Let's Get Started, Coder!!
Fill the following Document

1. Which one of the following is an Imperative Language?

1. HTML
2. CSS
3. Java Script

Answer:

Imperative programming is a software development approach in computer science that employs statements to modify the state of a program. Functions are implicitly coded at every step necessary to solve a problem in imperative programming, hence pre-coded models aren't used. In contrast to declarative programming, which tells the computer "what" the program should do, imperative programming tells the machine "how" to do it.
Example of Imperative Language: ALGOL, C, C#, Assembly language, BASIC, Blue, Ada, C++, PHP, PROSE, Python Ruby, COBOL, D, FORTRAN, Go, Perl, Rust, Groovy, Java, Julia, Lua, MATLAB, Modula, MUMPS, Nim, Oberon, OCaml, Pascal.

2. Which one of the following is a Declarative Language?

1. HTML
2. CSS
3. Java Script

Answer: Java Script

3. Name two uses of a DIV tag?

Answer:

The <div> tag defines a division or a section in an HTML document.

The <div> tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript.

The <div> tag is easily styled by using the class or id attribute.

Any sort of content can be put inside the <div> tag!

4. What is the difference between relative positioning and absolute positioning in HTML?

Answer:

Relative Positioning

- When position:relative is given to an element without any other attributes (top, bottom, left, right) nothing will happen.

- But when an attribute left:20px is added the element moves 20px to the right from its normal position. The element effects no other element on the layout. Other content will not be adjusted to fit into any gap left by the element.
- The child of this element can be positioned within this block only. So position:relative has its limitations.

Absolute Positioning

- When position:absolute is given to an element the element can be placed precisely where one wants to.
- The element is positioned nearest to the first relatively or absolutely positioned parent element.
- When there is no parent element the absolutely positioned element is positioned directly to the HTML page itself.

5. What is the use of opacity in CSS?

Answer:

The **opacity** in CSS is the property of an element that describes the transparency of the element. It is the opposite of transparency & represents the degree to which the content will be hidden behind an element.

When using the opacity property to add transparency to the background of an element, all of its child elements become transparent as well. This can make the text inside a fully transparent element hard to read.

6. Which is the programming language used in the React Native Framework?

Answer:

Javascript is the programming language used in the React Native Framework?

7. Which online editor are we using for creating our apps in React Native Framework?

Answer:

- Nuclide
- Atom
- Sublime Text
- Visual Studio
- Vim Editor
- Spacemacs Editor
- GNU Emacs Editor

8. Write the steps to test your first designed app in the online editor on mobile.

Answer:

Steps of Mobile application testing

1. Documentation Testing

The beginning of mobile testing takes place from Documentation testing -preparatory stage. Even before the development of the app starts the testers are handed over screen layouts, navigational charts, and other requirements that are obscure on the design.

In this phase, you need to analyze the requirements for wholeness and discrepancy. All the discrepancies found in this stage are required to be resolved before the development begins Documentation phase marks the creations and analysis of requirements (Specification, PRD), Test Cases, Test Plan, Traceability Matrix.

2. Functional testing

It helps you test whether your mobile application works as expected and in accordance to the requirement specifications. While you are performing functional testing for your app keeps the following factors in mind:

- Business functionality of your app like banking, social networks, ordering and delivery of food, education, tickets, the game industry, etc.
- Target audience like companies, students, entrepreneurs, etc.
- Distribution channels like Google Play, direct delivery, App Store, etc.

The basic validations that you need to test in functional testing are:

- Installing and running the application
- Fields testing
- Business functionalities testing
- Interruptions testing
- Constant users feedback testing
- Update testing
- Device resources testing

3. Usability Testing

Usability testing ensures that your application offers convenient browsing to your customers and creates an intuitive interface that abides by industry standards. It promises fast and easy-to-use applications. Usability of your application is judged based on these basic three criteria:

- Satisfaction
- Efficiency
- Effectiveness

4. UI (User Interface) testing

User Interface (UI) testing ensures that your application's GUI meets all the required

specifications.

5. Compatibility (Configuration) testing

Compatibility (Configuration) testing validates the optimal performance of your application on different devices based on their size, screen resolution, version, hardware, etc.
Compatibility testing also takes care of

- OS Configuration
- Browser Configuration
- Database Configuration
- Device Configuration
- Network Configuration

Compatibility testing can be further divided into

Cross-platform testing: Testing your mobile application compatibility with different operating systems: Windows, iOS, Android, and BlackBerry, etc.

Cross-browser testing: Testing your mobile application compatibility in different browsers Google Chrome, Mozilla Firefox, Opera Mini, etc.

