Лабораторная работа № 13

Тема работы: Работа с элементами BOM средствами JavaScript. Обработка событий браузера

1. Цель работы

Формирование умений взаимодействия с программой просмотра веб-страниц с помощью BOM.

2. Задание

Выполнить задания в соответствии с порядком выполнения лабораторной работы.

3. Оснащение работы

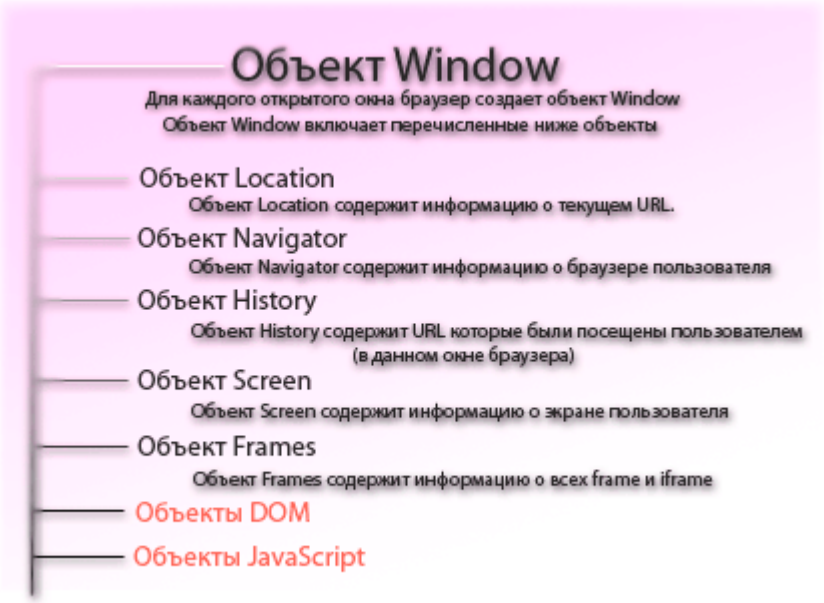
ПК, редактор исходного кода, браузер.

4. Основные теоретические сведения

BOM-интерфейс позволяет работать с окном браузера как с объектом, у которого имеются свои свойства и методы. Данным объектом является объект window (он же является глобальным объектом в JavaScript).

Некоторые браузеры открывают страницы не в отдельных окнах, а во вкладках одного окна. В этом случае каждая вкладка также имеет свой собственный объект window. Однако, при обращении к свойствам, которые не могут относиться к вкладкам (например, изменяющие размер окна браузера), автоматически используется общий объект window окна браузера. Например, если внутри отдельной вкладки происходит вызов метода window.resizeTo(), тогда window ссылается на общий объект окна браузера.

BOM включает в себя следующие объекты:



Объект **window** кроме глобальных объектов (document, screen, location, navigator и др.) имеет собственные свойства и методы.

К объекту Window нельзя обратиться напрямую, однако он имеет свойство window, ссылающееся на сам объект, которое можно использовать вместо ключевого слова this для ссылки на глобальный объект.

Свойства объекта Window

Свойство	Описание
closed	Возвращает логическое значение (true или false) в зависимости от того, закрыто указанное окно или отрыто.
frames	Возвращает массив всех фреймов на странице (включая iframe). `document.write(frames[0].name`
document	Возвращает объект Document данного окна.
history	Возвращает объект History данного окна.
length	Возвращает количество фреймов (включая iframe), которые находятся в данном окне. `document.write(length);`
location	Возвращает объект Location данного окна.

name	Устанавливает или возвращает имя данного окна.
	`win = window.open('','Новое окно','');`
	` win.name`

navigator	Возвращает объект Navigator данного окна.

