Grado en Ingeniería Informática del Software

Software y estándares para la Web

P6. ECMASCRIPT PURO O “VANILLA”

Escuela de Ingeniería Informática
Universidad de Oviedo

Contenido

Objetivos

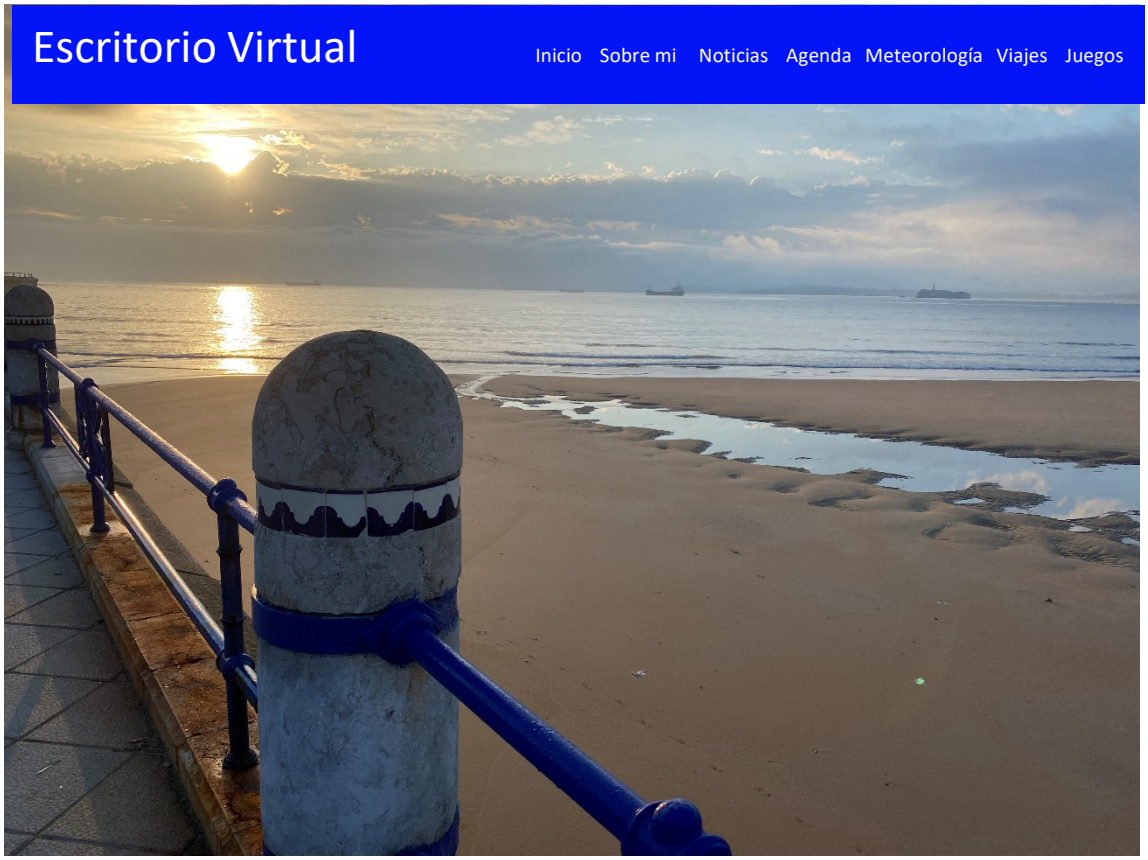
En esta práctica se va a realizar:

- La creación de un documento de script utilizando el estándar ECMAScript
- La creación de objetos para la realización de diferentes tareas en combinación con HTML.
- La generación de código HTML desde el documento de script y su posterior inserción en el documento HTML original
- La validación del código HTML estático y el código HTML generado

IMPORTANTE: Recuerda las pautas de trabajo establecidas en la primera sesión de prácticas (P0. Pautas de trabajo): valida todos los documentos HTML, valida todas las hojas de estilo CSS, comprueba la adaptabilidad y la accesibilidad con las herramientas proporcionadas.

