Теория

Домашнее задание

1. `localStorage` – это веб-хранилище данных, доступное в браузере. Данные, сохраненные в `localStorage`, остаются там даже после закрытия окна браузера. Они сохраняются на уровне домена и не имеют срока истечения. Однако они остаются доступными только для того домена, на котором были сохранены.

Так что, пока вы не удаляете данные из `localStorage` или не очищаете кеш браузера, они сохраняются постоянно.

2) Вы можете просмотреть содержимое `localStorage` прямо в консоли разработчика вашего браузера. Вот как это сделать в различных браузерах:

1. \*\*Google Chrome:\*\*

- Откройте инструменты разработчика, нажав правой кнопкой мыши на странице и выбрав "Исследовать элемент" или используя комбинацию клавиш `Ctrl+Shift+I` (Windows/Linux) или `Cmd+Opt+I` (Mac).

- Перейдите на вкладку "Application".

- В левой панели выберите "Storage" > "Local Storage".

2. \*\*Mozilla Firefox:\*\*

- Откройте инструменты разработчика, нажав правой кнопкой мыши на странице и выбрав "Исследовать элемент" или используя комбинацию клавиш `Ctrl+Shift+I` (Windows/Linux) или `Cmd+Opt+I` (Mac).

- Перейдите на вкладку "Storage".

3. \*\*Microsoft Edge:\*\*

- Откройте инструменты разработчика, нажав правой кнопкой мыши на странице и выбрав "Исследовать элемент" или используя комбинацию клавиш `Ctrl+Shift+I` (Windows/Linux) или `Cmd+Opt+I` (Mac).

- Перейдите на вкладку "Storage".

4. \*\*Safari:\*\*

- Откройте инструменты разработчика, перейдя в "Настройки" > "Расширенные" и включив "Показать меню "Разработка" в строке меню.

- Затем выберите "Разработка" в меню и перейдите в "Показать консоль".

- Перейдите на вкладку "Storage".

На вкладке "Local Storage" вы увидите все ключи и значения, сохраненные в `localStorage` для текущего домена.

3) Да, `sessionStorage` также является веб-хранилищем, но в отличие от `localStorage`, данные в `sessionStorage` хранятся только в течение сессии браузера. Это означает, что данные сохраняются только до тех пор, пока пользователь не закроет вкладку или окно браузера. Как только сессия завершается, данные из `sessionStorage` удаляются.

`sessionStorage` полезен, если вам нужно временно хранить данные, которые не должны сохраняться после закрытия вкладки браузера. Он также работает на уровне домена и доступен только для той вкладки, где был установлен.

Пример использования `sessionStorage`:

```javascript

// Установка значения

sessionStorage.setItem('key', 'value');

// Получение значения

var value = sessionStorage.getItem('key');

// Удаление значения

sessionStorage.removeItem('key');

// Очистка всех данных в sessionStorage

sessionStorage.clear();

```

Также, как и с `localStorage`, вы можете использовать инструменты разработчика браузера для просмотра содержимого `sessionStorage`.

4) Чтобы очистить локальное хранилище (`localStorage`) в браузере, вы можете использовать метод `clear()`:

```javascript

// Очистка локального хранилища

localStorage.clear();

```

Этот код удалит все данные, сохраненные в локальном хранилище для текущего домена. После выполнения этой команды все ключи и соответствующие значения будут удалены, и локальное хранилище будет пустым.

Будьте осторожны при использовании этой команды, так как она удалит все данные, и их нельзя будет восстановить.

5) Куки (`cookies`) и `localStorage`/`sessionStorage` являются механизмами хранения данных в веб-браузере, но у них есть различные применения и характеристики.

### Куки (`Cookies`):

1. \*\*Хранение данных на стороне клиента:\*\*

- Куки представляют собой небольшие текстовые файлы, хранящиеся на стороне клиента.

2. \*\*Серверная и клиентская сторона:\*\*

- Куки могут быть отправлены сервером и сохранены на стороне клиента, а затем отправлены обратно на сервер при каждом запросе.

3. \*\*Срок действия:\*\*

- Куки могут иметь определенный срок действия, после истечения которого они удаляются. Однако их можно также настроить на хранение в течение долгого времени.

4. \*\*Ограниченный объем данных:\*\*

- Куки ограничены по размеру (обычно 4 КБ), что делает их не самым подходящим для хранения больших объемов данных.

### `localStorage` и `sessionStorage`:

1. \*\*Хранение данных на стороне клиента:\*\*

- Оба этих механизма хранения данных работают на стороне клиента и предоставляют программным приложениям доступ к локальному хранилищу браузера.

2. \*\*Объем данных и срок действия:\*\*

- `localStorage` и `sessionStorage` имеют больший объем хранения данных (обычно 5 МБ), чем куки.

- `localStorage` сохраняет данные долгосрочно, даже после закрытия браузера.

- `sessionStorage` хранит данные только в течение сессии браузера (до закрытия вкладки).

