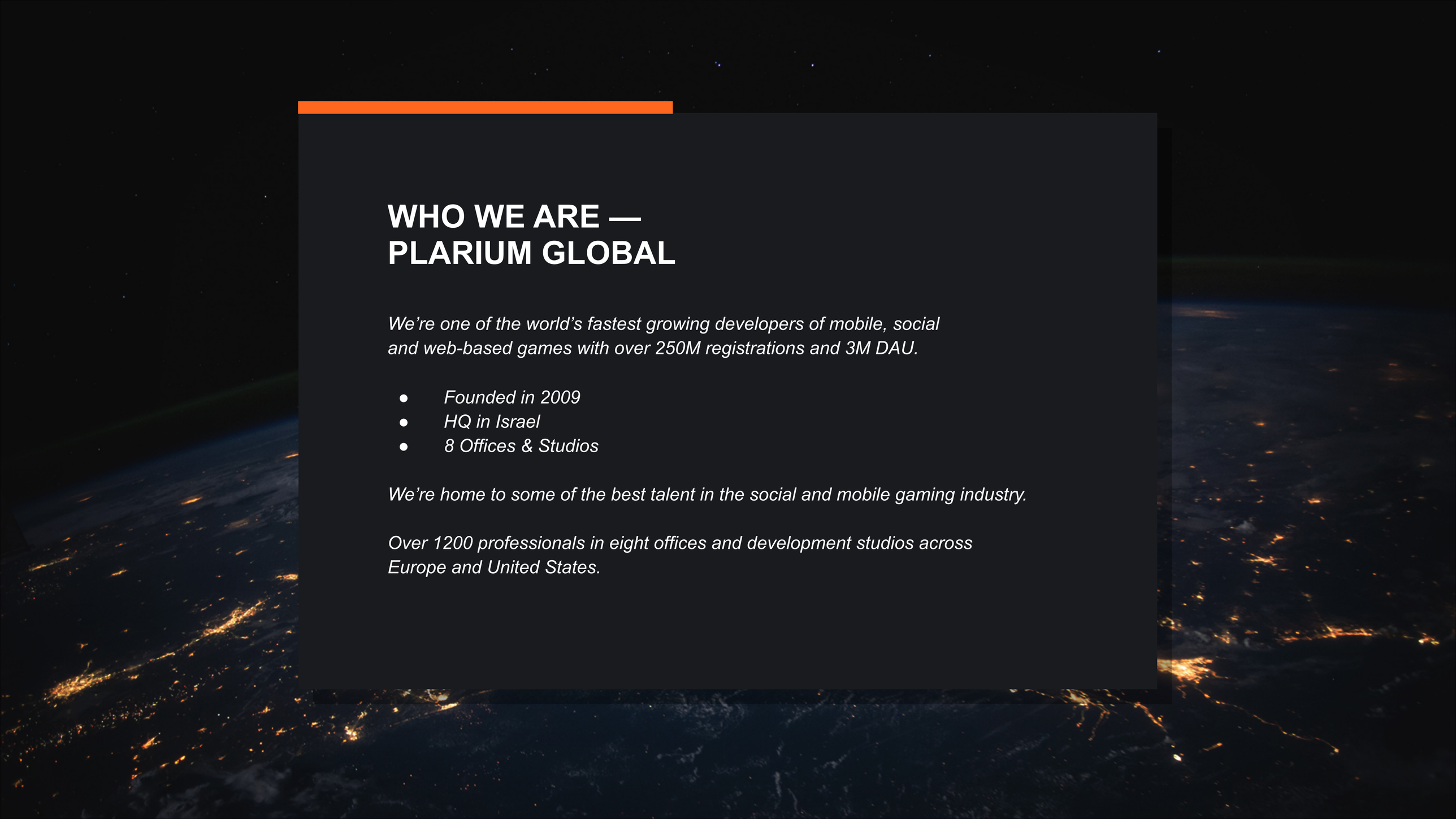

PLARIUM



WHO WE ARE — PLARIUM GLOBAL

We’re one of the world’s fastest growing developers of mobile, social and web-based games with over 250M registrations and 3M DAU.

- *Founded in 2009*
- *HQ in Israel*
- *8 Offices & Studios*

We’re home to some of the best talent in the social and mobile gaming industry.

Over 1200 professionals in eight offices and development studios across Europe and United States.

UX

User Experience

Разработка для пользователей

Забота о пользователях

DX

Developer Experience

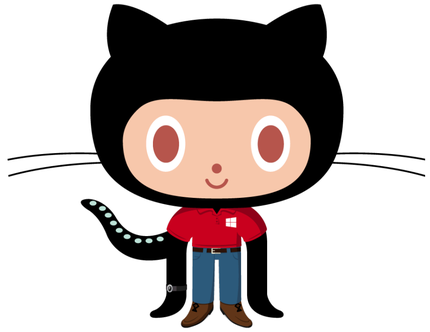
Ежедневная работа

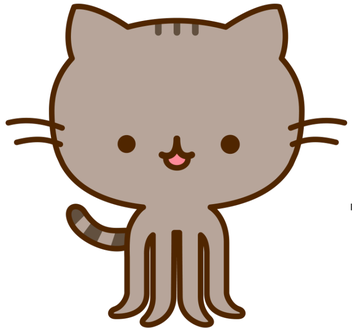
Комфорт в работе

Затрагивает весь процесс разработки

От написания кода до релиза

<code/>





<code/>

Мы пишем код, с которым будут работать другие разработчики

DX ~ UX

Разработка для разработчиков

DX ~ UX

Забота о разработчиках

Стараться держать все в хорошем состоянии

Все зависит от проекта
(архитектура / legacy / важность)

Но нюансы в мелочах

```
export default styled(
  compose(
    withData(
      new UsersData({
        cache: false,
      })
    ),
    withDefaultTheme(),
    withBaseCardCSS(),
    withUserCRUDActions({
      actions: [CRUD_ACTIONS.UPDATE, CRUD_ACTIONS.DELETE],
    })
  )(props => {
    return (
      <Card
        className={cx(props.className, styles.user)}
        title={props.user.name}
        actions={
          <React.Fragment>
            <IconButton
              icon="edit"
              onClick={() => {
                props.editUser(pros.user.id);
              }}
            />
            <IconButton
              icon="remove"
              onClick={() => {
                props.removeUser(pros.user.id);
```

```
export default styled(
  compose(
    withData(
      new UsersData({
        cache: false,
      })
    ),
    withDefaultTheme(),
    withBaseCardCSS(),
    withUserCRUDActions({
      actions: [CRUD_ACTIONS.UPDATE, CRUD_ACTIONS.DELETE],
    })
  )(props => {
    return (
      <Card
        className={cx(props.className, styles.user)}
        title={props.user.name}
        actions={
          <React.Fragment>
            <IconButton
              icon="edit"
              onClick={() => {
                props.editUser(pros.user.id);
              }}
            />
            <IconButton
              icon="remove"
              onClick={() => {
                props.removeUser(pros.user.id);
```

```
        title={props.user.name}
        actions={
          <React.Fragment>
            <IconButton
              icon="edit"
              onClick={() => {
                props.editUser(pros.user.id);
              }}
            />
            <IconButton
              icon="remove"
              onClick={() => {
                props.removeUser(pros.user.id);
              }}
            />
          </React.Fragment>
        }
      >
        <Avatar src={getUserAvatar(props.user)} />
      </Card>
    );
  })
)`
  border-color: black;
  padding: 8px;

  :hover {
    background: ${getThemeColor('normal200')};
  }
`;
```

Прочитать возможно, но сложно

Перед Бренданом Эйхом, нанятым в компанию Netscape 4 апреля 1995 года[17], была поставлена задача внедрить язык программирования Scheme или что-то похожее в браузер Netscape. Поскольку требования были размыты, Эйха перевели в группу, ответственную за серверные продукты, где он проработал месяц, занимаясь улучшением протокола HTTP[17]. В мае разработчик был переброшен обратно, в команду, занимающуюся клиентской частью (браузером), где он немедленно начал разрабатывать концепцию нового языка программирования. Менеджмент разработки браузера, включая Тома Пакина (Tom Paquin), Михаэля Тоя (англ.), Рика Шелла (Rick Schell), был убеждён, что Netscape должен поддерживать язык программирования, встраиваемый в HTML-код страницы

Перед Бренданом Эйхом, нанятым в компанию Netscape 4 апреля 1995 года[17], была поставлена задача внедрить язык программирования Scheme или что-то похожее в браузер Netscape. Поскольку требования были размыты, Эйха перевели в группу, ответственную за серверные продукты, где он проработал месяц, занимаясь улучшением протокола HTTP[17]. В мае разработчик был переброшен обратно, в команду, занимающуюся клиентской частью (браузером), где он немедленно начал разрабатывать концепцию нового языка программирования. Менеджмент разработки браузера, включая Тома Пакина (Tom Paquin), Михаэля Тоя (англ.), Рика Шелла (Rick Schell), был убеждён, что Netscape должен поддерживать язык программирования, встраиваемый в HTML-код страницы

```
<Dialog isOpen={true} />

<Popover isVisible={true} />

<FiltersToolbar isShown={true} />

{isVisible ? <UserDialog /> : null}
```

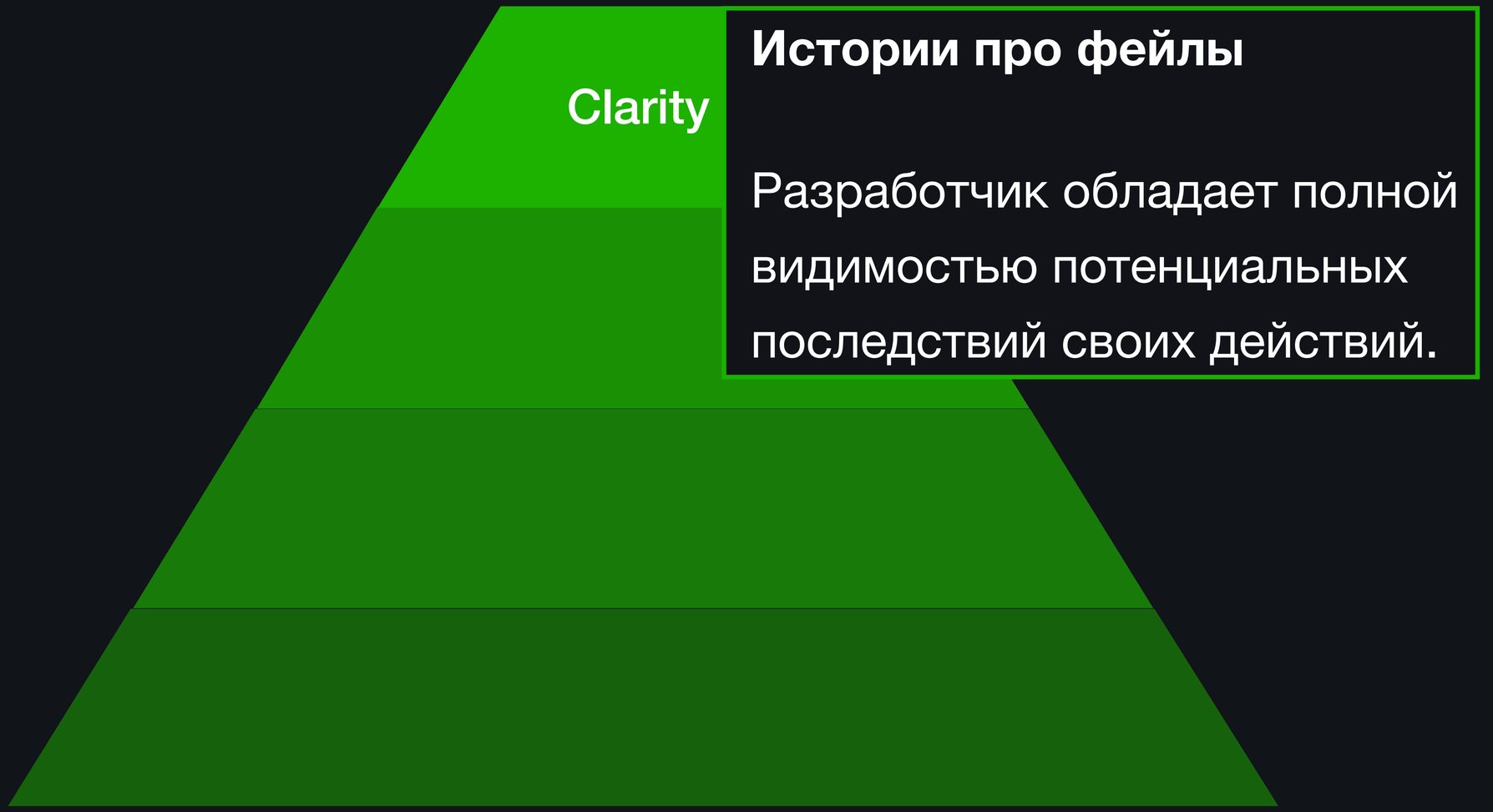
Тратишь целый день на мелочи и
пропадает желание делать
интересные и сложные части проекта


Business Value важнее. Часто
приходится с этим жить.

