

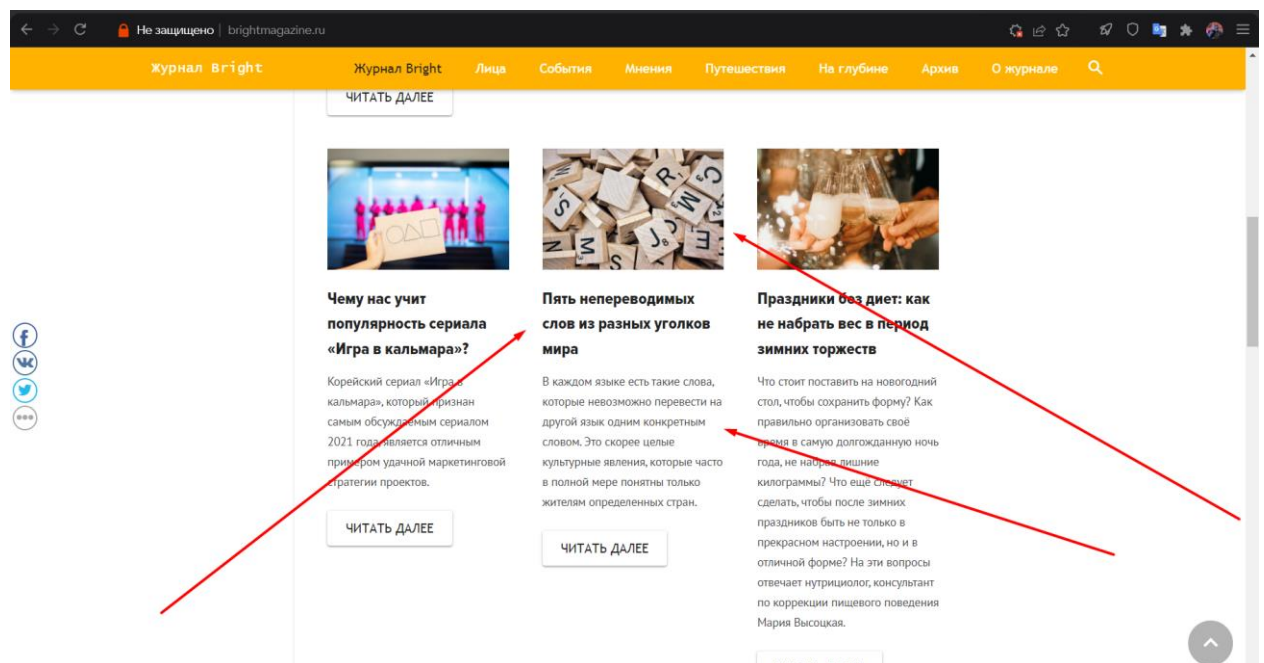
Отчёт по проделанной работе на семинаре по мобильной разработке (25.03.2022).

(выполнил Валяев Георгий Анатольевич)