Temática del proyecto: Escritorio virtual (Virtual Desktop)

El proyecto Escritorio Virtual (Virtual Desktop) es evolutivo y será creado, completado y modificado en las diferentes prácticas de la asignatura.



Ejercicio 1: Información de un país con HTML y JS

En este ejercicio se va a utilizar un documento JS para componer un fichero HTML5 que contenga información sobre un país del mundo.

El país sobre el que se debe trabajar viene determinado por todos los dígitos de su UO con la fórmula siguiente:

NÚMERO-PAIS = X MÓDULO 194 +1

siendo **X** todos los dígitos del UO y el **NÚMERO-PAÍS** resultante es de la clasificación del Anexo 1.

Tarea 1. Creación del documento JavaScript

Dentro de la carpeta js del directorio del proyecto Escritorio Virtual crea un nuevo fichero llamado pais.js

Tarea 2. Creación de la clase País

Dentro del fichero creado en la tarea anterior crea la clase País y añade a dicha clase los atributos necesarios para representar la siguiente información: nombre del país, nombre de la capital, cantidad de población, tipo de forma de gobierno, coordenadas de la capital y religión mayoritaria.

Guía para resolver la tarea 2

Para ver como se crea una clase en ECMAScript puede consultarse el ejercicio:

http://di002.edv.uniovi.es/~cueva/JavaScript/68-ES6-Ejemplo-uso-de-clases.html

Tarea 3. Creación de métodos en la clase País

Añade métodos a la clase País que permitan: construir un objeto de dicha clase, inicializar los valores de los atributos, acceder a la información de algunos atributos en forma de texto y escribir información en el documento html.

Guía para resolver la tarea 3

Se deben añadir, al menos, los siguientes métodos:

- Un método constructor que reciba como parámetros los datos de algunos de los atributos creados en el apartado anterior (por ejemplo: nombre del país, nombre de la capital y cantidad de población).
- Un método que rellene el valor del resto de atributos existentes.
- Varios métodos que devuelvan la información principal del país (nombre y capital) en forma de cadena de texto, uno por cada atributo.
- Un método que devuelva la información secundaria del país (población, forma de gobierno y religión mayoritaria) con la estructura de una lista de HTML5 dentro de una cadena.

- Un método que escriba en el documento la información de coordenadas de la capital del país.

Tarea 4. Rellenado inicial del documento meteorologia.html

Modifica el documento meteorologia.html para incluir en él una referencia al fichero pais.js y crea un objeto de la clase País con la información del país resultante de aplicar la fórmula explicada al principio del ejercicio.

Guía para resolver la tarea 4

Utiliza los métodos creados en la tarea anterior para escribir en el documento meteorologia.html toda la información del objeto de la clase País que has creado.

Observa en el siguiente ejemplo cómo se crea la instancia de la clase y como se llama al método correspondiente para escribir la información en el documento html utilizando el método document.write().

http://di002.edv.uniovi.es/~cueva/JavaScript/68-ES6-Ejemplo-uso-de-clases.html

Se aconseja consultar la especificación del método document.write() en:

https://developer.mozilla.org/es/docs/Web/API/Document/write

Recuerda que algunos métodos devuelven información y otros escriben directamente la información sobre el documento.

Observa el siguiente ejemplo para ver las diferentes formas en las que se puede invocar al método *write* del objeto *document* para añadir información al documento html.

http://di002.edv.uniovi.es/~cueva/JavaScript/74-InfoNavegador.html

Se aconseja consultar la especificación del método document.querySelector() en:

https://developer.mozilla.org/es/docs/Web/API/Document/querySelector

Resultado del ejercicio 1

Una vez completado el ejercicio deberían haberse añadido los siguientes ficheros al proyecto del Escritorio Virtual:

- Fichero pais.js en el directorio js del proyecto

Además, se debe haber modificado el contenido de los ficheros meteorologia.html, para incluir la información de la clase Pais, y estilo.css para modificar la presentación de los elementos del fichero anterior (en caso necesario).

