Методические указания для подготовки к экзамену

Форма проведения экзамена

Экзамен по дисциплине “Технология разработки приложений для мобильных устройств” проводится в письменной форме и призван проверить сформированные у студентов компетенции в части проектирования и использования мобильных приложений в экономике и финансах.

Экзаменационный билет состоит из двух теоретических и одного практического вопроса. Каждый вопрос оценивается в 20 баллов. При ответе на теоретический вопрос студент должен развернуто изложить основные сведения по теме вопроса, показать глубину теоретической подготовки и готовность применять полученные знания на практике.

Ответ на практический вопрос должен содержать подробное пояснение релевантного участка кода (либо данного в задании, либо сгенерированного студентом самостоятельно), сам программный код в виде фрагмента, а также необходимые комментарии, показывающие, что данный код действительно решает задачу, приведенную в вопросе.

Теоретические вопросы для подготовки к экзамену

При ответе на теоретические вопросы студент должен самостоятельно сформулировать краткое сообщение по теме, обозначенной в вопросе. В своем ответе необходимо раскрыть основные аспекты заданного вопроса, однако слишком углубляться в детали не следует.

Список теоретических вопросов на экзамене:

1. React Native - общая характеристика, применение, аналоги.
2. Для чего используется React Native?
3. Текущие стандарты JavaScript: ECMAScript2015 (он же ES6), ECMAScript2019 (он же ES10)
4. Расширенный набор JavaScript, который добавляет строгую типизацию.
5. React Native библиотеки: React-Router, Redux, Redux-Thunk, Redux-Saga, GraphQL.
6. Менеджеры пакетов: NMP, Yarn.

7. Инструменты, помогающие поддерживать стиль кода: ESLint, TSLint, Prettier.
8. Виды и общая характеристика компонентов в React Native. Отличия, область применения.
9. Повторное использование кода в приложении на react Native.
10. JSX в React Native. Общая характеристика и примеры использования.
11. Установка React Native и создание каркаса приложения.
12. Redux - общая характеристика, назначение, примеры использования.
13. Состояние приложения. Назначение, примеры использования.
14. React Native как инструмент создания общей кодовой базы для Android и iOS.
15. SetState — механизм использования, примеры.
16. Кросс-платформенная разработка приложений.
17. Redux в React Native. Основные компоненты Redux в приложении React Native.
18. UI-библиотека React Native — назначение, преимущества.
19. Особенности разработки приложений React Native с использованием IDE.
20. Свойства (пропсы) в приложении React Native. Назначение, примеры.
21. Компонент Scrollview. Назначение, примеры.
22. Функциональные и классовые компоненты в приложении React Native.
23. Методы жизненного цикла компонентов в React Native.
24. Управление состоянием в приложении React Native.
25. Redux store. Назначение, применение, примеры использования.
26. Практическое использование методов shouldComponenUpdate().
27. Сравнительная характеристика компонентов Scrollview и FlatList.
28. Хку в приложении React Native. Общий механизм работы.
29. Функция onPress в React Native View.
30. Передача информации между экранами в React Native?
31. Маршрутизация в приложениях React Native.
32. Универсальные компоненты React Native.
33. Платформенно-специализированные компоненты React Native.
34. Сравнительная характеристика props и state в React Native.
35. Способы задания стилей в React Native.
36. Навигация в приложениях React Native?
37. “Умные” и “глупые” компоненты в React Native. Сравнительная характеристика.
38. Обработка пользовательского ввода в React Native?
39. Основные компоненты в React Native и аналогичные html-объекты.

40. Компонент ListView и его использование в React Native.
41. Асинхронное хранилище в React Native — общее назначение, области применения.
42. Хранение конфиденциальных можно достичь данных в React.
43. Анимация в приложении React Native. Общий механизм, обеспечение плавности.
44. Гибкие макеты в React Native. Принципы создания, примеры.
45. Синхронные и асинхронные действия в Redux.
46. Линковка пользовательских шрифтов в React Native.
47. Задание стиля текста с помощью свойств в React Native.
48. Отображение контента React Native поверх Status Bar и System Bar.
49. Свойства стилей React Native, отвечающие за расположение объекта.
50. Задание относительных размеров объектов в React Native.
51. Компиляция кода React Native в платформенно-специфичный код.
52. Организация файловой системы в React Native.
53. Работу с БД в React Native.
54. Загрузка данных с сервера в React Native.
55. Component Update. Общий механизм работы.
56. Библиотеки обработки жестов, совместимые с React Native.

Примеры практических заданий на экзамене

1. Создайте функциональный компонент счётчик и поменяйте его состояние через хук (подсчет нажатий на кнопку).

2. Создайте собственный компонент в отдельном js-файле, вызовите его в основном App и передайте ему значения через props.

3. Что появится на экране при реализации фрагмента синтаксиса:

```
<View
  style={{
    marginTop: 30,
    flexDirection: 'row',
    flexWrap: 'wrap',
    justifyContent: 'space-between',
  }} >
  {categories.map((cat, idx) => (
    <View
      key={`categories ${idx}`}
```

```
        style={{
          width: '30%',
          marginBottom: 20,
        }} >
        <TouchableHighlight
          underlayColor={COLORS.secondary}
          style={{
            height: 100,
            justifyContent: 'center',
            borderRadius: SIZES.radius,
            paddingLeft: 5,
            paddingRight: 5,
            backgroundColor: COLORS.gray,
          }} >
```

4. Исправьте ошибки в коде:

```
import React from 'react'
import { View } from 'react-native'
import { COLORS } from './constants'
import Header from './src/components/Header'
import Popular from './src/components/Popular'

export default function App() {
  return (
    <View
      style={{
        Padding: 24,
        paddingTop: 55,
        paddingBottom: 75,
        BackgroundColor: COLORS.black,
      }}
    >
      <Header />
      <Categories />
      <Popul />
  )
}
```

5. Создайте кнопку и задайте ей анимацию с помощью компонента Animate