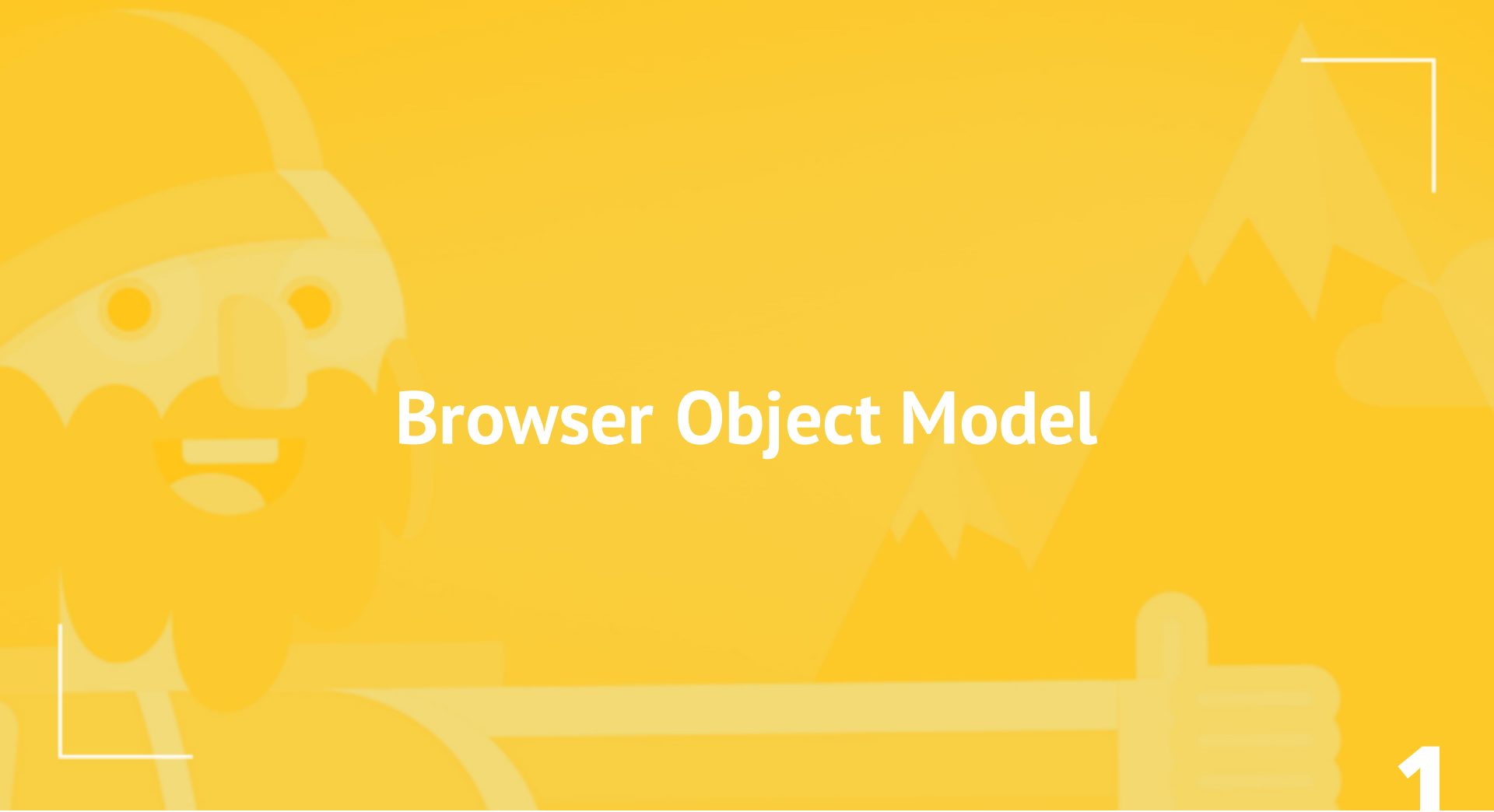
