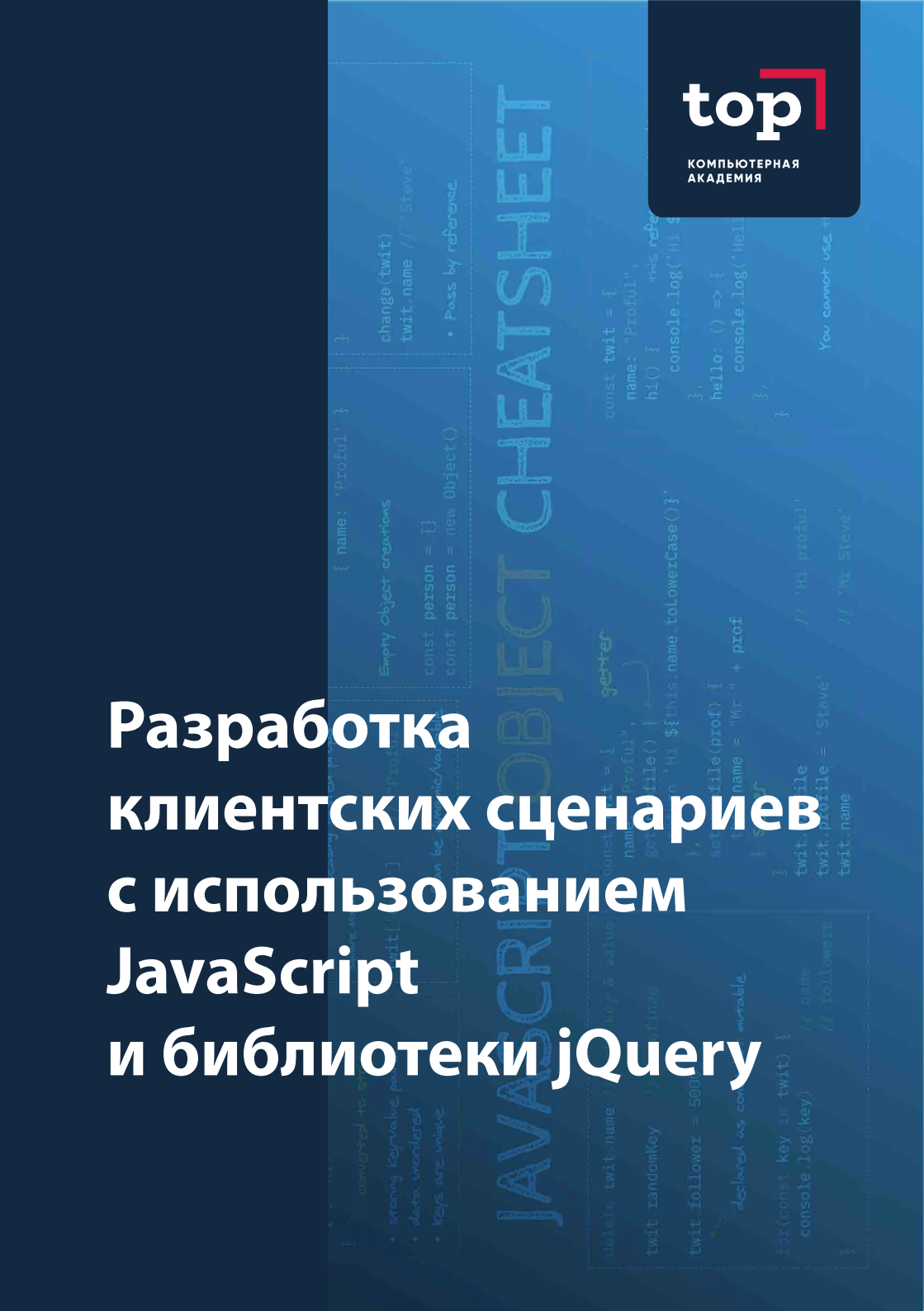
top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Разработка клиентских сценариев с использованием JavaScript и библиотеки jQuery



Unit 4.1

Browser Object Model

Содержание

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

1. Что такое Browser Object Model?

JavaScript является универсальным языком программирования, позволяющим создавать решения для различных задач, например, оконные приложения или серверные модули. Однако при использовании его для разработки веб-страниц появляются некоторые ограничения. Возможности программ, написанных на JavaScript внутри веб-страницы, не могут выйти за пределы этой страницы, ее HTML структуры. Поскольку коды JavaScript обрабатываются и выполняются браузером, они не могут прямо управлять самим браузером, лишь в той степени, в которой браузер это позволяет. В свою очередь браузер является отдельной программой в операционной системе и доступ к параметрам этой системы также лимитирован и для браузера, как программы, и для JavaScript, как ее части.

Основные ограничения, накладываемые на возможности JavaScript в HTML странице, можно сформулировать следующим образом:

- При помощи JavaScript нельзя запускать другие приложения, давать команды операционной системе. В противном случае веб-страницы могли бы перезагружать компьютер, отключать антивирусы и т.п.
- Инструкции JavaScript не имеют прямого доступа к файлам компьютера. Без разрешения пользователя невозможно даже открыть файл для чтения. Иначе разные сайты могли бы читать Ваши файлы и передавать их содержимое по сети.

- Из скриптов одной вкладки нельзя управлять содержимым других вкладок. Единственная возможность — закрыть вкладку, если она была создана из данного скрипта (эту возможность мы рассмотрим далее в уроке).
- Обращаться к серверу при помощи JavaScript можно только в том случае, если сайт был загружен именно с этого сервера. В конце урока мы более подробно поговорим о кросс-доменных ограничениях, влияющих на эту особенность JavaScript.
- Командами JavaScript нельзя закрыть или запустить браузер, свернуть его главное окно или изменить его размеры.

Взаимодействие с браузером и системой всё же необходимо для хорошо построенных сайтов. Например, могут возникнуть задачи вернуться на предыдущую просмотренную страницу, открыть новую страницу (или новый сайт) в текущей или новой вкладке браузера, а также получить данные о типе браузера или операционной системе, в которой он запущен. Для мобильных устройств могут понадобиться сведения о состоянии источника питания (аккумулятора) или данные о геолокации устройства.

Возможности, которые предоставляет браузер для использования кодами JavaScript, составляют основу Браузерной Объектной Модели (*Browser Object Model*, BOM). Согласно с этой моделью, работа с браузером и, через него, с операционной системой заменяется работой со специальным объектом «window». Из этого объекта можно получить (считать) информацию о параметрах и свойствах

браузера, системы или устройства, а также установить или изменить (записать) некоторые величины, влияющие на их работу.

