DWEC - REPASO JAVASCRIPT

JAVASCRIPT_EJ1.HTML

Diseña un programa que solicite a un usuario en una sola petición su nombre, dos apellidos y año de nacimiento (por ejemplo: «Pepe Perez Sanchez 1990». El nombre y los apellidos serán una única palabra cada uno, pero no hace falta que lo valides. Sobre ese nombre deberá mostrar la siguiente información para crear un nombre de usuario:

- Transformación de la cadena a todos mayúsculas.
- Muestra de nombre, apellido1, apellido2 y año, cada cuál en una línea.
- Nombre de usuario: estará formado por la inicial del nombre, el primer apellido, la inicial del segundo apellido y las dos últimas cifras del año de nacimiento, todo en minúsculas (ej. pperezs90).

JAVASCRIPT_EJ1.HTML

Amplia el ejercicio anterior para que además muestre los siguentes datos:

- Número de vocales que contiene la cadena de nombre y apellidos.
- Tamaño total de toda la cadena
- Tamaño de la cadena que contiene, únicamente, nombre y dos apellidos.

ADDEVENTLISTENER

```
elemento.addEventListener("<evento_sin_on>",<funcion>,<false|true>);
```

```
document.getElementById("parrafo").onclick = hazEsto;
document.getElementById("parrafo").onclick = hazLoOtro;
```

```
document.getElementById("parrafo").addEventListener("click", hazEsto);
document.getElementById("parrafo").addEventListener("click", hazLoOtro);
```

SELECCIONAR ELEMENTOS

Hay varias formas de seleccionar elementos:

- A partir de su identificador, mediante getElementById.
- A partir de su clase, mediante getElementsByClassName.
- A partir de su etiqueta, mediante getElementsByTagName.
- A partir de su selector CSS, mediante querySelector.

¡Pero ojo! No es lo mismo utilizar un método que devuelva un único elemento, como es el primer y último caso, que utilizar cualquiera de los otros dos, que devuelven arrays y, por lo tanto, deben ser procesados como tal.

CARTAS.HTML - EJERCICIO EN PAREJAS

Vas a diseñar un juego que consiste en encontrar parejas en 12 cartas con 6 parejas de los personajes de los Simpson. El juego consistirá en lo siguiente:

- La web deberá tener un grid con 3 filas y cuatro columnas de un color, deberá contar también con un GAP de 10px de otro color. Además habrá un cuadro de texto con el valor 0 pero no modificable, este cuadro se localizará encima del grid en el centro de la página, le añadiremos también un título que pondrá "Puntuación".
- Cada celda del grid tendrá 200px de altura.
- Cuando el usuario haga clic sobre una celda, se mostrará una imagen dentro de la celda. La imagen debe tener también 200px de altura.
- Cuando el usuario haga clic sobre otra celda, se mostrará otra imagen.
- Si las dos imágenes son iguales, se mantendrán visibles añadiendoles un pequeño borde de color y se añadirá 1 punto al cuadro de texto.
- Si las dos imágenes son diferentes, se ocultarán mostrando nuevamente el color inicial.
- Si el marcador llega a 6 puntos se mostrará un alert que dirá: "Felicidades has ganado el juego" y reseteará el valor de todas las celdas ocultando todas las imágenes.

COOKIES

```
todasLasCookies = document.cookie; //Mostrar todas las cookies

document.cookie = nuevaCookie; //Escribir una cookie
```

CARTAS.HTML - EXTRA

Si ya has acabado tu juego básico de cartas ahora es hora de mejorarlo añadiendole las siguientes funciones:

- Las cartas se deberán colocar aleatoriamente en diferentes posiciones cada vez que se recargue la página y cada vez que se gane el juego. **(Opcional para subir nota).**
- Habilita un nuevo contador con errores, este contador sumará uno cada vez que el jugador cometa un error al elegir cartas. Al terminar el juego, añadiremos al mensaje de finalización los errores que se han cometido antes de ganar.
- Al iniciar el juego se pedirá via prompt un nick o nombre de usuario. Este nick deberá aparecer cerca del contador de puntuación o del contador de errores.
- Añade un ranking donde saldrá el jugador que ha completado el juego con menos errores. Deberás guardar el nombre del jugador con menos errores (y los errores) haciendo uso de cookies para que aunque cerremos el navegador el ranking siga funcionando.
- Habilita dos botones en la esquina superior izquierda. Uno dirá "ES" otro "EN". Cada botón cambiará el idioma del juego a español o inglés respectivamente. (Solo tenemos que cambiar el título que decía puntuación/score). El idioma debe guardarse en una cookie durante 30 días para que cuando volvamos a entrar al juego este este en el idioma que dejamos guardado.

WEB STORAGE

Web Storage es una API de almacenamiento web que proporciona los mecanismos mediante los cuales el navegador puede almacenar información de tipo clave-valor de una forma mucho más intuitiva que utilizando cookies.

+ Seguro + Sencillo

+ Capacidad de almacenamiento

- sessionStorage
- localStorage

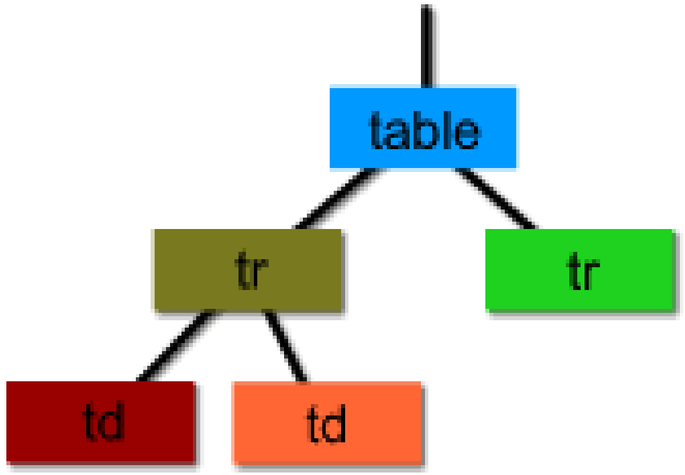
- setItem
- getItem
- removeItem
- clear

CARTAS.HTML - EXTRA WEB STORAGE

Sustituye la función que hacías con las Cookies por las nuevas funciones que hemos aprendido de WeStorage, recuerda usar localStorage en lugar de sessionStorage.

REPASO - MANEJO DEL DOM

```
...
<table>
  <tr>
    <td>
    </td>
    <td>
    </td>
  </tr>
  <tr>
  </tr>
</table>
...
```



Crear, modificar y borrar elementos en un documento:

- createElement
- createTextNode
- appendChild
- replaceChild
- removeChild
- setAttribute

- Seleccionar elemento padre: **parentElement**
- Acceder a todos los hijos: **childNodes children**
- Acceder a primer o último hijo: **firstChild** o **lastChild**
- Encontrar el siguiente hermano, con **nextSibling** y **nextElementSibling**

AJAX - ASYNCHRONOUS JAVASCRIPT AND XML

AJAX es el acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), una técnica de desarrollo web que nos permite crear aplicaciones interactivas que se ejecutan en el lado del cliente (navegador) mientras se realiza una comunicación con el servidor en segundo plano.

El resultado es una petición al servidor con una **obtención de datos sin necesidad de que la página se recargue**, es decir, los datos se cargan en una parte de la página mientras que el resto de información se mantiene invariable, de modo que esa carga no interfiere en la visualización ni comportamiento de la página.

Las ventajas de utilizar AJAX son muchas: se reduce la velocidad de carga, se mejora la interactividad y, sobre todo, la usabilidad de las aplicaciones web.

AJAX - XMLHTTPREQUEST

Mediante AJAX es posible cargar contenido de un archivo de texto de manera asíncrona utilizando una serie de métodos que debemos conocer y que son propios del objeto XMLHttpRequest.

