

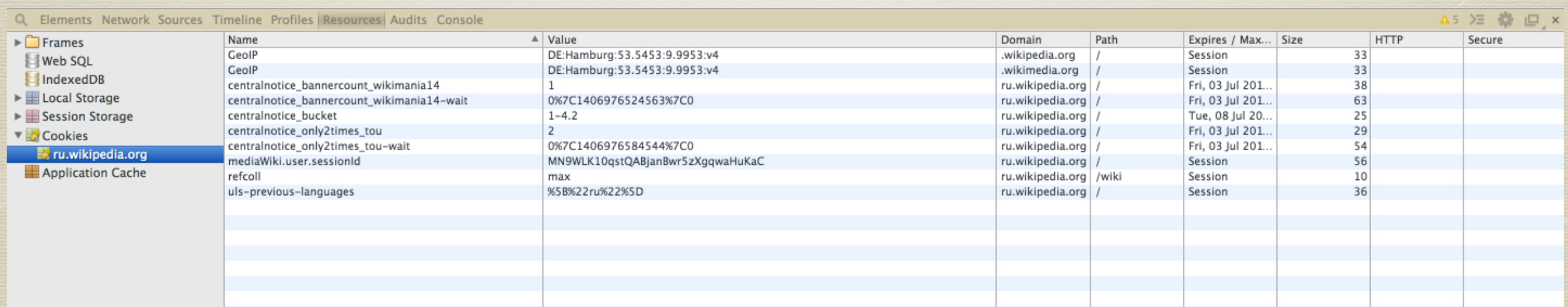
Урок 10

Cookie

Cookie - небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя. Веб-клиент (обычно веб-браузер) всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в виде HTTP-запроса. Cookie хранятся только на стороне клиента. Обычно с ключом записанным в cookie связаны определенные данные которые хранятся на сервере. Ограничения cookie

- Имя и значение (после encodeURIComponent) вместе не должны превышать 4кб.
- Общее количество cookie на домен ограничено 30-50, в зависимости от браузера.
- Сервер может поставить cookie с дополнительным флагом HttpOnly. Cookie с таким параметром передаётся только в заголовках, оно никак не доступно из JavaScript.

Просмотр данных cookie через консоль браузера



The screenshot shows a browser's developer console with the 'Resources' tab selected. On the left, a tree view shows 'Cookies' expanded for 'ru.wikipedia.org'. The main area displays a table of cookies.

Name	Value	Domain	Path	Expires / Max...	Size	HTTP	Secure
GeolP	DE:Hamburg:53.5453:9.9953:v4	.wikipedia.org	/	Session	33		
GeolP	DE:Hamburg:53.5453:9.9953:v4	.wikimedia.org	/	Session	33		
centralnotice_bannercount_wikimania14	1	ru.wikipedia.org	/	Fri, 03 Jul 201...	38		
centralnotice_bannercount_wikimania14-wait	0%7C1406976524563%7C0	ru.wikipedia.org	/	Fri, 03 Jul 201...	63		
centralnotice_bucket	1-4.2	ru.wikipedia.org	/	Tue, 08 Jul 20...	25		
centralnotice_only2times_tou	2	ru.wikipedia.org	/	Fri, 03 Jul 201...	29		
centralnotice_only2times_tou-wait	0%7C1406976584544%7C0	ru.wikipedia.org	/	Fri, 03 Jul 201...	54		
mediaWiki.user.sessionId	MN9WLK10qstQABjanBwr5zXgqwaHuKaC	ru.wikipedia.org	/	Session	56		
refcoll	max	ru.wikipedia.org	/wiki	Session	10		
uls-previous-languages	%5B%22ru%22%5D	ru.wikipedia.org	/	Session	36		

Работа с Cookie в JS

JavaScript поддерживает встроенный объект с именем `document.cookie` для работы с кукисам. Этот объект хранит все `cookie`, доступные для страницы, с которой запущен скрипт. **`document.cookie`** представляет собой строку в специальном формате. При записи нового `cookie` могут быть также указаны параметры через знак `;` и каждый параметр разделяется таким знаком. Параметры которые могут быть указаны :

- **`path=/mypath`** - Путь, внутри которого будет доступ к `cookie`. Если не указать, то имеется в виду текущий путь и все пути ниже него.
- **`domain=site.com`** - домен на котором будет доступна `cookie`. По умолчанию текущий домен.
- **`expires=Tue, 19 Jan 2038 03:14:07 GMT`** - Дата истечения куки в формате GMT. Если не указать то `cookie` будет жить до закрытия браузера.
- **`secure - cookie`** будет передаваться только через HTTPS

Чтение всех cookie привязанных к данной странице

```
console.log(document.cookie)
```

Запись нового ключа в cookie

```
document.cookie = «someKey=someValue; path=/; domain=somedomain.com »;
```


Работа с окнами в JS

JS позволяет работать и открывать новые вкладки и новые окна с помощью метода **window.open**. **window.open** имеет следующие параметры:

- **url** - URL для загрузки в новое окно.
- **name** - Имя нового окна. Может быть использовано в параметре target в формах.
- **params** - Строка с конфигурацией для нового окна. Состоит из параметров, перечисленных через запятую. Пробелов в ней быть не должно.

window.open возвращает ссылку на объект window открытого окна, но только в том случае если оно не было заблокировано. Также родительское окно имеет доступ ко всем переменным которые есть в дочернем окне, но только при условии что это окно открыто на том же самом домене. В дочернем окне есть свойство **window.opener** которое содержит ссылку на окно которое открыло его. По умолчанию **window.open** открывает новую вкладку. Для того чтобы метод открыл новое окно нужно указать позиционирование.

```
var params = 'scrollbars=no,resizable=no,status=no,location=no,toolbar=no,menubar=no';  
window.open('/', 'test', params);
```


Свойства и методы окна браузера

События окна:

- **onresize** - событие о изменении окна браузера
- **onscroll** - события прокрутки страницы
- **onload** - событие загрузки окна
- **onfocus** - событие фокусировки на окне

Свойство и методы окна:

- **open()** - метод для открытия нового окна
- **close()** - метод для закрытия окна
- **closed** - свойство в котором содержится индикатор было ли закрыто окно
- **title** - заголовок окна соответствует содержимому элемента TITLE в HEAD.
- **moveBy(x,y)** - перемещает окно на x пикселей вправо и y пикселей вниз
- **moveTo(x,y)** - передвигает окно в заданные координаты
- **resizeBy(width,height)** - изменяет размер окна на заданную величину
- **resizeTo(width,height)** - изменяет размер окна на заданное значение.
- **scrollBy(x,y)** - прокрутка окна на заданное число пикселей вперед или назад.
- **scrollTo(x,y)** - прокручивает окно к заданным координатам.
- **focus()** - метод для передачи фокуса на окно

Работа с iframe в JS

Элемент **iframe** является обычным узлом DOM, как и любой другой. Существенное отличие — в том, что с ним связан объект **window** внутреннего окна. Он доступен по ссылке **iframe.contentWindow**.

Другой способ получить ссылку на **iframe** это перебор свойства **window.frames** текущего объекта **window**. Доступ в **window.frames** возможен либо по имени либо по индексу внутри **window.frames**.

```
window.frames[o]
```

Внутри **iframe** ссылку на родительское окно можно получить через свойство **window.parent**. **window.top** всегда содержит ссылку на самое верхнее окно в текущей иерархии

У **iframe** есть событие **onload** которое говорит о том моменте когда он загрузился и оно никак не связано с **onload** страницы в которую он встроен.

Iframe и родительское окно имеют доступ к данным друг друга внутри только при условии что они открыты с одного домена и одного и того же порта.

Общение между вкладками с помощью `postMessage`

Для общения между вкладками открытыми с разных доменов и между `iframe` и родительским окном открытых с разных доменов в объекте есть интерфейс **`window.postMessage`**.

Отправка сообщения

Для отправки сообщения в другую вкладку или **`iframe`** открытый с другого домена нужно вызвать **`postMessage`** с 2 параметрами:

- **`data`** - Данные. По спецификации, это может быть любой объект, который будет клонирован с сохранением структуры при передаче.
- **`targetOrigin`** - Разрешить получение сообщения только окнам с данного источника. При указании «*» сообщение передается всем доменам.

```
win.postMessage("Привет!", "http://localhost");
```

Получение сообщения

Для получения сообщения нужно подписаться на событие **`message`**. Обработчик события принимает аргумент со следующими свойствами: **`data`** - присланные данные. **`origin`** - домен источник события. **`source`** - ссылка на объект окна который послал сообщение.

```
window.addEventListener("message", function listener(event) {  
    console.log(event.data, event.origin, event.source)  
}  
  
    , false);
```


localStorage

localStorage - это клиент-сайд база данных, которая позволяет сохранять данных в форме «ключ — значение». Преимуществами является возможность сохранения данных больших чем ограничено куками. Данные установленные в одной вкладке на 1 домене доступны на и в других вкладках. Позволяет сохранять данные для домена даже после закрытия браузера.

Методы и свойства:

- **localStorage.clear()** - удаление всех пар ключ-значение
- **localStorage.getItem(key)** - возвращение значения по ключу
- **localStorage.remainingSpace()** - возвращает оставшееся доступное место
- **localStorage.removeItem(key)** - удаляет значение по ключу из хранилища
- **localStorage.setItem(key, value)** - сохранение ключа и значения для него

События:

- **storage** - срабатывает при любом изменении хранилища (удаление, добавление , изменение значений) и обработчик события принимает аргумент со следующими свойствами: **key** - Ключ с которым произошло изменение, **oldValue** - старое значение ключа, **newValue** - новое значение или **null**, если удалено, **url** - страница, которая вызывает метод, приведший к изменению.