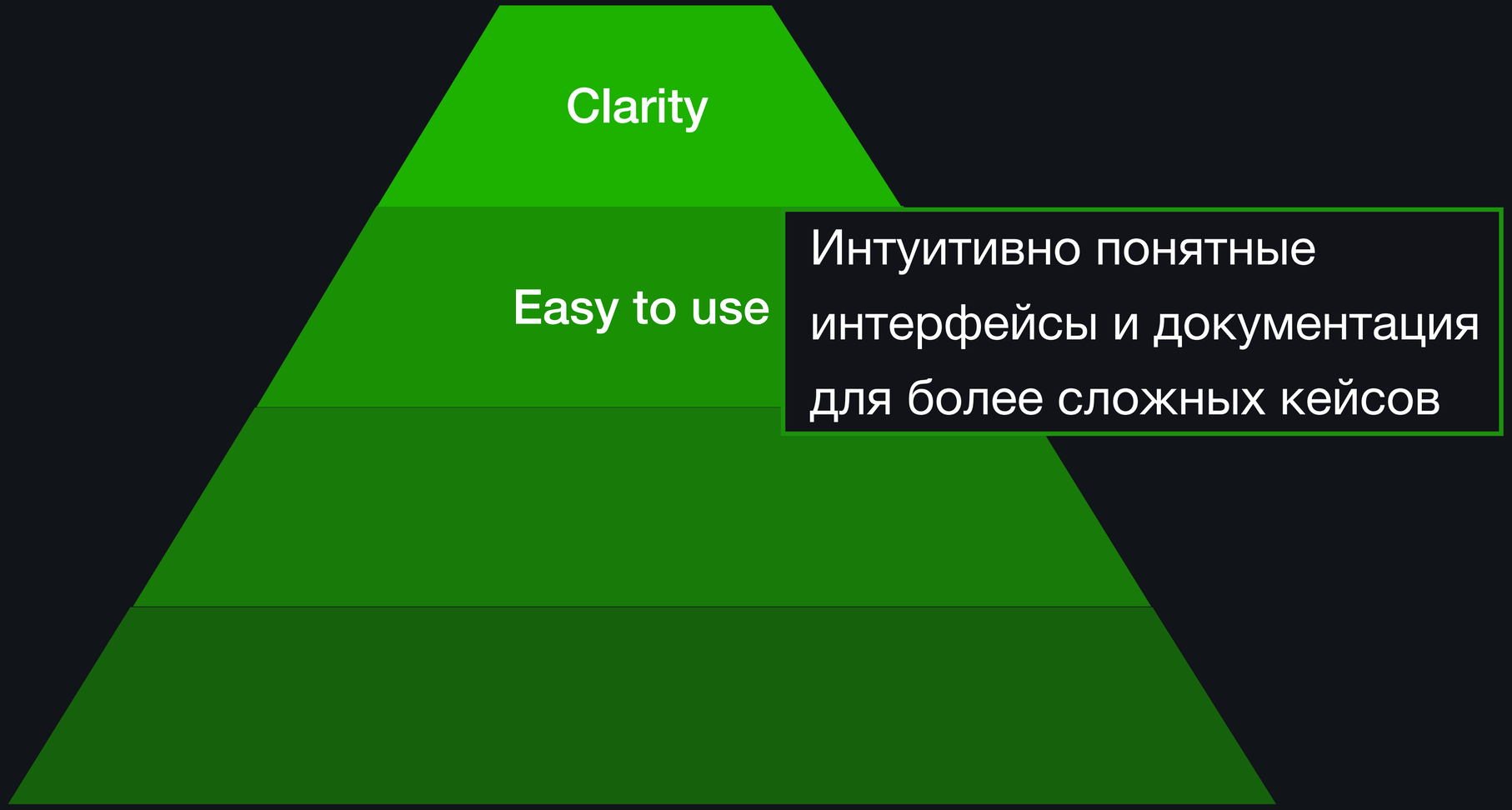
DX

DX

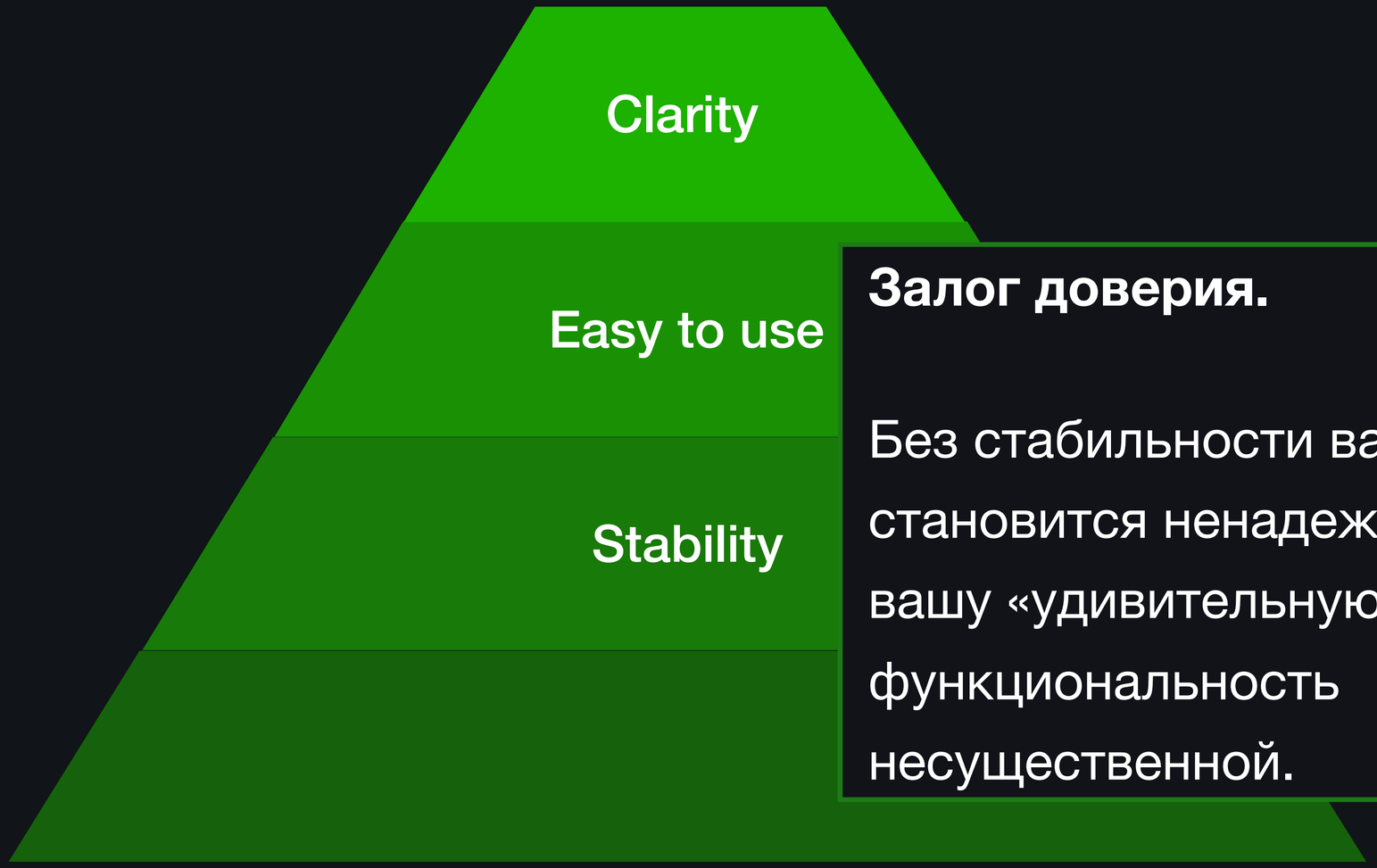
DX


Clarity
Истории про фейлы
Разработчик обладает полной видимостью потенциальных последствий своих действий.

DX


Clarity
Easy to use
Интуитивно понятные интерфейсы и документация для более сложных кейсов

DX



Залог доверия.

Без стабильности ваш продукт становится ненадежным, делая вашу «удивительную» функциональность несущественной.

DX

Если не работает, то это просто не работает.

DX

Чем меньше кода — тем лучше

Чем проще код — тем лучше

DX влияет на:

- Скорость разработки
- Сохранение спокойного состояния
- Комфорт
- …
- Качество

Что нужно делать?

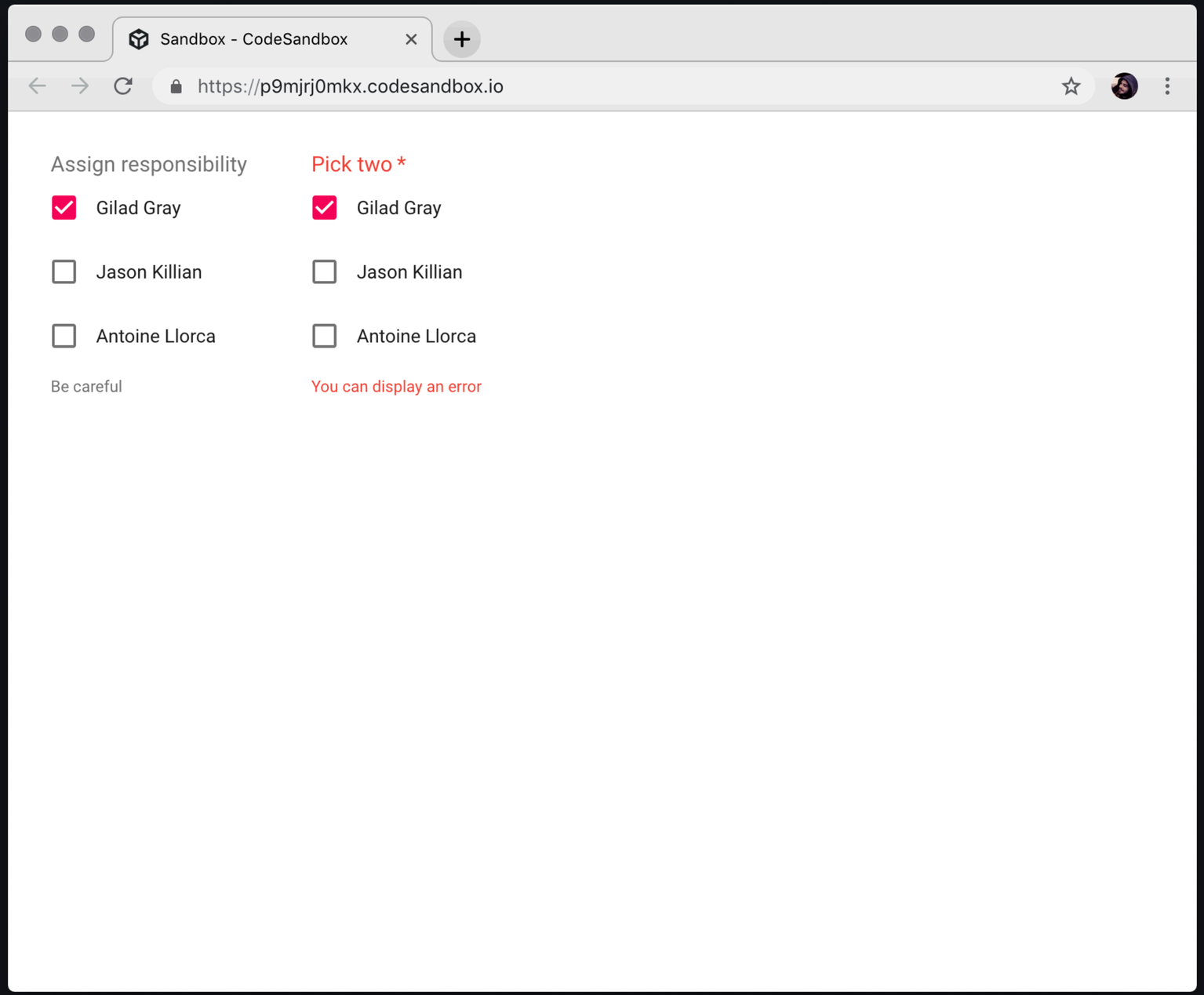
- Не писать лишний код
- Писать простой код
- Разделение кода — бизнес логика и инфраструктура
- Контролировать изменения
- Придерживаться гайдлайнов

Что нужно делать?

- Не писать лишний код
- Писать простой код
- Разделение кода — бизнес логика и инфраструктура
- Контролировать изменения
- Придерживаться гайдлайнов

React Components

Существующие решения


Sandbox - CodeSandbox
https://p9mjrj0mkx.codesandbox.io
Assign responsibility
Gilad Gray
Jason Killian
Antoine Llorca
Be careful
Pick two *
Gilad Gray
Jason Killian
Antoine Llorca
You can display an error

```
import React from 'react';
import PropTypes from 'prop-types';
import { withStyles } from '@material-ui/core/styles';
import FormLabel from '@material-ui/core/FormLabel';
import FormControl from '@material-ui/core/FormControl';
import FormGroup from '@material-ui/core/FormGroup';
import FormControlLabel from '@material-ui/core/FormControlLabel';
import FormHelperText from '@material-ui/core/FormHelperText';
import Checkbox from '@material-ui/core/Checkbox';

const styles = theme => ({
  root: {
    display: 'flex',
  },
  formControl: {
    margin: theme.spacing.unit * 3,
  },
});

class CheckboxesGroup extends React.Component {
  state = {
    gilad: true,
    jason: false,
    antoine: false,
  };

  handleChange = name => event => {
    this.setState({ [name]: event.target.checked });
  };
```

```
import React from 'react';
import PropTypes from 'prop-types';
import { withStyles } from '@material-ui/core/styles';
import FormLabel from '@material-ui/core/FormLabel';
import FormControl from '@material-ui/core/FormControl';
import FormGroup from '@material-ui/core/FormGroup';
import FormControlLabel from '@material-ui/core/FormControlLabel';
import FormHelperText from '@material-ui/core/FormHelperText';
import Checkbox from '@material-ui/core/Checkbox';

const styles = theme => ({
  root: {
    display: 'flex',
  },
  formControl: {
    margin: theme.spacing.unit * 3,
  },
});

class CheckboxesGroup extends React.Component {
  state = {
    gilad: true,
    jason: false,
    antoine: false,
  };

  handleChange = name => event => {
    this.setState({ [name]: event.target.checked });
  };
```

```
            />
            <FormControlLabel
              control={
                <Checkbox checked={jason} onChange={this.handleChange('jason')} value="jason" />
              }
              label="Jason Killian"
            />
            <FormControlLabel
              control={
                <Checkbox
                  checked={antoine}
                  onChange={this.handleChange('antoine')}
                  value="antoine"
                />
              }
              label="Antoine Llorca"
            />
          </FormGroup>
          <FormHelperText>You can display an error</FormHelperText>
        </FormControl>
      </div>
    );
  }
}

CheckboxesGroup.propTypes = {
  classes: PropTypes.object.isRequired,
};

export default withStyles(styles)(CheckboxesGroup);
```

```
import FormLabel from '@material-ui/core/FormLabel';
import FormControl from '@material-ui/core/FormControl';
import FormGroup from '@material-ui/core/FormGroup';
import FormControlLabel from '@material-ui/core/FormControlLabel';
import FormHelperText from '@material-ui/core/FormHelperText';
import Checkbox from '@material-ui/core/Checkbox';

function render() {
  return (
    <div className={classes.root}>
      <FormControl component="fieldset" className={classes.formControl}>
        <FormLabel component="legend">Assign responsibility</FormLabel>
        <FormGroup>
          <FormControlLabel control={<Checkbox />} label="Gilad Gray" />
          <FormControlLabel control={<Checkbox />} label="Jason Killian" />
          <FormControlLabel control={<Checkbox />} label="Antoine Llorca" />
        </FormGroup>
        <FormHelperText>Be careful</FormHelperText>
      </FormControl>
      <FormControl
        required
        error={error}
        component="fieldset"
        className={classes.formControl}
      >
        <FormLabel component="legend">Pick two</FormLabel>
        <FormGroup>
          <FormControlLabel control={<Checkbox />} label="Gilad Gray" />
          <FormControlLabel control={<Checkbox />} label="Jason Killian" />
          <FormControlLabel control={<Checkbox />} label="Antoine Llorca" />
```

```
import Box from '@plarium/ui';
import Text from '@plarium/ui';
import Checkbox from '@plarium/ui';

function render() {
  return (
    <Box>
      <Box vertical={true}>
        <Text variant="legend">Assign responsibility</Text>
        <Checkbox label="Gilad Gray" />
        <Checkbox label="Jason Killnan" />
        <Checkbox label="Antonie Llorca" />
        <Text variant="overline">Be careful</Text>
      </Box>
      <Box vertical={true}>
        <Text variant="legend">Pick two</Text>
        <Checkbox label="Gilad Gray" />
        <Checkbox label="Jason Killnan" />
        <Checkbox label="Antonie Llorca" />
        <Text variant="overline">You can display an error</Text>
      </Box>
    </Box>
  );
}
```

```
import Box from '@plarium/ui';
import Text from '@plarium/ui';
import Checkbox from '@plarium/ui';

function render() {
  return (
    <Box>
      <Box vertical={true}>
        <Text variant="legend">Assign responsibility</Text>
        <Checkbox label="Gilad Gray" />
        <Checkbox label="Jason Killnan" />
        <Checkbox label="Antonie Llorca" />
        <Text variant="overline">Be careful</Text>
      </Box>
      <Box vertical={true}>
        <Text variant="legend">Pick two</Text>
        <Checkbox label="Gilad Gray" />
        <Checkbox label="Jason Killnan" />
        <Checkbox label="Antonie Llorca" />
        <Text variant="overline">You can display an error</Text>
      </Box>
    </Box>
  );
}
```

```
import Box from '@plarium/ui';
import Text from '@plarium/ui';
import Checkbox from '@plarium/ui';

function render() {
  return (
    <Box>
      <Box vertical={true}>
        <Text variant="legend">Assign responsibility</Text>
        <Checkbox label="Gilad Gray" />
        <Checkbox label="Jason Killnan" />
        <Checkbox label="Antonie Llorca" />
        <Text variant="overline">Be careful</Text>
      </Box>
      <Box vertical={true}>
        <Text variant="legend">Pick two</Text>
        <Checkbox label="Gilad Gray" />
        <Checkbox label="Jason Killnan" />
        <Checkbox label="Antonie Llorca" />
        <Text variant="overline">You can display an error</Text>
      </Box>
    </Box>
  );
}
```

