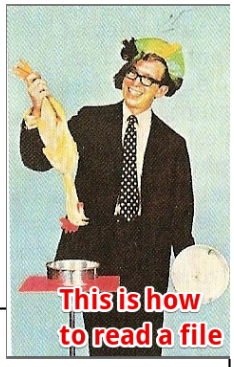
Reading file content

INTRODUCTION / TYPICAL USE



Step1: create a FileReader object

The file API proposes several methods for reading file content, each taken from the`FileReader` interface. Here is how you create a`FileReader` object:

```
var reader = new FileReader();
```

Steps 2 & 3: first call a method of the FileReader object for reading the file content, then get the file content in an `onload` callback

There are three different methods available for reading a file's content: `readAsText`, `readAsArrayBuffer` for binary data and also as `readAsDataURL` (the content will be a URL you will use to set the `src` field of an `<img src=...>`, `<audio>`, `<video>`, and also with all existing methods/properties that accept a URL).

All these methods take as a unique parameter a `File` object (for example, a file chosen by a user after clicking on a `<input type=file>` input field). Below, we use, as an example, the`readAsText` method:

```
function readFileContent(f) {
   // Executed last: called only when the file content is
loaded, e.target.result is
   // The content
   reader.onload = function(e) {
        var content = e.target.result;
        // do something with the file content
        console.log("File " + f.name + " content is:
" + content);
```

```
    };
10.    // Executed first: start reading the file asynchronously,
    will call the
11.    // reader.onload callback only when the file is read
    entirely
       reader.readAsText(f);
    }
```

The above code shows how a file can be read as text. The function is called, for example by clicking on the button corresponding to a `<input type="file" id="file" onchange="readFileContent(this.files)"/>`, and by choosing a file.

- *Line 12* is executed first, and asks the `Reader` object to read the file `f` as text. As this takes some time, it's an asynchronous operation that will be executed by the browser in the background. When the file is read, the`reader.onload` callback function is called.
- *Line 4* is executed after line 12, and is called only when the file content is available. This callback takes an event `e` as a unique parameter, and `e.target.result` is the file content.

Try a variation of the above code in your browser, that displays the file content in a text area. This example is detailed further in the course. Click and select a text file below:

Choose a text file: Choose File No file chosen

In following next course sections, we will look at different examples that read file contents as text, dataURL and binary.