Следует отметить, что на данный момент для BOM не существует официального стандарта. Связано это с постоянной конкуренцией на рынке браузеров — так называемой «войной браузеров». Для завоевания лидирующих позиций браузеры предоставляют новые возможности, отличающие их от других конкурентов. Вероятно, что стандарт BOM вообще не будет принят в окончательном виде, так как появление такого стандарта, во-первых, сделает браузеры «одинаковыми» и они тут же начнут снова расширять свои возможности сверх стандарта. И мы снова будем говорить об отсутствии стандарта, но уже на другие группы функций. Во-вторых, сама идея стандарта касается не столько языка JavaScript, сколько его взаимодействия с браузером и системой. В этом аспекте нужно принять решение, что какой-то или какие-то браузеры являются «образцовыми», а остальные должны переделать свои функции под них. Такие предпосылки могут завершиться неудачей из-за правил честной конкуренции.

Таким образом, складывается ситуация в которой разные браузеры содержат собственные, уникальные возможности, не повторяющиеся в других браузерах. Более того, эти возможности постоянно обновляются и добавляются. Когда Вы будете читать этот урок перечень возможностей различных браузеров, наверняка, будет отличаться от тех, которые существуют сейчас, на момент написания урока. В то же время, большинство современных браузеров реализуют определенный набор одинако-

вых методов, позволяющих скриптам на веб-страницах управлять одними и теми же свойствами. Его можно считать определенным неофициальным стандартом, позволяющим вести разговор о браузерной модели вообще, а не о функциях каждого отдельного браузера по отдельности. Исходя из сказанного, при использовании возможностей BOM в реальных проектах желательно проверить их поддержку и правильность работы в различных браузерах, причем не только в их новейших выпусках, а и в прошлых версиях, которые могут использовать поклонники данного семейства браузеров.

Ранее мы упоминали об еще одной роли объекта «window» — обеспечении глобальной области видимости для переменных (см. урок 1). Напомним, что именно благодаря этому объекту реализуется возможность обмена данными между различными объектами JavaScript.

Сам объект «window», в свою очередь, является составным. В его структуру входит несколько дочерних объектов, отвечающих за различные задачи (см. рис. 1)

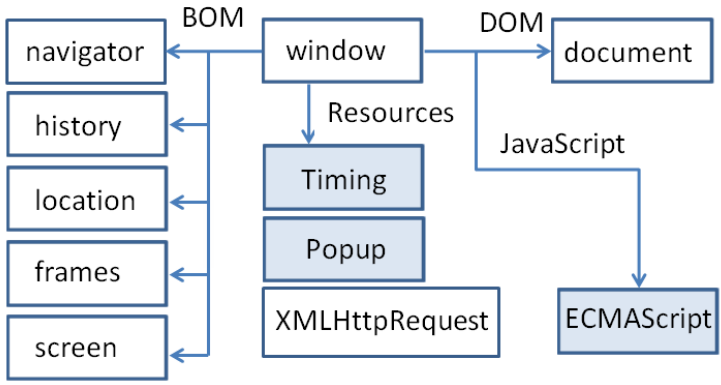


Рисунок 1

Среди дочерних элементов объекта «window» можно выделить несколько групп. Группу BOM составляют объекты, отвечающие за описанное выше взаимодействие с браузером и операционной системой. Чуть позже мы рассмотрим их более детально.

Группу ресурсов (Resources) формируют объекты и функции, обеспечивающие вспомогательные задачи, такие как:

- таймер и отложенный запуск функций (Timing);
- диалоговые окна (Popup), рассмотренные на первом уроке (alert, prompt, confirm);
- асинхронный обмен данными через объект «XMLHttpRequest».

Объект «document» является основой объектной модели документа (*Document Object Model*, DOM). В нем собирается информация о структуре HTML страницы. С его помощью можно управлять этой структурой посредством команд JavaScript. С этим объектом мы детальнее познакомимся во второй части урока.

Стандартизированные элементы языка JavaScript можно выделить в группу «ECMAScript». Здесь мы увидим, например, типы данных, такие как «Object» или «String», основные функции и определения языка. Эта группа обеспечивает работу JavaScript в соответствии с требованиями стандарта языка — ECMAScript.

Элементы, указанные на рисунке как Timing, Popup и ECMAScript не являются непосредственными дочерними узлами объекта «window», а обозначают группы дочерних функций, свойств и объектов, объединенных

по своему предназначению. К остальным элементам можно получить доступ, набрав их имя после имени объекта «window.» (добавив после него точку), либо непосредственно указав их имя, опустив запись «window.». Убедитесь в этом, включив консоль браузера:

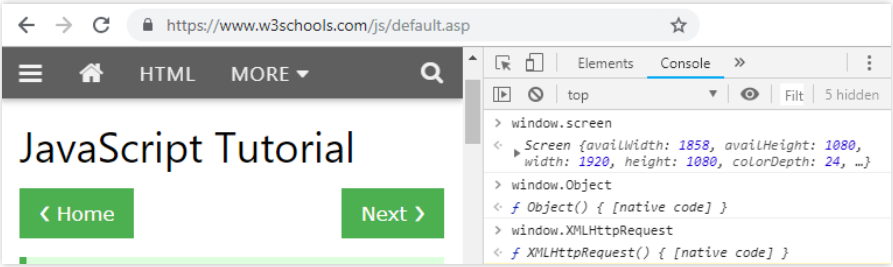


Рисунок 2

2. Объекты Browser Object Model

Объект Window. Открытие, перемещение и изменение размера окон

Объект «window» является основой браузерной объектной модели (BOM). Этот объект «представляет» окно браузера как главный контейнер для веб-страницы. Всё, что присутствует на этой веб-странице, так или иначе, попадает в объект «window» и становится доступным для программного управления. Кроме HTML элементов страницы в объекте «window» также собираются все переменные и функции, объявленные в различных скриптах страницы. Объект «window» поддерживается всеми видами браузеров.