Технология

Typescript

— Используете типы?
— Вы просто в голове не можете все удержать

Зачем использовать React, если можно:

`document.appendChild()`

```tsx
interface IButtonProps extends React.ButtonHTMLAttributes<HTMLButtonElement> {
  variant?: 'outlined' | 'text' | 'contained';
}

const Button = styled.button<IButtonProps>``;
```

```tsx
interface IButtonProps extends React.ButtonHTMLAttributes<HTMLButtonElement> {
  variant?: 'outlined' | 'text' | 'contained';
}

const Button = styled.button<IButtonProps>``;


const render = () => {
  return <Button variant="outlined">Kharkiv JS</Button>;
};
```

```tsx
interface IButtonProps extends React.ButtonHTMLAttributes<HTMLButtonElement> {
  variant?: 'outlined' | 'text' | 'contained';
}

const Button = styled.button<IButtonProps>``;


const render = () => {
  return <Button variant="oulined">Kharkiv JS</Button>;
};
```

```ts
interface IButtonProps extends React.ButtonHTMLAttributes<HTMLButtonElement> {
  variant?: 'outlined' | 'text' | 'contained';
}

const Button = styled.button<IButtonProps>``;


const render = () => {
  return <Button variant="oulined">Kharkiv JS</Button>;
};
```

- Мелочи экономят кучу времени
- Лишний раз не нужно идти за документацией
- Исправление ошибок до запуска приложения
- Скорость разработки

Как сделать компонент?

— Создаем файл {component}.tsx
— Пишем код

Как нарисовать сову

1.

1. Рисуем кружочки

2.

2. Рисуем остаток совы

Нужно выделять время на

Проектирование

Как вы хотите использовать компонент в коде?

Tooltip

Нужен anchor, что бы рассчитать позицию

```
const Tooltip = (props: { anchor: HTMLElement, text: sting }) => {
  const styles = getPositionAccordingToAnchor(props.anchor);

  return <Text style={styles}>{props.text}</Text>;
};
```

```
const Tooltip = (props: { anchor: HTMLElement, text: sting }) => {
  const styles = getPositionAccordingToAnchor(props.anchor);

  return <Text style={styles}>{props.text}</Text>;
};
```

```
const Tooltip = (props: { anchor: HTMLElement, text: sting }) => {
  const styles = getPositionAccordingToAnchor(props.anchor);

  return <Text style={styles}>{props.text}</Text>;
};
```

```
const Tooltip = (props: { anchor: HTMLElement, text: sting }) => {
  const styles = getPositionAccordingToAnchor(props.anchor);

  return <Text style={styles}>{props.text}</Text>;
};
```

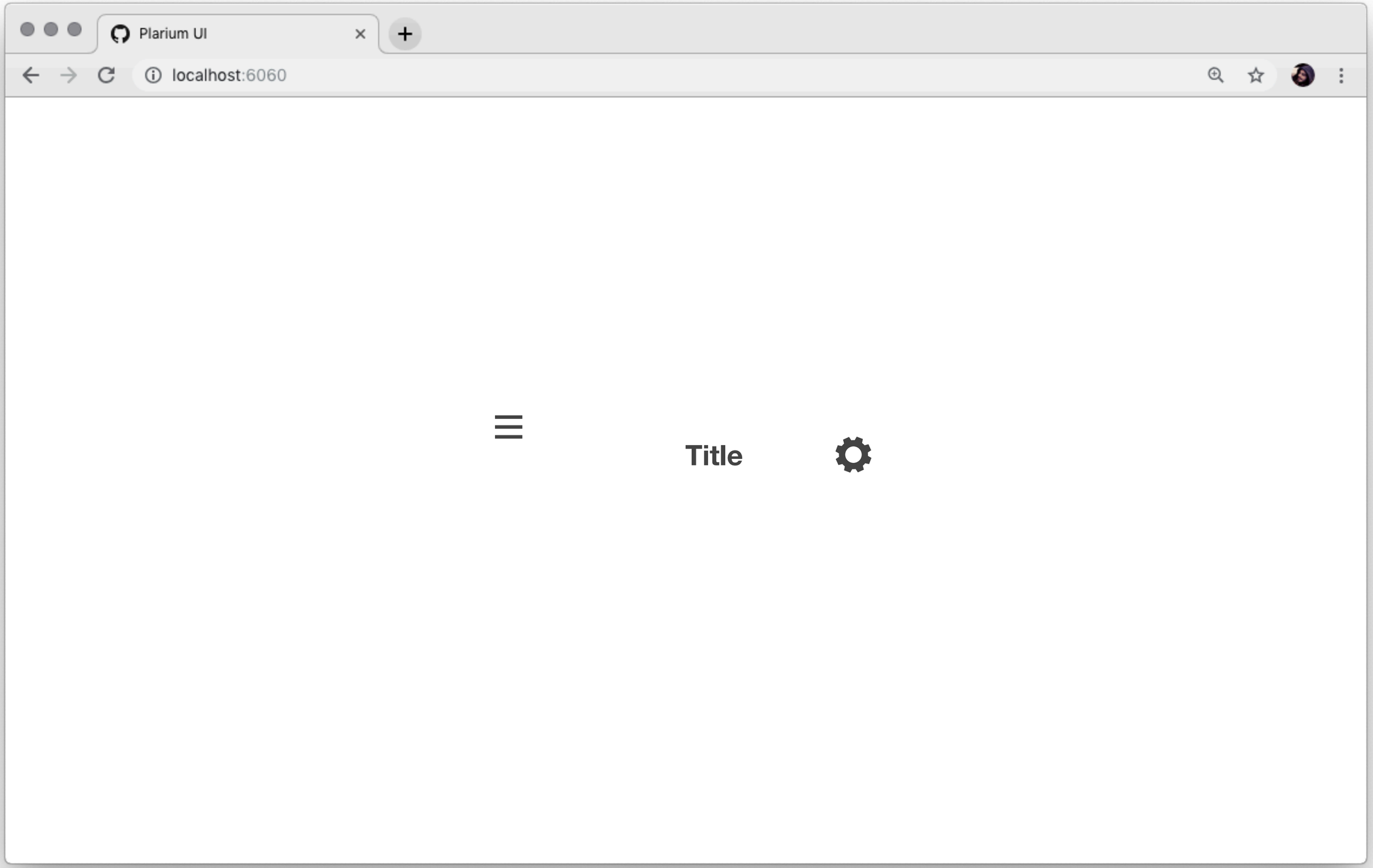
```
class UserEditIconButton extends React.Component {
  ref = React.createRef();

  render() {
    return (
      <React.Fragment>
        <IconButton icon="edit" ref={this.ref} />
        <Tooltip text="Edit User" anchor={this.ref.current} />
      </React.Fragment>
    );
  }
}
```

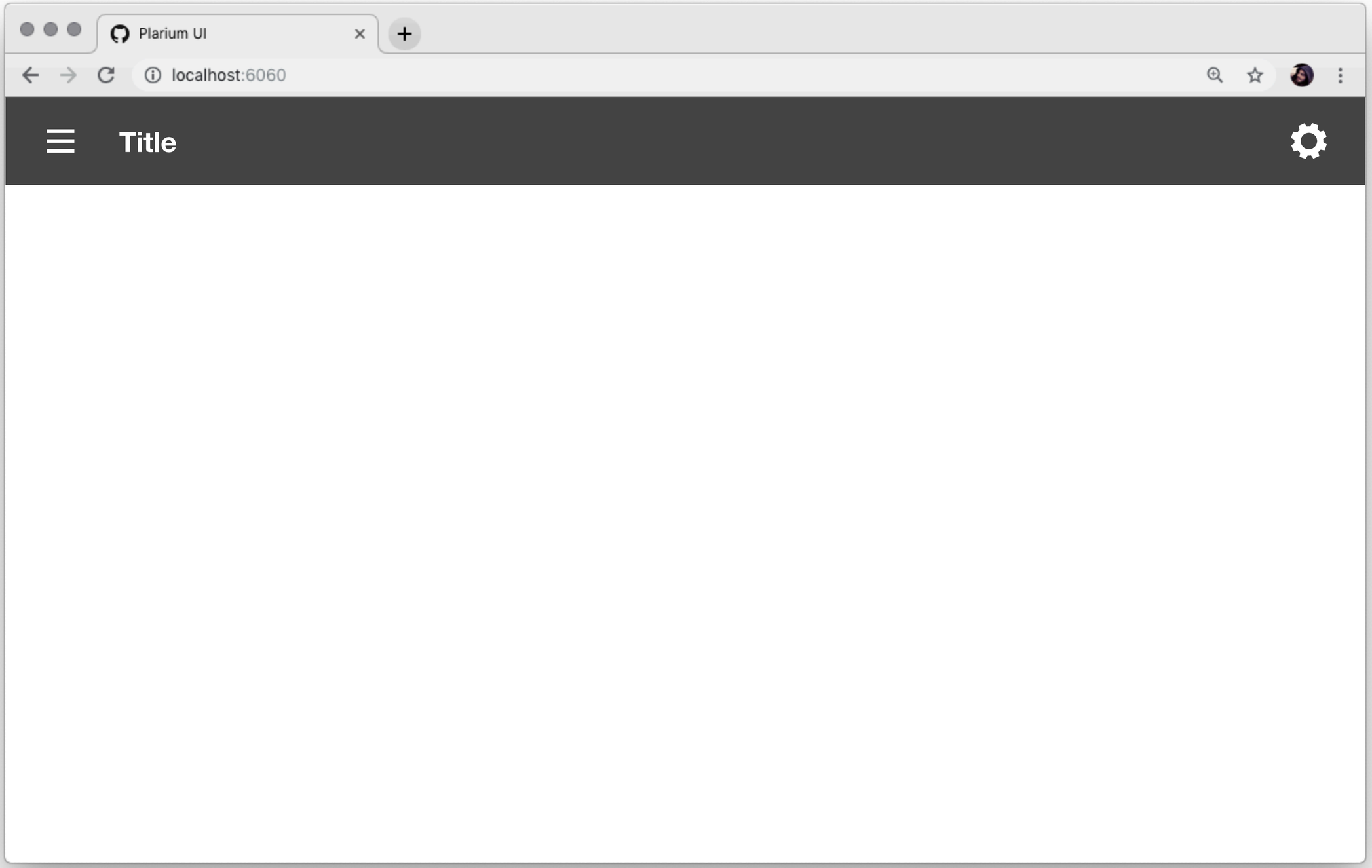
```
class UserEditIconButton extends React.Component {
  ref = React.createRef();

  render() {
    return (
      <React.Fragment>
        <IconButton icon="edit" ref={this.ref} />
        <Tooltip text="Edit User" anchor={this.ref.current} />
      </React.Fragment>
    );
  }
}
```

