# Macintosh HD:Users:gloria:Desktop:TRABAJOS:La Florida:@ Florida AF :Florida - Marcas AF:300dpi:Florida - Marca 300dpi-Universitària_c.png GUIÓN DE ACTIVIDAD

# Desarrollo web en entorno cliente

## Actividad Práctica 4: Puzzles

## Objetivos

* *Emplear instrucciones para la generación dinámica de contenido HTML desde el cliente*
* *Emplear instrucciones para la búsqueda de elementos dentro del DOM*
* *Modificación de atributos de elementos del DOM desde JavaScript*

## Temporalización

###### Aproximadamente 10 horas

## Proceso de desarrollo

En esta actividad vamos a desarrollar un pequeño juego de rompecabezas desde JavaScript. Para ello, emplearemos principalmente los métodos y funciones disponibles para manipular el DOM. También emplearemos alguna instrucción para la interacción con el usuario, aunque estas las estudiaremos más adelante.

A continuación os guiaremos en el desarrollo de la actividad práctica. La programación de esta práctica se realizará íntegramente desde JavaScript (fichero puzzle.js) y se emplearán las funciones estándar del DOM y de JavaScript. **El empleo de frameworks, modificación del fichero HTML/CSS, y el uso de módulos externos conllevará una penalización en la nota (-3 puntos).**

1. **Selección de la dificultad del puzzle (0.5 puntos)**

El puzzle será más o menos complicado dependiendo del número de piezas que seleccione el usuario. Nada más comenzar la aplicación del puzzle, al usuario se le solicitará un número de piezas para jugar al puzzle.

* Construye una función llamada getNumberPiecesFromUser:
  + La función no contará con ningún parámetro de entrada
  + La función devolverá un número indicando el número de piezas del puzzle
  + El número de piezas del puzzle deberá de ser un número con raíz cuadrada perfecta (no decimal o entero).
  + Se le solicitará al usuario el número de piezas del puzzle hasta que provea de un número correcto.
  + Se le informará al usuario si el tipo de datos introducido es incorrecto o si el valor introducido no es un entero.
  + <https://mzl.la/2PCJHYG>
  + <https://mzl.la/2OWzTM7>

1. **Funciones de manipulación de la puntuación (1 punto)**

Ahora construiremos una serie de funciones que nos ayudarán a manipular los puntos de la partida. En principio, el usuario partirá con un número de puntos determinados y por cada intercambio de piezas, su puntuación se reducirá en un número determinado de puntos. Por tanto, cuántos menos movimientos realice el usuario, más puntos recibirá.

* Construye una función llamada getMaxScore
  + La función tomará como parámetro de entradas el número de piezas
  + Devolverá el doble del parámetro de entrada
* Construye una función llamada getScore
  + No tomará parámetros de entrada
  + La función recogerá el score actual de la partirda, en formato numérico y lo devolverá
* Construye una función llamada updateScore
  + Tomará como parámetros de entrada la nueva puntuación
  + Cambiará en el HTML la puntuación actual por la nueva puntuación
* Construye una función llamada decreaseScore
  + Tomará como parámetro de entrada un número
  + Actualizará la puntuación en el HTML decrementando su valor actual por el valor del parámetro de entrada (restándolo).

1. **Funciones auxiliares (1 punto)**

Ahora desarrollaremos una serie de funciones auxiliares que nos ayudarán en algunas partes del proyecto.

* Desarrolla una función llamada getNewSizes
  + Tomará como parámetro la anchura y la altura de un elemento
  + Devolverá una nueva anchura y altura teniendo en cuenta que la dimensión (anchura o altura) más grande medirá exactamente 200 y que se debe mantener la relación de aspecto entre la anchura y altura actual.
* Desarrolla una función llamada shuffle
  + Tomará como parámetro un array de objetos
  + Barajará los elementos del array de forma aleatoria (<https://bit.ly/2QUSPIl>)
* Desarrolla una función llamada pieceNumberToRowsColumns
  + Tomará como parámetro un número de pieza del puzzle (del 0 a N-1) y el número total de piezas del puzzle.
  + Devolverá el número de fila y columna correspondiente teniendo en cuenta que el puzzle se organiza en un número de filas y columnas igual a la raíz cuadrada del número de piezas. Además, las piezas irán posicionadas por filas empezando desde arriba y leyendo dichas filas de izquierda a derecha.

1. **Dibujado del puzzle (2 puntos)**

Ahora construiremos una serie de funciones para preparar la partida del puzzle. Estas funciones prepararán los elementos HTML necesarios para mostrar el puzzle al usuario.