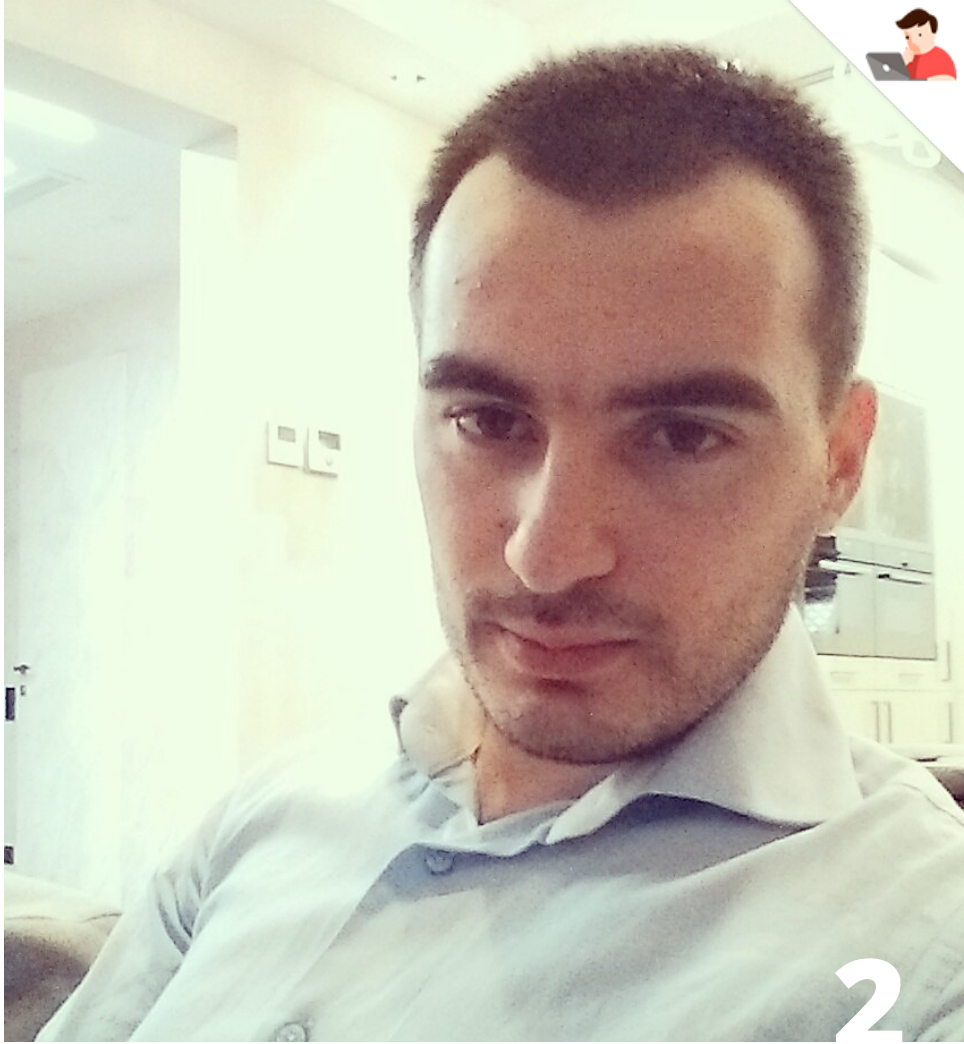