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method, url, async, user, psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

AJAX - XMLHTTPREQUEST

Property	Description
onload	Defines a function to be called when the request is recieved (loaded)
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

AJAX - XMLHTTPREQUEST

```
<script>
    document.getElementById("cambiaContenido").addEventListener("click", cambiaContenido);

    function cambiaContenido() {
        var xhr = new XMLHttpRequest();
        xhr.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("texto").innerHTML = this.responseText;
            }
        };
        /* .open: especifica la solicitud
         - GET/POST.
         - Archivo: txt, php, xml, json, etc.
         - true/false: método de envío. */
        xhr.open("GET", "holamundo.txt", true);
        /* .send: envía la solicitud al servidor.
            Si utilizamos POST debemos pasar los datos por parámetro */
        xhr.send();
    }
</script>
```

CARTAS.HTML - AJAX

Si ya has acabado tu juego básico de cartas ahora es hora de mejorarlo añadiendole las siguientes funciones:

- El idioma se deberá leer desde un archivo XML que estará en nuestro servidor subido como idioma.xml. Debe tener una estructura parecida a la siguiente:
- Dicho XML debemos almacenar al menos lo siguiente:
  - Puntuacion/Score
  - Errores/Errors
  - Clasificación/Top Player/Ranking
  - Mensaje acierto (Genial, las cartas son iguales)
  - Mensaje fallo (Ops, has fallado)
  - Mensaje victoria (Has ganado el juego)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LANGUAGE>
  <EN>
    <SCORE>Score</SCORE>
    <ERRORS>Errors</ERRORS>
  </EN>
  <ES>
    <SCORE>Puntuación</SCORE>
    <ERRORS>Errores</ERRORS>
  </ES>
</LANGUAGE>
```

- Además la descripción del juego se deberá leer de un fichero "descripcion_es.txt" o "descripcion_en.txt" en función del idioma que tengamos. Esta deberá cambiar también si cambiamos el idioma de nuestro juego.

AJAX - JSON - JAVASCRIPT OBJECT NOTATION

Dentro de un JSON nos podemos encontrar 2 tipos de elementos:

- **Arrays**: listas de valores entre corchetes y separadas por comas.
  - **["hola", 3, "Antonio", 1.234]**
- **Objetos**: listas de parejas nombre:valor separados por dos puntos y a su vez cada pareja por comas. Los objetos se escriben entre llaves y los nombres de las parejas siempre entre comillas dobles.
  - **{"nombre": "Antonio", "nacimiento": 1815 }**

- Un documento JSON únicamente contiene un elemento (objeto o array).
- El último elemento de objetos y arrays no puede ir seguido de coma.
- Los espacios en blanco y saltos de línea son irrelevantes.

AJAX - JSON

Los valores en objetos y arrays pueden ser:

- números: enteros, decimales, exponencial, etc.
- cadenas: entre comillas dobles.
- valores true, false y null (sin comillas).
- objetos y arrays: pueden contener a su vez más objetos y matrices sin límite de anidamiento.

+ Corto que XML

+ Rápido de leer y escribir que XML

+ Permite Arrays

AJAX - JSON

Los valores en objetos y arrays pueden ser:

- números: enteros, decimales, exponencial, etc.
- cadenas: entre comillas dobles.
- valores true, false y null (sin comillas).
- objetos y arrays: pueden contener a su vez más objetos y matrices sin límite de anidamiento.

+ Corto que XML

+ Rápido de leer y escribir que XML

+ Permite Arrays

CARTAS.HTML - AJAX - JSON

Si ya has acabado de trabajar con ficheros de texto y XML, es hora de renovar nuestro código y empezar a trabajar con ficheros JSON:

- Ahora leeremos tanto los textos del idioma como los de la descripción del juego del mismo fichero JSON al que llamaremos lang.json. Este archivo tendrá una estructura parecida a la siguiente:

