# Лабораторная работа № 9 Объектная модель браузера (BOM). Таймеры

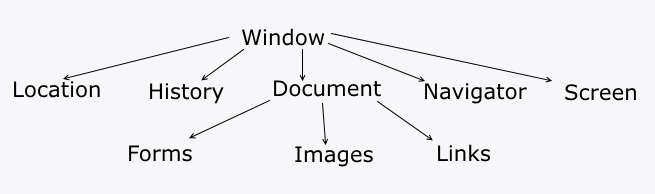
**Цель работы:** изучить объекты BOM, научиться использовать таймеры в JavaScript.

**Краткие теоретические сведения:**

Большое значение в JavaScript имеет работа с веб-браузером и теми объектами, которые он предоставляет. Например, использование объектов браузера позволяет манипулировать элементами html, которые имеются на странице, или взаимодействовать с пользователем.

Все объекты, через которые JavaScript взаимодействует с браузером, описываются таким понятием как **Browser Object Model** (Объектная Модель Браузера).

Browser Object Model можно представить в виде следующей схемы:



В вершине находится главный объект – объект window, который представляет собой браузер. Этот объект в свою очередь включает ряд других объектов, в частности, объект document, который представляет отдельную веб-страницу, отображаемую в браузере.

**Объект window**

Объект window представляет собой окно веб-браузера, в котором размещаются веб-страницы. Window является глобальным объектом, поэтому при доступе к его свойствам и методам необязательно использовать его имя.

Все объявляемые в программе глобальные переменные или функции автоматически добавляются к объекту window.

Ранее были рассмотрены 3 методы объекта window для взаимодействия с пользователем:

* **alert(str)**: показывает сообщение в диалоговом окне.
* **prompt(str, default)**: показывает сообщение и запрашивает ввод текста от пользователя. Возвращает напечатанный текст в поле ввода или null, если была нажата кнопка «Отмена» или Esc с клавиатуры.
* **confirm(str)**: показывает сообщение и ждёт, пока пользователь нажмёт OK или Отмена. Возвращает true, если нажата OK, и false, если нажата кнопка «Отмена» или Esc с клавиатуры.

Все эти методы являются модальными: останавливают выполнение скриптов и не позволяют пользователю взаимодействовать с остальной частью страницы до тех пор, пока окно не будет закрыто.

Объект window также предоставляет методы для работы с окном браузера:

* **window.open ()**: открыть новое окно;
* **window.close ()**: закрыть текущее окно;
* **window.moveTo ()**: переместить текущее окно;
* **window.resizeTo ()**: изменить размер текущего окна.

**Объект history**

Объект history предназначен для хранения истории посещений веб-страниц в браузере. Нам этот объект доступен через объект window.

Все сведения о посещении пользователя хранятся в специальном стеке (history stack). С помощью свойства **length** можно узнать, как много веб-станиц хранится в стеке.

Для перемещения по страницам в истории в объекте history определены методы **back()** (перемещение к прошлой посмотренной странице) и **forward()** (перемещение к следующей просмотренной странице)

Также в объекте history определен специальный метод **go()**, который позволяет перемещаться вперед и назад по истории на определенное число страниц. Положительное число определет перемещение вперед, а отрицательное – назад.

**Объект location**

Объект location содержит информацию о расположении текущей веб-страницы: URL, информацию о сервере, номер порта, протокол. С помощью свойств объекта мы можем получить эту информацию:

* **href**: полная строка запроса к ресурсу;
* **pathname**: путь к ресурсу;
* **origin**: общая схема запроса;
* **protocol**: протокол;
* **port**: порт, используемый ресурсом;
* **host**: хост;
* **hostname**: название хоста;
* **hash**: если строка запроса содержит символ решетки (#), то данное свойство возвращает ту часть строки, которая идет после этого символа;
* **search**: если строка запроса содержит знак вопроса (?), например, то данное свойство возвращает ту часть строки, которая идет после знака вопроса.

Также объект location предоставляет ряд методов, которые можно использовать для управления путем запроса:

* **assign(url)**: загружает ресурс, который находится по пути url.
* **reload(forcedReload)**: перезагружает текущую веб-страницу. Параметр forcedReload указывает, надо ли использовать кэш браузера. Если параметр равен true, то кэш не используется.
* **replace(url)**: заменяет текущую веб-станицу другим ресурсом, который находится по пути url. В отличие от метода assign, который также загружает веб-станицу с другого ресурса, метод replace не сохраняет предыдущую веб-страницу в стеке истории переходов history, поэтому мы не сможем вызвать метод history.back() для перехода к ней.