Наряду с тем, что «window» является контейнером для других элементов, сам объект «window» также содержит методы и свойства. В основном, они нацелены на возможность получить информацию о параметрах окна браузера, а также на управление окнами. Основные свойства объекта «window», определяющие размеры и положение текущего окна (вкладки) браузера приведены в следующей таблице:

Свойство	Описание
innerHeight	Высота контента окна (без панелей браузера)
innerWidth	Ширина контента окна
outerHeight	Высота окна браузера (включая панели)
outerWidth	Ширина окна браузера

Свойство	Описание
screenLeft, screenX	Отступ от левого края окна браузера до начала содержимого страницы (контента)
screenTop, screenY	Отступ от верхнего края окна браузера до начала содержимого страницы (контента)
scrollX, scrollY	Величина (в пикселях) сдвига контента за счет полос прокрутки

Все эти свойства доступны только для чтения, то есть можно узнать их величину из JavaScript, но изменения их значений не будут иметь эффекта на отображении окна.

Основные методы объекта «window», позволяющие управлять самими окнами, приведены в следующей таблице:

Метод	Описание
open()	Открывает новое пустое окно браузера
open(addr,id,attr)	Открывает новое окно и загружает страницу с адресом addr. Окну присваивается имя «id» и применяются аттрибуты «attr»
stop()	Прекращает загрузку окна
close()	Закрывает окно
moveTo(X,Y)	Перемещает окно в точку экрана с заданными координатами
moveBy(dX,dY)	Сдвигает позицию окна по каждой координате
resizeTo(W,H)	Устанавливает размеры окна на заданные
resizeBy(dW,dH)	Прибавляет к размерам окна переданные значения

Метод «open» после создания нового окна возвращает ссылку на это новое окно. При помощи этой ссылки можно управлять окном, вызывая остальные методы из таблицы.

Рассмотрим пример. Откройте в браузере произвольную страницу (в текущем примере это страница с адресом «https://www.w3schools.com») и вызовите консоль

разработчика. Напишите в консоли инструкцию по созданию нового окна:

```
window.open()
```

После выполнения команды (нажатия кнопки «Enter») в браузере появится новая пустая вкладка. Очевидно, что изменить размеры отдельной вкладки невозможно, т.к. она встроена в браузер наравне с остальными вкладками.

Для того чтобы иметь возможность управлять размерами и положениями нового окна необходимо создавать не вкладку, а именно отдельное окно. Это достигается путем передачи аттрибута «resizable» при вызове метода создания окна. Выполним в консоли следующую команду:

```
newWin=window.open("https://itstep.org", "StepWin",
                   "resizable");
```

Как результат ее выполнения появится новое окно (а не вкладка), в котором начнется загрузка страницы «https://itstep.org». Второй аргумент «StepWin», переданный в функцию создания окна, задает имя нового окна как программного объекта. По этому имени окно может быть найдено среди множества других окон. Это имя не является названием вкладки или окна и непосредственно нигде не отображается. Для того чтобы закрыть новое окно введем в консоли команду:

```
newWin.close()
```

Окно, которое было ранее создано, закроется и исчезнет с экрана компьютера без каких-либо предупреждений.

Задавать положение и размер окна необходимо при его создании. После того как новое окно будет создано и управление вернется в консоль старого окна, возможность их взаимного влияния будет ограничена. Как мы уже видели, закрыть новое окно из старого можно, поскольку оно его и создавало. А вот поменять размеры и положение может оказаться затруднительным. Попробуйте ввести в консоли команду изменения размеров «newWin.resizeTo(500,500)». В результате будет «выброшено» исключение, запрещающее прямой доступ сайтов одного домена к сайтам другого домена (*cross-origin access*):

Рисунок 3

Задать положение и размер нового окна все-таки можно, только делать это необходимо сразу после его создания, не ожидая возврата в консоль разработчика. Команды нужно передать в одной инструкции, разделив их точкой с запятой. Введите или скопируйте в консоль следующую инструкцию:

```
newWin=window.open("https://itstep.org", "StepWin",
                   "resizable");
newWin.resizeTo(500,500);
newWin.moveTo(50,50);
```

После нажатия «Enter» появится новое окно с заданными размерами (500x500 пикселей) в левом верхнем

углу экрана (точнее, с отступами 50x50 пикселей от левого верхнего угла). Как уже отмечалось, возможность закрыть новое окно сохраняется и после возврата в консоль предыдущего окна браузера.

Попробуйте самостоятельно создавать новые окна с другими адресами. Используйте методы относительного характера «moveBy» и «resizeBy» для управления положением и размерами новых окон.

Объект Navigator. Параметры браузера

Объект «navigator» используется для сбора информации от операционной системы, параметрах, настройках и состоянии браузера. Этот объект доступен только для чтения, установка (запись) новых параметров не будет иметь эффекта на дальнейшей работе браузера или системы. Отметим еще раз, что объект «navigator» не является полностью стандартизированным, тем не менее, большинство браузеров обеспечивают его поддержку.

Информацию, содержащуюся в объекте «navigator», не следует использовать как однозначно достоверную. В силу отсутствия стандартов многие браузеры заполняют поля объекта одинаковыми значениями, просто чтобы не оставлять их пустыми. Также, браузеры могут неправильно подавать сведения об операционной системе, если она обновлялась позже, чем был установлен браузер.

Полезную информацию в объекте «navigator» содержат поля, отвечающие за настройки браузера такие, как предпочтительный язык, наличие сетевого подключения или установки разрешений на использование Cookie.

Основные свойства и методы объекта «navigator» приведены в следующей таблице

Свойство / метод	Описание	Особенности
appCode-Name	Кодовое имя браузера	Содержит «Mozilla» для браузеров Chrome, Edge, Firefox, IE, Safari, и Opera.
appName	Имя браузера	Содержит «Netscape» для браузеров Chrome, Edge, Firefox, IE11 и Safari.
appVersion	Данные о версии браузера	Включает совместимые версии
battery или getBattery()	Данные об аккумуляторе	Возвращает объект Battery-Manager. Поддерживается не всеми браузерами
connection	Данные о сетевом подключении	Содержит объект Connection
cookie-Enabled	Установки разрешений Cookie	true или false
geolocation	Данные геолокации	Содержит объект Geolocation
language	Основной язык браузера	Кодовое имя языка
languages	Допустимые языки	Массив кодовых имен
onLine	Наличие подключения к сети	true или false
platform	Платформа (операционная система) браузера	Строка с кодовым названием системы
product	Имя ядра браузера	Все браузеры содержат «Gecko»
userAgent	Данные заголовка «User-Agent», отсылаемого браузером	Объединяет appCodeName и appVersion
getUser-Media()	Доступ к медиа-ресурсам	Получить доступ к микрофону или камере (если они есть)
vibrate()	Управление устройством вибрирования (если есть)	Включает вибрацию на заданное время

Некоторые свойства объекта «navigator» рассчитаны на использовании в составе мобильных устройств. Например, свойство «battery» или метод «getBattery()» (в зависимости от типа браузера) позволяет получить данные о состоянии аккумулятора устройства и прогнозированном времени его работы.

