Upcoming APIs (Part 2)

Cpt S 489

Washington State University

Storage API

- window.localStorage object
  - instance of Storage object
- MDN status (as of April 2017): (REC) W3C Recommendation
  - Currently supported in all major browsers and even several older ones
- Idea: Storage that persists after closing/leaving the page
  - Much like just saving a file to the local file system
  - But it’s using a file system managed by the browser

Storage API

- To store data: localStorage.setItem(keyName, keyValue);
  - Key name and value are expected to be strings
- Example:

```
localStorage.setItem("mydata”, “This is a saved string”);
```

Storage API

- To retrieve data: localStorage.getItem(keyName);
  - Returns null if the key does not exist
- Example:

```
var savedData = localStorage.getItem("mydata");
if (savedData) {…}
```

Storage API

- Can be used to save user data when you can't or don't want to send data to a server
- Simple to use, but keep in mind all local storage for your app/page is GONE once you call localStorage.clear()
- Probably best to not use as primary or exclusive method of data storage
  - But very simple to use, so it has its place

Typed Arrays

- Typed arrays are standard in ES 6
- Much more like arrays in C/C++
  - In fact, you can even simulate “casting a pointer” to a different type
- Wrap around ArrayBuffer object, which needs to be discussed first

ArrayBuffer

- According to MDN:

"The ArrayBuffer object is used to represent a generic, fixed-length raw binary data buffer. You cannot directly manipulate the contents of an ArrayBuffer; instead, you create one of the typed array objects or a DataView object which represents the buffer in a specific format, and use that to read and write the contents of the buffer."

- ArrayBuffer is allocated with a size, in bytes, and does not change size after that
  - No push, pop, or other operations that would change length/size

Typed Arrays

- Typed arrays are backed by an ArrayBuffer for their storage
- C++ code:

```
void* data = malloc(8);
int* i32Arr = (int*)data;
unsigned char* u8Arr = (unsigned char*)data;
```

- Equivalent JavaScript:

```
var data = new ArrayBuffer(8);
var i32Arr = new Int32Array(data);
var u8Arr = new Uint8Array(data);
```

Typed Arrays

- Given the information from the previous slide, and the code:

```
var data = new ArrayBuffer(8);
var i32Arr = new Int32Array(data);
var u8Arr = new Uint8Array(data);
```

- What do you expect are the values of the following? (discuss)
  - i32Arr.length
  - u8Arr.length

Typed Arrays

- Given the information from the previous slide, and the code:

```
var data = new ArrayBuffer(8);
var i32Arr = new Int32Array(data);
var u8Arr = new Uint8Array(data);
```

- What do you expect are the values of the following?
  - i32Arr.length == 2
  - u8Arr.length == 8

Typed Arrays

```
var data = new ArrayBuffer(8);
var i32Arr = new Int32Array(data);
var u8Arr = new Uint8Array(data);
i32Arr[0] = 17;
```

- What are the values of the following?
  - u8Arr[0]
  - u8Arr[1]
  - u8Arr[2]
  - u8Arr[3]

Typed Arrays

- To answer the previous question, you actually need to know the platform byte order because, as the MDN documentation for Int32Array states: “The **Int32Array** typed array represents an array of twos-complement 32-bit signed integers in the platform byte order.”
- So what are two options for the values of the following? (discuss)
  - u8Arr[0]
  - u8Arr[1]
  - u8Arr[2]
  - u8Arr[3]

Typed Arrays

- First 32-bit integer was set to 17
- If platform used least significant byte first (most likely these days)
  - u8Arr[0] == 17
  - u8Arr[1] == 0
  - u8Arr[2] == 0
  - u8Arr[3] == 0
- If platform used most significant byte first
  - u8Arr[0] == 0
  - u8Arr[1] == 0
  - u8Arr[2] == 0
  - u8Arr[3] == 17

Typed Arrays

- Several different types, any of which can be created from an ArrayBuffer*
  - Int8Array
  - Uint8Array
  - Uint8ClampedArray
  - Int16Array
  - Int32Array
  - Uint32Array
  - Float32Array
  - Float64Array
- * = they also have constructors that can create from a length. You don't have to create the ArrayBuffer first if you use one of these constructors.

Typed Arrays

- Uint8Array vs Uint8ClampedArray
  - Assignments in the clamped array constrain values to the appropriate range
- First question: what is the range of an unsigned 8-bit value? (discuss)

Typed Arrays

- Uint8Array vs Uint8ClampedArray
  - Assignments in the clamped array constrain values to the appropriate range
- Range is [0,255], so assignment like this:

```
var clamped8 = new Uint8ClampedArray(1);
clamped8[0] = 257;
```

makes clamped8[0] == 255

Typed Arrays

- So what if we did the same thing with the non-clamped?

```
var arr8= new Uint8Array(1);
arr8[0] = 257;
```

- What is the value of arr8[0]? (discuss)
  - Should be able to determine what it is and understand why. It's exactly like what would happen with C/C++ code.

Typed Arrays / ArrayBuffer

- Some uses for typed arrays / ArrayBuffer
  - Processing pixel data on a canvas image
  - Loading file data as “raw binary” using FileReader.readAsArrayBuffer()
  - Sending binary data with XMLHttpRequest.send(ArrayBuffer)
- Recall earlier mention: standard in ES6
  - Works fine in client-side JavaScript as well as Node.js

Media Devices

- *var promise = navigator.mediaDevices*.getUserMedia(*constraints*);
  - Editor's draft status, so only a few browsers support it right now
  - MDN Page
- Used to get audio and/or video devices
- If the *constraints* parameter is { audio: false, video: true } then it just requests video devices ("webcam" on most machines)

Webcam

- When the Promise is fulfilled this comes in the form of a MediaStream
  - Let’s take a look at the MDN page
  - See if we can find out how we actually make use of a MediaStream object

Webcam

- MDN page probably didn't give us much
- First simple thing we can do with the media stream is to put a <video> tag on the page
  - While this can be used for playing video files (MP4 and others) it can also serve as a host for live video
- Notice inheritance on MDN page
  - HTMLVideoElement inherits from HTMLMediaElement
  - HTMLMediaElement has srcObject member, which can be set to a MediaStream object!

Webcam

- Demo page shows that this gives us the image from the webcam, but what if we want to do more?
- What if we want to send the image to an external source?
- MDN sample page has several concepts we need to look at…