opener	Возвращает ссылку на окно, которое открыло данное.
	`//Создадим новое окно win`
	`win = window.open();`
	`//Выведем в новое окно текст`
	`win.document.write("Текст нового окна");`
	`//Теперь выведем текст в родительское окно с помощью свойства opener`
	`win.opener.document.write("Текст родительского окна");`

parent	Возвращает родительское окно данного окна.
	`//Создадим новое окно win`
	`win = window.open();`
	`//Выведем в новое окно адрес родительского окна`
	`win.document.write(parent.location.href);`

screen	Возвращает объект Screen данного окна.
self	Возвращает текущее окно. `/* Выведем адрес текущего окна */` `document.write(self.location.href + " ");` `//Теперь выведем адрес родительского окна фрейма` `document.write(self.parent.location.href);`
top	Возвращает верхнее браузерное окно для данного окна. `document.write(top.location.href+' ');` `/* Отобразим адрес текущего окна (т.к результат выполнения кода отображается во фрейме он и является текущим окном) */` `document.write(location.href);`

```
//Определим переменную newwin
```

```
var newwin;
//Зададим функцию winop, которая создает окно newwin по нажатию кнопки
function winop()
{
newwin=window.open();
}
//Зададим функцию checkwin, которая проверяет в каком состоянии находится окно
newwin
function checkwin()
{
/* Если переменная newwin false это значит, что окно newwin не было создано
и переменная осталась не задействована, в этом случае выведем соответствующее
сообщение */
if (!newwin)
document.getElementById("mes").innerHTML = "Окно newwin не было открыто.";
else
/* Если переменная newwin не равна false проверим его свойство closed если оно
true
значит окно уже закрыли если true то окно открыто в данный момент времени */
if (newwin.closed)
document.getElementById("mes").innerHTML = "Окно newwin было закрыто.";
else
document.getElementById("mes").innerHTML = "Окно newwin открыто в данный
момент.";
}
```

Методы объекта Window

Метод	Описание
alert()	Вызывает окно оповещения, которое содержит текст сообщения и клавишу ОК.

blur()	Делает окно неактивным.
	`win = window.open();`
	`win.document.write("Я новое окно");`
	`win.blur()`

clearInterval()	Прекращает повторное выполнение кода заданного setInterval().
clearTimeout()	Отменяет запланированное методом setTimeout() выполнение кода.

close()	Закрывает окно.
	`win = window.open();`
	`if (win != null) { win.close(); }`

confirm()	Вызывает окно подтверждения содержащее текст сообщения и клавиши ОК и Отмена.
focus()	Делает окно активным. `win.focus()`
moveBy(x,y)	Смещает окно относительно его текущей позиции. `win.moveBy(200,100);`
moveTo(x,y)	Перемещает окно на указанную позицию. x - указывает координату, на которую будет перемещен верхний левый угол окна по оси x (по горизонтали) y - указывает координату, на которую будет перемещен верхний левый угол окна по оси y (по вертикали).

	`win.moveTo(500,400);`

open(url,имя,параметры)	Открывает новое окно браузера.
	Имя:
	имя устанавливает имя окна
	_blank URL загружается в новом окне
	_parent URL загружается в родительском фрейме
	_self URL заменяет текущую страницу
	Параметры
	height устанавливает высоту окна в пикселях. Минимальное значение 100.
	location отвечает за отображение адресной строки (yes - отображать, no - неотображать).
	menubar URL отвечает за отображение верхнего меню (yes - отображать, no - неотображать).
	resizable отвечает за возможность изменения размеров окна (yes - можно изменять размер, no - нельзя изменять размер).
	scrollbars отвечает за отображения полосы прокрутки (yes - отображать, no - неотображать).
	width устанавливает ширину окна в пикселях. Минимальное значение 100.
	`win = window.open('','','width=300,height=100');`
	`var win = window.open("http://mrk-bsuir.by ", "My Window", "width=400,height=500,menubar=yes,toolbar=no,location=yes,scrollbars=yes");`

print()	Распечатывает содержимое текущего окна.
prompt()	Вызывает окно запроса, побуждающее посетителя ввести в него определенные данные.
scrollBy(x,y)	Прокручивает содержимое окна на указанное количество пикселей. `scrollBy(0,-100);`
scrollTo()	Прокручивает содержимое окна до указанных координат. `scrollTo(0,960);`
setInterval()	Вызывает функцию или выполняет код через определенные промежутки времени (указанные в миллисекундах).
setTimeout()	Вызывает функцию или выполняет код после указанного количества миллисекунд один раз.

Метод **setTimeout** предназначен для вызова функции один раз через указанный интервал времени.

