



# Lesson - 22

**Closures.**  
**Storing data on client side**

# План занятия



1. Вопросы по домашнему заданию.
2. Лекции **9.05.2018 => 26.04.2018 18:00** или **19:00**.
3. Последняя лекция (экзамен): 16.05.2019.
4. Closure.
5. Cookie.
6. Web storage API.

# Lexical environment



Все переменные внутри функции – это свойства специального внутреннего объекта `LexicalEnvironment`, который создаётся при её запуске.

```
1 function sayHi(name) {  
2   var phrase = "Привет, " + name;  
3   alert( phrase );  
4 }  
5  
6 sayHi( 'Вася' );
```

1. Создает пустой объект `LexicalEnvironment` и заполняет его (name = "Вася", phrase = undefined).
2. Функция выполняется.
3. В конце выполнения функции объект с переменными обычно выбрасывается и память очищается.

# [[Scope]]



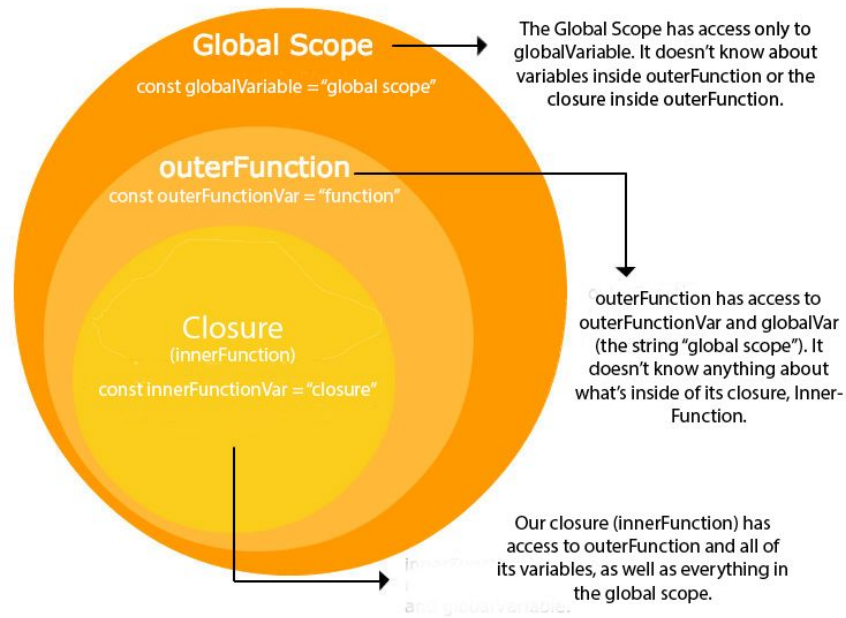
При создании функция получает скрытое свойство `[[Scope]]`, которое ссылается на лексическое окружение, в котором она была создана:

```
1 var userName = "Вася";  
2  
3 function sayHi() {  
4   alert( userName ); // "Вася"  
5 }
```

```
sayHi. [[Scope]] = window
```

# Closures

*You can think of closure as a state retained between function calls. This state is created by using variables from function's outer scope.*