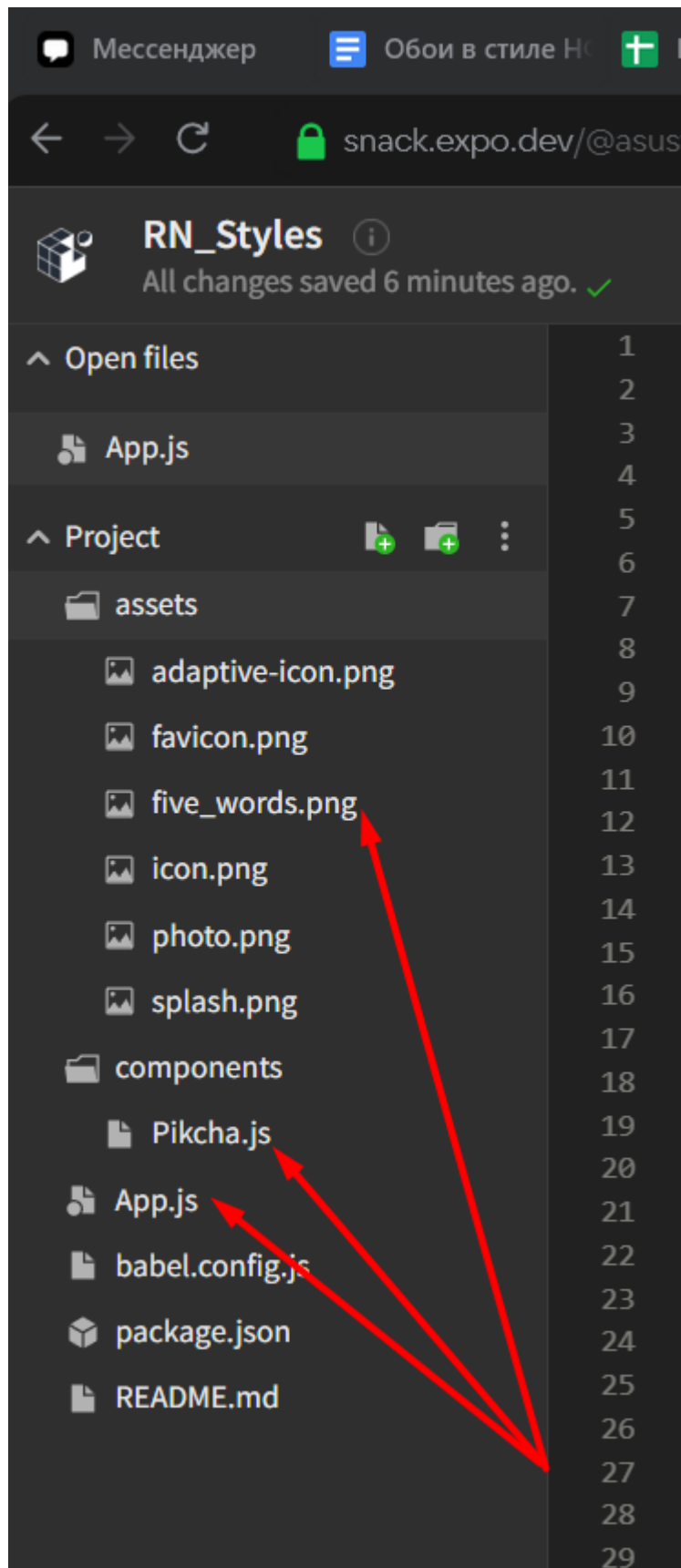
React-native – один из интересных и богатых фреймворков, наследующих большинство свойств фреймворка React (в рамках разработки нативных веб-приложений), разработанный командой Facebook для создания и написания приложений под мобильные устройства с поддержкой ОС Android, iOS и других систем.

В данной методичке мы научимся основам работы со стилизацией мобильных приложений, разработанных в React-Native.

