Typed Arrays

Arrays are JavaScript objects that can hold a lot of different elements.

```
var complexArr = [1, 5, "2", "Word", {"name": "James"}];
```

Basically what happens in the background is that your browser will automatically give the right amount of memory space for that array. It will also change as needed if you add or remove data.

However, in the world of high performance and different element types, sometimes you need to be more specific on how much memory is given to an array.

Typed arrays are the answer to this problem. You are now able to say how much memory you want to give an array. Below is a basic overview of the different types of arrays available and the size in bytes for each element in that array.

Type	Each element size in bytes
`Int8Aray`	1
`Uint8Array`	1
`Uint8ClampedArray`	1
`Int16Array`	2
`Uint16Array`	2
`Int32Array`	4
`Uint32Array`	4
`Float32Array`	4
`Float64Array`	8

There are two ways in creating these kind of arrays. One way is to create it directly. Below is how to create a 3 length `Int16Array`.

```
var i8 = new Int16Array(3);
console.log(i8);
// Returns [0, 0, 0]
```

You can also create a *buffer* to assign how much data (in bytes) you want the array to take up.

Note
To create typed arrays using buffers, you need to assign the number of bytes to be a multiple of the bytes listed above.

```
// Create same Int16Array array differently
var byteSize = 6; // Needs to be multiple of 2
var buffer = new ArrayBuffer(byteSize);
var i8View = new Int16Array(buffer);
buffer.byteLength; // Returns 6
i8View.byteLength; // Returns 6
console.log(i8View); // Returns [0, 0, 0]
```

Buffers are general purpose objects that just carry data. You cannot access them normally. To access them, you need to first create a *view*.

```
i8View[0] = 42;
console.log(i8View); // Returns [42, 0, 0]
```

Note
Typed arrays do not have some of the methods traditional arrays have such as `.pop()` or `.push()`. Typed arrays also fail `Array.isArray()` that checks if something is an array. Although simpler, this can be an advantage for less-sophisticated JavaScript engines to implement them.

First create a `buffer` that is 64-bytes. Then create a `Int32Array` typed array with a view of it called `i32View`.

⚗	Your buffer should be 64 bytes large.
⚗	Your i32View of your buffer should be 64 bytes large.
⚗	Your i32View of your buffer should be 16 elements long.