Ejercicio 2: Juego de memoria

En este ejercicio se utilizará JavaScript en combinación con HTML y CSS para programar un juego de memoria.

En este juego habrá un grupo de 12 cartas boca abajo y el usuario deberá ir pulsando en las diferentes cartas para voltearlas y hacer parejas. Para conseguir esta funcionalidad, las cartas pasarán por tres estados diferentes: el estado inicial (boca abajo), el estado "flip" (boca arriba) y el estado "revealed" (boca arriba y forma parte de una pareja ya descubierta).

Tarea 1. Creación del menú en juegos.html

Edita el documento juegos.html y añade una sección al inicio del documento que represente el menú de acceso a los diferentes juegos que se van a crear.

Todos los ficheros html en los que se enlace un juego deberán tener esta misma sección de menú en la parte superior, de tal manera que desde un juego se pueda acceder siempre a los demás.

Tarea 2. Creación del documento memoria.html

Crea el fichero memoria.html en el directorio raíz del proyecto de Escritorio Virtual.

Añade al menú creado en la tarea anterior un enlace que permita el acceso al fichero memoria.html

Tarea 3. Creación de un nuevo documento JavaScript

Dentro de la carpeta js del directorio del proyecto Escritorio Virtual crea un nuevo fichero llamado memoria.js.

Añade una referencia a este nuevo fichero en el documento memoria.html

Tarea 4. Creación de la clase Memoria

Dentro del fichero creado en la tarea anterior crea la clase Memoria.

Dentro de dicha clase crea un objeto JSON denominado *elements* que contenga la declaración de los elementos que luego representarán las tarjetas del juego de memoria. Cada tarjeta tendrá un valor *element* con el nombre del elemento y un valor *source* con la dirección url de una imagen que represente a dicho elemento.

Para hacer el juego temático con respecto a los contenidos de la asignatura, utilizaremos los siguientes valores para los nombres de los elementos y las imágenes para las tarjetas del juego:

- HTML5: https://upload.wikimedia.org/wikipedia/commons/3/38/HTML5_Badge.svg
- CSS3: https://upload.wikimedia.org/wikipedia/commons/6/62/CSS3_logo.svg
- JS: https://upload.wikimedia.org/wikipedia/commons/b/ba/Javascript_badge.svg
- PHP: https://upload.wikimedia.org/wikipedia/commons/2/27/PHP-logo.svg
- SVG: https://upload.wikimedia.org/wikipedia/commons/4/4f/SVG_Logo.svg
- W3C: https://upload.wikimedia.org/wikipedia/commons/5/5e/W3C_icon.svg

Guía para resolver la tarea 4

Consultar el siguiente enlace para conocer la estructura interior de un objeto JSON así como para comprender como recorrerlo y obtener la información que almacena:

https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON

Debido a que hay 6 elementos diferentes y el juego constará de 12 cartas, se deben crear dos entradas de cada elemento en el objeto JSON.

Tarea 5. Creación del constructor de la clase Memoria

Crea un constructor para la clase Memoria que cree los siguientes atributos:

- *hasFlippedCard*, el atributo que indica si ya hay una carta dada la vuelta; valor de inicialización: false.
- *lockBoard*, el atributo que indica si el tablero se encuentra bloqueado a la interacción del usuario; valor de inicialización: false.
- *firstCard*, el atributo que indica cuál es la primera carta a la que se ha dado la vuelta en esta interacción; valor de inicialización: null.
- *secondCard*, el atributo que indica cuál es la segunda carta a la que se ha dado la vuelta en esta interacción; valor de inicialización: null.

Tarea 6. Creación de métodos de utilidad en la clase Memoria

Para que el juego funcione correctamente es necesario crear una serie de métodos de utilidad que realicen ciertas actividades: barajar los elementos del objeto JSON, voltear las tarjetas cuando termina la interacción del usuario, resetear el tablero, comprobar si se ha descubierto una pareja y deshabilitar la interacción sobre las cartas que forman parte de una pareja ya descubierta.

