Browser Object Model

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:

■■ 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 другихвкладок.

■■ Обращаться к серверу при помощи JavaScript можно только в том случае, если сайт был загружен именно с этого сервера

■■ Командами JavaScript нельзя закрыть или запустить браузер, свернуть его главное окно или изменить его размеры.

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

Объект Window. Открытие, перемещение

и изменение размера окон

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 методы объекта .window., позволяющие управлять самими окнами,

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

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()**

**newWin=window.open("https://google.com", “G","resizable");**

Как результат ее выполнения появится новое окно

(а не вкладка), в котором начнется загрузка страницы

.https://itstep.org.. Второй аргумент .StepWin., передан-

ный 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.

**newWin.close()**

Задавать положение и размер окна необходимо при

его создании. После того как новое окно будет создано

и управление вернется в консоль старого окна, возмож-

ность их взаимного влияния будет ограничена. Как мы уже

видели, закрыть новое окно из старого можно, посколь-

ку оно его и создавало. А вот поменять размеры и положение может оказаться затруднительным.

**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 (500х500 пикселей) в левом верхнем углу экрана (точнее, с отступами 50х50 пикселей от левого верхнего угла).

**Попробуйте самостоятельно создавать новые окна**

**с другими адресами. Используйте методы относительного характера .moveBy. и .resizeBy. для управленияположением и размерами новых окон.**

Объект Navigator. Параметры браузера

Полезную информацию в объекте .navigator. содер-

жат поля, отвечающие за настройки браузера такие, как

предпочтительный язык, наличие сетевого подключения

или установки разрешений на использование Cookie.

appCodeName appName connection - типе сетевого подключения

languages

**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.

Наиболее достоверными и часто применимыми свой-

ствами 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, соответствен-

но), а также .availWidth. и .availHeight. — отвечающие

за доступные (от англ. available) размеры, то есть за сво-

бодную часть окна, без учета панели задач, строк состояния, док-панелей

**Объекты Location и History. Перемещение по 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 теку-

щей вкладке страницу с другим адресом.

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

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

2) <http://portquiz.net:8080/?page=1>

У объекта .location. также существует несколько ме-

тодов, предназначенных для управления адресом отобра-

жаемой страницы. Они используются для перезагрузки

(обновления) страницы или для переходов на другие

адреса. Основные методы объекта .location. приведены

в следующей таблице

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")

Объект браузерной модели .history. хранит в себе

историю посещенных ранее страниц в данной вкладке.

При помощи этого объекта можно программным спосо-

бом управлять переходами к ранее просмотренным стра-

ницам или возвратами от них к тем, что были просмо-

трены позже. Другими словами, этот объект позволяет

.листать. историю просмотров. Также, объект .history.

позволяет менять историю — добавлять или модифицировать записи в истории просмотров.

Length state back() forward() go(n) - 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

файл js4\_1.html

Во-первых, подменяется запись текущей страницы

командой .replaceState.. В качестве аргументов указы-

ваем:

1. Объект-состояние с одним полем .{‹record':'0'}.

2. Название страницы .page 0.. Это название использу-

ется только для журнала переходов и не является за-

головком страницы, не отображается на вкладке бра-

узера. 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 дол-

жен совершить переход на предыдущую страницу из жур-

нала истории данной вкладки.

Проверьте работоспособность кнопок .Back. и .Forward

.. При движении назад по истории крайней записью

(первой страницей в истории) будет страница с адре-

сом .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.

Коллекция Frames. Управление фреймами

Все фреймы страницы хранятся в специальном объ-

екте-коллекции .window.frames.. Количество фреймов

можно узнать из величины .window.frames.length.. До-

ступ к элементам коллекции осуществляется по индек-

сам: .window.frames[0]., .window.frames[1]. и т.д.

**файл js4\_2.html**

Упомянутые в предыдущем уроке тенденции различ-

ных производителей создать наилучший браузер (так

называемые войны браузеров) приводят к тому, что каж-

дый браузер стараются создать особенным, не таким как

все. Сделать собственную систему управления контен-

том и различный набор ее возможностей. В результате

страдает единообразие и правильность одинакового ото-

бражения страниц на различных браузерах. Для введе-

ния общих правил обработки HTML кода и построения

из него страницы, World Wide Web Consortium (W3C)

подготовил и опубликовал требования, известные нам

под названием .объектная модель документа. (Document

Object Model, DOM).

Что же такое DOM? Это набор требований к тому, как

веб-страница должна быть представлена в виде управляе-

мой информационной системы, каким образом элементы

HTML должны превратиться в программные элементы

(элементы DOM), при помощи каких команд ими мож-

но управлять, на какие события они должны реагиро-

вать и так далее. DOM — это стандарт, требующий от

различных браузеров соблюдать одни и те же правила

обработки HTML-кода, что позволяет разработчикам

писать универсальные инструкции не сильно беспокоясь

о том, что данная часть кода не будет выполнена на других браузерах.

Отличия DOM от BOM

Модели DOM и ВОМ, несмотря на похожие опреде-

ления и аббревиатуры, имеют существенные отличия.

Рассмотренная ранее объектная модель браузера (ВОМ)

является программным .представителем. браузера, вклю-

чая операционную систему, в которой он работает. В свою

очередь, модель документа (DOM) .представляет. саму

веб-страницу, открытую в браузере.

Если одна и та же страница будет несколько раз открыта в разных браузерах или даже в разных вкладках одного браузера, то для них:

а) значения объектов ВОМ могут отличаться. И, скорее всего, будут отличаться — разные браузеры могут иметь различное название, различные версии. В различных вкладках может отличаться история ранее открываемых страниц,

б) значения объектов DOM будут одинаковыми во всех

браузерах, во всех вкладках. Поскольку в них отображается одна и та же страница, ее параметры просто не могут быть различными

Основное предназначение ВОМ — взаимодейство-

вать с браузером и операционной системой, получать

данные о размерах окна, состоянии аккумуляторных

батарей, положении (геолокации) устройства, а также

управлять историей просмотров, переходами между раз-

личными адресами страниц и т.д.

Задачи DOM практически никак не касаются са-

мого браузера и нацелены на построение содержимого

веб-страницы как связанной совокупности отдельных

элементов (блоков, списков, рисунков и т.п.). Изменения,

вносимые в DOM, должны сразу приводить к перестро-

ению страницы, в результате чего пользователь их сразу

увидит на экране своего устройства.

Основой каждой из моделей является один объект.

В ВОМ — .window., в DOM — .document.. В составе

каждого из объектов присутствуют другие объекты, которые, в свою очередь, могут иметь внутреннюю структуру. В итоге схема модели представляет собой некоторое дерево или граф, где от каждого объекта (узла) отходят

ветви к дочерним объектам, от них — к своим и т.д.

* объектная модель документа (Document Object Model, DOM) – для управления содержимым HTML-документа;
* объектная модель браузера (Browser Object Model, BOM) – дополнительные объекты, предоставляемые браузером.