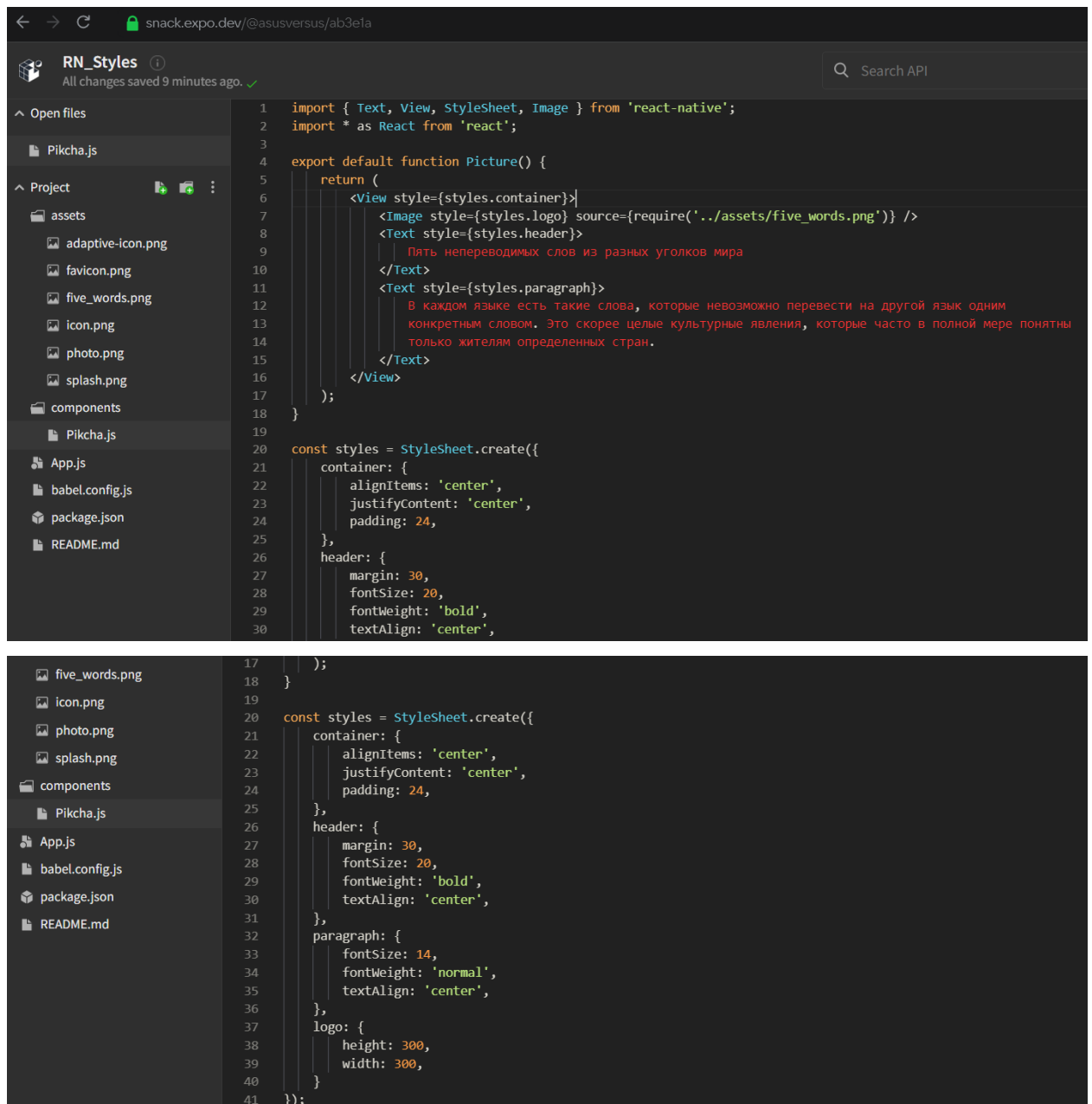
Начнём подгрузки медиа-файлов для отображения картинки и текста с сайта (<http://brightmagazine.ru/>):



После подгрузки получаем такую файловую структуру приложения:
(стрелками отмечены файлы, с которыми мы и работаем)