```
{
    "lang": {
        "ES": {
            "Score": "Puntuacion",
            "GameDescription": "Este es un juego de cartas que funciona...",
            "Errors": "Errores"
        },
        "ES": {
            "Score": "Score",
            "GameDescription": "This is a card game that works...",
            "Errors": "Errors"
        }
}
```

Otra alternativa sería crear 2 ficheros:
lang_es.json
lang_en.json
Únicamente con los strings Score, Errors...
Sin necesidad de utilizar objetos.
Usa la opción que más te guste.

REFACTORIZACIÓN DE CÓDIGO + CLEANCODE

Code Refactor o **refactorización** es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

Clean Code, o **Código Limpio,** es una filosofía de desarrollo de software que consiste en aplicar técnicas simples que facilitan la escritura y lectura de un código, volviéndolo más fácil de entender.

https://github.com/andersontr15/clean-code-javascript-es
https://softwareestrategico.com/es/2021/05/26/codigo-limpio-aplicado-a-javascript/

VAR - LET - CONST

```
console.log (greeter);
var greeter = "say hello"
```

```
var greeter;
console.log(greeter); // greeter is undefined
greeter = "say hello"
```

```
var greeter = "hey hi";
var times = 4;

if (times > 3) {
    var greeter = "say Hello instead";
}

console.log(greeter) // "say Hello instead"
```

VAR - LET - CONST

```
let greeting = "say Hi";
let times = 4;

if (times > 3) {
     let hello = "say Hello instead";
     console.log(hello);// "say Hello instead"
 }
console.log(hello) // hello is not defined
```

```
let greeting = "say Hi";
greeting = "say Hello instead";
```

```
let greeting = "say Hi";
let greeting = "say Hello instead"; // error:
```

```
let greeting = "say Hi";
if (true) {
    let greeting = "say Hello instead";
    console.log(greeting); // "say Hello instead"
}
console.log(greeting); // "say Hi"
```

VAR - LET - CONST

```
const greeting = "say Hi";
greeting = "say Hello instead";// error: Assignment to constant variable.
```

```
const greeting = "say Hi";
const greeting = "say Hello instead";// error: Identifier 'greeting' has already been declared
```

```
const greeting = {
    message: "say Hi",
    times: 4
}
```

```
greeting = {
    words: "Hello",
    number: "five"
} // error:  Assignment to constant variable.
```

```
greeting.message = "say Hello instead";
```

DWEC - JQUERY

JQUERY

JQuery es la biblioteca de Javascript más utilizada a día de hoy.
Permite interactuar con los documentos HTML de una manera muy sencilla, pudiendo así manipular el DOM, manejar eventos, añadir animaciones, etc. y facilitando enormemente el trabajo con Javascript.

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
```

```
<head>
<script src="jquery-3.5.1.min.js"></script>
</head>
```

JQUERY VS JAVASCRIPT

```javascript
document.getElementById("cambiar").addEventListener("click", function() {
  document.getElementById("texto").innerHTML = "Hola Javascript";
});

$("#cambiar2").click(function(){
  $("#texto").html("Hola Jquery");
});
```

+ CORTO QUE JS + SENCILLO
+ RÁPIDO DE LEER Y ESCRIBIR

JQUERY - SELECTORES

Syntax	Description
$("*")	Selects all elements
$(this)	Selects the current HTML element
$("p.intro")	Selects all <p> elements with class="intro"
$("p:first")	Selects the first <p> element
$("ul li:first")	Selects the first element of the first
$("ul li:first-child")	Selects the first element of every
$("[href]")	Selects all elements with an href attribute
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"
$(":button")	Selects all <button> elements and <input> elements of type="button"
$("tr:even")	Selects all even <tr> elements
$("tr:odd")	Selects all odd <tr> elements

JQUERY - EVENTOS

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

JQUERY - DOM + CSS

GET Y SET

- **.html()** extrae el código HTML.
- **.html("valor")** añade a un elemento el código html introducido entre paréntesis.
- **.attr("atributo")** extrae la información del atributo cuyo nombre está indicado entre paréntesis.
- **.attr("atributo", "valor")** añade el valor del segundo parámetro al atributo cuyo nombre está indicado en el primer parámetro.

