Практическое занятие №14 Глобальный объект Window

Задание №1

Проверьте запущен ли у Вас Open Server. Папка с материалами урока (js14.loc) должна быть размещена в директории /domains сервера. Заходим на http://js14.loc и начинаем урок.

Занятие №2

По сути, понятие «клиентский JavaScript» состоит из нескольких вещей:

1. Объект Window в качестве глобального объекта JavaScript и глобального контекста исполнения сценариев
2. Объект Document, иерархия элементов HTML-страниц и работа с элементами (DOM)
3. Модель событий

Ключевую роль играет объект Window. Именно он является тем самым глобальным объектом Global, в контексте которого и функционируют все сценарии на странице.

Объект Window очень важен для нас, без некоторых его свойств связь JavaScript с нашими страницами была бы невозможна. Среди всех перечисленных свойств особенно важно свойство, что ссылается на объект Document. Именно этот объект позволяет работать с документом, загруженным в окно.

Свойства объекта Window	
Название	Что означает
document	Ссылка на объект Document, в рамках которого реализована иерархия загруженного в окно документа
location	Ссылка на объект Location, отвечающий за адрес открытого в окне документа
history	Ссылка на объект History, связанный с кнопками «Вперед» и «Назад» браузера
navigator	Ссылка на объект Navigator, предоставляющий информацию о браузере
screen	Ссылка на объект Screen, предоставляющий информацию об экране монитора
frames	Ссылка на HTML-коллекцию фреймов, расположенных в загруженном документе (элементы frame и iframe)

Screen. С помощью этого объекта можно получить информацию о разрешении монитора и размерах рабочей области (например, размер экрана за вычетом панелей задач в Windows).

Свойства объекта Screen	
Название	Что означает
width	Ширина экрана монитора в пикселях
height	Высота экрана монитора в пикселях
availWidth	Ширина рабочей области экрана монитора в пикселях

availHeight	Высота рабочей области экрана монитора в пикселях

Задание №3

Location. Этот объект позволяет получить данные об адресе документа, который загружен в текущее окно, и загрузить новый документ.

Свойства объекта Location на примере URL http://www.google.com:80/search?q=javascript&test=test#test		
Название	Что означает	Данные примера
protocol	Протокол	http:
host	Хост и порт	www.google.com:80
hostname	Хост (без порта)	www.google.com
port	Номер порта	80
pathname	Строка пути (относительно хоста)	/search
search	Часть адреса после символа “?”, включая символ “?”. GET запрос.	?q=javascript&test=test
hash	Часть URL, которая идет после символа решетки “#”, включая символ “#”	#test
href	Весь URL	http://www.google.com:80/search?q=javascript&test=test#test

Также объект location имеет несколько методов:

1. assign – загружает в окно новый документ, в истории посещений окна появляется новый переход. В качестве параметра принимает URL нового документа.
2. replace - загружает в окно новый документ, в истории нет нового документа. В качестве параметра принимает URL нового документа.
3. reload – перезагружает текущую страницу.

На практике вместо метода assign(url) применяют более простой подход – присваивают свойству location требуемый URL нового документа.

```
window.location = 'http://google.com';
```

Несмотря на некорректность присваивания объекту строкового значение, этот способ укоренился в практике. Это связано, в том числе, и с тем, что политика безопасности JavaScript запрещает доступ к свойствам и методам глобального объекта другого окна, если он загружен с другого домена.

Также значение нового URL можно присвоить атрибуту href:

```
window.location.href = 'http://google.com';
```

Задание: напишите скрипт, который перенаправляет на эту же страницу, с добавлением в search часть параметра string, который равен 20. Если параметр уже есть, и он равен заданному числу, перенаправление не нужно.

Задание №4

Для того чтобы открыть ссылку в новом окне, нужно воспользоваться методом window.open(strUrl), где strURL – ссылка.

Метод open создает новое окно браузера, аналогично команде "Новое окно" в меню браузера.