Guía para resolver la tarea 6

Se deben crear los siguientes métodos:

- Método *shuffleElements*: este método coge el objeto de JSON y baraja los elementos. Se puede utilizar cualquier método de ordenación para recorrer y barajar los elementos, como por ejemplo el algoritmo Durstenfeld.
- Método *unflipCards*: este método bloquea el tablero en primer lugar y luego voltea las cartas que estén bocarriba y resetea el tablero.
    - **NOTA**: para que la ejecución del volteo de las tarjetas y el reseteo del tablero se realice con cierto margen temporal después del volteo de la segunda tarjeta se puede meter dentro de un delay utilizando el método setTimeout de ECMAScript.
- Método *resetBoard*: pone a null las variables *firstCard* y *secondCard* y pone a false las variables *hasFlippedCard* y *lockBoard*.
- Método *checkForMatch*: comprueba si las cartas volteadas son iguales. Si lo son, llama al método *disableCards* y si no lo son llama al método *unflipCards*.
    - **NOTA**: se puede usar un operador ternario de ECMAScript para simplificar la programación de este método
- Método *disableCards*: este método deshabilita las interacciones sobre las tarjetas de memoria que ya han sido emparejadas. Para ello modifica el valor del atributo *data-state* a *revealed* y después invoca al método *resetBoard*.

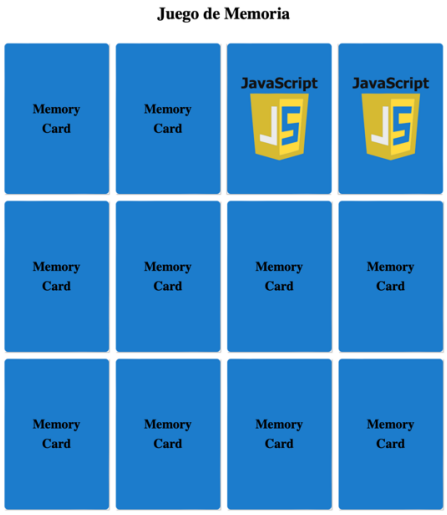
Para consultar las particularidades de cada una de las características de ECMAScript utilizadas en los métodos de utilidad de la clase Memoria se recomienda utilizar los siguientes enlaces:

- Método setTimeout
- Operador ternario

Tarea 7. Creación de los elementos del juego

Crea, dentro del body del documento memoria.html, una sección que contenga un encabezado con el texto "Juego de Memoria". En esta sección se mostrarán las tarjetas del juego.

El tablero del juego se debe ver como una sucesión de tarjetas dispuestas de tal manera que haya 4 elementos por fila y existan 3 filas de elementos, tal y como se observa en la siguiente imagen.



NOTA: De momento, las tarjetas de memoria del juego no se verán correctamente, el encabezado estará tapado por la imagen. Esto es debido a que será necesario añadirles transformaciones y transiciones a nivel de CSS para que la imagen esté oculta inicialmente y la tarjeta "se dé la vuelta" cuando el usuario la seleccione.

Guía para resolver la tarea 7

Cada tarjeta del juego estará representada por un nodo article dentro de esta sección.

Se debe crear un método llamado *createElements* que recorra el objeto JSON creado en la tarea 3 del ejercicio y cree, por cada elemento existente en el JSON, un nodo article en el documento html para representar cada tarjeta del juego de memoria.

El contenido de cada nodo article será:

- Un atributo "data-element" cuyo valor sea igual al valor de la variable element extraída del JSON.
- Un encabezado de orden 3 con el texto "Tarjeta de memoria" que se visualizará cuando la tarjeta esté bocabajo (situación inicial).
- Una imagen cuyo atributo *src* apunte a la imagen correspondiente al logo del elemento representado en la tarjeta y cuyo atributo *alt* sea igual al nombre del elemento. Esta

imagen solamente se visualizará cuando la tarjeta sea clicada por el usuario y se dé la vuelta.