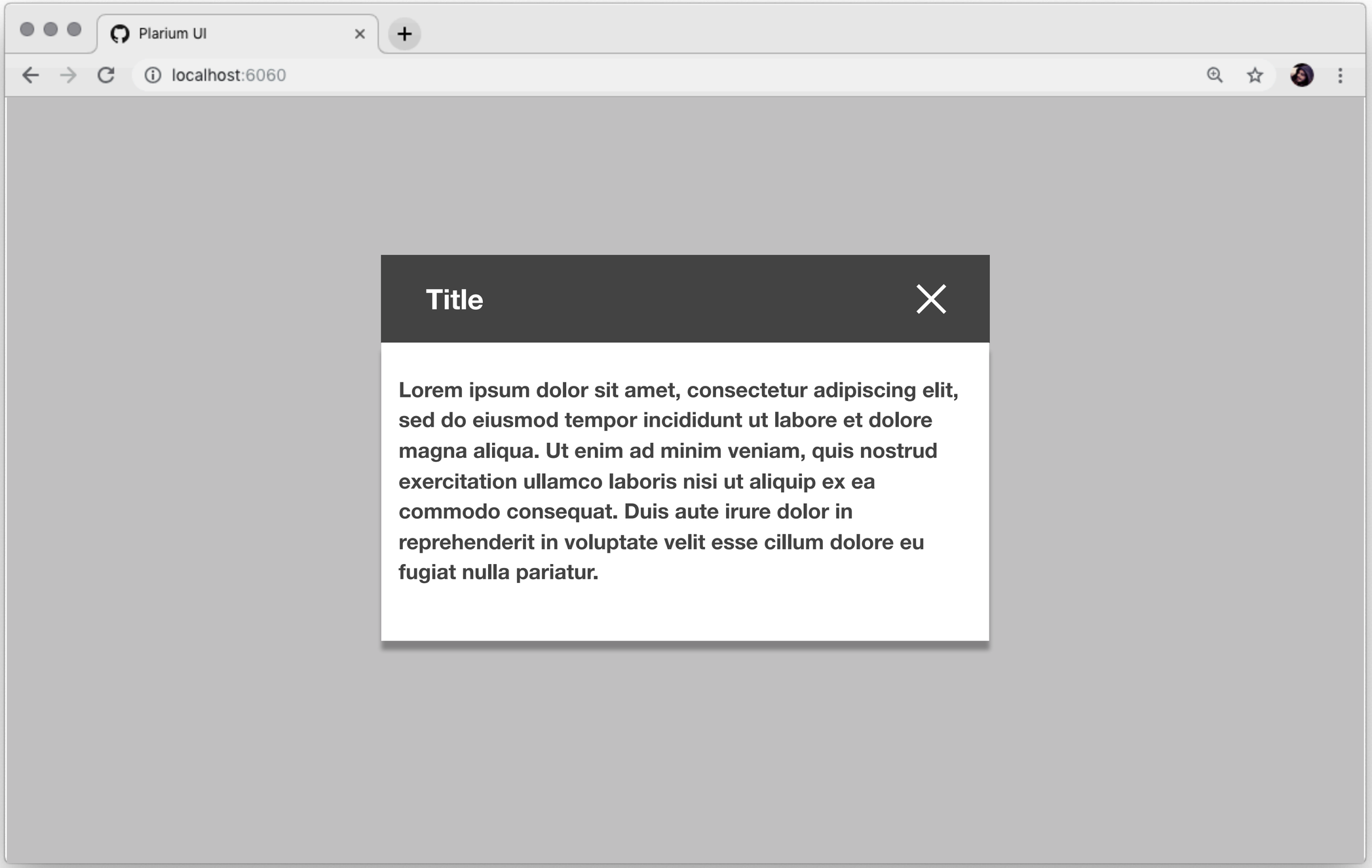
```
<Tooltip text="Edit User">
  <IconButton icon="edit" />
</Tooltip>
```

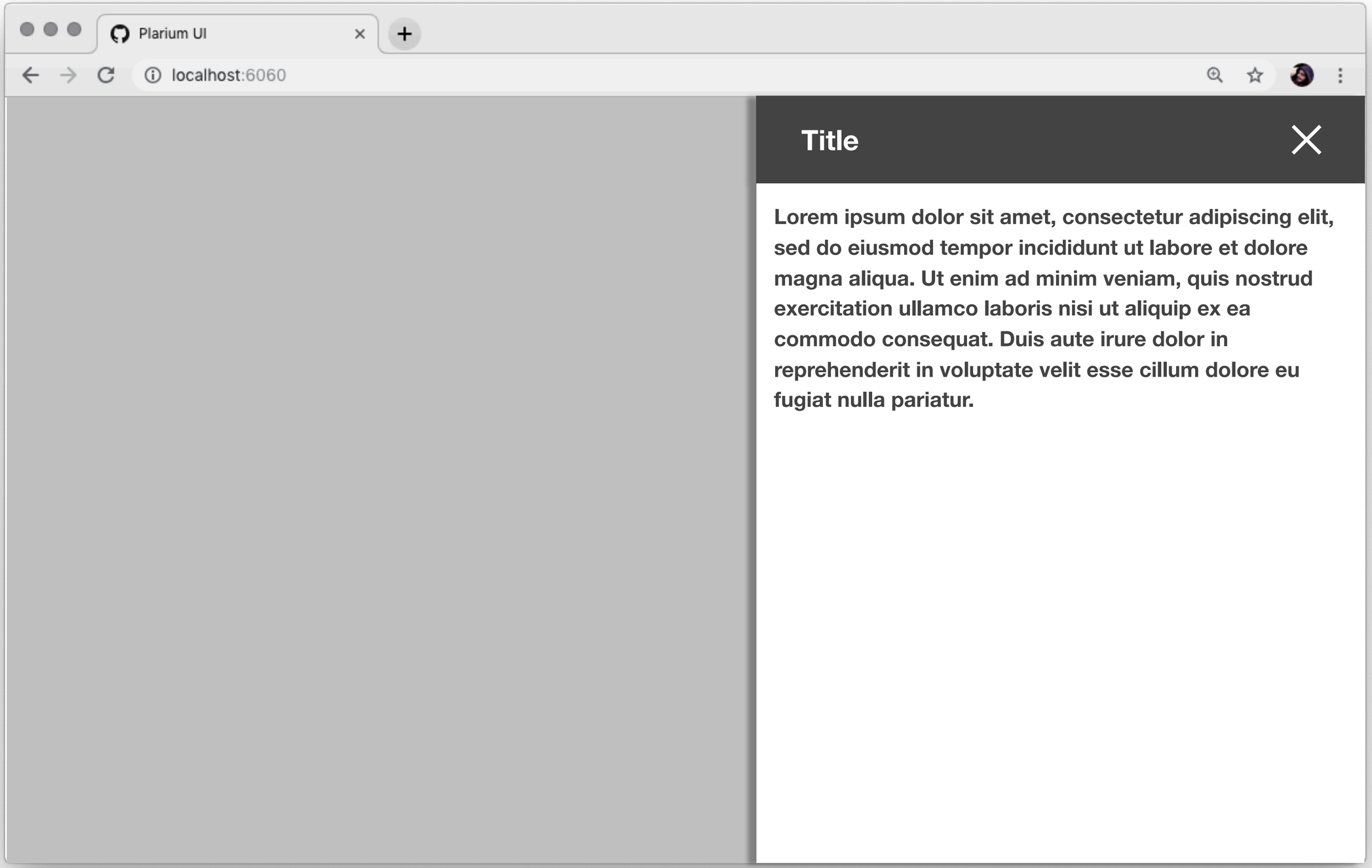
Как вы хотите использовать компонент в коде?

Где мы можем еще использовать компонент?


Plarium UI
localhost:6060
Title


Plarium UI
localhost:6060
Title


Plarium UI
localhost:6060
Title
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.


Plarium UI
localhost:6060
Title
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Plarium UI

localhost:6060

Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
<Toolbar
  title="Title"
  prefix={<IconButton icon="menu" />}
  actions={<IconButton icon="settings" />}
/>
```

```
<Dialog.Head
  title="Title"
  prefix={<IconButton icon="menu" />}
  actions={<IconButton icon="settings" />}
/>
```

```
<Card.Head
  title="Title"
  prefix={<IconButton icon="menu" />}
  actions={<IconButton icon="settings" />}
/>
```

```
<Toolbar
  title="Title"
  prefix={<IconButton icon="menu" />}
  actions={<IconButton icon="settings" />}
/>
```

```
<Dialog.Head
  title="Title"
  prefix={<IconButton icon="menu" />}
  actions={<IconButton icon="settings" />}
/>
```

```
<Card.Head
  title="Title"
  prefix={<IconButton icon="menu" />}
  actions={<IconButton icon="settings" />}
/>
```

Как вы хотите использовать компонент в коде?

Где мы можем еще использовать компонент?

Убедиться, что будет легко интегрировать

Описать нужные props и начать реализацию

Нужно определиться, как писать стили

CSS

Нужно определиться, как писать стили

CSS in JS

2 файла / 14 строк

```
// Button.css
.root {
  background: 1px solid --var(primaryColor);
  padding: 0 4px;
  margin: 4px;
}


// Button.js
import cx from 'classnames';
import styles from './Button.css';

const Button = ({ children, ...props }) => {
  return (
    <button {...props} className={cx(className, stales.root)}>
      {children}
    </button>
  );
};
```

1 файл / 5 строк

```
const Button = styled.button`
  background: 1px solid ${props => props.theme.colors.primary};
  padding: 0 4px;
  margin: 4px;
`;
```

Доступ к связанным компонентам через static fields

```
import {
  Table,
  TableBody,
  TableHead,
  TableHeadCell,
  TableRow,
  TableCell,
} from '@plarium/material-ui/Table';

function render() {
  return (
    <Table>
      <TableHead>
        <TableRow>
          <TableHeadCell>10</TableHeadCell>
        </TableRow>
      </TableHead>
      <TableBody>
        <TableRow>
          <TableCell>10</TableCell>
        </TableRow>
      </TableBody>
    </Table>
  );
}
```

```
import Table from '@plarium/material-ui/Table';

function render() {
  return (
    <Table>
      <Table.Head>
        <Table.Row>
          <Table.HeadCell>10</Table.HeadCell>
        </Table.Row>
      </Table.Head>
      <Table.Body>
        <Table.Row>
          <Table.Cell>10</Table.Cell>
        </Table.Row>
      </Table.Body>
    </Table>
  );
}
```

```
import Table from '@plarium/material-ui/Table';

function render() {
  return (
    <Table>
      <Table.Head>
        <Table.Row>
          <Table.HeadCell>10</Table.HeadCell>
        </Table.Row>
      </Table.Head>
      <Table.Body>
        <Table.Row>
          <Table.Cell>10</Table.Cell>
        </Table.Row>
      </Table.Body>
    </Table>
  );
}
```


KHARKIV JS

KHARKIV JS


```
import Button from '@plarium/ui/Button';

Button.Icon
Button.Text
```

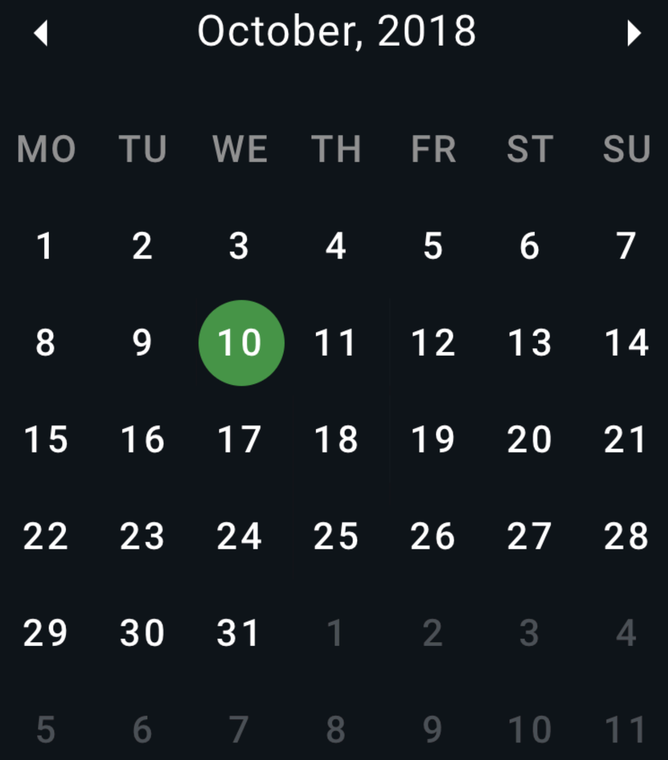

```
import styled from 'styled-components';
import Button from '@plarium/ui/Button';

const MyButton = styled(Button)`
  ${ Button.Icon }{
    color: yellow;
  }
`;
```


```
import styled from 'styled-components';
import Button from '@plarium/ui/Button';

const MyButton = styled(Button)`
  ${ Button.Icon }{
    color: yellow;
  }
`;
```

Autocomplete

```
const MyButton = styled(Button)`
  ${Button.Ic}
`;
```

Icon (property) IButton.Icon: StyledCom…

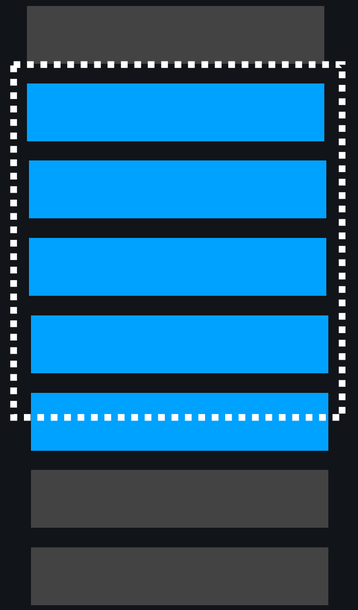



```
import Calendar from '@plarium/ui/Calendar';

Calendar.Day
Calendar.SelectedDay
Calendar.DisabledDay
Calendar.PrevMonthButton
Calendar.NextMonthButton
Calendar.MonthName
Calendar.Weekday
```

Что еще нужно?

Темизация

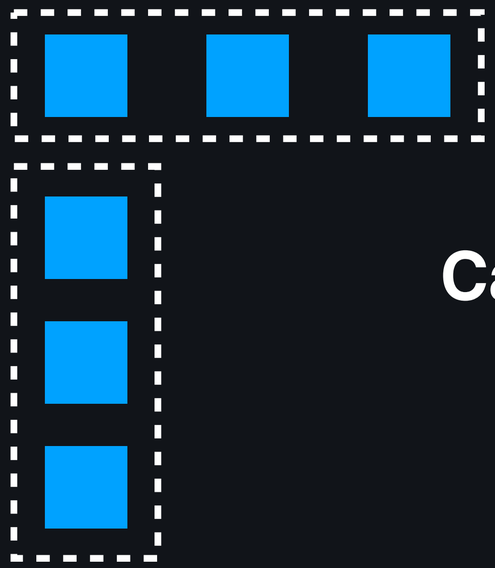
Виртуализация



Порталы

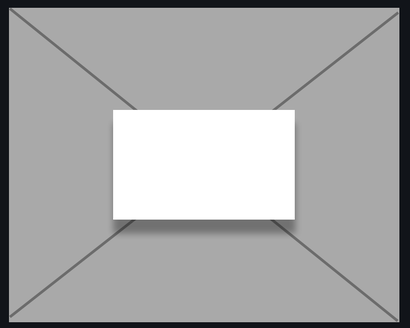
Triggers

Layout Система



Самодостаточность

Overlay



#1 Самодостаточность

Для использования компонента, не писать дополнительный код в приложении

Проблема Snackbar

Успех!

Беда =(

Должен быть виден {n} секунд или закрыт пользователем

Беда =(

Проблема:

— Рендерим на странице Snackbar
— Провисел 2 секунды из 10
— Пользователь перешил на другую страницу
— Snackbar пропал раньше времени

Решение:

— Берем Redux или другую подобную бибилотеку
— Добавляем модуль для сообщений (reducer)
— Реализуем методы для чтения и записи (actions)
— Берем первое добавленное сообщение и рендерим в руте приложения (connect)
— После скрытия показываем следующее сообщение
— Для рендера Snackbar используем Redux actions а не JSX

<SnackbarProvider/>