Для стационарных ПК, подключенных к сети, можно увидеть, например, полный заряд «level: 1» и бесконечное время разряда «dischargingTime: Infinity». Для устройств, работающий от батареи, свойство «dischargingTime» позволяет определить прогнозируемое время работы до полного разряда (в секундах), а свойство «level» содержит данные об уровне оставшегося заряда батареи. Также свойство «charging» уведомляет о наличии (или отсутствии) подключения к зарядному устройству.

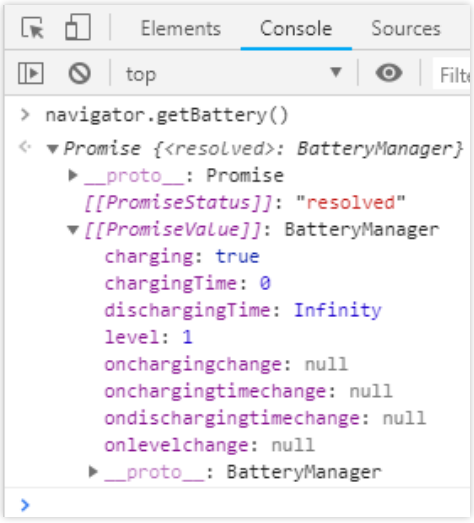


Рисунок 4

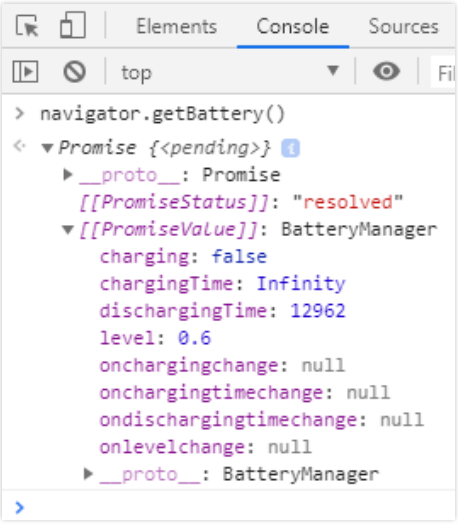


Рисунок 5

Свойство «connection» может быть использовано для получения информации о типе сетевого подключения.

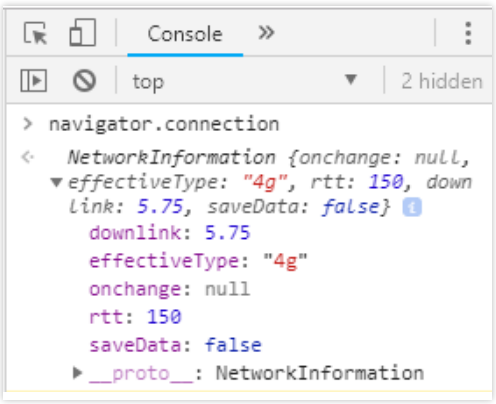


Рисунок 6

В составе объекта, возвращенного свойством «connection», присутствует информация о скорости под-

ключения (downlink) в Мб за секунду, эквивалентный тип подключения (effectiveType) — "slow-2g", "2g", "3g" или "4g", а также время приема-передачи (round-trip time, rtt), выраженное в миллисекундах.

Свойство «geolocation» предоставляет данные о геолокации устройства, на котором установлен браузер. Данные не передаются, если на устройстве нет модуля геолокации или этот модуль отключен.

Методы объекта «navigator», такие как «getUserMedia()» или «vibrate()» дают возможность получить доступ к видеокамере и микрофону устройства, а также к средствам вибрации, если они, конечно, в устройстве присутствуют. В основном, подобные устройства распространены в мобильных телефонах или планшетах. Хотя и в стационарных ПК видеокамера может быть установлена и зарегистрирована в операционной системе.

Сведения о языках, используемых браузером, и о предпочтительном языке содержат в себе массив «languages» и строка «language»:

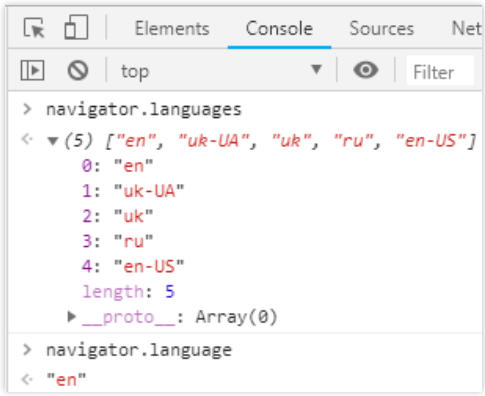


Рисунок 7

В массиве «languages» содержится набор строк, каждая их которых обозначает кодовое название языка. Порядок элементов соответствует приоритету языков, установленному пользователем в настройках операционной системы. Строка «language», по сути, является копией первого (нулевого) элемента массива «languages», то есть содержит кодовое название основного предпочитаемого языка.

Объект Screen. Свойства экрана

Информация, содержащаяся в объекте «window.screen», определяет параметры экрана монитора, такие как глубина цвета и ориентация, а также его размеры в пикселях.

Основные свойства и методы объекта «screen» приведены в следующей таблице:

Свойство, метод	Описание	Особенности
availTop	Первый доступный пиксель от верхнего края экрана	Поддерживается не всеми браузерами. Обычно содержит «0»
availLeft	Первый доступный пиксель от левого края экрана	
availHeight	Доступная высота экрана	Без учета нижней (и/или верхней) строки состояния
height	Полная высота экрана	Заданная монитором
availWidth	Доступная ширина экрана	Без учета боковых панелей
width	Полная ширина экрана	Заданная монитором
colorDepth	Глубина цвета активной палитры (бит на пиксель)	Некоторые браузеры, независимо от настроек, всегда содержат «24» (только для совместимости)

Свойство, метод	Описание	Особенности
pixelDepth	Глубина цвета монитора (бит на пиксель)	
orientation / msOrientation	Ориентация экрана	В браузере Edge используется свойство msOrientation
lockOrientation()	Закрепление ориентации экрана	Устаревшие. Исключаются из современных браузеров.
unlock-Orientation()	Освобождение закрепления	

Объект «screen» также не имеет стандартизированного описания, в следствие чего различные браузеры по-разному реализуют его свойства и методы. В частности, в целях совместимости скриптов некоторые браузеры задают значения свойств объекта «screen» независимо от реальных показателей экрана. До введения стандартов на объекты BOM не стоит полагаться на эти значения при разработке программ на JavaScript.