Aplica FLEXBOX a la sección del documento memoria.html que contiene los elementos del juego. Recuerda que la disposición de los elementos debe ser en filas de 4 elementos. Debes hacer la adaptación correspondiente para que el primer elemento de dicha sección (el encabezado con el título del juego) ocupe toda la primera fila por sí mismo.

Referencia: https://www.w3.org/Style/CSS/current-work

Más información:

- https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

Tarea 8. Añadiendo las pulsaciones al juego

Para que las tarjetas de memoria respondan a la interacción del usuario es necesario asociarles el evento click a través de JavaScript.

Crea un método denominado *addEventListeners* que recorra todas las tarjetas creadas en la tarea anterior y que provoque una llamada al método *flipCard* de la clase Memoria cuando se lance dicho evento.

- **NOTA:** El método *flipCard* se debe invocar utilizando la característica bind de JavaScript de la siguiente manera: *this.flipCard.bind(card, this)*

Guía para resolver la tarea 8

Consultar el evento global onclick:

https://developer.mozilla.org/es/docs/Web/API/GlobalEventHandlers/onclick

Consultar el evento keydown, que se produce cuando se pulsa una tecla

https://developer.mozilla.org/es/docs/Web/API/Document/keydown_event

Para comprender el manejo del evento onclick puede consultarse el ejercicio:

http://di002.edv.uniovi.es/~cueva/JavaScript/13Botones.html

Consulta los siguientes enlaces sobre bind para comprender el uso del mismo:

http://di002.edv.uniovi.es/~cueva/JavaScript/94-EC6-Ejemplo-clase-Cronometro.html

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Function/bind

Tarea 9. Transformación y transición de los elementos del juego

Para que las tarjetas de memoria se vean correctamente en pantalla e interaccionen a las pulsaciones del usuario es necesario incluir fragmentos de código tanto a nivel de CSS como a nivel de JavaScript.

Modificaciones a nivel de CSS

Para que el logo de cada tarjeta de memoria esté oculto y no se vea cuando la tarjeta esté boca abajo es necesario rotar la imagen en el eje Y 180 grados, utilizando el método *rotate* de CSS dentro de la propiedad *transform*. Esto se completará además con las propiedades *backface-visibility* al valor *hidden* tanto en la imagen como en el encabezado de orden 3 existentes dentro de cada tarjeta de memoria *transform-style* al valor *preserve-3d* para que el efecto funcione correctamente.

Cuando el usuario pulse sobre una tarjeta y esta tenga que ser volteada, se utilizará el método rotate de CSS para voltear la tarjeta completa 180 grados en el eje Y, dentro de la propiedad *transform*. De esta forma pasará a verse la imagen con el logo y se ocultará el encabezado, dando la sensación de que la tarjeta se da la vuelta.

Para completar todo lo anterior es necesario declarar la propiedad *transition* de CSS en cada artículo al valor "*transform .5s*". Esto hace que cada vez que se ejecute una transformación sobre uno de los artículos esta se ejecute en un espacio de tiempo de medio segundo.

Modificaciones a nivel de JavaScript

Dentro de la clase memoria es necesario crear el método *flipCard* que se encargue de dar la vuelta a las tarjetas cuando estas sean pulsadas por el usuario. Este método recibe como parámetro una variable *game* que representa el juego.

En primer lugar se deben realizar tres comprobaciones:

1. Si la tarjeta pulsada por el usuario ya estaba revelada y formaba parte de una pareja ya descubierta (atributo *data-state* a *revealed*), el método retorna y no hace nada más.
2. Si la propiedad *lockBoard* del juego estaba al valor true, el método retorna y no hace nada más.
3. Si la tarjeta pulsada por el usuario coincide con la tarjeta pulsada anteriormente como primer elemento de la pareja actual (variable *firstCard* del juego), el método retorna y no hace nada más.