```
function render() {
  return (
    <SnackbarProvider>
      <App/>
    </SnackbarProvider>
  );
}
```

```
class Snackbar extends React.Component {
  componentDidMount() {
    this.props.addMessage({
      message: this.props.message,
    });
  }

  render() {
    return null;
  }
}
```

```
class Snackbar extends React.Component {
  componentDidMount() {
    this.props.addMessage({
      message: this.props.message,
    });
  }

  render() {
    return null;
  }
}
```

Из контекста

```
class Snackbar extends React.Component {
  componentDidMount() {
    this.props.addMessage({
      message: this.props.message,
    });
  }

  render() {
    return null;
  }
}
```

В настоящей реализации кода явно больше

Выглядит сложновато

Не важно, насколько сложный и запутанный код в node_modules

#2 Темизация

Структура

color?

margin?

padding?

border-radius?

Структура

ButtonBorderRadius?

CardBorderRadius?

DialogBorderRadius?

...

...

...

...

...

...

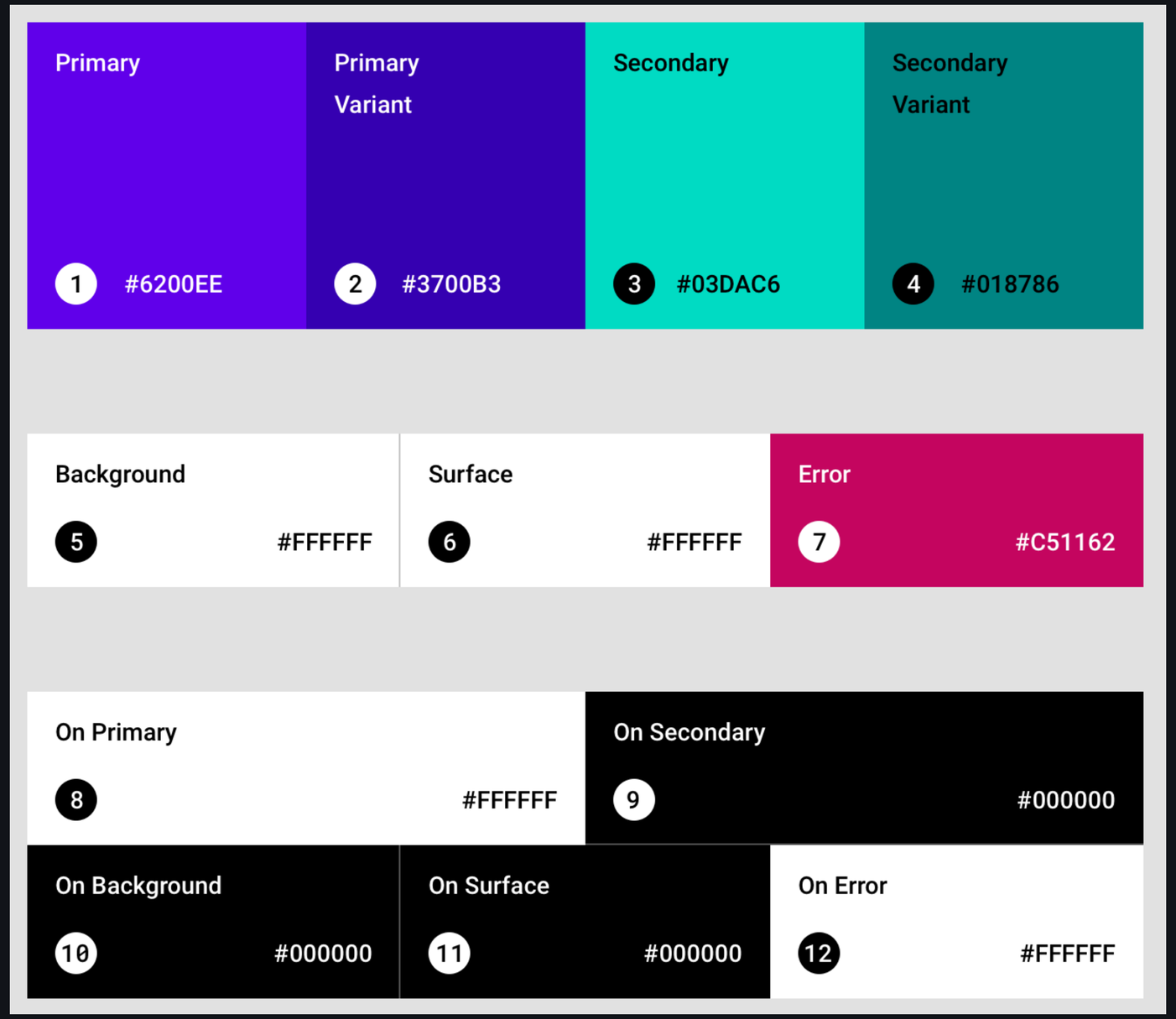
...

...

...

background?

Цветовая схема


Primary
1 #6200EE
Primary Variant
2 #3700B3
Secondary
3 #03DAC6
Secondary Variant
4 #018786
Background
5 #FFFFFF
Surface
6 #FFFFFF
Error
7 #C51162
On Primary
8 #FFFFFF
On Secondary
9 #000000
On Background
10 #000000
On Surface
11 #000000
On Error
12 #FFFFFF

```ts
interface IThemeColors {
  primary: string;
  onPrimary: string;

  secondary: string;
  onSecondary: string;

  accent: string;
  onAccent: string;

  surface: string;
  onSurface: string;

  error: string;
  onError: string;

  normal100: string;
  ...
  normal600: string;

  gray100: string;
  ...
  gray700: string;
}
```

Autocomplete

```
interface IThemeColors {
  primary: string;
  onPrimary: string;

  secondary: string;
  onSecondary: string;

  accent: string;
  onAccent: string;

  normal100: string;
  ...
  normal600: string;

  gray100: string;
  ...
  gray700: string;
}
```

```
export const MyButton = styled(Button)`
  color: ${props => props.theme.colors.p}
`;
```

```
primary  (property) IThemeColors.primary…
onPrimary
```

```ts
interface IThemeColors {
  primary: string;
  onPrimary: string;

  secondary: string;
  onSecondary: string;

  accent: string;
  onAccent: string;

  surface: string;
  onSurface: string;

  error: string;
  onError: string;

  normal100: string;
  ...
  normal600: string;

  gray100: string;
  ...
  gray700: string;
}
```

```ts
interface IThemeColors {
  primary: string;
  onPrimary: string;

  secondary: string;
  onSecondary: string;

  accent: string;
  onAccent: string;

  surface: string;
  onSurface: string;

  error: string;
  onError: string;

  normal100: string;
  ...
  normal600: string;

  gray100: string;
  ...
  gray700: string;
}
```

```tsx
import Surface from '@plarium/ui/Surface';
import Text from '@plarium/ui/Text';
import Icon from '@plarium/ui/Icon';

function render() {
  return (
    <Surface>
      <Text>Take The World!</Text>
      <Icon>face</Icon>
    </Surface>
  );
}
```

```ts
interface IThemeColors {
  primary: string;
  onPrimary: string;

  secondary: string;
  onSecondary: string;

  accent: string;
  onAccent: string;

  surface: string;
  onSurface: string;

  error: string;
  onError: string;

  normal100: string;
  ...
  normal600: string;

  gray100: string;
  ...
  gray700: string;
}
```

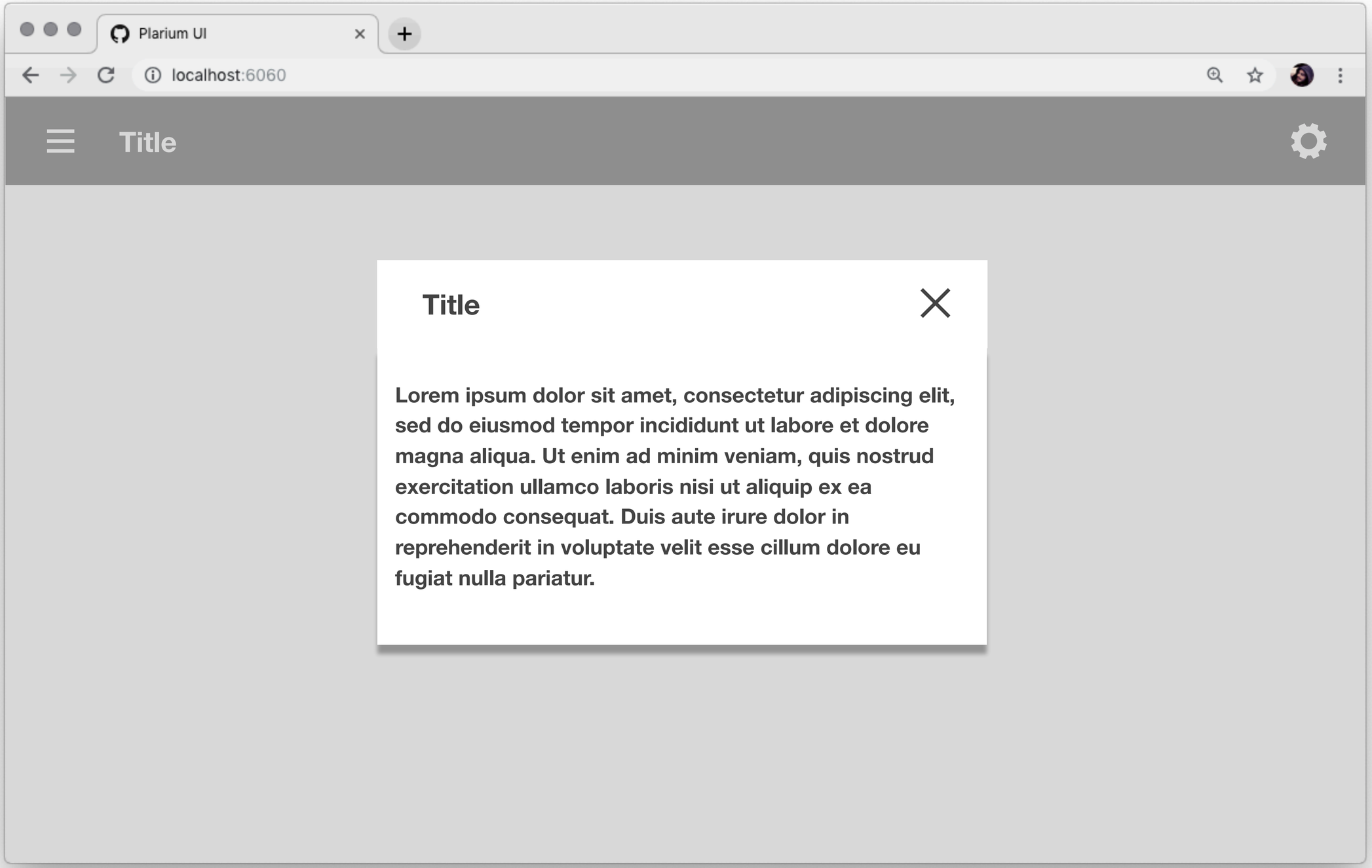
```ts
import Surface from '@plarium/ui/Surface';
import Text from '@plarium/ui/Text';
import Icon from '@plarium/ui/Icon';

function render() {
  return (
    <Surface>
      <Text>Take The World!</Text>
      <Icon>face</Icon>
    </Surface>
  );
}
```

```ts
interface IThemeColors {
  primary: string;
  onPrimary: string;

  secondary: string;
  onSecondary: string;

  accent: string;
  onAccent: string;

  surface: string;
  onSurface: string;

  error: string;
  onError: string;

  normal100: string;
  ...
  normal600: string;

  gray100: string;
  ...
  gray700: string;
}
```

```ts
import Surface from '@plarium/ui/Surface';
import Text from '@plarium/ui/Text';
import Icon from '@plarium/ui/Icon';

function render() {
  return (
    <Surface color="primary">
      <Text>Take The World!</Text>
      <Icon>face</Icon>
    </Surface>
  );
}
```

```ts
interface IThemeColors {
  primary: string;
  onPrimary: string;

  secondary: string;
  onSecondary: string;

  accent: string;
  onAccent: string;

  surface: string;
  onSurface: string;

  error: string;
  onError: string;

  normal100: string;
  ...
  normal600: string;

  gray100: string;
  ...
  gray700: string;
}
```

```ts
import Surface from '@plarium/ui/Surface';
import Text from '@plarium/ui/Text';
import Icon from '@plarium/ui/Icon';

function render() {
  return (
    <Surface color="accent">
      <Text>Error Happened</Text>
      <Icon>fire</Icon>
    </Surface>
  );
}
```


Plarium UI
localhost:6060
Title
Title
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Изменение темы

<Palette/>

```
const Button = styled.button`
  border-radius: 4px;
  border: 1px solid ${props => props.theme.colors.primary};
`;
```

KHARKIV JS

```
const Button = styled.button`
  border-radius: 4px;
  border: 1px solid ${props => props.theme.colors.primary};
`;
```

<Button>Kharkiv JS</Button>

KHARKIV JS

```
<Palette
  colors={{
    primary: 'red',
  }}
>
  <Button>Kharkiv JS</Button>
</Palette>
```

KHARKIV JS

