1. React: React - это JavaScript-библиотека для создания пользовательских интерфейсов.

2. JSX: JSX - это расширение синтаксиса JavaScript, которое позволяет вам описывать структуру пользовательского интерфейса внутри JavaScript-кода.

3. Babel: Babel - это инструмент, который позволяет использовать новейшие возможности языка JavaScript, которые могут не быть поддерживаемыми во всех браузерах, путем преобразования кода в совместимую версию JavaScript.

4. CRA (Create React App): Create React App - это инструментарий, который позволяет создавать новые проекты React с предварительно настроенной конфигурацией и средой разработки. Он автоматически настраивает сборку, конфигурацию Babel, Webpack и другие инструменты, чтобы упростить начало разработки в React.

5. SPA (Single Page Application): Single Page Application (одностраничное приложение) - это тип веб-приложения, которое загружает единственную HTML-страницу и динамически обновляет ее содержимое без полной перезагрузки страницы.

6. Рендеринг: Рендеринг - это процесс преобразования компонентов React в дерево виртуальных элементов (Virtual DOM) и дальнейшая отрисовка этого дерева на экране. React использует виртуальный DOM для эффективного обновления только необходимых частей пользовательского интерфейса.

7. Компоненты: Компоненты - это основные строительные блоки в React. Они представляют из себя независимые и переиспользуемые части пользовательского интерфейса.

8. Пропсы: Пропсы (props) - это механизм передачи данных от родительских компонентов к дочерним компонентам в React.

9. Состояние: Состояние (state) - это внутреннее состояние компонента, которое может изменяться во время выполнения. Изменение состояния вызывает перерисовку компонента и обновление пользовательского интерфейса.

10. Жизненные циклы компонента: Жизненные циклы компонента представляют собой методы, которые вызываются в различные моменты жизни компонента (например, при создании, обновлении и удалении). Эти методы позволяют вам выполнять определенные действия на разных этапах жизни компонента.

11. Принцип единственственной ответственности (Single Responsibility Principle): Принцип единственной ответственности - это принцип проектирования, который гласит, что каждый модуль, класс или компонент должен иметь только одну причину для изменения. Это означает, что каждый элемент должен быть ответственным только за одну конкретную задачу или функциональность.

12. Односторонний поток данных: Односторонний поток данных (One-Way Data Flow) - это концепция в React, где данные передаются от родительских компонентов к дочерним компонентам в одном направлении. Изменения данных происходят только вверх по иерархии компонентов через обратные вызовы (callbacks).

13. Иммутабельность: Иммутабельность - это концепция, которая гласит, что данные должны быть неизменяемыми (immutable), то есть после создания их нельзя изменить. В React рекомендуется использовать неизменяемые данные для обновления состояния компонентов, так как это помогает обнаруживать изменения и эффективно обновлять пользовательский интерфейс.

14. Чистая функция: Чистая функция (Pure Function) - это функция, которая всегда возвращает одинаковый результат при одинаковых входных данных и не имеет побочных эффектов

15. Функциональная компонента: Функциональная компонента (Functional Component) - это тип компонента в React, который определяется как обычная функция. Он принимает пропсы в качестве аргумента и возвращает элементы или компоненты, которые следует отобразить.

16. Хук (Hook): Хуки (Hooks) - это новая возможность в React 16.8, которая позволяет использовать состояние и другие возможности React в функциональных компонентах.

17. useState: useState - это хук, предоставляемый React, который позволяет функциональным компонентам иметь внутреннее состояние. Он возвращает пару значений: текущее состояние и функцию для его обновления.

18. useEffect: useEffect - это хук, предоставляемый React, который позволяет выполнить побочные эффекты в функциональных компонентах. Побочные эффекты могут быть, например, подпиской на события, запросами к серверу или изменением DOM. useEffect выполняет переданную функцию каждый раз, когда происходят изменения, указанные в зависимостях.