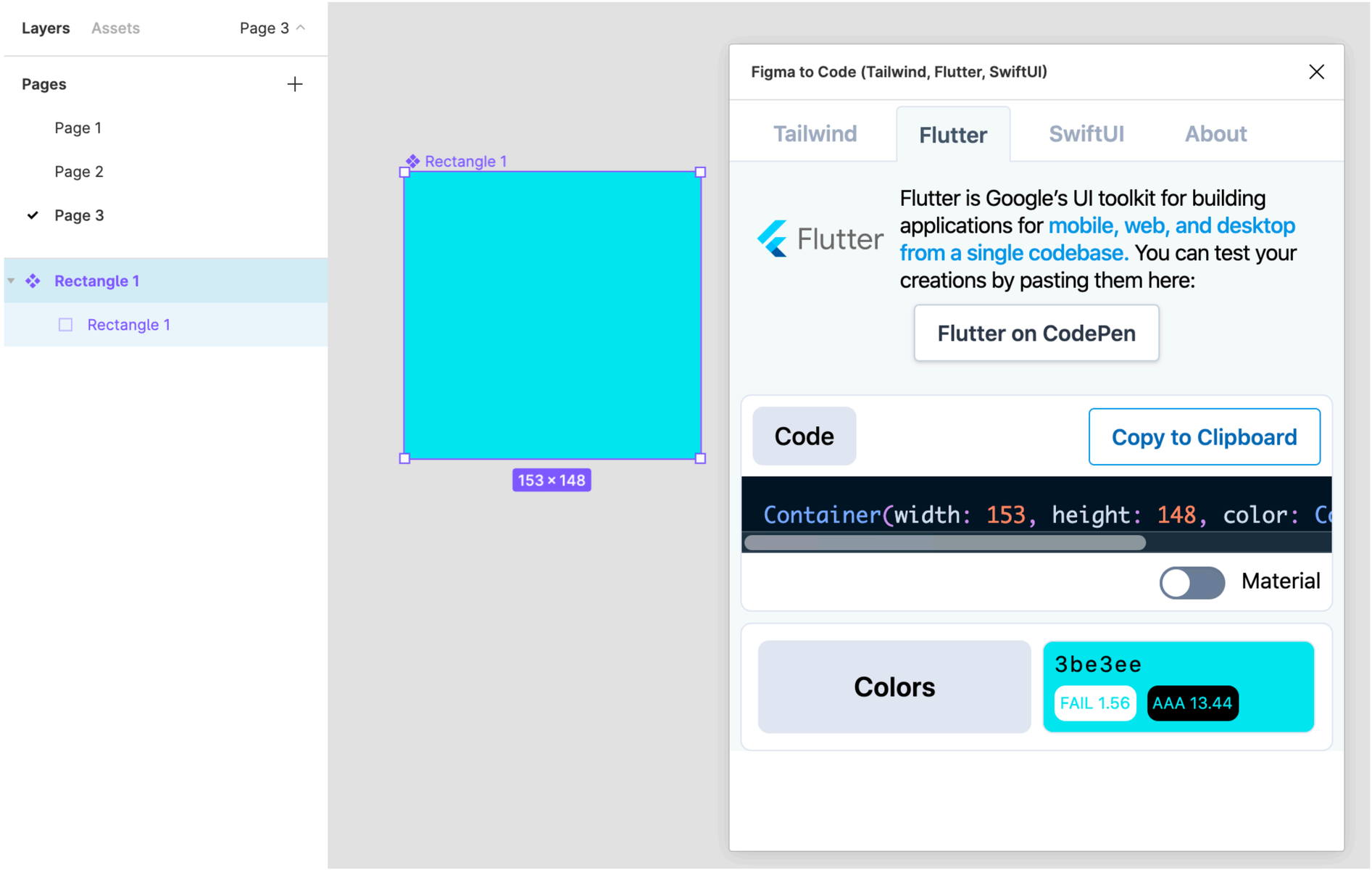


Figma to Code

CI passing | codecov 98% | Twitter @bernaferrari

Available as a Figma Plugin

github.com/bernaferrari/FigmaToCode


Layers
Assets
Page 3
Pages
Page 1
Page 2
Page 3
Rectangle 1
Rectangle 1
Rectangle 1
153 × 148
Figma to Code (Tailwind, Flutter, SwiftUI)
Tailwind
Flutter
SwiftUI
About
Flutter
Flutter is Google's UI toolkit for building applications for mobile, web, and desktop from a single codebase. You can test your creations by pasting them here:
Flutter on CodePen
Code
Copy to Clipboard
Container(width: 153, height: 148, color: C
Material
Colors
3be3ee
FAIL 1.56
AAA 13.44

react-native / react-figma / react-primitives supporting #11

Open **ilyalesik** opened this issue 4 days ago · 4 comments

ilyalesik commented 4 days ago • edited

Proposal: react-native / react-figma / react-primitives supporting

The react-primitives family has several universal interfaces:

- View
- Text
- StyleSheet (with Yoga Layout-like properties)
- Platform

bernaferrari commented 4 days ago Owner

My biggest issue is that I don't know react native. How do I make rectangles, flex, etc in it? If you are willing to help, we can do it. It is not hard, there are just too many details.