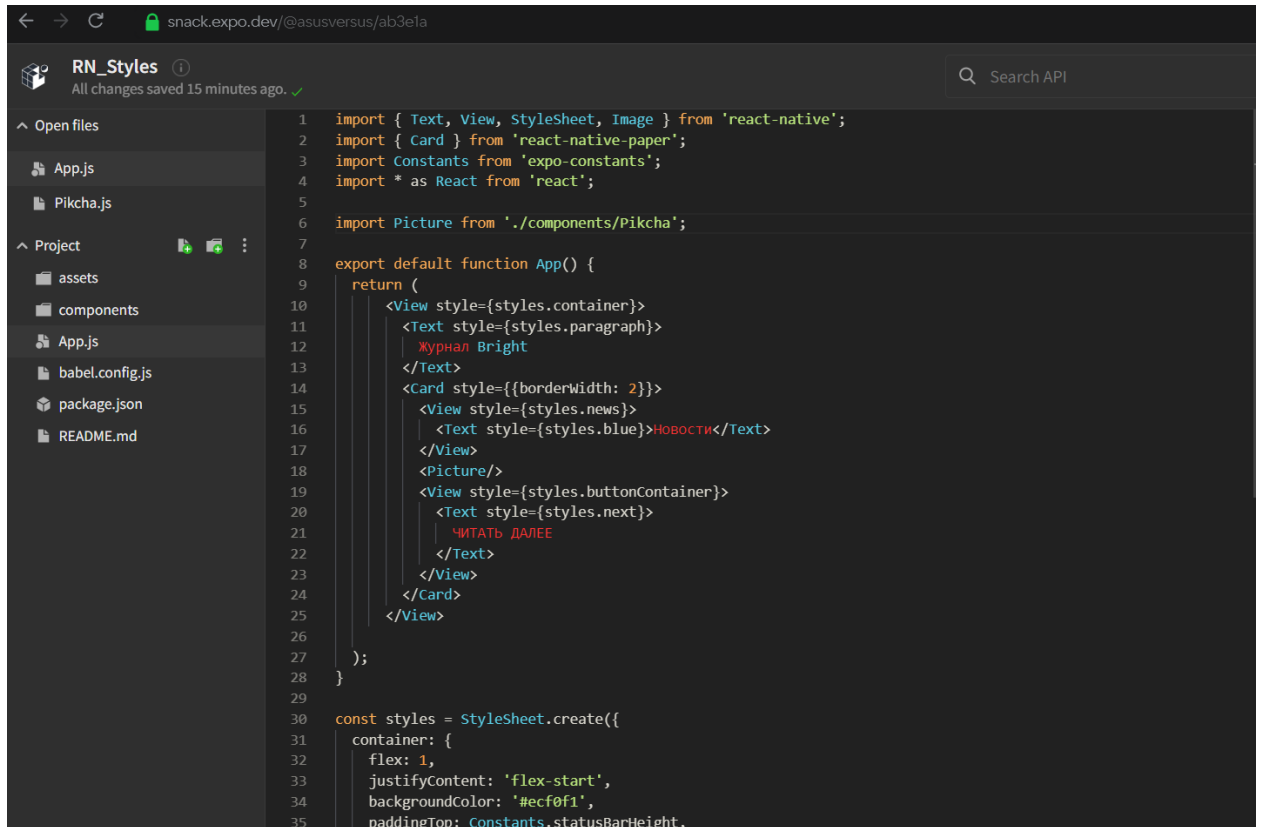


Начнём с самого «детского» компонента `Pikcha.js`, отвечающего за структуры подгруженной картинки и текста с сайта журнала:

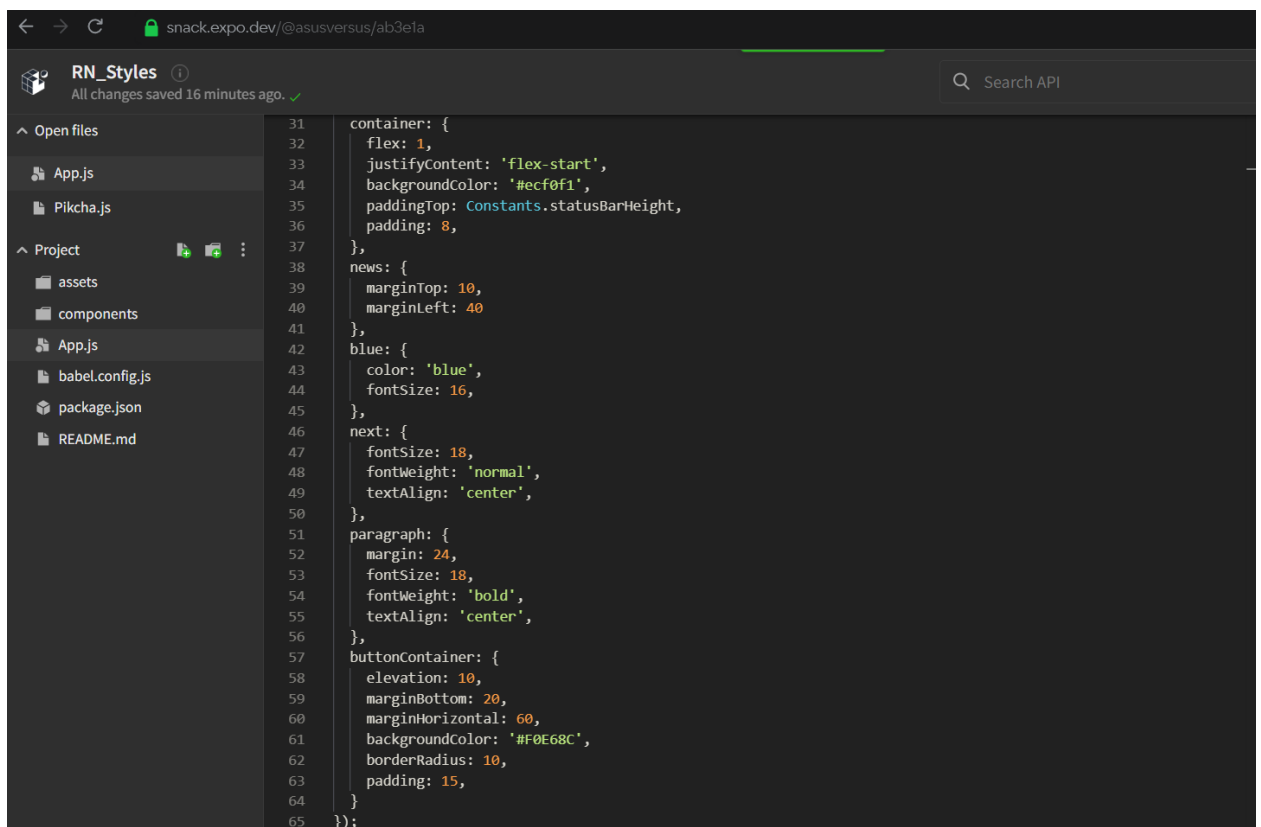


В большом объекте View, отвечающем за центральное расположение структуры и отступы от краёв экрана приложения, мы задаём объекты Image и Text. В первом мы указываем стилизацию и расположение файла в виртуальной файловой структуре, а во втором разделяем текстовое содержимое на заголовок и наполнение страницы, придавая каждому свои стили (например, для заголовка мы создаём внешний отступ объекта, жирно выделенный шрифт, центральное расположение и 20 размер шрифта текста).

Плавнo перейдём к описанию главного «родительского» компонента приложения – App.js:



```
1 import { Text, View, StyleSheet, Image } from 'react-native';
2 import { Card } from 'react-native-paper';
3 import Constants from 'expo-constants';
4 import * as React from 'react';
5
6 import Picture from './components/Pikcha';
7
8 export default function App() {
9   return (
10     <View style={styles.container}>
11       <Text style={styles.paragraph}>
12         Жyрнал Bright
13       </Text>
14       <Card style={{borderWidth: 2}}>
15         <View style={styles.news}>
16           <Text style={styles.blue}>НОВОСТИ</Text>
17         </View>
18         <Picture/>
19         <View style={styles.buttonContainer}>
20           <Text style={styles.next}>
21             ЧИТАТЬ ДАЛЕЕ
22           </Text>
23         </View>
24       </Card>
25     </View>
26   );
27 }
28
29
30 const styles = StyleSheet.create({
31   container: {
32     flex: 1,
33     justifyContent: 'flex-start',
34     backgroundColor: '#ecf0f1',
35     paddingTop: Constants.statusBarHeight,
```