3. \*\*Безопасность:\*\*

- Данные в `localStorage` и `sessionStorage` доступны только для того же домена, что и веб-страница, что делает их безопасными с точки зрения безопасности.

### Использование:

- \*\*Куки:\*\*

- Часто используются для отслеживания состояния сеанса, аутентификации пользователя, а также для хранения предпочтений пользователя.

- \*\*`localStorage` и `sessionStorage`:\*\*

- Часто используются для хранения настроек пользователя, кэширования данных, сохранения состояния приложений веб-приложений.

В общем, выбор между куками и `localStorage`/`sessionStorage` зависит от конкретных требований вашего веб-приложения и характеристик хранения данных, которые вам нужны.

6) Работа с куками в веб-разработке в основном включает в себя установку, чтение и удаление куков на стороне клиента. Вот несколько примеров на JavaScript:

### Установка куки:

```javascript

// Установка куки с именем "user" и значением "John" с сроком действия 1 день

document.cookie = "user=John; expires=" + new Date(new Date().getTime() + 24 \* 60 \* 60 \* 1000).toUTCString() + "; path=/";

```

### Чтение куки:

```javascript

// Чтение значения куки по имени

function getCookie(name) {

var nameEQ = name + "=";

var cookies = document.cookie.split(';');

for(var i = 0; i < cookies.length; i++) {

var cookie = cookies[i];

while (cookie.charAt(0) == ' ') cookie = cookie.substring(1, cookie.length);

if (cookie.indexOf(nameEQ) == 0) return cookie.substring(nameEQ.length, cookie.length);

}

return null;

}

// Пример чтения значения куки "user"

var username = getCookie("user");

```

### Удаление куки:

```javascript

// Удаление куки по имени

function eraseCookie(name) {

document.cookie = name + '=; Max-Age=-99999999;';

}

// Пример удаления куки "user"

eraseCookie("user");

```

Обратите внимание, что работа с куками может быть сложной из-за ограничений безопасности и приватности. Например, куки могут быть ограничены по размеру, и их можно установить только для того домена, на котором выполняется веб-страница.

Также, с появлением новых механизмов хранения данных, таких как `localStorage` и `sessionStorage`, использование кук может быть заменено в некоторых случаях. Однако куки остаются полезным инструментом для определенных сценариев, таких как аутентификация.

7) Для добавления и получения значений из веб-хранилища (например, `localStorage` или `sessionStorage`) вам нужно использовать соответствующие методы JavaScript. Вот примеры:

### Использование `localStorage`:

#### Добавление значения:

```javascript

// Установка значения в localStorage

localStorage.setItem('key', 'value');

```

#### Получение значения:

```javascript

// Получение значения из localStorage

var value = localStorage.getItem('key');

```

### Использование `sessionStorage`:

#### Добавление значения:

```javascript

// Установка значения в sessionStorage

sessionStorage.setItem('key', 'value');

```

#### Получение значения:

```javascript

// Получение значения из sessionStorage

var value = sessionStorage.getItem('key');

```

В этих примерах `'key'` является именем ключа, а `'value'` - значением. Замените их на соответствующие значения для вашего случая использования.

Обратите внимание, что значения в `localStorage` и `sessionStorage` хранятся в виде строк, поэтому, если вам нужно сохранить объект или массив, сначала преобразуйте его в строку с помощью `JSON.stringify()`, а при получении используйте `JSON.parse()`:

```javascript

// Сохранение объекта в localStorage

var obj = { key: 'value' };

localStorage.setItem('obj', JSON.stringify(obj));

// Получение объекта из localStorage

var storedObj = JSON.parse(localStorage.getItem('obj'));

```

Таким образом, вы можете легко добавлять и получать значения из веб-хранилища в вашем веб-приложении.

8) 1.Формы исследований и опросы:

- Если пользователь заполнил форму исследования или опроса на сайте, можно сохранить его ответы в локальном хранилище. Это позволяет предотвратить потерю данных, если пользователь случайно закроет вкладку или перезагрузит страницу до завершения опроса.

2. Состояние виджетов и интерактивных элементов:

- При работе с веб-приложениями, содержащими интерактивные элементы или виджеты (например, календари, панели инструментов), сохранение состояния этих элементов может улучшить общий пользовательский опыт. Например, сохранение выбранной даты в календаре между переходами по страницам.

6. 3. Закладки и предпочтения контента:

* + Позволить пользователям сохранять свои закладки или отмечать контент как "понравившийся". Эти данные могут быть сохранены в локальном хранилище и использованы для предоставления персонализированных рекомендаций или быстрого доступа к предпочитаемому контенту.

9) JSON (JavaScript Object Notation) - это формат обмена данными, который стал широко используемым в веб-разработке. Вот несколько преимуществ использования JSON:

1. \*\*Легкость чтения и записи:\*\*

- JSON имеет простой и легко читаемый синтаксис, основанный на парах "ключ-значение". Это делает его удобным для чтения и написания как людьми, так и машинами.