# Что такое понимать “замыкание”?



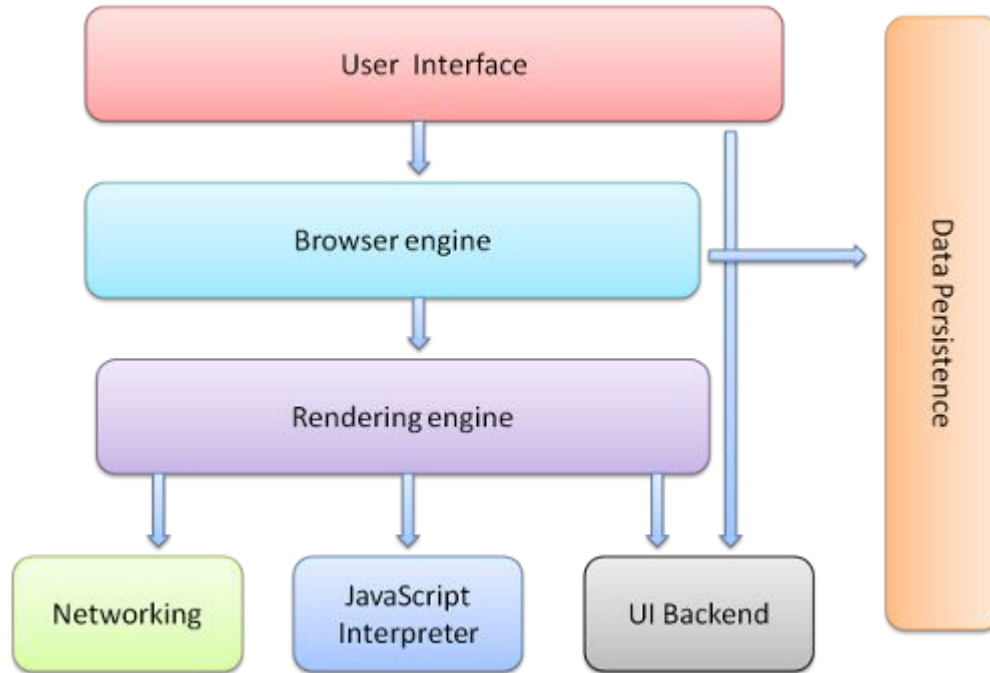
1. Все переменные и параметры функций являются свойствами объекта переменных `LexicalEnvironment`. Каждый запуск функции создает новый такой объект. На верхнем уровне им является «глобальный объект», в браузере – `window`.
2. При создании функция получает системное свойство `[[Scope]]`, которое ссылается на `LexicalEnvironment`, в котором она была создана.
3. При вызове функции, куда бы её ни передали в коде – она будет искать переменные сначала у себя, а затем во внешних `LexicalEnvironment` с места своего «рождения».

# Summary



- Область видимости как “полупрозрачная сумка”
- Каждая функция создает собственную область видимости.
- Глобальная область видимости - область видимости, к которой есть доступ из функций созданных в ней. Таким образом все функции имеют доступ к глобальной области видимости.
- Если внутренняя функция создается внутри другой функции, тогда внутренняя функция имеет доступ к области видимости внешней функции. Обратное не верно.
- При каждом вызове функция получает новую чистую область видимости так что нет сохраненного состояния
- Замыкание с другой стороны позволяет сохранить состояние между вызовами функции, используя переменную из внешней области видимости функции.

# Data persistence component





# Data persistence types

---

- Cookie
- Web Storage API
  - Session Storage
  - Local Storage
  - WebSQL Database (deprecated)
  - Indexed Database (still doesn't have great support)



# Cookie



Cookies - небольшой фрагмент данных, отправленный веб-сервером и сохраняется на компьютере пользователя. Веб клиент (обычно веб-браузер) каждый раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб в составе HTTP-запроса.

Применяется для сохранения данных на стороне пользователя, на практике обычно используется для:

- аутентификации пользователя;
- хранения персональных предпочтений и настроек пользователя;
- отслеживание состояния сеанса доступа пользователя;
- ведения статистики о пользователях.

# Cookies



Cookies - небольшой фрагмент данных, отправленный веб-сервером и сохраняется на компьютере пользователя. Веб клиент (обычно веб-браузер) каждый раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб в составе HTTP-запроса.

Применяется для сохранения данных на стороне пользователя, на практике обычно используется для:

- аутентификации пользователя;
- хранения персональных предпочтений и настроек пользователя;
- отслеживание состояния сеанса доступа пользователя;
- ведения статистики о пользователях.

# Cookies



- Неточная идентификация
- Кража
- Подмена
- Межсайтовые куки
- Нестабильность (клиент - сервер)
- Срок действия

# Cookies - неточная идентификация



Если на компьютере используется более одного браузера, то, как правило, каждый имеет отдельное хранилище для cookies. Поэтому cookies идентифицируют не человека, а сочетание учетной записи компьютера, и браузера.

Таким образом, любой человек, который использует несколько учетных записей, компьютеров или браузеров, имеет несколько наборов cookies.

# Cookies - кража



Куки могут быть украдены с помощью анализа трафика - это называется взломом сессии.

Сетевой трафик может быть перехвачен и прочитан не только его отправителем и получателем (особенно в публичных сетях Wi-Fi).

Этот трафик включает в себя и куки, передаваемые через не зашифрованы HTTP-сессии.

Там, где сетевой трафик не шифруется, злоумышленники могут прочитать сообщения пользователей сети, в том числе их куки, используя программы, называемые sniffером.

