APIs de HTML5

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

gsyc-profes (arroba) gsyc.urjc.es

Abril de 2022

HTML 5 define algunas nuevas APIS. Podemos englobar en este conjunto otras tecnologías que aunque en rigor no son parte de HTML5, están muy relacionadas y definidas por el W3C

- Web Storage
- Web Workers
- Geolocation
- Canvas
  API para dibujar imágenes vectoriales 2D y *bitmap*.
  Hay librerías como jCanvas que facilitan su uso
- Web Messaging
  Para comunicación entre documentos de distintos orígenes, de manera razonablemente segura y sin las limitaciones de la *same-origin policy*
- y algunas otras
  `https://en.wikipedia.org/wiki/HTML5#New_APIs`

Web Storage

Web Storage es una API del W3C que permite almacenar información en el navegador en una forma que generalmente resulta más conveniente que las cookies

- Espacio reservado para este almacenamiento depende del navegador, normalmente entre 2.5Mb y 5 Mb por origen
  - Mucho más que las cookies, limitadas a 4 K
- La información de *Web Storage* la escribe el navegador y permanece en el navegador, solo se envía al servidor si se programa explícitamente
  - Las cookies las escribe el servidor y luego viajan en cada petición, consumen ancho de banda.

Por todo ello

- Si la información la necesita el cliente, *Web Storage* es mucho más conveniente
- Si la información la necesita el servidor, *Web Storage* no es tan útil, porque el servidor no tiene acceso directamente a ella

`https://stackoverflow.com/questions/3220660/local-storage-vs-cookies`

Web Storage es una estructura de diccionario: clave-valor
Incluye dos mecanismos:

- *sessionStorage*
  Un diccionario que dura tanto como la sesión. Se borra al cerrar el navegador
- *localStorage*
  Un diccionario persistente, el diccionario se mantiene aunque se cierre el navegador. Solo se borra si el usuario o la página lo borran explícitamente

Cada uno de estos diccionarios es distinto para cada origen (protocolo, host, puerto)

Almacenamiento de un valor

```
localStorage.setItem(clave) = valor;
sesionStorage.setItem(clave) = valor;
```

- El valor y la clave han de ser cadenas
- Si se intenta usar otro tipo de datos para el valor, muchos navegadores lo convierten a cadena
- Es preferible convertir el dato en JSON explícitamente

Recomendación:

- Normalmente un programa tendrá que almacenar varias variables. Nada impide guardar cada una de ellas por separado un una llamada a `setItem`
- Pero problemente es más conveniente almacenar todos esos valores como propiedades de un único *plain object*, y guardar ese objeto en una única llamada `setItem`

Recuperación de un valor

```
localStorage.getItem(clave);
sesionStorage.getItem(clave);
```

- Estos métodos devuelven el valor asociado a la clave
- O *undefined* si la clave no existe

Borrado de valores

```
localStorage.removeItem(clave)
sesionStorage.removeItem(clave)
```

- Borran la clave
- Devuelven *undefined*, tanto si la clave existía como si no

```
localStorage.clear()
sesionStorage.clear()
```

- Borran el diccionario completo del origen actual (protocolo, host, puerto)

Este programa pregunta al usuario su nombre solamente la primera vez que se ejecuta

```
'use strict'
let nombreUsuario = localStorage.getItem('nombreUsuario');
if (!nombreUsuario) {
  let input= prompt("¿Cómo te llamas?");
  localStorage.setItem('nombreUsuario',input);
} else {
  alert("Hola " + nombreUsuario);
}
for (let clave in localStorage) {
  let valor = localStorage[clave];
  console.log(clave+": "+valor)
}
```

http://ortuno.es/localStorage.html

Este programa borra el nombre de usuario, si estaba definido

```
let clave = 'nombreUsuario';
let valor=localStorage.getItem(clave);
if (valor){
    localStorage.removeItem(clave);
    alert(clave+ ' valía '+valor+ '. Ahora lo he borrado');
}else{
    alert(clave+" no definido");
};
```

http://ortuno.es/localStorage2.html

Referencias

- https://en.wikipedia.org/wiki/Web_storage
- https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API

Web Workers

Un *web worker* es un programa en JavaScript que se ejecuta en el navegador web en segundo plano, independiente de la página principal

- Es útil para realizar tareas intensivas en CPU
- Es útil para aprovechar los ordenadores multi núcleo (prácticamente todos los actuales)
- Se crea como instancia del objeto/la función `Worker()`, que recibe como argumento el nombre del script
- El web worker se comunica con el documento mediante el paso de mensajes: basta asignar el manejador a la propiedad *onmessage* del worker
- Se puede finalizar desde la página principal invocando al método `.terminate()` del worker
- Por motivos de seguridad, algunos navegadores como Chrome no permiten que se ejecuten en páginas cargadas localmente, solo en aquellas servidas desde un web

El siguiente ejemplo calcula dos números aleatorios, y cuando coinciden, envía un mensaje

```
'use strict'
function random(x){
    return Math.floor((Math.random() * x) + 1);
}

let tamanio=1000000;
let x,y;
let c=100;
while (c>0){
    x=random(tamanio);
    y=random(tamanio);
    if (x===y) {
          postMessage(x);
          c-=1;
    }
}
```

http://ortuno.es/web_worker.js.txt

```
// Chrome no permite ejecutar un web worker localmente,
// tiene que estar servido desde web
'use strict'
let w;

let texto_salida = document.querySelector("#texto_salida");

if (typeof(Worker) !== "undefined") {
  if (typeof(w) == "undefined") {
    w = new Worker("js/web_worker.js");
  }
  w.onmessage = function(event) {
    texto_salida.textContent = event.data;
  };
} else {
  console.log("Web workers no soportados");
}
```

http://ortuno.es/web_worker.html

Geolocation

Los navegadores modernos pueden conocer, si el usuario lo permite, su ubicación geográfica
Posiblemente el más preciso es Google Chrome

- En dispositivos con GPS, la información proviene del GPS
- En conexiones cableadas, obtiene información a partir de la dirección IP
- En conexiones WiFi
  - Obtiene las MAC de los Access Point WiFi colindantes
  - La compara con la base de datos de Google de la posición de los Access Point
- En conexiones de telefonía móvil
  - Compara el identificador de la celda con su base de datos de posiciones de celdas de telefonía

¿Cómo obtiene Google la base de datos de Access Point y celdas de telefonía móvil?

- Con los coches que capturan datos para Google Maps. (Aunque esto le causó problemas legales)
- Con la información de los millones de dispositivos Android con GPS

Para aceder a las coordenadas basta usar el método `navigator.geolocation.getCurrentPosition()` pasándole como parámetro

- La función que procesará las coordenadas
- La función que se invocará en caso de error
- Un objeto con opciones

```
let options = {
  enableHighAccuracy: true,
  timeout: 5000,
  maximumAge: 0
};
function success(pos) {
  let x = pos.coords;
  let mensaje = 'Posición actual\n';
  mensaje += 'Latitud :' + x.latitude;
  mensaje += '\nLongitud :' + x.longitude;
  mensaje += '\nPrecisión :' + x.accuracy + " metros";
  alert(mensaje);
}
function error(err) {
  console.warn(`ERROR(${err.code}): ${err.message}`);
};
navigator.geolocation.getCurrentPosition(success, error, options);
```

http://ortuno.es/geoloc.html