Browser Object Model

Ведущий вебинара

Сергей Мелюков

Круг интересов: Backend, Frontend, GameDev, MobileDev

Место работы: Frontend разработчик профессиональных инструментов Avito



Содержание

Что такое BOM?

Что такое BOM?

JavaScript это просто **язык** программирования, который **не знает** о том, где и как его будут использовать.

BOM - это всего лишь **набор объектов** и методов в них, для работы с **браузером**.

Это своеобразная **прослойка** между JS-кодом и браузером

Основные компоненты BOM

Основные компоненты BOM

window	работа с окнами и контейнер для львиной доли WEB API
screen	работа с экраном
location	работа адресной строкой
history	работа с историей переходов
navigator	информация о браузере
cookie	работа с cookie-хранилищем
таймеры	
диалоги	

window

Помимо прочего...

open(url, name, params)	открывает новое окно по указанному адресу, с указанным именем и параметрами
close()	закрывает окно

Если необходимо открыть **пустое** окно и наполнить его произвольным контентом, то необходимо указать пустой **url** и использовать метод:

document.write(text)	запишет новое содержимое в окно

screen

Содержит информацию об экране:

width	ширина экрана
availWidth	ширина области, в которой ОС позволяет поместить окно
height	высота экрана
availHeaght	высота области, в которой ОС позволяет поместить окно
colorDepth	глубина цвета

location - работа с адресной строкой

href	полное содержимое адресной строки
hostname	имя хоста
pathname	строка *относительно* имени хоста
search	строка запроса (знак **“?”** и всё, что *после* него в адресной строке)
hash	знак “**#**” и всё, что *после* него в адресной строке
assign(url)	перейти по указанному **адресу**
reload()	перезагрузить текущую страницу

history - работа с историей переходов

back()	вернуться на предыдущую страницу истории переходов
forward()	перейти на следующую страницу истории переховод
go(amount)	перейти вперед или назад по истории посещений на указанное **количество** шагов. Можно указывать отрицательное число
Length	сколько переходов сохранено в истории

Так же, у **history** имеется специальный набор методов, для сохранения произвольных **состояний**. Данный набор методов является частью **HTML 5 API** и будет рассмотрен в одном из следующих занятий.

navigator - информация о браузере

По сути, самыми важными и полезными свойствами являются:

userAgent	содержит информацию об ОС и браузере
language	текущий язык браузера

Значения остальных свойств не претендуют на точность, либо существуют только в целях совместимости.

cookie - локальное хранилище данных

cookie - небольшие текстовые файлы, хранящиеся на устройстве пользователя.

HTTP-протокол устроен таким образом, что веб-сервер **не умеет** запоминать своих клиентов и информацию о них. Соответственно, необходим механизм, который позволяет “напоминать” серверу о клиенте. Таким механизмом является **cookie**.

Размер каждого cookie **ограничен** небольшим значением(значение зависит от браузера).

cookie - локальное хранилище данных

У каждого cookie-элемент есть следующие свойства:

name	имя cookie-элемента
value	значение cookie-элемента (только текст)
domain	имя домена, для которого ставится cookie
path	относительный путь, для которого ставится cookie
expires	время жизни (указывается дата в **UTC**). Если не указать, то cookie-элемент удалится сразу после *закрытия* окна со страницей

cookie - локальное хранилище данных

Для того, чтобы **создать** cookie-элемент, необходимо написать:

```
document.cookie  =
‘cookieName=cookieValue’;
```

Это не перезапишет все cookie, а лишь создаст **новый** cookie-элемент.

Дополнительные свойства можно указать, разделив их через “; ”

```
document.cookie  =
‘cookieName=cookieValue;
 path=/some-path’;
```

Получить все **не** просроченные cookie, соответствующие данному хосту и данному пути, можно обратившись к тому же свойству.

В ответ получим список cookie-элементов, разделенных “;”

Таймеры

setTimeout(fn, milliseconds)	выполнить функцию **один раз**, через определенный промежуток времени (возвращает **идентификатор** таймера)
clearTimeout(timerId)	удалить таймер
setInterval(fn, milliseconds)	выполнять функцию, с указанным **интервалом**, пока таймер не будет **удален** (возвращает **идентификатор** таймера)
clearInterval(timerId)	удалить таймер

Диалоги

alert(message)	выводит простейшее окно с сообщением
confirm(question)	показать окно с вопросом. Вернет **true** или **false**, в зависимости от того, что нажал пользователь
prompt(question, default)	показать окно с полем для ввода текста. Если указан параметр **default**, то по умолчанию, в текстовом поле, будет именно это значение. Вернет введенное **значение** или **null**, в случае отмены ввода

Время ваших вопросов