# Получение/изменение cookies in JS

Хотя теоретически куки должны храниться и отправляться обратно на сервер неизменными, злоумышленник может изменить их содержимое перед отправкой.

Например, куки могут содержать общую сумму, которую пользователь должен оплатить свои покупки; изменив это значение, злоумышленник сможет заплатить меньше установленной суммы.

Процесс изменения содержания куки называется подменой куки.

```
var x = document.cookie;
```

```
document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00  
UTC; path="/;
```

# Термин действия cookie



Постоянные куки критикуются экспертами за свой долгий срок хранения, который позволяет веб-сайтам отслеживать пользователей и создавать их профиль с течением времени. Здесь затрагиваются и вопросы безопасности, поскольку украденные постоянные куки могут использоваться в течение значительного периода времени.

Кроме того, правильно составленная вредоносная программа, которая может быть запущена после аутентификации пользователя, сможет перенести сессионные куки на компьютер злоумышленника, в первом приближении позволит посещать защищенный сайт без ввода имени пользователя и пароля сколько угодно долгое время.



# Web Storage API


















Документация Web Storage была разработана консорциумом W3C для улучшения работы по хранению данных на стороне клиента:

- Память сеанса (Session Storage)
- Локальная память (Local Storage).
- WebSQL Database (deprecated)
- Indexed Database

# Session Storage

Память сеанса (Session Storage) имеет задачу хранить данные вашей сессии при просмотре веб-страницы, и удалять их как только вы закрыли эту страницу в которой работали.

Свойство `sessionStorage` позволяет получить доступ к объекту [Storage](#) текущей сессии. Свойство `sessionStorage` очень похоже на свойство [Window.localStorage](#), единственное различие заключается в том, что все данные, сохраненные в `localStorage` не имеют определенного времени жизни, а данные в `sessionStorage` очищаются в момент окончания сессии текущий страницы. Сессия страницы остается активной все время пока окно браузера открыто и сохраняется между перезагрузками страниц. **Открытие той же страницы в новом окне браузера или новой вкладке приводит к созданию новой сессии страницы**, что отличается от поведения `session cookies`.

													
	 Chrome	 Edge	 Firefox	 Internet Explorer	 Opera	 Safari	 Android webview	 Chrome for Android	 Edge Mobile	 Firefox for Android	 Opera for Android	 Safari on iOS	 Samsung Internet
Basic support	5	Yes	2	8	10.5	4	Yes	Yes	Yes	Yes	11	3.2	?

# Using Session Storage

\*show on save click counter example
















```
1 // Save data to sessionStorage
2 sessionStorage.setItem('key', 'value');
3
4 // Get saved data from sessionStorage
5 var data = sessionStorage.getItem('key');
6
7 // Remove saved data from sessionStorage
8 sessionStorage.removeItem('key');
9
10 // Remove all saved data from sessionStorage
11 sessionStorage.clear();
```

# Local Storage

Локальная память данных (Local Storage) используется если вам необходимо, чтобы данные хранились дольше сессии просмотра веб-документа.

Свойство `sessionStorage` хранит данные в течение сеанса (до закрытия браузера), в отличие от данных, находящихся в свойстве `localStorage`, которые не имеют ограничений по времени хранения и могут быть удалены только с помощью JavaScript.

Простым примером использования может быть подсчет количества посещений веб-страницы.

													
	 Chrome	 Edge	 Firefox	 Internet Explorer	 Opera	 Safari	 Android webview	 Chrome for Android	 Edge Mobile	 Firefox for Android	 Opera for Android	 Safari on iOS	 Samsung Internet
Basic support	5	Yes	2	8	10.5	4	Yes	Yes	Yes	Yes	11	3.2	?