```
<Palette
  colors={{
    primary: 'red',
  }}
>
  <Button>Kharkiv JS</Button>
</Palette>
```

#3 Layout компоненты

<Box/>

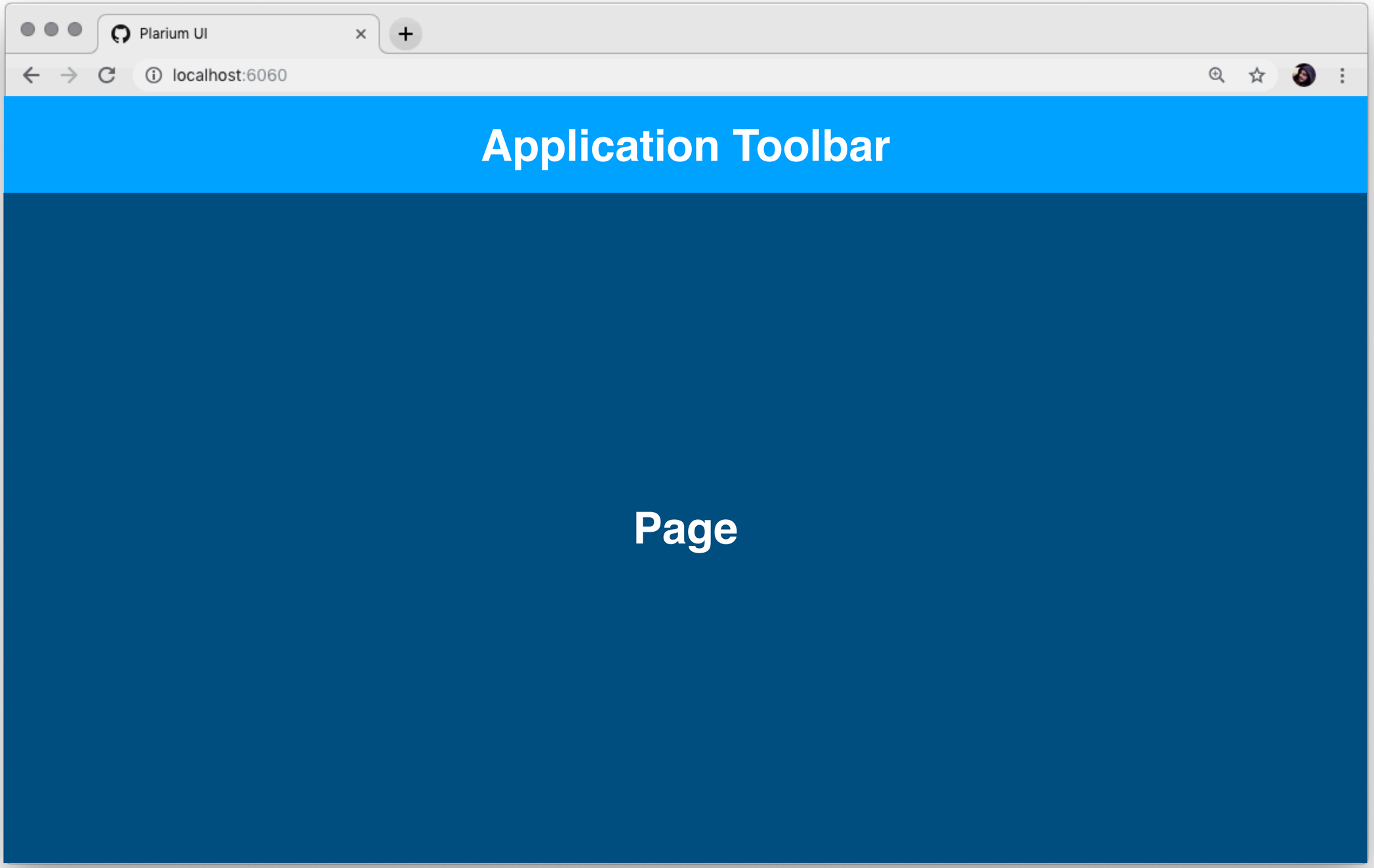
```
<Box>
  <Button>Take</Button>
  <Button>The</Button>
  <Button>World!</Button>
</Box>
```

```
<Box vertical={true}>
  <Button>Take</Button>
  <Button>The</Button>
  <Button>World!</Button>
</Box>
```

```
<Box grow={1} align="center" justify="center" {...otherFlexRelatedProps}>
  ...
</Box>
```

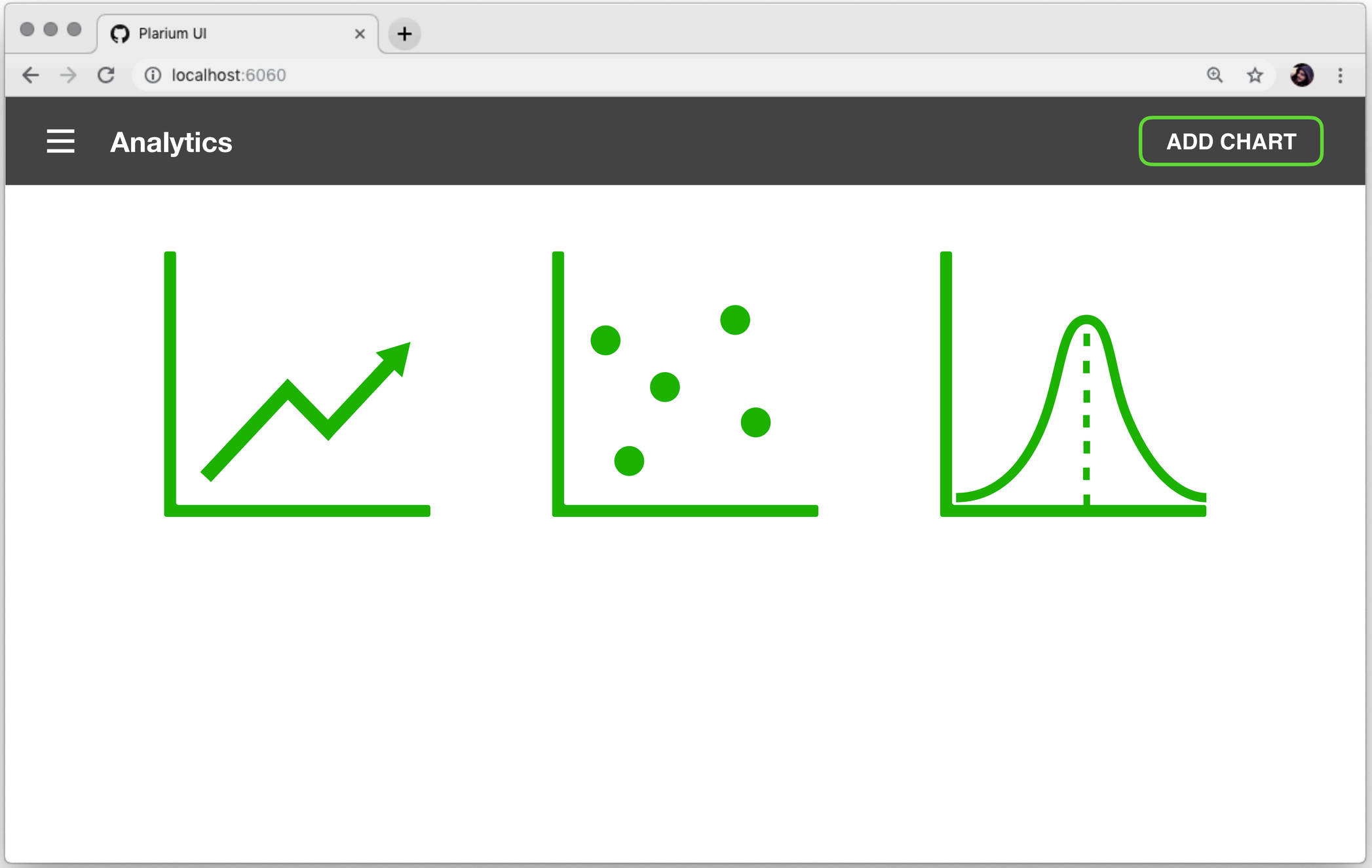
Очень легко позиционировать компоненты

#4 Порталы


Plarium UI
localhost:6060
Application Toolbar
Page

```
function render() {
  return (
    <React.Fragment>
      <ApplicationToolbar />
      <AnalyticsScreen />
    </React.Fragment>
  );
}
```

```
function render() {
  return (
    <React.Fragment>
      <ApplicationToolbar />
      <AnalyticsScreen />
    </React.Fragment>
  );
}
```


Plarium UI
localhost:6060
Analytics
ADD CHART

```
ReactDOM.createPortal(child, container)
```

```
ReactDOM.createPortal(
  <Button>Add Chart</Button>,
  container
);
```

```
ReactDOM.createPortal(
  <Button>Add Chart</Button>,
  container
);
```

DOM Element

```
ReactDOM.createPortal(
  <Button>Add Chart</Button>,
  document.getElementById('#chart-add-button')
);
```

```
ReactDOM.createPortal(
  <Button>Add Chart</Button>,
  document.getElementById('#chart-add-button')
);
```

```
import { createPortal } from '@plarium/ui/Portals';

const Portal = createPortal();
```

```
import { createPortal } from '@plarium/ui/Portals';

const Portal = createPortal();


Portal.Placeholder
Portal.Content
```

Application Toolbar

```
import { createPortal } from ‘@plarium/ui/Portals';
const ToolbarActionsPortal = createPortal();

const ApplicationToolbar = () => {
  return (
    <Toolbar>
      <ToolbarActionsPortal.Placeholder />
    </Toolbar>
  );
};

export const ToolbarActions = ToolbarActionsPortal.Content;
```

Application Toolbar

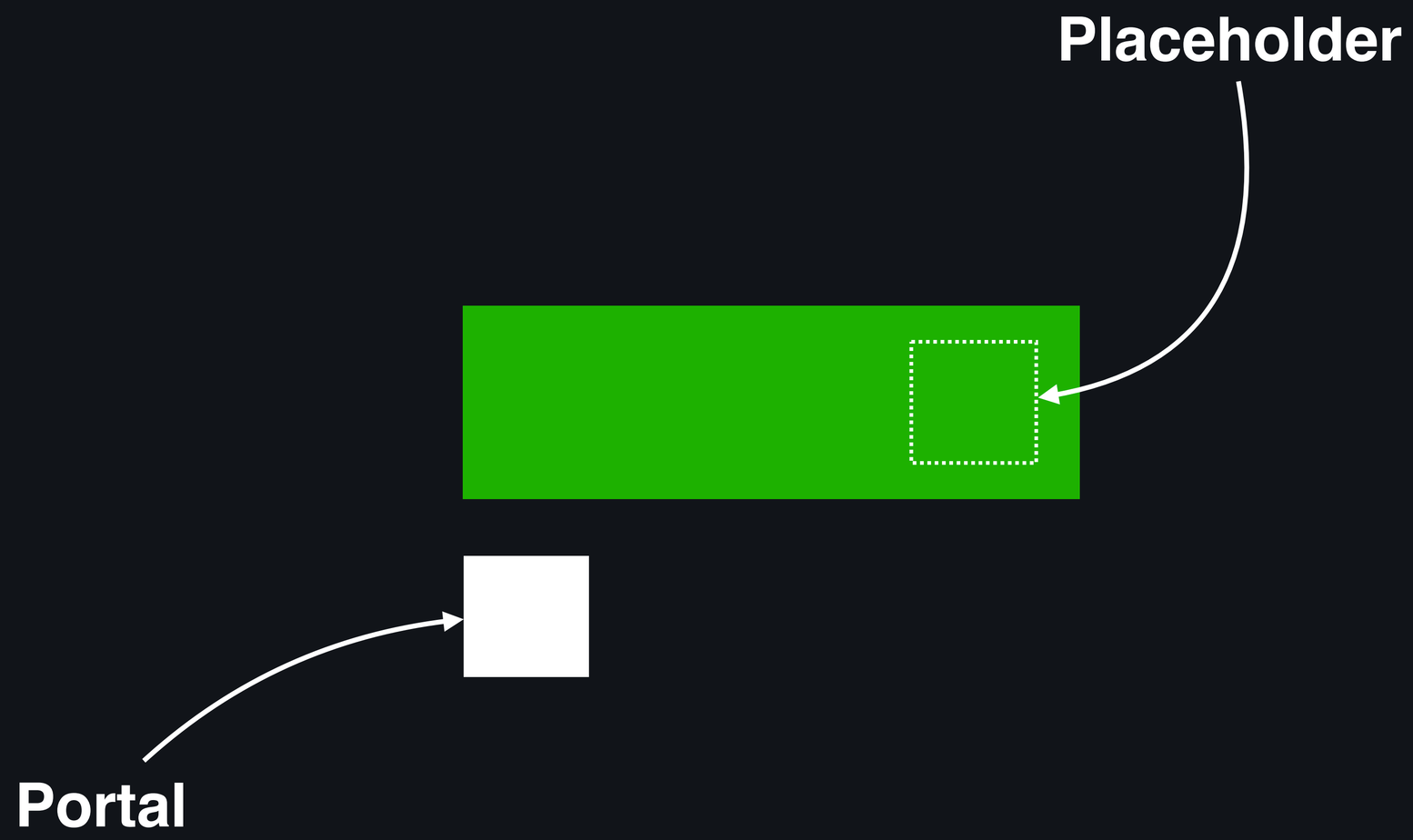
```
import { createPortal } from '@plarium/ui/Portals';
const ToolbarActionsPortal = createPortal();

const ApplicationToolbar = () => {
  return (
    <Toolbar>
      <ToolbarActionsPortal.Placeholder />
    </Toolbar>
  );
};

export const ToolbarActions = ToolbarActionsPortal.Content;
```

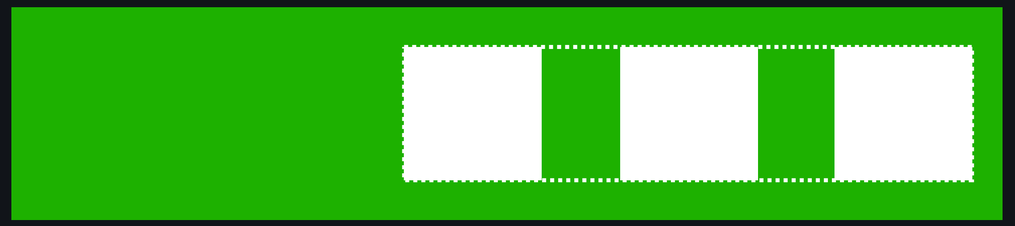
Screen