Метод open возвращает ссылку на новое окно – глобальный объект window нового окна. Новое окно же ссылается на открывающее в свойстве window.opener.

```
var googleWindow = window.open("https://www.google.com.ua/");
console.log(googleWindow); //глобальный объект нового окна
```

Также open может принимать еще 2 параметра:

winName – имя нового окна для использования в параметре target форм и ссылок.

winParams – необязательный список настроек, с которыми открывать новое окно

window.open(strUrl, winName, winParams);

```
var googleWindow = window.open(
      "https://www.google.com.ua/"
      , "Google"
      , 'toolbar=0,status=0,scrollbars=1,width=626,height=512'
      );
```

Если окно с именем winName уже существует, то вместо открытия нового окна, strUrl загружается в существующее и возвращается ссылка на него. При этом строка параметров не применяется.

Задание №5

History. Этот объект позволяет получить общую информацию об истории посещений в рамках окна. Эта информация мало информативна и просто бесполезна. На заре появления этого объекта он представлял собой массив ссылок посещенных страниц, однако очень скоро такая информация из объекта исчезла в силу конфиденциальности личной информации пользователя.

Объект имеет свойство history.length которое равняется количеству страниц, посещенных в данном окне, а также методы:

1. back - загружает предыдущий документ. Переход в истории посещений на 1 шаг назад. Если переходить некуда, ничего не происходит.
2. forward - загружает следующий документ. Переход в истории посещений на 1 шаг вперед. Если переходить некуда, ничего не происходит.
3. go - загружает предыдущий или последующий документ. Принимает параметр n - количество шагов в истории вперед или назад. go(-1) равносилен методу back, go(1) равносилен методу forward. Если переходить некуда, ничего не происходит.

Задание: напишите скрипт, который возвращает на предыдущую страницу. Если предыдущей страницы нет, перенаправляет на страницу http://js14.loc/task-list/.

Задание №6

Navigator. Объект предоставляет информацию о браузере, в котором открыто окно. Не так давно этот объект использовался активно, однако, в настоящее время его значение для программистов очень и очень небольшое.

Свойства объекта Navigator	
Название	Что означает
appCodeName	Кодовое имя браузера Практически все браузеры вернут “Mozilla”
appName	Имя браузера appVersion Внутренний номер версии браузера и другая служебная информация Внутренний номер, как правило, содержит не тот номер, который принят для обозначения и виден пользователю
userAgent	Информация, содержащаяся в HTTP заголовке USER-AGENT Представляет собой комбинацию appCodeName (или appName) и appVersion
cookieEnabled	Если браузер поддерживает запись cookie, то вернет true В обратном случае false
platform	Аппаратная платформа, на которой работает браузер
onLine	Если включена опция «работать автономно» (меню File браузера), вернет false В противном случае вернет true
plugins	Массив подключенных плагинов
mimeTypes	Массив поддерживаемых mime-types

Задание: если страница загружена в браузере IE (в свойстве userAgent будут подстроки “MSIE” или “ rv:”) перенаправить пользователя на страницу https://www.google.ru/chrome/browser/desktop/

Задание №7

Глобальные переменные и Window. Глобальные переменные представляют собой свойства глобального объекта Window, а функции, декларированные в глобальном контексте (то есть вне других функций), являются методами этого глобального объекта.

```
var x = 5;
function y() {
    var x = 3;
    console.log(x);
    console.log(window.x);
}
y();
```

Задание №8

У объекта Window есть 3 вида диалоговых окон, являющихся модальными. Модальными эти окна называются по причине перехвата управления – невозможно воспользоваться ни одним элементом управления браузера, пока не будет закрыто такое окно. Кроме того, исполнение программного кода «замораживается», пока модальное окно не будет закрыто.

Эти диалоговые окна предназначены для вывода информации, ввода ее и выбора пользователем одного из вариантов. Каждое из этих окон формируется с помощью вызова одного их методов объекта Window.