* Crea una función llamada createPuzzleLayout:
  + Esta función tomará como entrada el número de piezas del puzzle, la anchura del puzzle, la altura del puzzle, y la dirección de la imagen.
  + La función creará una tabla HTML conformada por un número igual de filas y columnas. El número de las mismas será igual a la raíz cuadrada del número de piezas del puzzle.
  + Cada celda de la tabla tendrá un atributo html de tipo id igual a ‘pieceN’ donde N es el número de pieza del puzzle (comenzando a contar desde 0 y contando por filas de izquierda a derecha).
  + El borde de cada celda será negro, tendrá un grosor de 3 píxeles, y será sólido.
  + Dado que es complicado particionar una imagen en subimágenes empleando JavaScript estándar, cargaremos toda la imagen como fondo en cada celda de la tabla. Conseguiremos más tarde el efecto de pieza real haciendo que cada una muestre una parte de la imagen al desplazar el fondo en cada celda.
  + La tabla se añadirá al cuerpo del documento html.
* Crea una función llamada pieceToOffset
  + Esta función tomará como entrada el número de pieza (0 a N-1), la anchura de la imagen, su altura, y el número total de piezas del puzzle.
  + La función devolverá el desplazamiento del fondo para una pieza determinada del puzzle. Para ello se deberá tener en cuenta que cada pieza del puzzle tiene un ancho y alto determinado, y que además la esquina superior izquierda de la imagen es el punto de referencia (0,0).
* Crea una función llamada createReferenceSolution
  + Esta función tomará como entrada la anchura y altura del puzzle, y el número de piezas del mismo.
  + La función devolverá un array con el desplazamiento (offset) de cada una de las piezas del puzzle
* Crea una función llamada drawContentPuzzle
  + La función tomará como parámetro un array de desplazamientos para cada una de las piezas del puzzle
  + La función cambiará el fondo de cada una de las celdas de la tabla con el desplazamiento indicado.

1. **Lógica del juego (1.5 puntos)**

* Crea una función llamada checkIfSolution
  + La función tomará como entrada la solución del puzzle y el estado actual
  + La función devolverá si el puzzle está terminado
* Crea una función llamada initGame
  + La función tomará como entrada la URL de una imagen y el número de piezas del puzzle
  + El código de la función será:

|  |
| --- |
| function initGame(imageURL,numberOfPieces){  let img = new Image();  img.addEventListener(‘load’,function(){  gameLogic(img,numberOfPieces);  });  img.src = imageURL;  } |

* + Esta función carga dinámicamente una imagen en JavaScript a partir de una URL. Cuando la imagen está cargada en el objeto, se dispara un evento que ejecuta la lógica del juego.
* Crea una función llamada gameLogic
  + La función tomará como entrada un objeto de tipo Image y el número de piezas del puzzle
  + La función inicializará los elementos HTML del juego, creando un punto de inicio aleatorio.
  + Se creará una función (sin parámetros) que se asociará a los eventos de click en cada una de las celdas de la tabla HTML: element.addEventListener(‘click’, f) donde f es la función creada. Dentro de dicha función, this hace referencia al elemento HTML sobre el cual se ha realizado click.
    - Si no se ha seleccionado todavía ninguna pieza, la pieza quedará seleccionada. Esto quedará marcado con un color de borde rojo
    - Si se ha pulsado sobre la pieza ya seleccionada, esta se deseleccionará
    - Si ya hay pieza seleccionada, se procederá a intercambiar las pieza seleccionada con la pieza sobre la cual se acaba de pulsar. Se decrementará la puntuación y se comprobará si el juego ha terminado, en cuyo caso se avisará al usuario y se desasociará el evento de click asociado a todas las celdas de la tabla (e.g., element.removeEventListener(‘click’, f), donde f es la función desarrollada).
* Desde el script principal llama a getNumberOfPiecesFromUser y initGame.

1. **Ampliaciones del juego (2.5 puntos)**

Llegados a este punto, el juego ya debería ser funcional. A continuación proponemos una serie de ampliaciones:

* Cuando una pieza se coloque en su sitio, cambiar el borde a verde e impide que la pieza vuelva a ser seleccionable de nuevo
* Modifica el juego para que, si el usuario no ha cambiado dos piezas en 10 segundos, el juego sugiera un intercambio correcto al usuario visualmente (no emplees alert ni cuadros de diálogos).
  + <https://mzl.la/2ErXmRg>
  + <https://mzl.la/2OqJg78>
* Crea un contador de tiempo de cuenta atrás que llegue de al usuario N minutos para terminar el juego. La partida se pierde si no se completa el puzzle en el tiempo dado.
  + <https://mzl.la/2Cjec1X>
* Permite que se pueda jugar varias veces al puzzle
  + Recuerda como hemos añadido eventos de click anteriormente.
* Crea una tabla de mejores puntuaciones en las que aparezcan las 10 mejores partidas

1. **Profesionalidad del código**

* Realiza testing unitario de las funciones implementadas (aquellas que no manipulan el DOM o HTML) **(0.75 puntos)**
* Indentación de código, estructuración, comentarios **(0.75 puntos)**

1. **Puntos extra: (1 punto)**

* Investiga como puedes emplear Jest para testear código que manipula el DOM y emplea el navegador. Realiza testing unitario para las funciones desarrolladas que manipulan el DOM