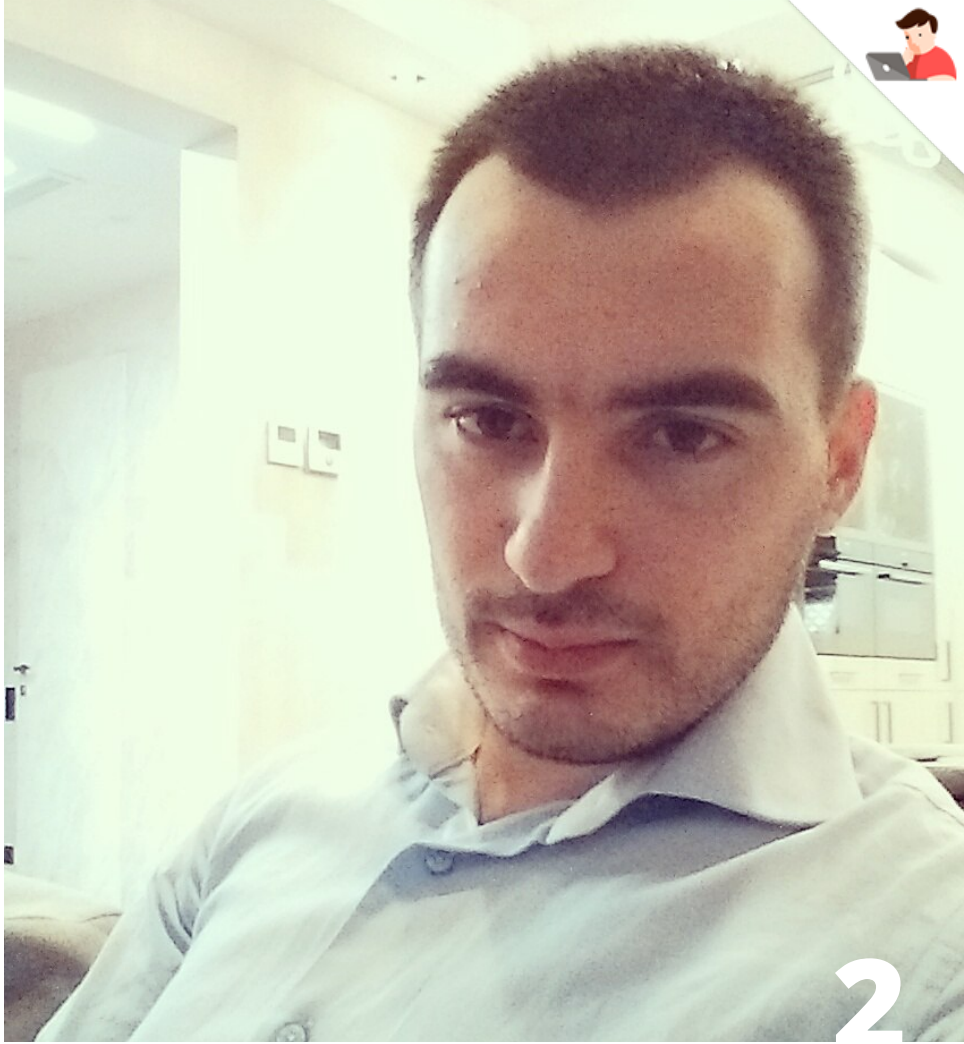

HTML 5 API

Ведущий вебинара

Сергей Мелюков

Круг интересов: Backend, Frontend, GameDev, MobileDev

Место работы: Frontend разработчик профессиональных инструментов Avito



HTML 5 API

HTML 5 API - это набор объектов, расширяющих возможности разработчика, при разработке веб-приложений. Вот некоторые из таких расширений:

DataSet	привязка произвольных данных к HTML-элементам
History states	фиксация состояний приложения и переход между ними
Local storage	key/value хранилище
FileAPI	чтение файлов из файловой системы пользователя
Canvas	рисование и доступ к графическому ускорителю
AudioAPI	низкоуровневое управление звуковыми данными
WebSockets	постоянная связь с сервером
WebWorkers	выполнение кода в отдельном процессе
Geolocation	определение местоположения пользователя

DataSet

Позволяет привязывать к элементу произвольные данные и получать к ним доступ.