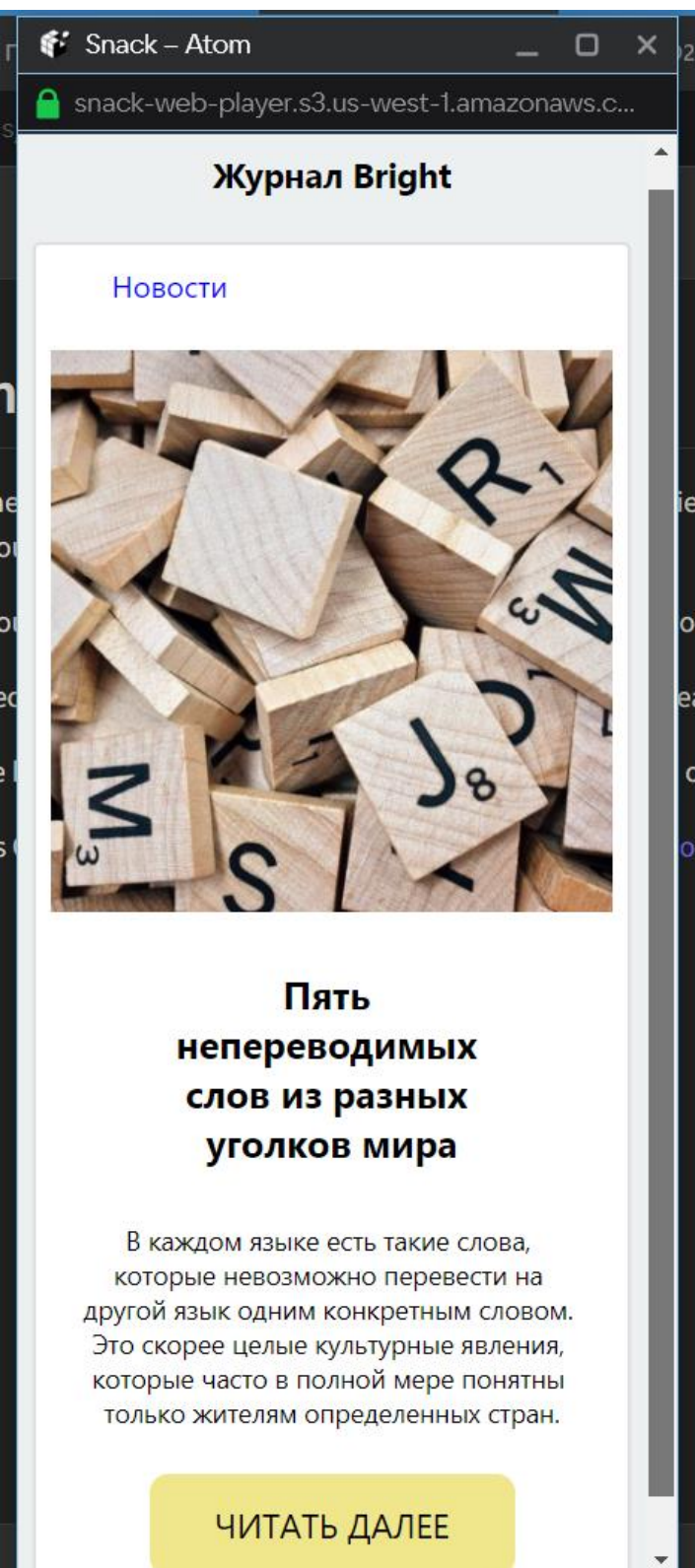


```
36     padding: 8,
37   },
38   news: {
39     marginTop: 10,
40     marginLeft: 40
41   },
42   blue: {
43     color: 'blue',
44     fontSize: 16,
45   },
46   next: {
47     fontSize: 18,
48     fontWeight: 'normal',
49     textAlign: 'center',
50   },
51   paragraph: {
52     margin: 24,
53     fontSize: 18,
54     fontWeight: 'bold',
55     textAlign: 'center',
56   },
57   buttonContainer: {
58     elevation: 10,
59     marginBottom: 20,
60     marginHorizontal: 60,
61     backgroundColor: '#F0E68C',
62     borderRadius: 10,
63     padding: 15,
64   }
65 });
```

В данном компоненте мы создаём большой контейнер-объект View, в котором располагаем в объекте View основной заголовок страницы (Журнал Bright), карточку (объект Card). В неё уже мы закладываем меньший заголовок с тематикой «Новости» и весь компонент, который создали ранее (Pikcha.js). А также простой вариант кнопки с текстом «ЧИТАТЬ ДАЛЕЕ» в виде объекта View со стилизацией под кнопку.

Прорабатывая стили, мы закладываем главные свойства для всех элементов экрана приложения: расположение по центру экрана (textAlign,), внешние и внутренние отступы относительно самого объекта (margin & padding), цвет фона (backgroundColor) и переработка отображения текста как более жирного варианта (fontWeight), тени кнопки и закругления по краям объекта View (elevation & borderRadius).

Таким образом, мы получаем следующий результат в веб-версии приложения и в мобильном варианте на собственном телефоне:



Страница журнала Bright в мобильном варианте реализована успешно.

Контрольные вопросы

1. Приведите аналоги тегов html в React Native:

- `<div>`
- `<p>`
- ``
- `<button>`

1. Теги, имеющие похожий функционал, но работающие в веб-разработке и в качестве объектов React-Native: `<div>` – `<View>`, `<p>` – `<Text>`, `` – `<Image>`, `<button>` – `<Button>`