Наиболее достоверными и часто применимыми свойствами объекта «screen» являются те, которые отвечают за размеры экрана монитора. Среди них можно выделить две группы свойств: «width» и «height» — определяющие полные размеры экрана (ширину и высоту, соответственно), а также «availWidth» и «availHeight» — отвечающие за доступные (*от англ. available*) размеры, то есть за свободную часть окна, без учета панели задач, строк состояния, док-панелей и т.п.

Например, если на экране монитора есть одна панель задач, и она размещена по левому краю, то можно получить следующее соотношение параметров (см. рис. 8).

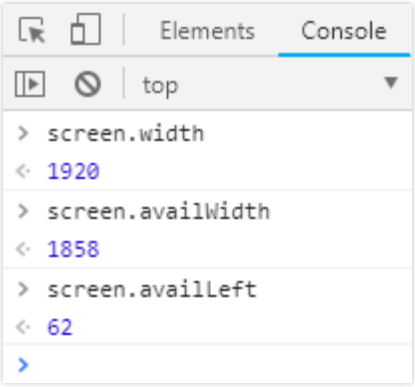


Рисунок 8

Полная ширина экрана соответствует его техническим характеристикам и составляет 1920 пикселей (screen.width). Свободная (доступная) ширина равна 1858 пикселей (screen.availWidth), поскольку часть экрана занимает панель задач. Первый доступный пиксель имеет отступ 62 от левого края (screen.availLeft), что равно ширине панели задач. Несложно убедиться, что 1858+62=1920.

Последний использованный параметр (screen.availLeft) не имеет гарантированной поддержки во всех браузерах, поэтому его применение должно сопровождаться дополнительными проверками на возможность чтения.

Полностью аналогично можно получить данные о высоте экрана — полной и доступной.

Объекты Location и History. Перемещение по страницам

Объекты «location» и «history» в структуре BOM обеспечивают управление адресами веб-страниц, которые отображаются на данной вкладке браузера. При помощи этих объектов можно перемещаться по различным страницам.

Объект «location» собирает в себе данные об веб-адресе текущей страницы, открытой в данный момент в данной вкладке. С его помощью можно узнать об этом адресе и его составных частях, а также загрузить в текущей вкладке страницу с другим адресом.

Перед тем как приступить к рассмотрению свойств и методов объекта «location», напомним о том, что такое веб-адрес и из чего он состоит.

В сети Интернет существует множество разнообразных сайтов. Мы уже знаем, что веб-сайт представляет собой некоторую программу, которая хранится на определенном компьютере (сервере). Для того чтобы браузер отобразил нам нужный сайт, он должен найти сервер, на котором этот сайт располагается и запустить ответственную за сайт программу.

При этом программа должна понимать, что от нее требуется — передать файл, выдать определенные данные или вернуть HTML код страницы. Необходимое действие задается протоколом (или схемой) — первой составной частью адреса сайта. HTTP протокол предусматривает, что от программы требуется HTML код. В таком случае, адрес сайта будет начинаться с «http://». Если при этом необходимо обеспечить шифрование данных, указывают протокол «https://» (s означает «secure» — защищенный). Если требуется получить (загрузить) сам файл, а не HTML код, используют файловый протокол FTP, а адрес будет начинаться как «ftp://». Когда мы выполняем упражнения и открываем локальные html-файлы браузер использует протокол «file://», ссылаясь на локальную файловую систему, а не к Интернет-ресурсам. Есть и другие виды протоколов.

Далее за протоколом следует доменное имя сайта. Как правило, именно эту часть адреса называют «именем сайта» или «адресом сайта». Примерами являются «itstep.org», «www.google.com» или «D:/Sources/js4_1.html».

Если на одном компьютере (сервере) обслуживаются несколько сайтов, то они могут быть разделены при помощи разных портов. Порт (сетевой порт) — это целое число, определяющее программу, которой адресуется сетевой запрос. При помощи различных портов можно запускать несколько программ на одном сетевом узле, и они не будут мешать друг другу при работе с сетью.

Как правило, порт для веб-ресурсов имеет значение «80» и явно в адресе не указывается. Однако некоторые сайты специально размещают на нестандартном порту. Существует мнение, что это более безопасно, но это утверждение спорно. В таком случае после доменного имени необходимо указать номер сетевого порта, отделив его от имени двоеточием. Следующий сайт использует 8080-й порт: «http://portquiz.net:8080»

В составе адресной строки могут также передаваться дополнительные параметры. При помощи параметров можно уточнить номер или название товара, который запрашивает пользователь, желаемый язык, день, начиная с которого нужно вывести данные об оплатах, поисковый запрос и множество других данных. Параметры позволяют создать одну веб-страницу, в которой они будут анализироваться, вместо создания множества страниц под каждый отдельный товар, язык или опорный день.

Существует два основных способа передачи параметров для веб-страницы. Метод «POST» скрывает данные

в составе HTTP пакета, тогда как метод «GET» добавляет их непосредственно в адрес ресурса. Детальнее методы передачи данных мы рассмотрим в следующем уроке «Формы», пока остановимся на том, что параметры могут являться частью адреса веб-страницы и, как следствие, будут частью объекта «location» BOM.

Параметры отделяются от остального адреса знаком вопроса «?». Каждый из параметров задается по схеме «имя=значение». Если требуется передать несколько параметров, то их определения разделяются знаком «&». Например, в следующем адресе передается один параметр «page»: «http://portquiz.net:8080/?page=1». Пример передачи двух параметров «q» и «oq» иллюстрирует такой адрес «https://www.google.com/search?q=itstep&oq=itstep».

Для того чтобы при переходе на страницу была возможность ее «прокрутить» к определенному элементу, используются ссылки-якоря (anchors). Имена этих ссылок также добавляются к полному адресу страницы, отделяясь от него символом «#». Например, при переходе по адресу «https://www.w3schools.com/js/#demo» страница будет отображена, начиная от блока с именем «demo».