👍 1 👀 1

Сложности

Этап конвертации Figma-дерева в **AST**

Как работать с изменениями (например, **props** у компонентов)

Все хотят **разного**

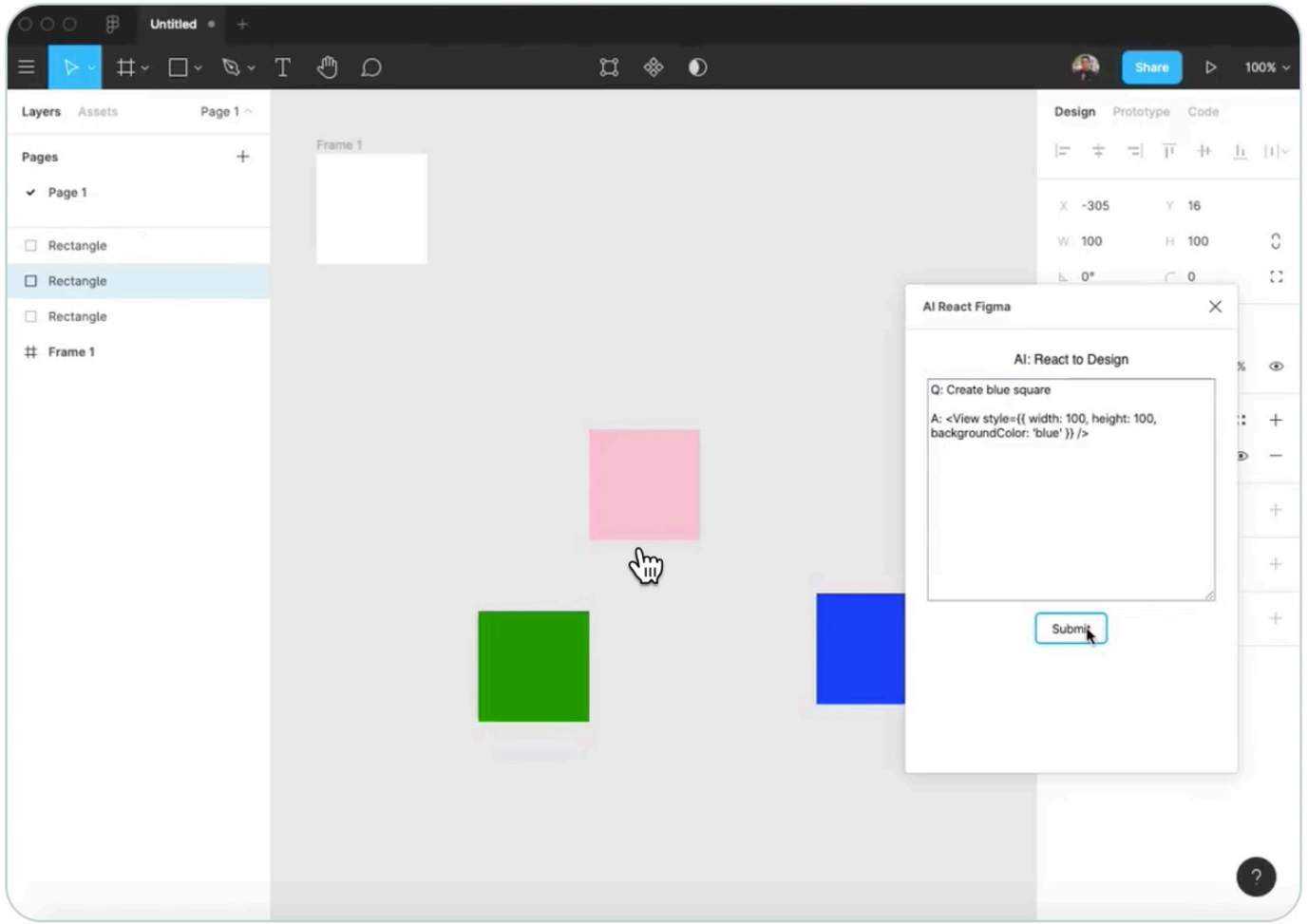
OpenAI GPT-3 💡

GPT-3

Sonny Lazuardi
@sonnylazuardi

experimenting @openAI to @reactjs component to @figmadesign

Перевести твит



Мы открыты для:

Новых **контрибуторов**

Для **early adopters**

welcome

Выводы

Принцип от простого - к сложному

У хранения дизайн-системы в коде есть **преимущества**

Пишите **кроссплатформенно** и процветайте!

Thanks

Спасибо за внимание!

Особая благодарность

Александра Калинина

Nix

korery

HOLY JS /...

Kudos

ilyalesik/multiplatform-design-systems-talk

@ilyalesik

@ilialesik

lesik.dev