Привязка данных осуществляется через атрибуты с именами **data-***

Получить доступ к списку таких атрибутов и их значениям можно через свойство **dataset** любого элемента

History states

Механизм, который позволяет запоминать состояния приложения и восстанавливать их без перезагрузки страницы. Используется для создания SPA(Single Page Application).

history.pushState(data, title, url) создать новое состояние

Для отслеживания перехода по состояниям, необходимо добавить обработчик события **popstate**.

Local/session storage

Key-value	хранилище данных, привязанное к домену.
Local storage	постоянное хранилище. Время жизни данных не ограничено.
Session storage	временное хранилище, которое очищается при закрытии страницы (при завершении сессии)

Доступ осуществляется через свойства **localStorage** и **sessionStorage** соответственно.

FileAPI

Дает возможность читать содержимое файла в режиме реального времени.

File	представляет собой файл
FileReader	набор методом для чтения файла.

Для обработки результата, необходимо использовать обработчики событий:

- **load**
- **progress**
- **error**
- **abort**

FileAPI

FileReader.readAsDataURL(file)	прочитать содержимое файла как URL embded
FileReader.readAsText(file)	прочитать содержимое файла как текст
FileReader.readAsArrayBuffer	прочитать содержимое файла как ArrayBuffer
FileReader.readAsBinaryString(file)	прочитать содержимое файла как строку с двоичными данными
File.slice(startByte, endByte)	получить из файла кусок байт, которые потом можно обработать

WebSocket

Средство постоянного соединения с сервером и обмена сообщениями.

WebSocket работает поверх HTTP, следовательно, HTTP-сервер должен поддерживать работу через протокол WebSocket.

Чтобы начать работу через протокол WebSocket, браузер посылает серверу дополнительные заголовки:

- **Upgrade: websocket**
- **Connection: Upgrade**
- **Sec-WebSocket-Key: aXQgaXMgbG9mdHNjaG9vbA==**
- **Sec-WebSocket-Version: 13**

WebSocket

Создание нового постоянного соединения:

```
var socket = new WebSocket('ws://…..');
```

WebSocket

WebSocket.send(data)	отправить данные на сервер
WebSocket.close()	закрыть соединение
событие **open**	соединение открыто
событие **message**	получено новое сообщение от сервера
событие **error**	возникла ошибка
событие **close**	соединение закрыто

Geolocation

Набор функций для определения местоположения пользователя:

navigator.geolocation.getCurrentPosition(callback) - определить местоположение пользователя и вызвать callback, в который будет передана информация о местоположении.

navigator.geolocation.watchPosition(callback) - отслеживать местоположение пользователя и периодически вызывать callback с обновленными данными.

navigator.geolocation.clearWatch(watchId) - отменить отслеживание местоположения.

WebWorkers

Это механизм, позволяющий запускать какой-либо js-код параллельно основному.

Для использования механизма, необходимо создать worker, указав скрипт, в котором содержится исходный код воркера:

```
var worker = new Worker('worker.js');
```

А затем, отправлять ему сообщения с произвольным содержимым при помощи метода **worker.postMessage(...);**

Получать сообщения от воркера можно через обработску события message:

```
worker.addEventListener('message', function() {....});
```

SQL-подобная база данных

Механизм, при помощи которого, можно получить доступ к управлению SQL-подобное БД внутри браузера. Используется одна из реализаций языка запросов SQL - SQLite.

Поддерживаются все основные CRUD-операции.

CRUD - **C**reate, **R**ead, **U**pdate, **D**elete

Время ваших вопросов