CSS

- **.addClass("nombre de la clase")** añade una clase al elemento.
- **.removeClass("nombre de la clase")** borra una clase del elemento.
- **.toggleClass("nombre de la clase")** añade una clase si no la tiene; de lo contrario, la elimina.
- **.hasClass("nombre de la clase")** devuelve true o false si el elemento tiene la clase.

JQUERY - TRAVERSING (RECORRER)

Antecesores:

- .**parent()**: padre directo.
- **.parents()**: todos los antecesores hasta html.
- .**parentsUntil()**: todos los antecesores hasta uno en concreto.
- **.closest()**: antecesor más cercano que tiene esa etiqueta.

Añadir elementos:

- **.add():** añade elementos (¡ojo! No los crea).

Descendientes:

- **.children()**: hijos directos.
- **.find()**: hijos que cumplen con una condición.
- **.siblings()**: todos los hermanos.
- **.next()** y .**prev()**: siguiente hermano y hermano anterior.
- .**nextAll()** y .**prevAll()**: todos los siguientes hermanos y todos los hermanos anteriores.
- .**nextUntil()** y .**prevUntil()**: el siguiente o anterior hermano hasta uno en concreto.

JQUERY - TRAVERSING (RECORRER)

- **.append, .appendTo .prepend**: añade y mueve elementos como hijos.
- **.before y .after**: añade y mueve elementos como hermanos.
- **.clone()**: clona un elemento.
- **.insertAfter(), .insertBefore()**: inserta un elemento antes o después de otra.
- **.remove()**: borra el elemento y sus hijos.
- **.empty()**: borra el contenido y los hijos del elemento.

CARTAS - JQUERY

- Si tenías 2 repositorios en git diferentes para DIW y DWEC, fusionalos en uno. Los proyectos de cartas deben estar en la raíz de tu repositorio. (No dentro de una carpeta dentro de otra carpeta dentro de otra carpeta...)
- Crea una copia de tu proyecto cartas, a la que llamaras CartasV2 / CartasBS / CartasJ / CartasPro ....
- Sube a tu GIT esta copia para tener claro desde donde empezamos.
- Ahora empezaremos a mejorar nuestro juego de cartas con ayuda de JQuery.
- Sustituye todos tus manejadores "addEventListener" por manejadores JQuery.
- Sustituye tus "getElementById, getElementsByClass, querySelector..." por código JQuery
- **Atento, a medida vas sustituyendo comprueba que todo siga funcionando, no dejes la comprobación para el final.**
- Sustituye los "innerHTML, setAttribute... " por código JQuery.
- Refactoriza tu código siguiendo las indicaciones del profesor.

CARTAS - JQUERY - SONIDO

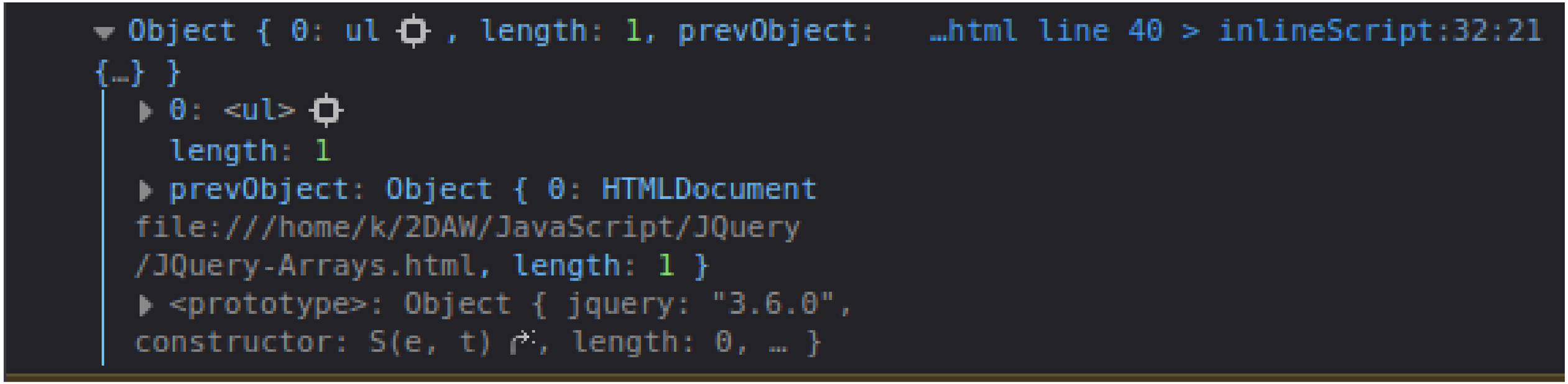
Es hora de poner un poco de sonido a nuestro juego:

- Añade un sonido al seleccionar una carta.
- Añade un sonido para fallo, que se reproducirá si la segunda carta es diferente, es la misma carta, o es una carta que ya esta hacia arriba.
- Añade un sonido para acierto que se reproducirá si la segunda carta es igual.
- Todos los archivos de audio deben estar en la carpeta /audio

JQUERY - OBJETO JQUERY

$("ul")

$("ul")[0]



JQUERY - ESPACIO DE NOMBRES

Objeto JQuery

$.fn

$("div").XXX

Nucleo JQuery

$

$.each

JQuery.each

Hay métodos con el mismo nombre

JQUERY - ARRAYS

Métodos:

- **$.each** y **$.map**: nos permiten recorrer y manipular elementos de un array u objeto.
- **$.inArray:** devuelve el índice de un valor en un array, o `-1` si el valor no se encuentra en el array.
- **$.merge:** permite unir dos arrays.
- **$.grep:** busca un elemento que cumpla con una característica y devuelva true o false.
- **$.makeArray**: convierte un array jQuery en un array Javascript.

Operaciones sobre un selector que devuelve uno o más elementos:

- **.each**: recorre los elementos seleccionados y ejecuta para cada uno de ellos la función pasada por parámetro.
- **.length**: devuelve el número de elementos de un array.
- **.get()**: extrae el elemento de una posición indicada por parámetro.
- **.index()**: extrae el número de un elemento respecto a sus hermanos.

CARTAS - JQUERY - CARTA BOMBA

Es hora de añadir un poco de dificultad a nuestro juego, hasta ahora era muy fácil ganar.

- Añade una pareja de cartas más
- Añade una carta individual a la que llamaremos "bomba" o "cartaBomba", esta carta debe tener imagen de bomba, explosión...
- Si levantamos esta carta, se reiniciará todo el juego, pero sin modificar el contador de errores.
- En total ahora el juego debe tener 15 cartas.
- Añade un sonido de explosión al levantar esta carta.

CARTAS - EXTRA - OPCIONAL

Si se selecciona la dificultad fácil al empezar el juego:

- Añade un botón arriba/al lado/debajo del botón de iniciar/reiniciar juego.
- Este botón tendrá una imagen o un texto del tipo "Ayuda" o "Desvelar".
- Se podrá usar únicamente una vez.
- Al pulsarlo levantará todas las cartas durante 2 segundos y luego las volverá a ocultar.
- Si ya había cartas hacia arriba estas deberán seguir como estaban.

Alternativa para el funcionamiento del botón:

- Al pusar el botón este abrira un modal de boostrap donde aparecerán todas las cartas levantadas. Este modal se cerrará a los 2 segundos.

En el caso de que se seleccione otro tipo de dificultad este botón no debe aparecer.

JQUERY - DATA

```
//MÉTODO DATA
//$.data (nombre,valor) / (nombre) / (objeto)
//$.removeData(nombre)
$("#nombre").click(function () {
    $("div").data("nombre", "ANTONIO");
});

$("#nombre2").click(function () {
    alert($("div").data("nombre"));
});

$("#nombre3").click(function () {
    $("div").removeData("nombre");
});
```

- Guarda información sobre un elemento **en un elemento**.
- No modifica el DOM.
- La info no se ve en el HTML.
- Gasta menos recursos y menos memoria.

