**Day 4 03 October2023**

**ArrayBuffer**

Basically ArrayBuffer is used to keep binary data. It can be the binary data of an image for example.

The basic binary object is ArrayBuffer – a reference to a fixed-length contiguous memory area.

This is how it is created:

let buffer = new ArrayBuffer(16); // create a buffer of length 16

alert(buffer.byteLength); //16

This allocates a contiguous memory area of 16 bytes and pre-fills it with zeroes.

**NOTE!!** ArrayBuffer has nothing common with Array.

* It has a fixed length, we cant increase or decrease it.
* It takes exactly that much space in the memory.
* To access individual bytes, another view object is needed, not buffer[index].

ArrayBuffer is a memory area. What’s stored in it? It has no clue. Just a raw sequence of bytes.

**TypedArray**

Typed arrays are a special kind of array-like object introduced in JavaScript to efficiently work with binary data. They are used for handling raw binary data, such as audio buffers, image data, or networking protocols.

TypedArray is the common term for these views (Uint8Array, Uint16Array, Uint32Array and Float64Array), because they share the same set of methods and properties.

There are 5 variants of arguments:

* new TypedArray(buffer, [byyteOffset], [length]);
* new TypedArray(object);
* new TypedArray(typedArray);
* new TypedArray(length);
* new TypedArray();

To access the ArrayBuffer, there are properties:

* arr.buffer – references the ArrayBuffer.
* Arr.byteLength – the length of the ArrayBuffer.

Here’s the list of typed arrays:

* Int8Array, Int16Array, Int32Array – for signed integer numbers (can be negative).
* Float32Array, Float64Array – for signed floating-point numbers of 32 and 64 bits.

**Out-Of-Bounds Behaviour**

Uint8ClampedArray is special in this aspect, its behavior is different. It saves 255 for any number that is greater than 255, and 0 for any negative number. That behavior is useful for image processing.

**TypedArray Methods**

TypedArray has regular Array methods and with notable exceptions.

We can iterate, map, slice, find, reduce, etc.

These methods allows us to copy typed arrays, mix them, create new arrays from existing ones, and so on.

**Data View**

DataView is a special super-flexible untyped view over ArrayBuffer. It allows from accessing data on any offset in any format.

The syntax:

new DataView(buffer,[bytOffset],[byteLength])

* Buffer – the underlying ArrayBuffer. Unlike typed arrays, DataView doesn’t create buffer on its own. We need to have it ready.
* byteOffset – the starting byte of the view (0 bydefault).
* byteLength – the byte length of the view (by default till the end)