Функция setTimeout имеет следующий синтаксис:

```
let timeoutID = window.setTimeout(func, [delay, param1, param2, ...]);
let timeoutID = window.setTimeout(code, [delay]);
```

где:

timeoutID – числовой идентификатор, который может использоваться функцией clearTimeout() для отмены таймера;

func – функция, которая будет выполнена. Ссылка на функцию указывается без оператора вызова «()», иначе в setTimeout будет передан результат выполнения функции;

code (альтернативный синтаксис) – блок кода, который будет выполнен. Код в виде строки передавать не рекомендуется, т.к. такой способ затрудняет чтение (а также сопровождение и/или отладку), внутри используется eval(), что является небезопасным и, наконец, этот способ медленнее остальных, т.к. вызывает интерпретатор JS;

delay – задержка в миллисекундах (1000 мс = 1 с), по истечении которой будет выполнена функция. По умолчанию равна 0;

param1, param2 – параметры, которые будут переданы в качестве аргументов указанной функции.

```
function WOW() {
  console.log('WOW');
}

setTimeout(WOW, 1500);
```

```
function sum(mem1, mem2) {
    sum= mem1+mem2;
  console.log (`${sum}`);
}

setTimeout(sum, 1000, "2", "3");  // 23
```

```
setTimeout(()=>console.log('WOW'), 1500);
```

Код, выполняемый в setTimeout, выполнятся в отдельном контексте выполнения. Так как в функции setTimeout значением this является глобальный объект window.

```
let hockeyplayer = {
    firstName: "Slava",
    getamplua: function(){
        console.log(`My name is ${this.firstName}" , I'm forward`);
    }
};

hockeyplayer.getamplua();                 // My name is Slava" , I'm
forward
setTimeout(hockeyplayer.getamplua, 1000); // My name is undefined",
I'm forward
```

Для явного задания значения this можно использовать функцию bind().

```
setTimeout(hockeyplayer.getamplua.bind(hockeyplayer), 500);
```

Функция setTimeout возвращает числовой **id**, который может использоваться функцией **clearTimeout()** для отмены таймера.

```
let timernum = setTimeout(...);
```

```
clearTimeout(timernum);

let timernum = setTimeout(() => con-
sole.log("таймер будет отменен"), 1000);
console.log(timernum);

clearTimeout(timernum);
console.log(timernum);
```

Функция **setInterval()** неоднократно вызывает функцию или выполняет переданный код с указанной временной задержкой между каждым вызовом. Функция будет вызываться на исполнение до тех пор, пока не будет закрыто окно с веб-страницей или вызвана функция **clearInterval()**, прерывающая работу **setInterval()**.

Метод setInterval имеет такой же синтаксис как setTimeout:

```
let timernum = setInterval(func[, delay, param1, param2, ...]);
let timernum = setInterval(code[, delay]);
```

Все аргументы имеют такое же значение. Но отличие этого метода от setTimeout в том, что функция запускается не один раз, а периодически через указанный интервал времени.

```
let timernum = setInterval(() => alert('tick-takc'), 5000);

let timernum = setInterval(() => alert('tick-takc'), 5000);
clearInterval(timernum); // код из таймера не выполнится
```

Чтобы отменить таймер после выполнения нескольких итераций.

```
let timernum = setInterval(() => alert('tick-takc'), 1000);
setTimeout(() => { clearInterval(timernum); }, 5000);

  <p><button onclick="showAlert();">Вызывать alert каждые 3
секунды</button></p>
  <p><button onclick="clearAlert();">Отменить вызов alert</button></p>
```

Помимо **setInterval** регулярный запуск можно реализовать с помощью, так называемого, рекурсивного **setTimeout.**

Свойства объекта window могут записываться без префикса window.

Объект **window.location** может использоваться для получения адреса (URL) текущей страницы и перенаправления браузера на новую страницу.

Методы объекта Location

Метод	Описание
assign()	Загружает новый документ в текущее окно (вкладку) браузера. window.location.assign("https://msiter.ru")
reload()	Перезагружает документ с текущем URL-адресом. Может принимать один аргумент булева типа: true - перезагрузка страницы с сервера, false - перезагрузка страницы из кэша браузера.

replace()	Заменяет текущий документ на документ, расположенный по адресу, переданному методу в качестве аргумента. Отличие от метода assign() заключается в том, что после использования replace() текущая страница удаляется из истории посещения текущего окна, это значит что пользователь не сможет вернуться на нее нажав кнопку назад в браузере.