**Объект navigator**

Объект navigator содержит информацию о браузере и операционной системе, в которой браузер запущен. Он определяет ряд свойств и методов, основным из которых является свойство **userAgent**, представляющее браузер пользователя.

Объект navigator хранит свойство **geolocation**, с помощью которого можно получить географическое положение пользователя. Для получения положения используется метод **getCurrentPosition()**. Этот метод принимает два параметра: функцию, которая срабатывает при удачном запуске, и функцию, которая срабатывает при ошибке запроса геоданных.

function success(position) {

let latitude = position.coords.latitude;

let longitude = position.coords.longitude;

let altitude = position.coords.altitude;

let speed = position.coords.speed;

document.write("Широта: " + latitude + "<br/>");

document.write("Долгота: " + longitude + "<br/>");

document.write("Высота: " + altitude + "<br/>");

document.write("Скорость перемещения: " + speed + "<br/>");

};

function error(obj) {

document.write("Ошибка при определении положения");

};

navigator.geolocation.getCurrentPosition(success, error);

**Таймеры**

Для выполнения действий через определенные промежутки времени в объекте window предусмотрены функции таймеров. Есть два типа таймеров: одни выполняются только один раз, а другие постоянно через промежуток времени.

Для одноразового выполнения действий через промежуток времени предназначена функция **setTimeout()**. Она может принимать два параметра:

let timerId = setTimeout(someFunction, period)

Параметр period указывает на промежуток, через который будет выполняться функция из параметра someFunction. А в качестве результата функция возвращает id таймера.

Для остановки таймера применяется функция **clearTimeout(id)**.

Функции **setInterval()** и **clearInterval()** работают аналогично функциям setTimeout() и clearTimeout() с той лишь разницей, что setInterval() постоянно выполняет определенную функцию через промежуток времени.

Функция **requestAnimationFrame()** действует аналогично setInterval() за тем исключением, что он больше заточен под анимации, работу с графикой и имеет ряд оптимизаций, которые улучшают его производительность. В метод window.requestAnimationFrame() передается функция, которая будет вызываться определенное количество раз (обычно 60) в секунду.

**Практическая часть:**

**Задание 1.** «Написать свой браузер»! На странице в верхней части размещается форма, в нижней части – плавающий фрейм. Форма обеспечивает следующие возможности:

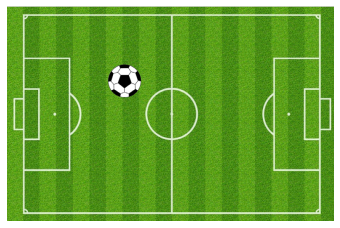
* ввод адреса;
* загрузка страницы в нижний фрейм или новое окно;
* переход назад и вперед по истории страниц нижнего фрейма;
* повторная загрузка страницы в нижнем фрейм.



**Задание 2.** Создать HTML-документ с кнопкой. При подведении курсора мыши к кнопке с любой стороны кнопка должна исчезать и через 3 секунды появляться в случайном месте окна.

**Задание 3.** Создать HTML-документ с футбольным полем, которое занимает всю ширину и высоту экрана, и мячом размером 100 на 100 пикселей.

Сделать так, чтобы при клике мышкой по полю, мяч плавно перемещался на место клика. Учтите: необходимо, чтобы центр мяча останавливался именно там, где был совершен клик мышкой.



**Задание 4.** Создать HTML-документ с кнопкой или блоком div небольшого размера. Написать сценарий, вращающий элемент вокруг текущего положения курсора мыши. При изменении положения курсора, центр вращения так же должен смещаться следом за ним.

**Требования по оформлению отчета:**

Отчет должен включать следующие пункты:

1. Титульный лист.
2. Цель работы.
3. Задание.
4. Листинг программного кода.
5. Результаты выполнения сценариев.
6. Выводы.

**Контрольные вопросы для защиты:**

1. Как определить размеры окна?
2. Как создать новое окно?
3. Как определить адрес загруженного документа?
4. Как с помощью сценария перегрузить документ в текущем окне?
5. Как определить размеры экрана?
6. Какие свойства объекта **location** Вы знаете?
7. Какие свойства объекта **history** Вы знаете?
8. Какие методы объекта **history** Вы знаете?
9. В чем отличие функций **setInterval** и **setTimeout**?
10. Как сбросить таймер?
11. Какие свойства объекта **navigator** Вы знаете?