Database testing: Testing your mobile application compatibility in different database configurations: DB2, Oracle, MSSQL Server, MySql, Sybase.

Device Configuration testing: Testing your mobile application compatibility on different devices based on

- Device type: smartphone, tablet, etc.
- Device configuration: processor type, RAM, battery capacity, screen resolution, etc.

Network configuration testing: Testing your mobile application compatibility in different network configurations (TDMA, GSM) and standards (2G, 3G, 4G).

6. Performance testing

Performance testing helps you test your application reaction and constancy under the specific workload.

Performance testing attributes

- Load Testing: It is done to check the application's behavior under normal and extreme loads.
- Stress Testing: It is done to test the application's ability to sustain stress. It ensures that your application is capable to bear undue stress.
- Stability Testing: It tests if your application can work well for a longer period within normal loads.
- Volume Testing: It is conducted to test your application's performance when subjected to a huge volume of data.
- Concurrency testing: It tests the performance of your application when multiple users are logged in.

7. Security testing

Security testing validates the security features of your application. It also analyzes the risks of application hackers, protection, viruses, and unauthorized access to extremely sensitive data.

8. Recovery testing

Recovery testing tests the ability of your application to withstand and successfully recover from possible and potential failures caused by software issues, hardware failures, or communication problems.

9. Localization testing

Localization testing tests the adaptability of your application for a specific target audience based on cultural specifics.

10. Change related testing

Finally, when complete testing is done, you might find some bugs, resulting in a certain piece of code to change to eliminate those bugs. After these code changes you again need to carry out a round of testing. This basically includes:

- **Re-testing or Confirmation testing**: to test that all the detected defects are successfully fixed.
- **Regression testing:** sometimes code changes can even disturb the working of existing and properly working functions. Regression testing is done to ensure that the new changes did not lead to the appearance of new bugs.

11. Beta testing

Beta testing is done by real users on real devices to validate usability, functionality, compatibility, and reliability testing.

Before pushing your app forward for beta testing, take account of the following factors

- A number of testing participants
- Testing duration
- Shipping
- Demographic coverage
- Testing costs

Beta testing is good investment ensuring a better quality of your mobile app.

12. Certification testing

Certification testing tests whether your application meets the standards, licensing agreements, terms of use and requirements of stores like the App Store, Google Play, and Windows Phone.

9. What is the use of the render function in React Native Framework?

Answer:

React renders HTML to the web page by using a function called render(). The purpose of the function is to display the specified HTML code inside the specified HTML element. In the render() method, we can read props and state and return our JSX code to the root component of our app.

10. What is the use of the return function in the React Native Framework?

Answer:

Whatever a function component returns is rendered as a **React element.** React elements let you describe what you want to see on the screen.

Here the `Cat` component will render a `<Text>` element:

```
const Cat = () => {
  return <Text>Hello, I am your cat!</Text>;
                        };
```

11. What are the various components in your first app that you designed?

Answer:

App components

App components are the essential building blocks of an Android app. Each component is an entry point through which the system or a user can enter your app. Some components depend on others.

There are four different types of app components:

- Activities
- Services
- Broadcast receivers
- Content providers

Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed.

Activities

An activity is the entry point for interacting with the user. It represents a single screen with a user interface. An activity facilitates the following key interactions between system and app:

- Keeping track of what the user currently cares about (what is on screen) to ensure that the system keeps running the process that is hosting the activity.
- Knowing that previously used processes contain things the user may return to (stopped activities), and thus more highly prioritize keeping those processes around.
- Helping the app handle having its process killed so the user can return to activities with their previous state restored.
- Providing a way for apps to implement user flows between each other, and for the system to coordinate these flows.

Services

A service is a general-purpose entry point for keeping an app running in the background for all kinds of reasons. It is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface.

There are two types of services that tell the system how to manage an app: started services and bound services.

Started services tell the system to keep them running until their work is completed. This could be to sync some data in the background or play music even after the user leaves the app.

Bound services run because some other app (or the system) has said that it wants to make use of the service. This is basically the service providing an API to another process. The system thus knows there is a dependency between these processes, so if process A is bound to a service in process B, it knows that it needs to keep process B (and its service) running for A.

Broadcast receivers

> A broadcast receiver is a component that enables the system to deliver events to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements. Because broadcast receivers are another well-defined entry into the app, the system can deliver broadcasts even to apps that aren't currently running.

Content providers

> A content provider manages a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access. Through the content provider, other apps can query or modify the data if the content provider allows it.