Свойства объекта Location

Свойство	**Описание**
Hash	Задает или получает часть URL-адреса, следующую за знаком #.
Host	Задает или возвращает имя хоста и порт URL-адреса.
hostname	Задает или возвращает имя хоста из URL-адреса.
Href	Задает или возвращает полный URL-адрес документа.
Origin	Возвращает строку, содержащую URL-адрес, включающий протокол, домен и порт.
pathname	Строка, содержащая имя файла или путь к текущему файлу от корневой папки.
Port	Содержит строку с портом.
protocol	Задает или получает протокол.
Search	Задает или возвращает часть URL-адреса, следующую за знаком вопроса (?), включая сам знак вопроса.

```
window.location = 'http://www.puzzleweb.ru';
```

Объект **window.navigator** содержит информацию о браузере посетителя страницы.

Объект navigator – это чисто информационный объект, он дает информацию о самом браузере (название, версия, установленные компоненты). Получить объект **navigator** можно как одноименное свойство window:

```
window.navigator
```

Свойства объекта navigator

Свойство	**Описание**
appName	Возвращает имя браузера. Свойство доступно только для чтения. `document.write("Вы используете браузер " + navigator.appName);`
appVersion	Возвращает название браузера, его версию и платформу (ОС) на которой он установлен. Свойство доступно только для чтения. `document.write("Версия Вашего браузера " + navigator.appVersion);`
cookieEnabled	Показывает доступно или не доступно использование cookie файлов в браузере.
geolocation	Получает ссылку на объект Geolocation, который сообщает географическое расположение устройства, на котором запущено приложение.
onLine	Возвращает значение булева типа, значение true для работы браузера в онлайн режиме, значение false для автономного режима работы.

Platform	Возвращает тип платформы (ОС) под которую был скомпилирован данный браузер.
systemLanguage	Показывает, какой язык по умолчанию используется операционной системой.
userAgent	Возвращает дополнительную информацию о браузере, так же определяет, какой движок использует ваш браузер.

Методы объекта navigator

Свойство	**Описание**
appName	Возвращает имя браузера. Свойство доступно только для чтения. `document.write("Вы используете браузер " + navigator.appName);`
appVersion	Возвращает название браузера, его версию и платформу (ОС) на которой он установлен. Свойство доступно только для чтения. `document.write("Версия Вашего браузера " + navigator.appVersion);`
cookieEnabled	Показывает доступно или не доступно использование cookie файлов в браузере.
geolocation	Получает ссылку на объект Geolocation, который сообщает географическое расположение устройства, на котором запущено приложение.
onLine	Возвращает значение булева типа, значение true для работы браузера в онлайн режиме, значение false для автономного режима работы.
Platform	Возвращает тип платформы (ОС) под которую был скомпилирован данный браузер.
systemLanguage	Показывает, какой язык по умолчанию используется операционной системой.
userAgent	Возвращает дополнительную информацию о браузере, так же определяет, какой движок использует ваш браузер.

Объект **window.history** содержит историю посещенных браузером страниц.

У объекта **History** есть только одно свойство – **length**. Оно возвращает целое число, указывающее количество страниц в истории сеанса работы с текущим окном браузера, включая текущую загруженную страницу. Это свойство можно использовать следующим образом:

```
alert("Количество страниц, просмотренных в текущем окне, равно " +
history.length);
```

Метод **back()** загружает предыдущую страницу из истории сеанса (выполняет тоже действие, что и кнопка назад в браузере). Эквивалентен **history.go(-1)**.

```
history.back();
```

Метод **forward()** загружает следующую страницу в истории сеанса (выполняет тоже действие, что и кнопка вперед в браузере). Эквивалентен history.go(1).

```
history.forward();
```

Метод **go()** загружает страницу из истории посещения текущего окна или вкладки браузера. Этот метод принимает один аргумент: строку или целое число.

Целое положительное число указывает на какое количество страниц вперед нужно перейти, отрицательное число указывает на сколько страниц назад нужно вернуться. С помощью строкового аргумента можно указать точный URL-адрес из истории посещения, на который нужно перейти.

```
history.go(-1);
```

Объект **window.screen** содержит информацию об экране пользователя. Он просто предоставляет сведения о графических параметрах клиентской системы вне окна браузера, такие как ширина и высота в пикселях.

