**Day 1: File and FileReader**

A [File](https://www.w3.org/TR/FileAPI/#dfn-file) object inherits from Blob and is extended with filesystem-related capabilities.

There are two ways to obtain it.

First, there’s a constructor, similar to Blob:

new File(fileParts, fileName, [options])

* **fileParts** – is an array of Blob/BufferSource/String values.
* **fileName** – file name string.
* **options** – optional object:
  + **lastModified** – the timestamp (integer date) of last modification.

Second, more often we get a file from <input type="file"> or drag’n’drop or other browser interfaces. In that case, the file gets this information from OS.

As File inherits from Blob, File objects have the same properties, plus:

* name – the file name,
* lastModified – the timestamp of last modification.

That’s how we can get a File object from <input type="file">:

<input type="file" onchange="showFile(this)">

<script>

function showFile(input) {

let file = input.files[0];

alert(`File name: ${file.name}`); // e.g my.png

alert(`Last modified: ${file.lastModified}`); // e.g 1552830408824

}

</script>

**[FileReader](https://javascript.info/file" \l "filereader)**

[FileReader](https://www.w3.org/TR/FileAPI/#dfn-filereader) is an object with the sole purpose of reading data from Blob (and hence File too) objects.

It delivers the data using events, as reading from disk may take time.

The constructor:

let reader = new FileReader(); // no arguments

The main methods:

* **readAsArrayBuffer(blob)** – read the data in binary format ArrayBuffer.
* **readAsText(blob, [encoding])** – read the data as a text string with the given encoding (utf-8 by default).
* **readAsDataURL(blob)** – read the binary data and encode it as base64 data url.
* **abort()** – cancel the operation.

The choice of read\* method depends on which format we prefer, how we’re going to use the data.

* readAsArrayBuffer – for binary files, to do low-level binary operations. For high-level operations, like slicing, File inherits from Blob, so we can call them directly, without reading.
* readAsText – for text files, when we’d like to get a string.
* readAsDataURL – when we’d like to use this data in src for img or another tag. There’s an alternative to reading a file for that, as discussed in chapter [Blob](https://javascript.info/blob): URL.createObjectURL(file).

As the reading proceeds, there are events:

* loadstart – loading started.
* progress – occurs during reading.
* load – no errors, reading complete.
* abort – abort() called.
* error – error has occurred.
* loadend – reading finished with either success or failure.

When the reading is finished, we can access the result as:

* reader.result is the result (if successful)
* reader.error is the error (if failed).