# Using Local Storage



```
1 | localStorage.setItem('myCat', 'Tom');
```

The syntax for reading the localStorage item is as follows:

```
1 | var cat = localStorage.getItem('myCat');
```

The syntax for removing the localStorage item is as follows:

```
1 | localStorage.removeItem('myCat');
```

The syntax for removing all the localStorage items is as follows:

```
1 | // clear all items  
2 | localStorage.clear();
```

# Summary

Name	Size limitation	Send to server *Less traffic	Lifetime support and limitations	Parsing object	Security	Support
cookie	<b>4KB</b> The number of cookie a particular host can send to you is 20. The cookie size is limited to 4KB. The total number of cookie (from all hosts) is 300.	Yes	Exists by using expires	string	<b>httpOnly</b>  No access from JS	<b>About full support</b>  *can be disabled by user
sessionStorage	<b>5MB ~ unlimited</b>  *Depends on device and browser	No	<b>No expiration</b> *save for reload page *cleared on close *created new for new tab	Depends	Access from JS with same origin policy	<b>About full support</b>
LocalStorage	<b>5MB ~ unlimited</b>  *Depends on device and browser	NO	<b>No expiration</b> persists even when the browser is closed and reopened.	Depends	Access from JS with same origin policy	<b>About full support</b>

# References



1. Замыкания, области видимости:
  - a. RU: <https://learn.javascript.ru/functions-closures>
  - b. EN: <https://javascript.info/advanced-functions>
2. Управление памятью:
  - a. [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Memory\\_Management](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Memory_Management)
  - b. <https://blog.sessionstack.com/how-javascript-works-memory-management-how-to-handle-4-common-memory-leaks-3f28b94cfbec>
3. Understanding scope:
  - a. <https://scotch.io/tutorials/understanding-scope-in-javascript>
  - b. <https://float-middle.com/what-is-scope-and-closure-in-javascript/>
4. Cookie:
  - a. [http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP\\_StateManagement.html](http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_StateManagement.html)
  - b. [https://www.w3schools.com/js/js\\_cookies.asp](https://www.w3schools.com/js/js_cookies.asp)
5. Web Storage API:
  - a. [https://developer.mozilla.org/en-US/docs/Web/API/Web\\_Storage\\_API/Using\\_the\\_Web\\_Storage\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API)
6. Session storage:
  - a. <https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage>
7. Local storage:
  - a. <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>
8. Storage spaces test:
  - a. <https://www.html5rocks.com/en/tutorials/offline/quota-research/>
  - b. <http://dev-test.nemikor.com/web-storage/support-test/>

# Домашнее задание - теоретическая часть

1. Прочитать главу про области видимости и замыкания:
  - 1.1. RU: <https://learn.javascript.ru/functions-closures> (все)
  - 1.2. EN: <https://javascript.info/advanced-functions> (подразделы 3-7)
2. Cookies:
  - 2.1. [http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP\\_StateManagement.html](http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_StateManagement.html)
  - 2.2. [https://www.w3schools.com/js/js\\_cookies.asp](https://www.w3schools.com/js/js_cookies.asp)
3. sessionStorage:
  - 3.1. <https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage>
4. localStorage:
  - 4.1. <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>





# Домашнее задание - практическая часть (замыкание)

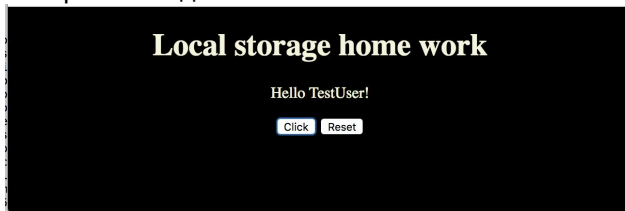


1. Создать функцию счетчик с возможностью задавать incrementValue и initialValue.

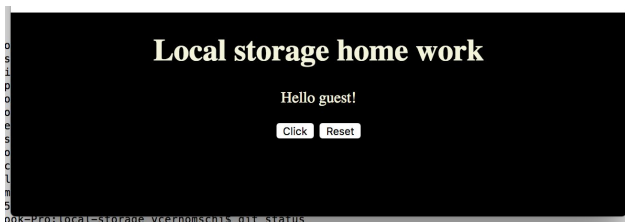
**\*Advanced part:** написать unit test.

# Домашнее задание - практическая часть (WebStorage API)

1. Создать страницу которая будет:
  - a. Выводить prompt и спрашивать имя пользователя <name>.
  - b. Сохранять <name> в localStorage если юзер ввел больше 2х символов: `localStorage.setItem(key, value);`
  - c. При открытии страницы нужно считать значение `localStorage.getItem(key, value);` и если полученное значение существует она должно быть показывать на экране в виде: Hello <name>!



- d. Если в localStorage нет данного элемента показывать: Hello guest!



- e. Добавить кнопку reset, которая будет удалять <name> из localStorage

**\*Advanced part:** добавить unit tests