**Лабораторная работа № 19-20**

**Контрольные вопросы**

JSON

1. API (Application Programming Interface) — это интерфейс для взаимодействия между программами, через который одно приложение может запрашивать данные у другого. JSON (JavaScript Object Notation) — формат обмена данными, представляющий собой текст на основе синтаксиса JavaScript. XML (eXtensible Markup Language) — гибкий формат данных для хранения и обмена, похожий на HTML, но с настраиваемыми тегами. Front-End — это интерфейс пользователя, то, что он видит и с чем взаимодействует, обычно написанное на HTML, CSS, JavaScript. Back-End — серверная часть приложения, которая отвечает за обработку логики, запросов и данных, часто использует языки вроде Python, Node.js и другие. AJAX (Asynchronous JavaScript and XML) — технология для асинхронного обновления данных на веб-странице без перезагрузки.

Связь между ними: API часто возвращают данные в формате JSON или XML, которые затем обрабатываются на стороне Front-End с помощью AJAX-запросов для отображения пользователю. JSON и XML — популярные форматы для передачи данных между Front-End и Back-End.

2. JSON — это текстовый формат для представления структурированных данных. Его принцип действия основан на синтаксисе JavaScript для передачи данных в виде пар "ключ-значение". JSON нужен для удобного обмена данными между клиентом и сервером, так как он легко читается и обрабатывается.

3. Синтаксис и структура JSON представлены как пары "ключ-значение" внутри фигурных скобок, где значения могут быть строками, числами, массивами, объектами, логическими значениями или null. Массивы в JSON обозначаются квадратными скобками.

4. В JavaScript данные можно сериализовать (преобразовать) в JSON с помощью JSON.stringify и парсить (распаковать из строки) обратно в объект с помощью JSON.parse.

5. Преимущества JSON: он лёгок в использовании и чтении как людьми, так и компьютерами, имеет малый объём, что ускоряет передачу данных, поддерживается большинством языков программирования и является стандартом для обмена данными.

6. XML и JSON. JSON проще, быстрее обрабатывается и лучше поддерживается JavaScript, что делает его идеальным для современных веб-приложений. XML может быть полезен в случаях, когда требуется сложная структура данных или есть необходимость в использовании настраиваемых тегов.

Cookie и Web Storage

1. Известные способы сохранения данных на клиенте: Cookie, Local Storage, Session Storage, IndexedDB.

2. Cookie предназначены для хранения данных, которые передаются на сервер при каждом запросе, часто используются для аутентификации. Недостаток — небольшой размер и меньшая безопасность. Web Storage (Local Storage и Session Storage) предназначен для хранения данных только на клиенте и не отправляется на сервер. Local Storage имеет неограниченный срок действия, а Session Storage стирается после закрытия вкладки браузера. Web Storage позволяет хранить больше данных и повышает безопасность, так как данные не передаются на сервер.

3. Установка куки: `document.cookie = "username=Alice; expires=Fri, 31 Dec 2024 12:00:00 UTC; path=/"`. Для получения всех куки можно использовать `document.cookie`. Для извлечения конкретного куки удобно написать функцию, которая находит куки по ключу.

4. Local Storage — долговременное хранилище для данных, которые остаются даже после закрытия браузера. Используется для данных, которые не нужно отправлять на сервер, например, для сохранения настроек пользователя. Session Storage — временное хранилище, данные в котором удаляются при закрытии вкладки браузера. Подходит для временных данных, таких как данные формы.

5. Local Storage: для установки значения используем `localStorage.setItem("username", "Alice")`, а для получения `localStorage.getItem("username")`. Session Storage: для установки значения используем `sessionStorage.setItem("sessionData", "Temporary data")`, а для получения `sessionStorage.getItem("sessionData")`.

Эти примеры покрывают ключевые моменты использования JSON и Cookie/Web Storage в JavaScript для обмена и хранения данных на клиенте.