2. \*\*Легкость разбора и создания:\*\*

- Парсинг (разбор) и создание JSON легко выполняются с использованием различных языков программирования. Это облегчает передачу данных между клиентом и сервером, а также интеграцию с различными технологиями.

3. \*\*Поддержка различных типов данных:\*\*

- JSON поддерживает различные типы данных, включая строки, числа, булевы значения, массивы, объекты и null. Это позволяет эффективно представлять разнообразные данные.

4. \*\*Независимость от языка:\*\*

- JSON является независимым от языка форматом данных, что означает, что он может быть использован с любым языком программирования. Это упрощает взаимодействие между различными частями веб-приложений, написанными на разных языках.

5. \*\*Поддержка в большинстве языков программирования:\*\*

- Почти все современные языки программирования предоставляют библиотеки или встроенные средства для работы с JSON. Это облегчает использование JSON в различных технологических стеках.

6. \*\*Эффективность в передаче данных:\*\*

- JSON обеспечивает эффективность в передаче данных благодаря своей компактной структуре. Это особенно важно при передаче данных по сети, таких как веб-запросы и ответы.

7. \*\*Поддержка в браузерах:\*\*

- Поддержка JSON встроена в большинство современных браузеров. Это упрощает взаимодействие веб-приложений с серверами и обмен данными в браузерной среде.

Из-за этих преимуществ JSON широко используется для обмена данными между клиентской и серверной частями веб-приложений.

10) JSON (JavaScript Object Notation) поддерживает несколько типов данных. Вот основные значения, которые могут быть использованы в JSON:

1. \*\*Объекты (Objects):\*\*

- Объекты в JSON представляют собой неупорядоченные наборы пар "ключ-значение". Ключи (имена) являются строками, а значения могут быть другими объектами, массивами, строками, числами, булевыми значениями или `null`.

Пример объекта JSON:

```json

{

"name": "John",

"age": 30,

"city": "New York"

}

```

2. \*\*Массивы (Arrays):\*\*

- Массивы в JSON представляют собой упорядоченные списки значений. Значения в массиве могут быть строками, числами, булевыми значениями, объектами, массивами или `null`.

Пример массива JSON:

```json

["apple", "banana", "orange"]

```

3. \*\*Строки (Strings):\*\*

- Строки в JSON должны быть заключены в двойные кавычки. Они представляют собой последовательность символов Unicode.

Пример строки JSON:

```json

"Hello, World!"

```

4. \*\*Числа (Numbers):\*\*

- Числа в JSON представлены в стандартной числовой форме без кавычек.

Пример числа JSON:

```json

42

```

5. \*\*Булевы значения (Boolean):\*\*

- Булевы значения в JSON представлены как `true` или `false` (без кавычек).

Пример булевого значения JSON:

```json

true

```

6. \*\*`null`:\*\*

- Значение `null` в JSON представляет собой отсутствие значения.

Пример использования `null` в JSON:

```json

null

```

Эти типы данных могут быть комбинированы и вложены друг в друга для представления сложных структур данных в формате JSON.

11) Чтение данных из JSON в языке программирования, таком как JavaScript, обычно включает в себя процесс, называемый "парсинг" (разбор). Вот как это делается:

### В JavaScript:

1. \*\*JSON.parse():\*\*

- Встроенный метод `JSON.parse()` используется для преобразования строки JSON в объект JavaScript.

Пример:

```javascript

var jsonString = '{"name": "John", "age": 30, "city": "New York"}';

var jsonObject = JSON.parse(jsonString);

// Теперь jsonObject является объектом JavaScript

console.log(jsonObject.name); // Выводит: John

```

2. \*\*Работа с массивами в JSON:\*\*

- Аналогично, если у вас есть массив JSON, вы можете использовать `JSON.parse()` для преобразования его в массив JavaScript.

Пример:

```javascript

var jsonArrayString = '["apple", "banana", "orange"]';

var jsonArray = JSON.parse(jsonArrayString);

// Теперь jsonArray является массивом JavaScript

console.log(jsonArray[0]); // Выводит: apple

```

3. \*\*Обработка вложенных объектов:\*\*

- Если ваш JSON содержит вложенные объекты, `JSON.parse()` автоматически преобразует их во вложенные объекты JavaScript.

Пример:

```javascript

var nestedJsonString = '{"person": {"name": "Alice", "age": 25}}';

var nestedJsonObject = JSON.parse(nestedJsonString);

// Теперь nestedJsonObject.person является объектом JavaScript

console.log(nestedJsonObject.person.name); // Выводит: Alice

```

4. \*\*Обработка ошибок:\*\*

- Не забывайте обрабатывать ошибки при разборе JSON. Например, если JSON недействителен, `JSON.parse()` может вызвать ошибку.

Пример:

```javascript

var invalidJsonString = '{name: "John", age: 30}';

try {

var invalidJsonObject = JSON.parse(invalidJsonString);

console.log(invalidJsonObject);

} catch (error) {

console.error('Ошибка при разборе JSON:', error.message);

}

```

Помните, что формат JSON должен быть корректным, иначе разбор может завершиться ошибкой.