```
import { ToolbarActions } from 'toolbar';

const Screen = () => {
  return (
    <div>
      <Charts/>
      <ToolbarActions>
        <Button>Add Chart</Button>
      </ToolbarActions>
    </div>
  );
};
```

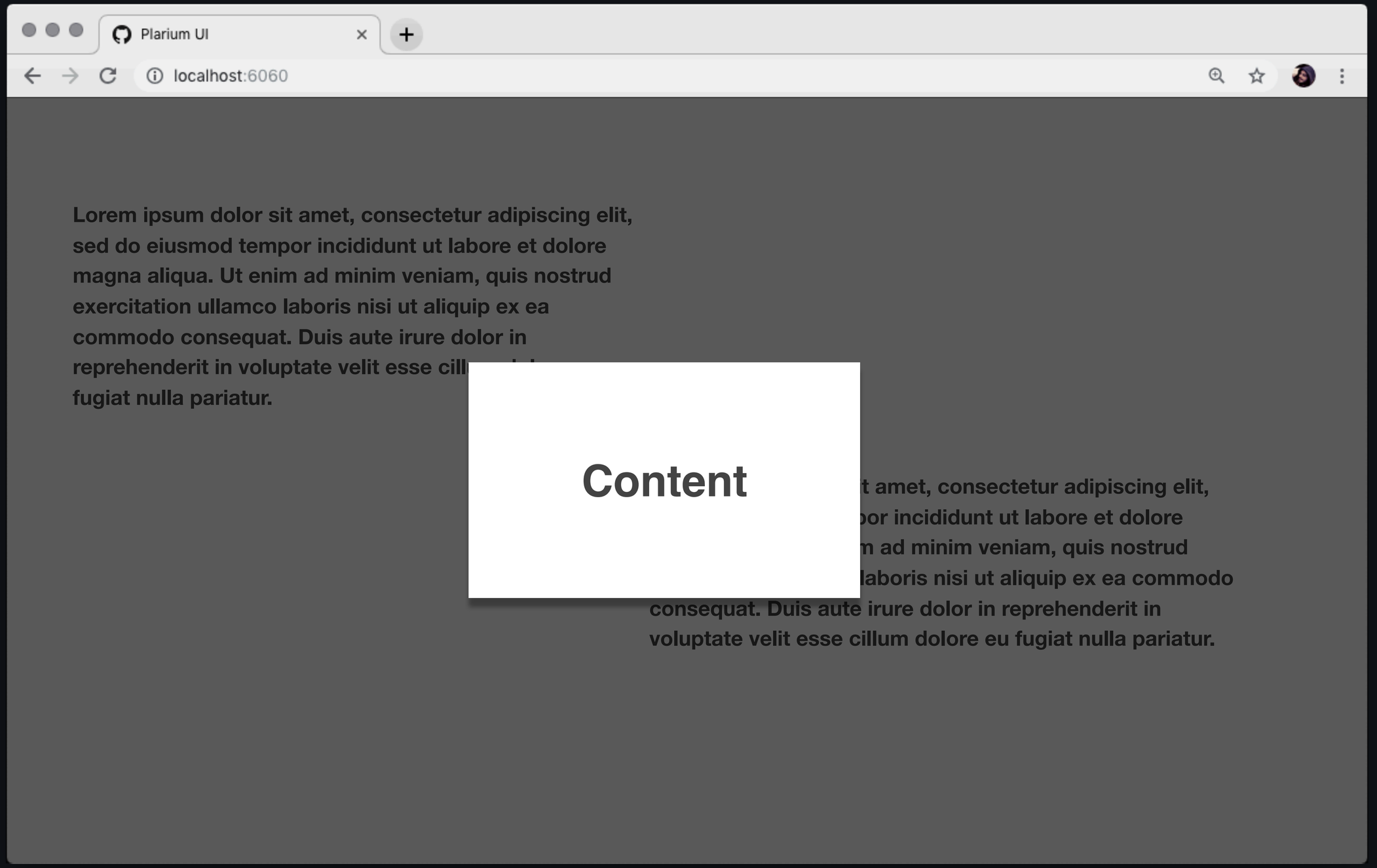
Screen

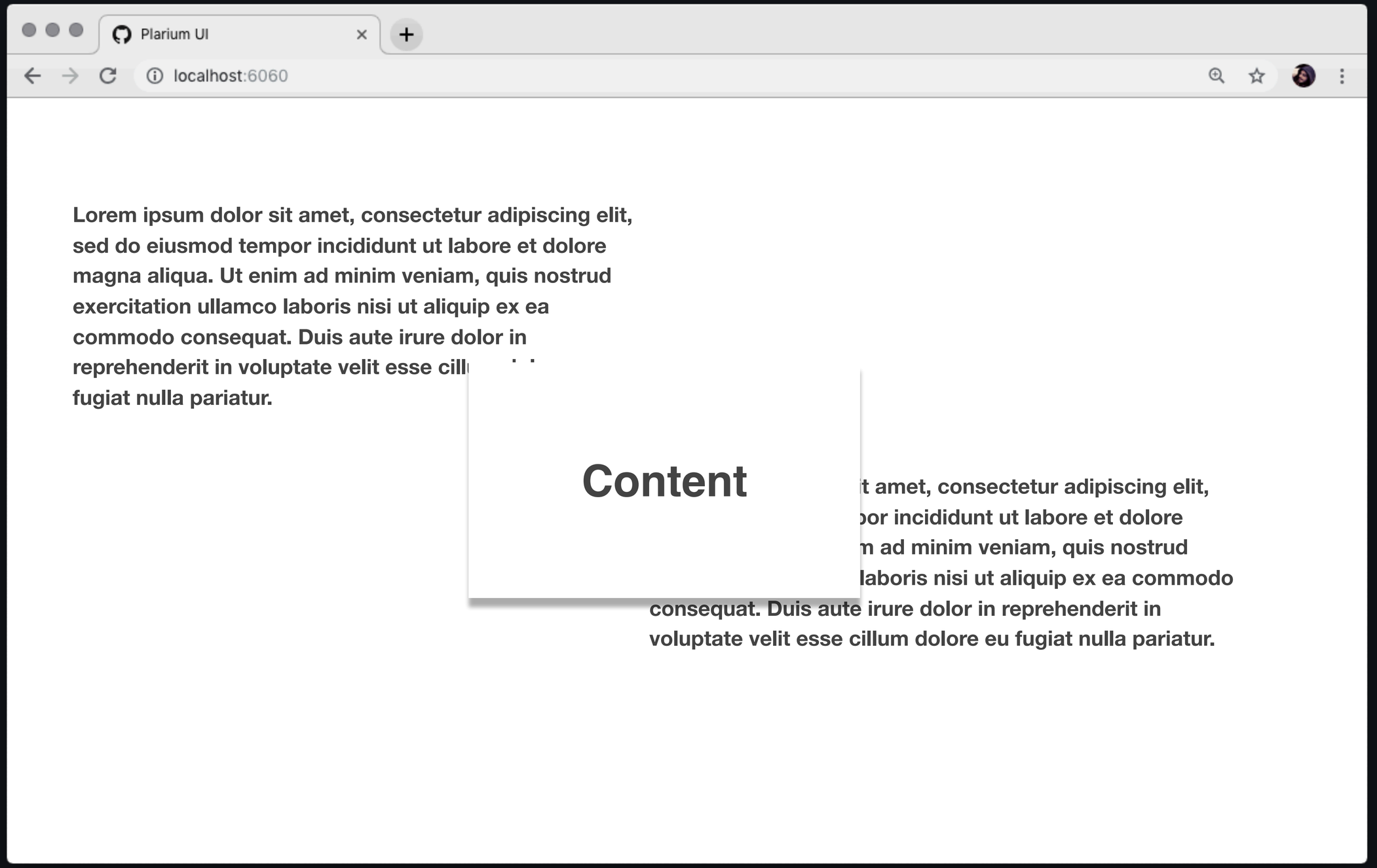
```
import { ToolbarActions } from 'toolbar';

const Screen = () => {
  return (
    <div>
      <Charts/>
      <ToolbarActions>
        <Button>Add Chart</Button>
      </ToolbarActions>
    </div>
  );
};
```

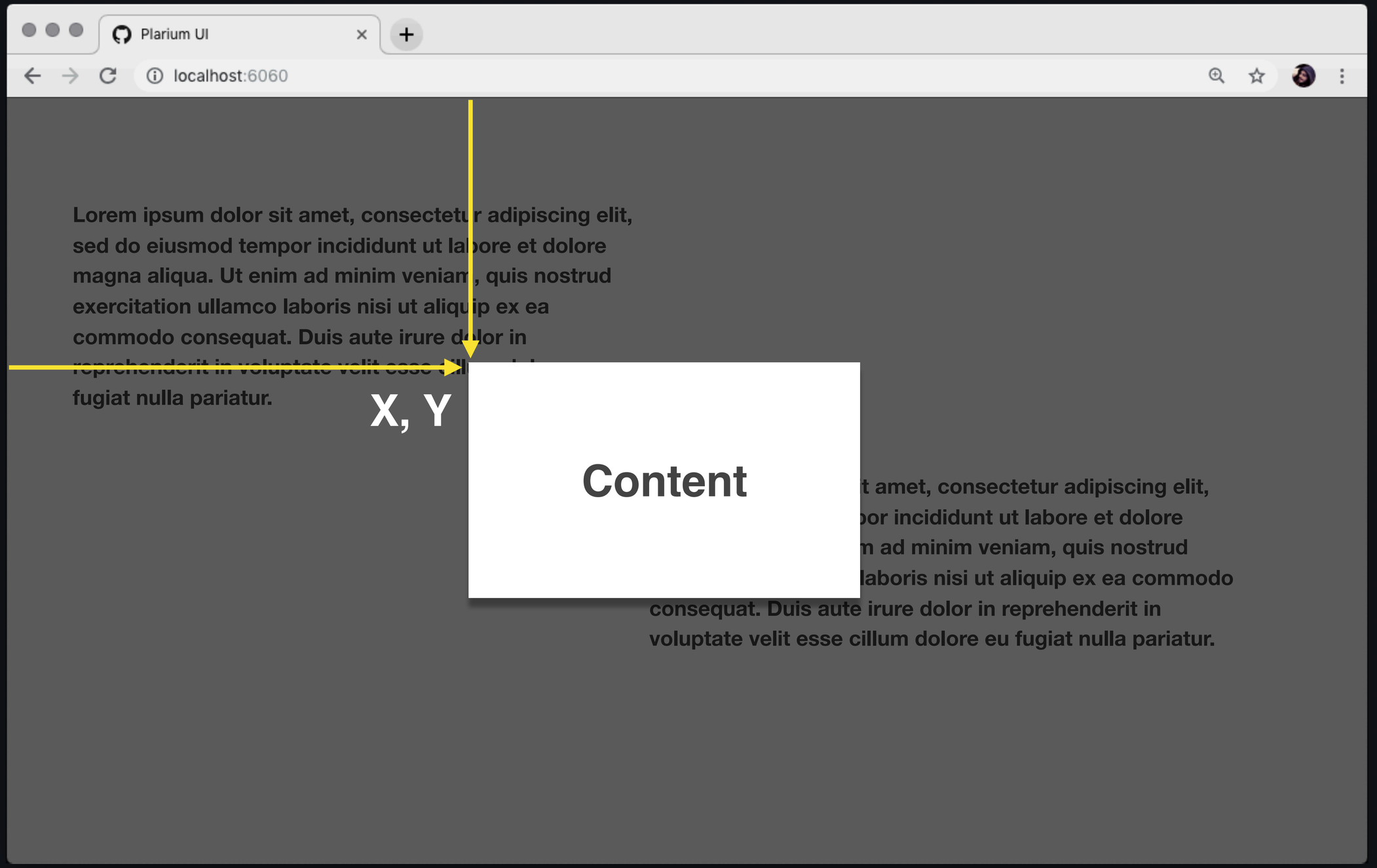

Placeholder
Portal



#5 Overlay


Plarium UI
localhost:6060
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cill
fugiat nulla pariatur.
Content
t amet, consectetur adipiscing elit,
or incididunt ut labore et dolore
m ad minim veniam, quis nostrud
laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.


Plarium UI
localhost:6060
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cill
fugiat nulla pariatur.
Content
t amet, consectetur adipiscing elit,
or incididunt ut labore et dolore
m ad minim veniam, quis nostrud
laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.


Plarium UI
localhost:6060
Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur.
X, Y
Content
t amet, consectetur adipiscing elit,
or incididunt ut labore et dolore
m ad minim veniam, quis nostrud
laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
import Overlay from ‘@plarium/ui/Overlay';

function render() {
  return (
    <Overlay>{children}</Overlay>
  );
}
```

```
import Overlay from '@plarium/Overlay';

function render() {
  return (
    <Overlay position={handlePosition}>{children}</Overlay>
  );
}

function handlePosition(content: HTMLElement) {
  return {
    top: 0,
    left: 0,
  }
}
```

```
import Overlay from '@plarium/Overlay';

function render() {
  return (
    <Overlay position={handlePosition}>{children}</Overlay>
  );
}

function handlePosition(content: HTMLElement) {
  return {
    top: 0,
    left: 0,
  }
}
```

<Select/>

<MultiSelect/>

<Dropdown/>

<Menu/>

<Dialog/>

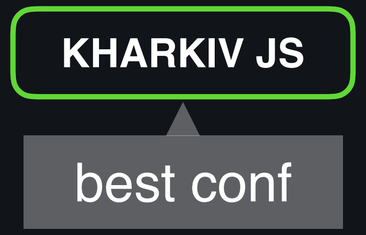
<SidePage/>

<Snackbar/>

<Tooltip/>

<DatePicker/>

#6 Dropdown II
Popover




```
<Tooltip text=“best conf”>
  <Button>Kharkiv JS</Button>
</Tooltip>
```

KHARKIV JS

```
<Popover content={<div>{content}</div>}>
  <Button>Kharkiv JS</Button>
</Popover>
```

KHARKIV JS

```
<Popover content={<div>{content}</div>}>
  <Button>Kharkiv JS</Button>
</Popover>
```

KHARKIV JS

```
<Popover
  content={
    <div>
      <Box>
        <Box vertical={true}>
          <Text variant="legend">Assign responsibility</Text>
          <Checkbox label="Gilad Gray" />
          <Checkbox label="Jason Killnan" />
          <Checkbox label="Antonie Llorca" />
          <Text variant="overline">Be careful</Text>
        </Box>
        <Box vertical={true}>
          <Text variant="legend">Pick two</Text>
          <Checkbox label="Gilad Gray" />
          <Checkbox label="Jason Killnan" />
          <Checkbox label="Antonie Llorca" />
          <Text variant="overline">You can display an error</Text>
        </Box>
      </Box>
      ;
    </div>
  }
>
  <Button>Kharkiv JS</Button>
</Popover>
```

