PCW

PROGRAMACIÓN DEL CLIENTE WEB

API File

Dept. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Universitat d'Alacant
Universidad de Alicante

API File

- Proporciona una **API para representar objetos de tipo fichero** en aplicaciones web.
- Incluye las siguientes **interfaces**:
  - `FileList`. Lista de ficheros, como la que proporciona el elemento `<input type="file">`.
  - `Blob`. Representa datos *raw* en binario. Permite acceder a rangos de bytes dentro del fichero.
  - `File`. Proporciona acceso a la información de los atributos de sólo lectura del fichero: *nombre* y *fecha de última modificación*.
  - `FileReader`. Proporciona métodos para leer un fichero y un modelo de eventos para obtener la información leída.
  - `URL`. Permite referenciar los ficheros en aplicaciones web.

API File

Interfaz `FileList`

```
Interface FileList{
    readonly attribute unsigned long length;
    getter File? item(unsigned long index);
};
```

- **`length`**: Devuelve el número de ficheros en el objeto `FileList`.
- **`item(`*`index`*`)`**: Devuelve el fichero que se encuentra en la posición indicada por *`index`* en el array de ficheros representado por el objeto `FileList`.

Ejemplo:

```
let fichero = document.querySelector('form>input[type="file"]').files[0];

// También se puede acceder mediante una sintaxis alternativa:
// let fichero = document.querySelector('form>input[type="file"]').files.item(0);

if(fichero)
{
  // Se realizan las operaciones sobre el fichero
}
```

API File

Interfaz Blob

```
Interface Blob{
  constructor(optional sequence<BlobPart> blobParts,
              optional BlobPropertyBag options = {});

  readonly attribute unsigned long long size;
  readonly attribute DOMString type;

  // obtener un rango de bytes (desde start a end) del objeto Blob
  Blob slice(optional [Clamp] long long start,
             optional [Clamp] long long end,
             optional DOMString contentType);

  // leer desde el objeto Blob
  [NewObject] ReadableStream stream();
  [NewObject] Promise<USVString> text();
  [NewObject] Promise<ArrayBuffer> arrayBuffer();
};

enum EndingType { "transparent", "native" };

dictionary BlobPropertyBag {
  DOMString type = "";
  EndingType endings = "transparent";
};

typedef (BufferSource or Blob or USVString) BlobPart;
```

API File

Interfaz `Blob` *(y II)*

Ejemplo:

```
// Crear un nuevo objeto Blob
let a = new Blob();

// Crear un ArrayBuffer de 1024 bytes
let buffer = new ArrayBuffer(1024);

// Crear objetos ArrayBufferView objects basados en buffer
// (buffer también podría proceder de la lectura de un objeto File)
let shorts = new Uint16Array(buffer, 512, 128);
let bytes = new Uint8Array(buffer, shorts.byteOffset + shorts.byteLength);

let b = new Blob(["foobarbazetc" + "birdiebirdieboo"], {type: "text/plain;charset=utf-8"});

let c = new Blob([b, shorts]);

let a = new Blob([b, c, bytes]);

let d = new Blob([buffer, b, c, bytes]);
```

API File

Interfaz File

```
Interface File : Blob{
  constructor(sequence<BlobPart> fileBits,
              USVString fileName,
              optional FilePropertyBag options = {});
  readonly attribute DOMString name;
  readonly attribute long long lastModified;
};

dictionary FilePropertyBag : BlobPropertyBag {
  long long lastModified;
};
```

Ejemplo:

```
// Imprimir en consola el nombre y fecha de última modificación de un fichero seleccionado
let fichero = document.querySelector('input[type="file"]').files[0],
    fecha   = new Date(fichero.lastModified);
console.log("Fichero seleccionado: " + fichero.name);
console.log("Fecha última modificación: " + fecha.toDateString());

// Generar un fichero especificando la fecha de última modificación:
let d = new Date(2020, 12, 5, 16, 23, 45, 600);
let ficheroGenerado = new File(["Rough Draft ...."],
                               "Draft1.txt",
                               {type: "text/plain", lastModified: d});
```

API File

Interfaz FileReader

```
Interface FileReader : EventTarget{
  constructor();
  // métodos de lectura asíncronos
  undefined readAsArrayBuffer(Blob blob);
  undefined readAsBinaryString(Blob blob);
  undefined readAsText(Blob blob, optional DOMString encoding);
  undefined readAsDataURL(Blob blob);

  undefined abort();
  // estados
  const unsigned short EMPTY = 0; // no ha empezado la lectura
  const unsigned short LOADING = 1; // se está leyendo un objeto File o Blob
  const unsigned short DONE = 2; // se ha producido fin de lectura, error o abort

  readonly attribute unsigned short readyState;
  // datos tipo File o Blob
  readonly attribute (DOMString or ArrayBuffer)? result;

  readonly attribute DOMException? error;
  // atributos de manejadores de eventos
  attribute EventHandler onloadstart; // cuando empieza la carga
  attribute EventHandler onprogress; // mientras es está realizando la carga
  attribute EventHandler onload; // cuando se obtiene el resultado al finalizar la carga
  attribute EventHandler onabort; // cuando se aborta la operación
  attribute EventHandler onerror; // cuando se produce un error
  attribute EventHandler onloadend; // cuando se termina la carga
};
```

API File

Interfaz `FileReader` *(y II)*

Ejemplo: Previsualizar una imagen

HTML

```
...
<input type="file" onchange="previsualizarImagen(this);"><br>
<img id="preview" src="" height="200" alt="Previsualización de la imagen">
...
```

JavaScript

```
function previsualizarImagen(inpFile){
  let img    = document.querySelector('#preview'),
      file   = inpFile.files[0],
      reader = new FileReader();

  reader.onload = function(){
    // la operación de lectura ha finalizado correctamente
    img.src = reader.result;
  }

  if(file)
    reader.readAsDataURL(file);
  else
    img.src = "";
}
```

API File

Interfaz URL

```
partial interface URL{
  static DOMString createObjectURL((Blob or MediaSource) obj);
  static undefined revokeObjectURL(DOMString url);
};
```

- `createObjectURL(obj)`: Crea una url a partir del objeto que se le pasa. Esta url puede ser utilizada en una aplicación web.
- `revokeObjectURL(url)`: Destruye la url creada para el objeto, haciendo que ya no exista más en la aplicación web y liberando la memoria. El navegador elimina los elementos URL automáticamente cuando se cierra el documento html que los creó.

Ejemplo: Previsualizar una imagen

HTML

```
<input type="file" onchange="previsualizarImagen(this);"><br>
<img id="preview" src="" height="200" alt="Previsualización de la imagen">
```

JavaScript

```
function previsualizarImagen(inpFile){
  let img = document.querySelector('#preview);

  if(inpFile.files[0]){
    img.onload = function(){ URL.revokeObjectURL( img.src ); };
    img.src = URL.createObjectURL( inpFile.files[0] );
  }
}
```