Quiz 1- Components, Styles, Layouts

1. What's one of the main roles of the built-in `<View>` component?
    a. Directly output text.
    b. Structure / Group other child components.
    c. Handle touch events.

    Correct Answer: Option b - Structure / Group other child components.

    Explanation:

    - Option a – That's wrong. For text, there's a dedicated <Text> component which you need to use.
    - Option b – That's correct! The <View> allows you to group other components together, structure them (i.e. provide a layout) and (optionally) add some container styling.
    - Option c – That's wrong. There are other components that help you with handling touch events.

2. Can you use HTML elements (e.g. `<div>`, `<p>`, `<input>`) in React Native apps?
    a. Yes, React Native is able to handle HTML components, but native components are preferred.
    b. No, React Native doesn't provide any built-in components, you need to build all components from scratch.
    c. No, React Native doesn't recognize these components – it doesn't know how to compile them to native views.

    Correct Answer: Option c - No, React Native doesn't recognize these components – it doesn't know how to compile them to native views.

    Explanation:

    - Option a – That's wrong, React Native doesn't "understand" HTML elements.
    - Option b – That's wrong. React Native does offer built-in components which you use to build your own components.
    - Option c – That's correct!

3. What's the relation between React Native component styling and CSS (Cascading Style Sheets) for the Web?
    a. React Native styling is inspired by CSS (comparable / similar property names and values).
    b. React Native uses the exact same syntax but only supports a limited set of features.
    c. There is no connection.

    Correct Answer: Option a - React Native styling is inspired by CSS (comparable / similar property names and values).

    Explanation:

- Option a – That's correct. Whilst the exact property names and values you use don't always match the CSS alternative, the goal of RN is to get as close as possible.
- Option b – That's not entirely correct. The syntax doesn't match 100%.
- Option c – That's wrong, there absolutely is!

4. Which of the following example style rules does NOT work in React Native?
    a. borderColor: 'black'
    b. padding: 10
    c. 'background-color': '#ccc'

    Correct Answer: Option c - 'background-color': '#ccc'

    Explanation:

    - Option a - That's wrong - this property-value combination does work (e.g. if applied to a View)
    - Option b - That's wrong, this property-value combination does work (e.g. if applied to a View)
    - Option c - That's correct. This won't work in React Native because 'background-color' is not a supported property name (even though it's technically valid JS code).

5. Why would you use `const styles = StyleSheet.create({})` instead of a regular JavaScript object (`const styles = {}`)?
    a. There is no reason to use `StyleSheet`.
    b. Using a StyleSheet adds validation and potential performance improvements.
    c. Using a StyleSheet unlocks extra styling properties.

    Correct Answer: Option b - Using a StyleSheet adds validation and potential performance improvements.

    Explanation:

    - Option a – That's wrong, you should use a StyleSheet.
    - Option b - That's correct!
    - Option c – That's wrong, there are no extra properties to be unlocked.

6. What's "Flexbox"?
    a. A styling property that allows you to order items in a grid.
    b. A component built into React Native which you can use to structure content.
    c. A concept/ set of styling properties that allows you to structure content (i.e. create layouts).

    Correct Answer: Option c - A concept/ set of styling properties that allows you to structure content (i.e. create layouts).

    Explanation:

- Option a – That's wrong, it's neither a single styling property nor is it related to creating grids.
- Option b – That's wrong, it's not a component!
- Option c – That's correct!

7. What's the default styling/ layout behavior of a `<View>` component?'
    a. It organizes child components in a grid.
    b. It doesn't allow any ordering or structuring of child components.
    c. It uses Flexbox to organize its child components.

    Correct Answer: Option c - It uses Flexbox to organize its child components.

    Explanation:

    - Option a – That's not correct, it doesn't organize anything in a grid.
    - Option b – That's wrong. The opposite is the case - View is great when it comes to structuring child content.
    - Option c – That's correct!

8. If a `<View>` has `flexDirection: 'column'` (which is the default) - what does `alignItems: 'flex-end'` do in that case?
    a. It positions all child elements at the end of the column – on the horizontal axis.
    b. It positions all child elements at the end of the column – on the vertical axis.
    c. It has no effect.

    Correct Answer: Option a - It positions all child elements at the end of the column – on the horizontal axis.

    Explanation:

    - Option a – That is correct! alignItems positions elements along the cross axis. For flexDirection: 'column', the cross axis is the horizontal axis.
    - Option b – That's wrong - this would be achieved with justifyContent: 'flex-end'
    - Option c – That's wrong - there is an effect!

Quiz 2- More Components and Lists

1. How do you output a list (array) of content in React apps?
    a. You need a special component to output a list (array) of data.
    b. You use `map()` to map (=transform) the array of data into an array of components.

    Correct Answer: Option b - You use `map()` to map (=transform) the array of data into an array of components.

    Explanation:

    - Option a – That's incorrect. You don't need a special component to output an array of data.
    - Option b – This is correct.

2. Why might you need a `<ScrollView>` instead of a normal `<View>`?
    a. Unlike browser, mobile apps don't give you automatic scrolling. `<ScrollView>` adds it.
    b. A `<ScrollView>` gives you more efficient / optimized scrolling than you would've gotten without it.
    c. You can't `map()` array data to components without a `<ScrollView>`.

    Correct Answer: Option a - Unlike browser, mobile apps don't give you automatic scrolling. `<ScrollView>` adds it.

    Explanation:

    - Option a – That's correct. By default, the views you're building aren't scrollable. ScrollView changes that.
    - Option b – That's wrong, ScrollView doesn't optimize scrolling.
    - Option c – That's wrong, you don't need ScrollView to output a list of data.

3. What's the core difference between `<FlatList>` and `<ScrollView>`?
    a. ScrollView optimizes scrolling by only rendering what's required.
    b. FlatList optimizes scrolling by only rendering what's required.
    c. There is no big difference, FlastList is just a modern replacement for ScrollView.

    Correct Answer: Option b - FlatList optimizes scrolling by only rendering what's required.

    Explanation:

    - Option a – That's wrong. But close (kind of...)!
    - Option b – That's correct!
    - Option c – That's wrong - there is a real difference.