☹️

KHARKIV JS

☹️

```
class PopoverButton extends React.Component {
  ref = React.createRef();

  render() {
    return (
      <React.Fragment>
        <Button ref={this.ref}>Kharkiv JS</Button>
        <Popover anchor={this.ref.current}>{content}</Popover>
      </React.Fragment>
    );
  }
}
```

Усложняем код

```
import { createDropdown } from '@plarium/ui/createDropdown';

const Dropdown = createDropdown();

Dropdown.Trigger
Dropdown.Content
```

```
import { createDropdown } from '@plarium/ui/createDropdown';

const Dropdown = createDropdown();

function render() {
  return (
    <React.Fragment>
      <Dropdown.Trigger>
        {props => <Button onClick={props.handleOpen}>Open</Button>}
      </Dropdown.Trigger>
      <Dropdown.Content>{() => <div>{content}</div>}</Dropdown.Content>
    </React.Fragment>
  );
}
```

```
import { createDropdown } from '@plarium/ui/createDropdown';

const Dropdown = createDropdown();

function render() {
  return (
    <React.Fragment>
      <Dropdown.Trigger>
        {props => <Button onClick={props.handleOpen}>Open</Button>}
      </Dropdown.Trigger>
      <Dropdown.Content>{() => <div>{content}</div>}</Dropdown.Content>
    </React.Fragment>
  );
}
```

```
import { createDropdown } from '@plarium/ui/createDropdown';

const Dropdown = createDropdown();

function render() {
  return (
    <React.Fragment>
      <Dropdown.Trigger>
        {props => <Button onClick={props.handleOpen}>Open</Button>}
      </Dropdown.Trigger>
      <Dropdown.Content>{() => <div>{content}</div>}</Dropdown.Content>
    </React.Fragment>
  );
}
```

#7 Виртуализация

Рендерим только видимую часть

<Grid/> <Select/> <List/>

react-virtualized

```
import { MultiGrid } from 'react-virtualized';

function render() {
  return (
    <MultiGrid
      cellRenderer={cellRenderer}
      columnCount={headers.length}
      fixedColumnCount={1}
      fixedRowCount={1}
      columnWidth={100}
      height={300}
      rowHeight={40}
      rowCount={100}
      width={width}
    />
  );
}

function cellRenderer({ rowIndex, columnIndex }) {}
```

```
import { MultiGrid } from 'react-virtualized';

function render() {
  return (
    <MultiGrid
      cellRenderer={cellRenderer}
      columnCount={headers.length}
      fixedColumnCount={1}
      fixedRowCount={1}
      columnWidth={100}
      height={300}
      rowHeight={40}
      rowCount={100}
      width={width}
    />
  );
}

function cellRenderer({ rowIndex, columnIndex }) {}
```

```
      height={300}
      rowHeight={40}
      rowCount={100}
      width={width}
    />
  );
}

function cellRenderer({ rowIndex, columnIndex }) {
  const column = data[rowIndex][columnIndex];

  switch (column.col) {
    case 'avatar':
      return <Avatar src={column.avatar} />;

    default:
      return <Text>{column.toString()}</Text>;
  }
}
```

```
import { MultiGrid } from 'react-virtualized';

function render() {
  return (
    <MultiGrid
      cellRenderer={cellRenderer}
      columnCount={headers.length}
      fixedColumnCount={1}
      fixedRowCount={1}
      columnWidth={100}
      height={300}
      rowHeight={40}
      rowCount={100}
      width={width}
    />
  );
}

function cellRenderer({ rowIndex, columnIndex }) {}
```

```
import { MultiGrid } from 'react-virtualized';

function render() {
  return (
    <MultiGrid
      cellRenderer={cellRenderer}
      columnCount={headers.length}
      fixedColumnCount={1}
      fixedRowCount={1}
      columnWidth={getColumnHeight}
      height={300}
      rowHeight={getRowHeight}
      rowCount={100}
      width={width}
    />
  );
}

function cellRenderer({ rowIndex, columnIndex }) {}
```

Мешаем бизнес логику с инфраструктурой

```
import Grid from '@plarium/Grid';

function render() {
  return (
    <Grid data={data}>
      <Grid.Column width={38} when={props => props.col === 'avatar'}>
        {cell => <Avatar src={cell.avatar} />}
      </Grid.Column>
    </Grid>
  );
}
```

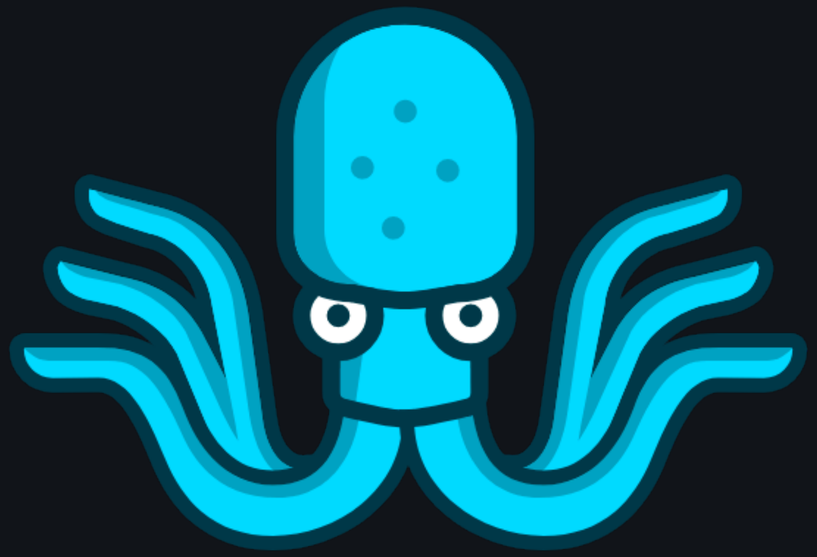
```
import Grid from '@plarium/Grid';

function render() {
  return (
    <Grid data={data}>
      <Grid.Column width={38} when={props => props.col === 'avatar'}>
        {cell => <Avatar src={cell.avatar} />}
      </Grid.Column>
    </Grid>
  );
}
```

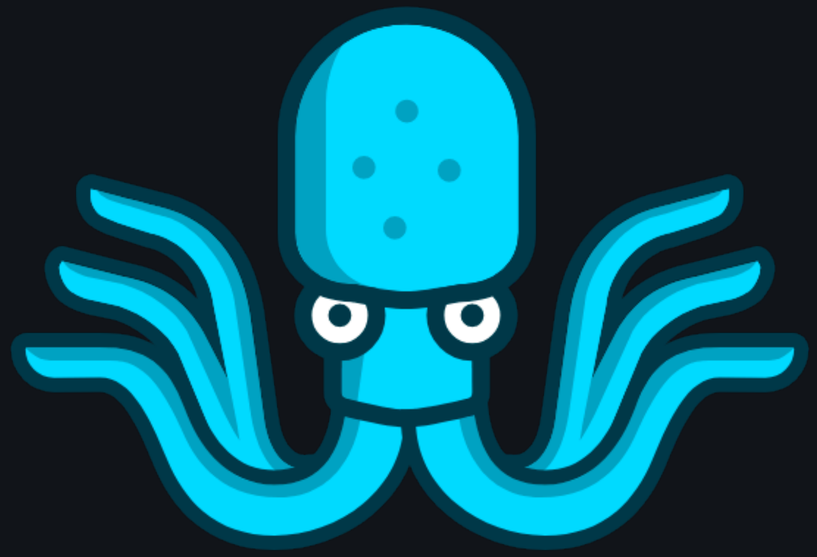
```
import Grid from '@plarium/Grid';

function render() {
  return (
    <Grid data={data}>
      <Grid.Column width={38} when={props => props.col === 'avatar'}>
        {cell => <Avatar src={cell.avatar} />}
      </Grid.Column>
    </Grid>
  );
}
```

```
import Grid from '@plarium/Grid';

function render() {
  return (
    <Grid data={data}>
      <Grid.Column width={38} when={props => props.col === 'avatar'}>
        {cell => <Avatar src={cell.avatar} />}
      </Grid.Column>
    </Grid>
  );
}
```

Превью компонентов

STORYBOOK





Button

```js static
import {
  OutlinedButton,
  ContainedButton,
  TextButton,
} from '@plarium/material-ui/Button';
```

```js
<Box>
  <Button>Take</Button>
  <Button variant="outlined">The</Button>
  <Button variant="contained">World!</Button>
  <Button variant="contained" icon="face">
    Face
  </Button>
</Box>
```

README.md

Button variants could be imported separately.

```
import {
  OutlinedButton,
  ContainedButton,
  TextButton,
} from '@plarium/material-ui/Button';
```

```
<Box>
  <Button>Take</Button>
  <Button variant="outlined">The</Button>
  <Button variant="contained">World!</Button>
  <Button variant="contained" icon="face">
    Face
  </Button>
</Box>
```

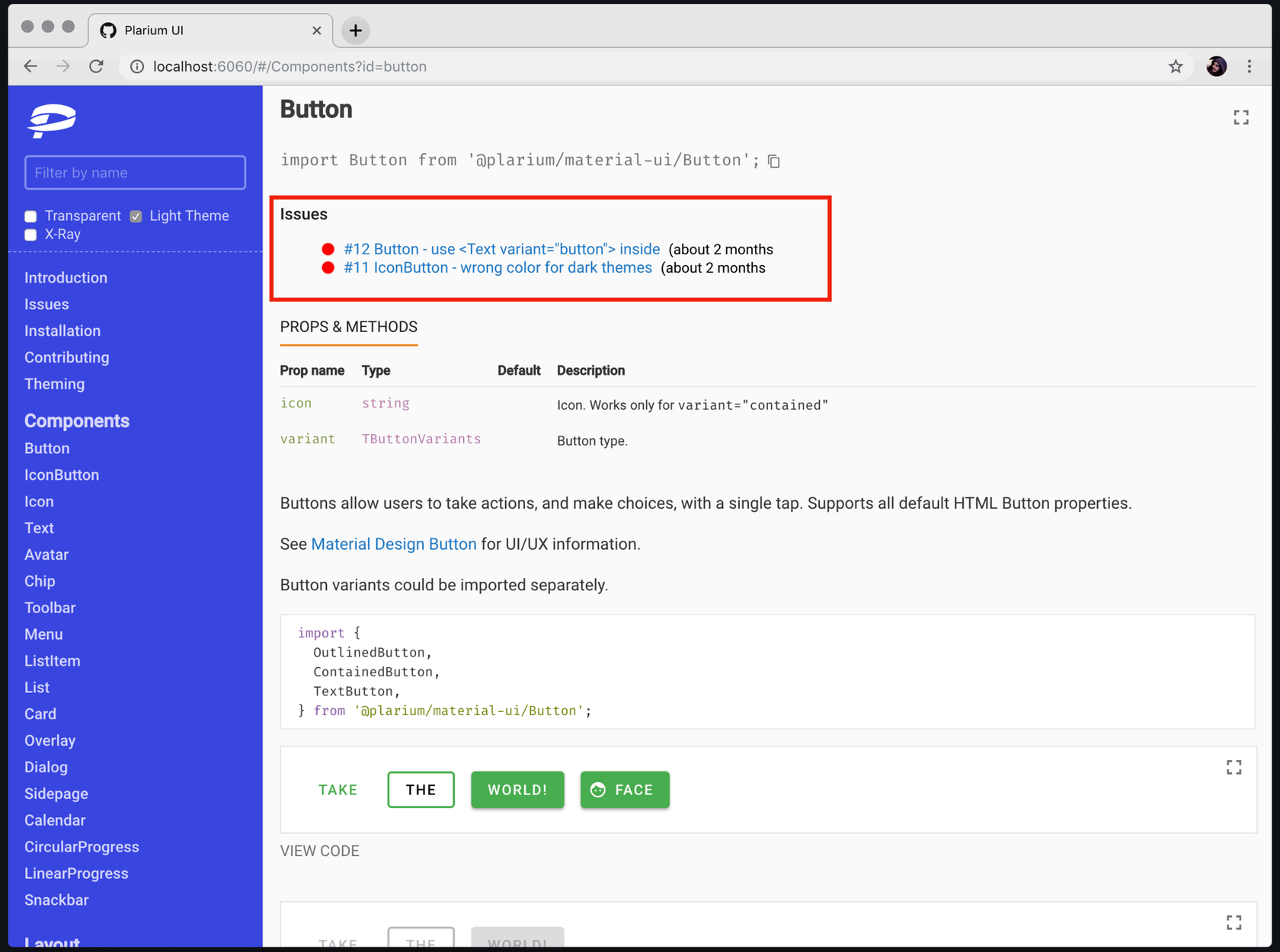
```
<Button disabled={true}>Take</Button>
<Button variant="outlined" disabled={true}>The</Button>
<Button variant="contained" disabled={true}>World!</Button>
```

Button variants could be imported separately.

```
import {
  OutlinedButton,
  ContainedButton,
  TextButton,
} from '@plarium/material-ui/Button';
```

VIEW CODE

```
<Box>
  <Button>Take</Button>
  <Button variant="outlined">The</Button>
  <Button variant="contained">World!</Button>
  <Button variant="contained" icon="face">
    Face
  </Button>
</Box>
```


Plarium UI
localhost:6060/#/Components?id=button
Filter by name
Transparent
Light Theme
X-Ray
Introduction
Issues
Installation
Contributing
Theming
Components
Button
IconButton
Icon
Text
Avatar
Chip
Toolbar
Menu
ListItem
List
Card
Overlay
Dialog
Sidepage
Calendar
CircularProgress
LinearProgress
Snackbar
Button
import Button from '@plarium/material-ui/Button';
Issues
#12 Button - use <Text variant="button"> inside (about 2 months
#11 IconButton - wrong color for dark themes (about 2 months
PROPS & METHODS
Prop name Type Default Description
icon string Icon. Works only for variant="contained"
variant TButtonVariants Button type.
Buttons allow users to take actions, and make choices, with a single tap. Supports all default HTML Button properties.
See Material Design Button for UI/UX information.
Button variants could be imported separately.
import {
OutlinedButton,
ContainedButton,
TextButton,
} from '@plarium/material-ui/Button';
TAKE
THE
WORLD!
FACE
VIEW CODE

THANK YOU

Copyright Plarium Global LTD. 2018
Do not distribute

TAKE THE WORLD