Первый из данных методов – метод alert(str). Он выводит переданную строку. Если передать объект (переменную), он будет трансформирован в строку:

```
//вывод строки
alert('Hello world');

//вывод объекта
//вызывается стандартный Object.prototype.toString()
alert({x:1});

//вывод массива
//вызывается Array.prototype.toString()
//который аналогичен Array.prototype.join(',')
alert([1,2,3]);
```

Второй - confirm(str). Выбор ответа на переданный вопрос – кнопки «Ok», «Нет». При нажатии на кнопку «Ok» возвращает true. При нажатии на кнопку «Отмена» возвращает false.

```
var a = confirm('Спасти мир?');
console.log(a);
```

Если передать объект (переменную), он будет трансформирован в строку, по тем же правилам что и для alert().

И третий - prompt(str1, str2). Предполагает ввод пользователем ответ на вопрос (str1). Возвращает введенную строку при нажатии на кнопку «Ok» (значение по умолчанию можно передать с помощью str2). При нажатии на кнопку «Отмена» вернет null.

```
var a = prompt('Какой сейчас год?', (new Date()).getFullYear());
console.log(a);
```

Задание №9

Напишите скрипт, который запрашивает у пользователя с использованием prompt имя, фамилию, телефон и e-mail. Запишите введенные данные в объект. После ввода уточните правильно ли введены данные с использованием метода confirm. Если данные введены верно выведите их с использованием alert, если нет, запросите данные повторно.

Задание №10

Таймеры - методы глобального объекта, которые позволяют запустить исполняемый код с задержкой, определяемой в них.

setTimeout(f, t) – где f это исполняемый код, а t – время задержки в миллисекундах. Выполняет исполняемый код один раз с указанной задержкой. Исполняемый код может иметь разные форматы, но наиболее предпочтительные это:

1. Передача анонимной функции

```
setTimeout(function(){
    console.log('hello');
}, 2000);
```

2. Передача функции по имени

```
function test(){
    console.log('hello');
}
setTimeout(test, 2000);
```

Если нужно выполнить функцию с параметрами можно использовать следующий вариант:

```
function test(x, y){
    console.log('x + y = ', x+y);
}

setTimeout(function(){
    test(5, 10);
}, 2000);
```

Или же можно передать аргументы самой функции setTimeout

```
function test(x, y){
    console.log('x + y = ', x+y);
}

setTimeout(test, 2000, 5, 10);
```

setTimeout возвращает timerId – идентификатор созданного таймаута.

Используя функцию clearTimeout и timerId таймаут можно отменить:

```
var timer = setTimeout(function(){
    console.log('hello');

}, 2000);

clearTimeout(timer);
```

Задание №11

Через 10 секунд после загрузки страницы спросите у пользователя хочет ли он быть перенаправленным на сайт партнера. Если он ответит положительно через 2 секунды перенаправьте его на сайт https://google.com.ua, если нет, через 5 секунд спросите хочет ли он чтобы сайт партнера был открыт в новой вкладке. Если он ответит положительно через 2 секунды откройте в новой вкладке сайт https://google.com.ua, если нет, больше не стоит беспокоить пользователя вопросами.

Задание №12

setInterval(f, t) – где f это исполняемый код, а t – время интервала в миллисекундах. Выполняет исполняемый код постоянно с указанным интервалом. Исполняемый код имеет те же форматы что и для функции setTimeout:

```
var i = 0;

setInterval(function(){
    console.log(i++);
}, 100);
```

Также, как и setTimeout, возвращает intervalId – идентификатор созданного интервала.

Используя функцию clearInterval и intervalId интервал можно отменить.

Задание №13

Каждую минуту спрашивайте у пользователя нужна ли ему консультация. Если он ответит положительно перенаправьте его на сайт https://google.com.ua. Если он ответит отрицательно 3 раза, перестаньте задавать ему вопросы.