Полный адрес сайта также называется термином URI (*англ. Uniform Resource Identifier*) — унифицированный идентификатор ресурса. Он стандартизирован документом RFC3986, который доступен в электронном виде по ссылке https://tools.ietf.org/html/rfc3986. В нем можно более подробно прочитать о принципах формирования адресов, видах протоколов и примерах адресов.

Как уже отмечалось, для получения информации об адресе страницы и его составных частях в браузерной

модели (BOM) предназначен объект «location». Его поля (свойства) заполняются при загрузке страницы в зависимости от наличия или отсутствия отдельных частей адреса.

Для примера рассмотрим адреса двух страниц, включающих в себя различные элементы:

1) «https://www.w3schools.com/js/#main» и

2) «http://portquiz.net:8080/?page=1».

Свойства объекта «location» с описаниями собраны в следующей таблице. В колонке «Значение» под номером «1» указано значение свойства, которое будет установлено при загрузке первой из указанных выше страниц, «2» — при загрузке второй.

Свойство	Описание	Значение
hash	часть адреса, начиная с символа '#'	1) "#test" 2) "" (пустая строка)
host	имя сайта и порт (если указан)	1) "www.w3schools.com" 2) "portquiz.net:8080"
hostname	имя сайта (без порта)	1) "www.w3schools.com" 2) "portquiz.net"
href	весь адрес	1) "https://www.w3schools.com/js/#main" 2) "http://portquiz.net:8080/?page=1"
pathname	строка пути (от имени сайта)	1) "/js/" 2) "/"
port	номер порта (если указан)	1) "" (пустая строка) 2) "8080"
protocol	имя протокола	1) "https:" 2) "http:"
search	часть адреса начиная с символа '? '	1) "" (пустая строка) 2) "?page=1"

Перейдите по указанным адресам, включите консоль разработчика в браузере и введите запросы на свойства объекта «location». Проверьте их значения в зависимости от модификаций адреса.

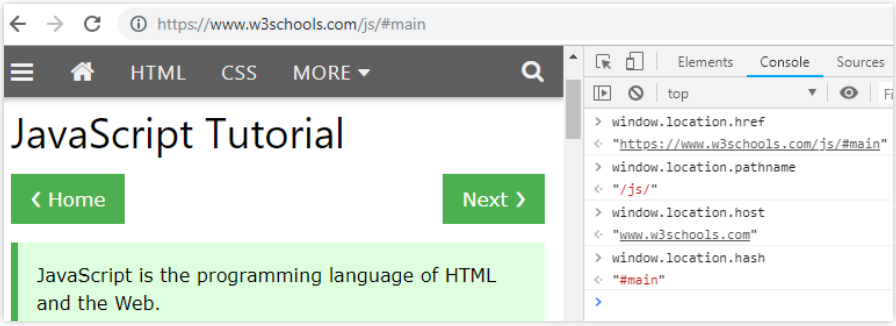


Рисунок 9

У объекта «location» также существует несколько методов, предназначенных для управления адресом отображаемой страницы. Они используются для перезагрузки (обновления) страницы или для переходов на другие адреса. Основные методы объекта «location» приведены в следующей таблице

Метод	Описание	Пример
assign(addr)	Перейти по адресу addr	window.location.assign("https://itstep.org")
reload()	Обновить (перезагрузить) страницу	reload(true) – перезагрузить с сервера reload(false) или reload() – из кеша браузера
replace(addr)	Сменить адрес на addr	window.location.replace("https://itstep.org")

Методы «assign» и «replace» производят практически одинаковые действия с тем отличием, что при использо-

вании метода «assign» сохраняется история переходов между страницами, тогда как применение «replace» не будет фиксироваться в истории и браузер будет запоминать лишь последний адрес в данной вкладке.

Объект «location» предусматривает возможность сокращенной формы вызова метода «assign» при помощи оператора присваивания:

```
window.location = "https://itstep.org"
```

После указанной инструкции произойдет переход на страницу с адресом «https://itstep.org» так же, как если бы был вызван метод «assign».

Также следует отметить, что изменение любого из свойств объекта «location» (кроме «hash») будет сопровождаться обновлением адреса путем вызова метода «assign». Например, инструкция

```
window.location.protocol = "https:"
```

после присваивания также приведет к перезагрузке страницы с учетом нового протокола.

Якорь страницы (или фрагмент, согласно RFC3986) хранится в свойстве «location.hash». Поскольку якоря обращаются к различным элементам одной страницы, смена якоря не требует перезагрузки ее содержимого. Этим часто пользуются для создания «одностраничных приложений», в которых смена страниц происходит без их реальной загрузки. Все данные обо всех страницах загружаются один раз при первом обращении к ресурсу. При необходимости отображения новых данных или новых страниц меняется их видимость, то есть старые данные становятся невидимыми,

а новые — видимыми. Для предотвращения перезагрузки страниц их адреса отличаются лишь после символа «#».

Объект браузерной модели «history» хранит в себе историю посещенных ранее страниц в данной вкладке. При помощи этого объекта можно программным способом управлять переходами к ранее просмотренным страницам или возвратами от них к тем, что были просмотрены позже. Другими словами, этот объект позволяет «листать» историю просмотров. Также, объект «history» позволяет менять историю — добавлять или модифицировать записи в истории просмотров.

Основные свойства и методы объекта «history» приведены в следующей таблице:

Свойство / метод	Описание	Особенности
length	Количество страниц в истории текущей вкладки	
state	Объект состояния, установленный для данной страницы	Задается методами pushState или replaceState
back()	Переход к предыдущей странице в истории просмотров	Аналогичен go(-1)
forward()	Переход к следующей странице в истории просмотров	Аналогичен go(1)
go(n)	Переместиться на n шагов в истории просмотров	n>0 — вперед по истории n<0 — назад по истории
pushState(state, title, addr)	Добавить запись в историю	После добавления происходит переход на добавленную страницу
replaceState(state, title, addr)	Заменить запись в истории	Заменяет текущее положение

Для иллюстрации работы методов объекта «history» рассмотрим следующую программу. Создайте новый HTML-файл, наберите или скопируйте в него следующее содержимое (*код также доступен в папке Sources, файл js4_1.html*)

```
<!DOCTYPE HTML>
<html>

<head>
  <title>History object</title>
</head>

<body>
  <button onclick="history.back()">Back</button>
  <button onclick="history.forward()">Forward</button>
  <script>
    window.history.replaceState({'record':'0'},
                                "page 0", "?record=0");
    window.history.pushState({'record':'1'},
                                "page 1", "?record=1");
    window.history.pushState({'record':'2'},
                                "page 2", "?record=2");
    window.history.pushState({'record':'3'},
                                "page 3", "?record=3");
    window.history.back();
  </script>
</body>
</html>
```

В теле документа создаются две кнопки «Back» и «Forward», в обработчиках нажатия которых применены соответствующие функции объекта «history». Далее в скриптовой части производятся манипуляции с историей браузера.