Свойства объекта Screen

Свойство	Описание
availHeight	Возвращает высоту рабочей области экрана системы в пикселях, за исключением элементов пользовательского интерфейса операционной системы (например, панели задач в Windows). `screen.availHeight;`
availWidth	Возвращает ширину рабочей области экрана системы в пикселях, за исключением элементов пользовательского интерфейса операционной системы.
colorDepth	Возвращает целое число: 1, 4, 8, 15, 16, 24 или 32, указывающее количество бит на пиксель для цветов.
Height	Возвращает общую высоту экрана в пикселях. `screen.height;`
pixelDepth	Возвращает целое число: 1, 4, 8, 15, 16, 24 или 32, указывающее количество бит на пиксель для цветов.
Width	Возвращает общую ширину экрана в пикселях. `screen.width`

5. Порядок выполнения работы

1. Создайте 2 элемента на странице (кнопки, текста или ссылки), при нажатии на которые будут создаваться 2 разных окна со следующими свойства-ми:

1-ое: ширина 200, высота 200, не отображение панели инструментов, не отображение панели меню, отображение строки состояния, не отображение адресной строки, не отображение кнопок браузера, отображение полос прокрут-ки;

2-ое: ширина 400, высота 400, отображение панели инструментов, отображение панели меню, отображение строки состояния, не отображение адресной строки, отображение кнопок браузера, не отображение полос прокрутки

2. Добавьте на HTML-страницу ссылку на другую страницу, кликом выполните переход по ссылке. Затем выполните возврат на исходную страницу с помощью JavaScript.

3. Выполните задание по варианту. Варианты представлены в таблице

13.1.

Таблица 13.1 – Варианты заданий

Вариант	Задания
1	1. На HTML-страницу добавьте кнопку. По нажатию на данную кнопку необходимо вывести высоту окна браузера. 2. Выведите на страницу всю информацию о браузере. Реализуйте вывод информации каждые 10 секунд.
2	1. На HTML-страницу добавьте кнопку. По нажатию на данную кнопку необходимо изменить высоту окна браузера. 2. Выведите на страницу версию и название браузера. Напишите сценарий, который выведет указанное сообщение 3 раза.
3	1. На HTML-страницу добавьте кнопку. По нажатию на данную кнопку необходимо вывести ширину окна браузера. 2. Выведите на страницу версию браузера. Добавьте кнопку, по нажатия на которую через 5 секунд указанная информация будет выведена на HTML-страницу любым способом.
4	1. На HTML-страницу добавьте кнопку. По нажатию на данную кнопку необходимо изменить ширину окна браузера. 2. Выведите на страницу название браузера. Выведите указанную информацию 3-мя разными способами, причем один из способов должен выводить информацию с задержкой в 7 секунд.
5	1. На HTML-страницу добавьте кнопку. По нажатию на данную кнопку необходимо узнать, есть ли у окна браузера вертикальная прокрутка 2. На HTML-страницу добавьте кнопку. По нажатию на данную кнопку необходимо изменить высоту окна браузера. Реализуйте изменение размера высоты окна с интервалом в 5 секунд.

4*. Реализуйте с помощью диалогового окна ввод имени пользователя и интервал, через который будет выводиться приветственное сообщение. Напишите сценарий, которой будет выводить указанное сообщение строго в соответствии с указанным интервалом.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

_____

_____

7. Контрольные вопросы и задания

1. Что представляет собой объектная модель браузера (BOM)?

2.Перечислите основные объекты BOM.

3.Опишите свойства объекта Window.

4.Укажите параметры окна, которое открывается при выполнении следующего кода:

msgWindow=window.open("1.html", "displayWindow", "width=300,height=300,location=no, status=yes,toolbar=yes,menubar=no, scrollbars=no,resizable=no")

Как можно закрыть данное окно?

5. В чем отличие между вызовом setInterval и рекурсивным вызовом setTimeout?

8. Рекомендуемая литература

1. **JAVASCRIPT.RU** [Электронный ресурс] / Современный учебник JavaScript – 2007—2020 Илья Кантор. – Режим доступа: https://learn.javascript.ru. – Дата доступа: 04.03.2020.

2. **Никсон, Р.** Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. 4-е изд. – СПб.: Питер, 2018.

3. **Симпсон, К.** ES6 и не только / К. Симпсон. – СПб.: Питер, 2017.

4. **Хавербеке, М.** Выразительный JavaScript. Современное веб-программирование / М. Хавербеке – СПб.: Питер, 2019.