CARTAS - JQUERY - STOP CHEAT

Hasta ahora cuando jugábamos a nuestro juego, podíamos encontrar una forma de hacer trampas para ganar. Usuarios más avanzados podrían analizar nuestro código y detectar alguna pista que les llevará a ver cuales cartas seleccionar sin errores, ya sea por medio de la id, background-img, src...
Es hora de evitar esto usando el método DATA que incorpora JQuery.

- Cambia la mecánica donde se comprueba si una carta es igual a otra, ahora usando el método DATA para que ningún jugador avanzado pueda hacer trampas.
- Recuerda que deberás asignar info a cada carta para comprobar si son iguales, igual que antes hacías asignando y comprobando la id, el background-img o el src...

JQUERY - EFECTOS - ANIMACIONES

- **.hide():** oculta el elemento.
- **.show():** muestra el elemento.
- **.toggle():** muestra el elemento si está oculto y viceversa.
- **.fadeIn():** muestra un elemento con un fundido de entrada.
- **.fadeOut():** oculta un elemento con un fundido de salida.
- **.fadeToggle():** oculta un elemento con un fundido de salida si está visible o lo muestra con un fundido de entrada si está oculto.
- **.fadeTo():** realiza un fundido hasta una opacidad indicada con un parámetro numérico entre 0 y 1.
- **.slideDown():** muestra un elemento con un efecto de persiana de arriba a abajo.
- **.slideUp():** oculta un elemento con un efecto de persiana de abajo a arriba.
- **.slideToggle():** muestra u oculta un elemento con un efecto persiana en función de si está o no visible.
- **animate():** https://www.w3bai.com/es/jquery/jquery_animate.html

CARTAS - JQUERY - EFECTOS Y ANIMACIONES

- Añade un efecto o animación para que la acción de "levantar carta" sea más suave, más despacio, o tenga un toque interesante. Es decir, evitar que sea simplemente/directamente cambiar el src. En clase se mostrará un ejemplo básico con un fadeOut y fadeIn.
  - El efecto o los efectos a elegir son totalmente libres.
  - También podéis hacer uso de animaciones.
  - Recuerda que también se pueden concatenar efectos.

- **OPCIONAL**: También puedes añadir efectos o animaciones al acertar, al seleccionar carta errónea, al iniciar el juego, al finalizar el juego... Así haras tu juego más dinámico y visual.

Puedes encontrar muchos más efectos haciendo uso de JQueryUI: **https://jqueryui.com/effect/**

CARTAS - JQUERY

Recuerdas que antes de iniciar el juego debemos fijar un nivel de dificultad:

- Si escogemos dificultad **fácil** iniciaremos el juego con dificultad normal. (Si tenemos realizado el anterior ejercicio opcional, iniciaremos el juego como se indica en este)

- Si escogemos dificultad normal, deberemos mostrar al iniciar el juego durante 2 segundos donde está la bomba.

- Si escogemos dificultad **difícil**, no se mostrará donde se encuentra la bomba.

- **OPCIONAL**: Habilita un cuarto nivel de dificultad, "Legenda", donde se mostrará donde esta la bomba, durante solo 1 segundo. Y si tienes más de 2 fallos pierdes el juego.

CARTAS - JQUERY - AJAX

- Cambia la forma en la que cargabas el lenguaje y hazlo ahora con AJAX de JQuery
  - Lo más sencillo sería que utilizaras el método **getJSON**

- **OPCIONAL:** Cambia el topplayer por un ranking que guardará los 5 mejores jugadores:
  - Puedes utilizar Arrays con el localStorage
  - Si no hay ningun jugador en el ranking lo iniciaremos con los siguientes datos:
    - Jugador : 10 errores
    - Jugador: 20 errores
    - Jugador: 40 errores
    - Jugador: 60 errores
    - Jugador: 80 errores

PROYECTO - TIENDA PRODUCTOS

- Login de usuario
- Login con Google

- Listado de productos paginado
- Insertar / Actualizar / Borrar productos