Во-первых, подменяется запись текущей страницы командой «replaceState». В качестве аргументов указываем:

1. Объект-состояние с одним полем «{‹record':'0'}»
2. Название страницы «page 0». Это название используется только для журнала переходов и не является заголовком страницы, не отображается на вкладке браузера. Более того, многими браузерами это название игнорируется, пока стандарт для объекта «history» не утвержден окончательно.
3. Адрес страницы, хранимый в журнале. В качестве адреса используем добавочные данные с указанием условного номера страницы «?record=0». В журнале истории разрешаются замены только на адрес того же домена, что и был в записи.

Во-вторых, в журнал истории добавляются три новые записи вызовами метода «pushState». Аргументы в этот метод передаются по тому же шаблону, что и для метода «replaceState».

В-третьих, вызывается метод «back()», который должен совершить переход на предыдущую страницу из журнала истории данной вкладки.

Сохраните файл и откройте его в браузере. Обратите внимание на адресную строку — в ней будет указано «js4_1.html?record=2». Из этого мы делаем вывод, что при добавлении записей в журнал истории методом «pushState», происходит автоматический переход на эту новую запись. Добавленные записи «page 1», «page 2» и «page 3» сопровождались сменой адреса текущей стра-

ницы на новый. Последняя команда «back()» вернула предпоследнюю запись из журнала, которую мы и увидели при загрузке страницы.

Проверьте работоспособность кнопок «Back» и «Forward». При движении назад по истории крайней записью (первой страницей в истории) будет страница с адресом «js4_1.html?record=0», а не «js4_1.html». Хотя первым мы открывали именно файл «js4_1.html», данная запись в журнале истории была заменена командой «replaceState» в скриптовой части страницы.

Небольшое замечание: если при работе с примером будут допущены ошибки, то после их исправления необходимо заново открыть файл в новой вкладке. Обновление текущей вкладки исправит ошибки, но не сотрет историю. При этом работа кнопок может отличаться от описанной выше.

Коллекция Frames. Управление фреймами

В своем составе HTML-страницы могут содержать фреймы — элементы, отображающие другие страницы.

При помощи фреймов на веб-странице часто реализуется взаимодействие со сторонними ресурсами, например, в отдельных фреймах размещается видео из ресурса «YouTube» или средства картографии «Google Maps» (детальнее о размещении плейеров или карт можно узнать по следующим ссылкам https://developers.google.com/youtube/player_parameters, https://developers.google.com/maps/documentation/embed/start). На рисунке 10 приведен пример размещения фрейма с видео в составе файла *js4_3.html* (*файл доступен в папке Sources*).

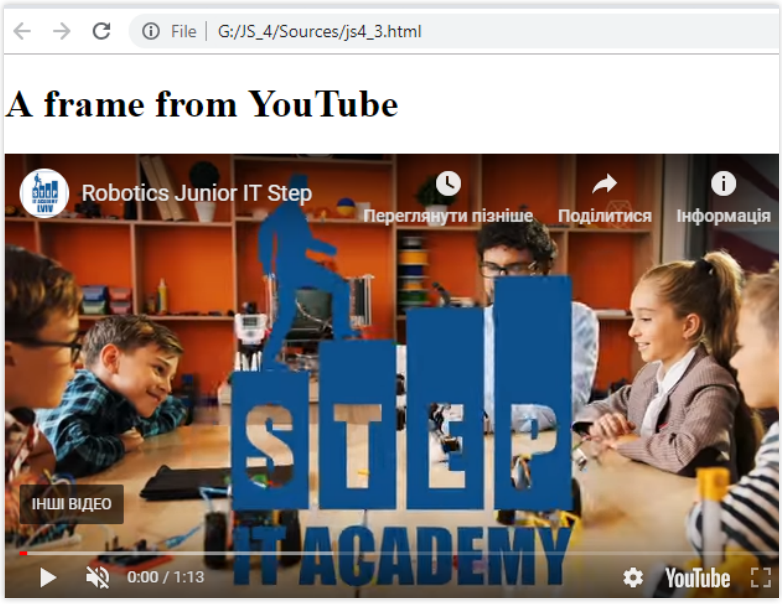


Рисунок 10

Напомним, что фреймы создаются тегом <iframe>, атрибут «src» которого указывает адрес страницы, загружаемой во фрейм. Тег является парным, то есть требует закрывающего тега:

```
<iframe src='https://itstep.org'></iframe>.
```

Фреймы являются «окнами в окне» — для них, так же как и для основной страницы, создается объект «window» со всеми свойствами и методами, со своей моделью BOM.

Все фреймы страницы хранятся в специальном объекте-коллекции «window.frames». Количество фреймов можно узнать из величины «window.frames.length». Доступ к элементам коллекции осуществляется по индексам: «window.frames[0]», «window.frames[1]» и т.д.

При создании фрейма ему может быть задано имя при помощи атрибута «name»:

```
<iframe name='itstepFrame'
        src='https://itstep.org'>
</iframe>
```

В таком случае доступ к элементу коллекции «window.frames» может быть осуществлен по имени фрейма в объектном стиле «window.frames.itstepFrame». Коллекция «window.frames» доступна только для чтения, изменять ее элементы не разрешается.

Для иллюстрации работы с коллекцией «window.frames» рассмотрим следующую программу. Создайте новый HTML-файл, наберите или скопируйте в него следующее содержимое (*код также доступен в папке Sources, файл js4_2.html*). При наборе обратите внимание на комбинирование двойных и одинарных кавычек при создании обработчиков «onclick».

```
<!DOCTYPE HTML>
<html>

<head>
    <title>Frames object</title>
    <style>
        iframe{
            border:1px solid navy;
            height:300px;
            margin:30px;
            width:300px;
        }
```

```
        button{
            margin:10px 150px;
        }
    </style>
</head>

<body>
    <iframe src="about:blank"></iframe>
    <iframe src="about:blank"></iframe>
    <iframe src="about:blank" name="frame3"></iframe>
    <button onclick="window.frames[0].
            location='https://itstep.org'">
        IT Step
    </button>
    <button onclick="window.frames[1].
            location='https://mystat.itstep.org'">
         Mystat
    </button>
    <button onclick="window.frames.frame3.
            location='https://quiz.itstep.org'">
         Quizes
    </button>
</body>

</html>
```

Откройте созданный файл в браузере. При загрузке страницы создаются три фрейма со специальной «пустой» страницей «src="about:blank"». Не рекомендуется создавать фреймы без указания источника, так как они могут быть проигнорированы браузерами при загрузке страницы.

Для наглядности фреймам установлены одинаковые размеры и рамки при помощи стилевых определений в заголовочной части программы. Ниже фреймов созда-

ются три кнопки. Внешний вид страницы должен соответствовать приведенному на рисунке 11.

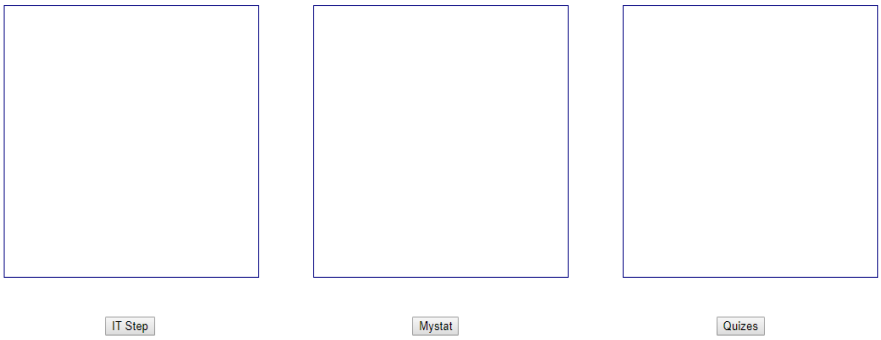


Рисунок 11

При нажатии на кнопки происходит обращение к коллекции «window.frames». Поскольку элементы этой коллекции обладают всеми атрибутами окна, используем их объект «location» для переходов на нужный адрес. Например, при нажатии первой кнопки для первого фрейма устанавливается адрес следующей командой:

```
window.frames[0].location='https://itstep.org'
```

Аналогичные команды созданы и для остальных кнопок. Для третьего фрейма использовано обращение по имени. При создании фрейма ему указан атрибут «name="frame3"», а в обработчике события нажатия кнопки его объект «location» меняется командой:

```
window.frames.frame3.location='https://quiz.itstep.org'
```

Нажмите на кнопки и убедитесь в том, что во фреймы загружаются указанные страницы.

Если есть такая необходимость, можно обеспечить обратный обмен данными: из фрейма к родительскому окну. Для этих целей предназначается объект «window.parent», который указывает на окно, в котором был создан данный фрейм. Таким образом, у всех элементов коллекции «window.frames» родительским элементом является текущее окно. Для проверки этого утверждения откройте консоль разработчика на странице с текущим примером и наберите в ней сравнение:

```
window.frames[1].parent === window
```

Как результат получим ответ «true». Проверьте данное условие для остальных членов коллекции.

В случае, если страницы, открытые в дочерних фреймах, имеют свои собственные фреймы, то реализуется более глубокая, иерархическая вложенность фреймов. Для определения самой первой, корневой страницы в таких структурах может быть использован объект «window.top».

Следует отметить, что не все сайты могут быть открыты во фреймах на других сайтах. Классически, считается что фреймы предназначены для отображения дополнительных страниц одного и того же сайта, то есть страниц с одним и тем же доменным именем. Описанная ситуация называется «кросс-доменными ограничениями» (*англ. «same origin policy»*). Подобные ограничения действуют не только на фреймы, но и на некоторые другие способы обмена данными, например, на AJAX-коммуникации, которые мы будем рассматривать в следующих уроках.

Кросс-доменные ограничения вводятся с целью безопасности, чтобы сайты не выдавали себя за другие

либо не пользовались их функциями для своих целей. Например, злоумышленник может создать фрейм размером на всю страницу с сайтом некоторого банка. При этом у него появляется возможность узнать о данных, которые пользователь будет вводить в любые поля, поскольку ввод осуществляется на самом деле на поддельной странице, содержащей фрейм.

Менее опасно, но также противоречит авторским правам использование лицензионных функций других сайтов, выдавая их за свои. Например, на сайте могут быть собраны прогнозы погоды, средства поиска, новостные анонсы от разных ресурсов, что будет поднимать популярность такого сайта за счет чужих разработок.

Если какой-либо сайт устанавливает кросс-доменное ограничение, то он не будет отображаться в фреймах сайтов с другими доменными именами. В то же время, как мы видели выше, определенные ресурсы не вводят ограничений и даже предлагают инструкции и указания по созданию фреймов для других сайтов. Перед тем как использовать фрейм необходимо убедиться, что загружаемая в него страница не содержит указанных ограничений.

Добавьте еще один фрейм с кнопкой к нашему примеру. Используйте в качестве источника фрейма пустую страницу, а в кнопке укажите новый адрес «https://www.google.com». После обновления страницы и нажатия на кнопку мы увидим предупреждение в консоли и страницу-ошибку во фрейме (рис. 12).

Из увиденного делаем вывод о том, что сайт «https://www.google.com» устанавливает кросс-доменные ограничения и не может быть помещен во фрейм стороннего сайта.


Google
Elements
Console
Sources
Network
top
Filter
Refused to display 'https://www.google.com/' in a frame because it set 'X-Frame-Options' to 'sameorigin'

Рисунок 12

Домашнее задание

1. Создайте HTML-страницу, разместите на ней две кнопки. По нажатию на первую должно открываться новое окно с адресом «https://itstep.org». Размеры нового окна должны быть 640x480. По нажатию на вторую кнопку новое окно должно закрываться.
2. Создайте HTML-страницу, при загрузке которой будет отображаться список установленных в системе языков. Предпочтительный язык выделите жирным шрифтом.
3. Создайте HTML-страницу, разместите на ней кнопку. По нажатию на кнопку должно выводиться количество записей в истории браузера. Проверьте работоспособность кнопки, переходя на страницу с различными параметрами (exc3.html?page = 1, exc3.html?page = 2, exc3.html?page = 3).
4. Модифицируйте пример из раздела «Фреймы». Добавьте кнопки управления историей («Назад» и «Вперед») для каждого фрейма под ним.