Si el método continúa su ejecución normal después de las tres comprobaciones anteriores porque ninguna de ellas se cumple, se deben realizar las siguientes acciones:

1. Modificar el atributo *data-state* de la tarjeta al valor *flip* para que la tarjeta se dé la vuelta y se vea la imagen.
2. Comprobar si el juego ya tenía una tarjeta volteada a través de la variable *flippedCard*
    a. En caso negativo, poner la variable *flippedCard* a verdadero y asignar a la variable *firstCard* el valor de la tarjeta actual (this).
    b. En caso afirmativo, asignar a la variable *secondCard* el valor de la tarjeta actual (this) e invocar al método *checkForMatch*.

Guía para resolver la tarea 9

Para la aplicación de las transformaciones y transiciones de CSS que permiten animar el volteo de las tarjetas del juego de memoria y hacer que la tarjeta se vea de forma diferente en función de si está boca arriba o boca abajo, consulta la información relativa al módulo [TRANSFORMS] a través de los siguientes enlaces:

- Referencia: https://www.w3.org/Style/CSS/current-work
- Ejemplo de tarjeta con volteo: https://jsfiddle.net/solisjaime/72ujf08a/10/

Tarea 10. Modificación del constructor de la clase Memoria

Una vez creados todos los componentes del juego, es necesario modificar el constructor de la clase Memoria para que el juego se inicialice cuando se construya la instancia de dicha clase y modificar el documento html para incluir la llamada que inicialice el juego.

Guía para resolver la tarea 10

Modifica el constructor de la clase Memoria añadiendo llamadas a los métodos *shuffleElements*, *createElements* y *addEventListeners* (en ese orden).

Posteriormente, añade una etiqueta script antes del cierre de la etiqueta html en el documento memoria.html y crea una instancia de la clase Memoria para que el juego se vea en pantalla y pueda ser utilizado.

Resultado del ejercicio 2

Una vez completado el ejercicio deberían haberse añadido los siguientes ficheros al proyecto del Escritorio Virtual:

- Fichero memoria.js en el directorio js del proyecto
- Fichero memoria.html en el directorio raíz del proyecto
- Fichero memoria.css en el directorio estilo del proyecto

Además, se debe haber modificado la sección de menú del fichero juegos.html para incluir el enlace de acceso al juego de memoria.

Ejercicio 3: Sudoku

En este ejercicio se utilizará JavaScript en combinación con HTML y CSS para realizar un Sudoku.

Sudoku es un juego matemático que se inventó a finales de la década de 1970 y que consiste en una cuadrícula de 81 celdas (9 x 9) en la que se introducen los números del 1 al 9 sin que puedan repetirse los mismos a nivel de fila, columna o cuadrícula de 3 x 3.

Utilizando Grid Layout se creará una estructura en celdas que represente el tablero del juego. En combinación con la estructura anterior se tratarán los eventos click y keydown que permitirán que el usuario seleccione una celda de dicha estructura y escriba un número en ella. Cada casilla del tablero podrá encontrarse en tres estados diferentes: *blocked* (contiene un número desde el inicio del sudoku), *clicked* (ha sido pulsada por el usuario) y *correct* (ha sido rellenada por el usuario).

Para llevar el control de las diferentes posiciones del tablero y los números que hay en ellas se utilizará un array de JavaScript de dos dimensiones.

Tarea 1. Creación del documento sudoku.html

Crea el fichero sudoku.html en el directorio raíz del proyecto de Escritorio Virtual.

Añade al menú creado en el ejercicio anterior dentro del fichero juegos.html un enlace que permita el acceso al fichero sudoku.html

Tarea 2. Creación de la clase Sudoku

Dentro de la carpeta js del directorio del proyecto Escritorio Virtual crea un nuevo fichero llamado sudoku.js. Añade una referencia a este nuevo fichero en el documento sudoku.html.

Dentro del fichero sudoku.js crea una clase Sudoku que tenga los siguientes atributos:

- Una cadena que represente el tablero del juego
- Un entero que represente el número de filas del sudoku (9),
- Un entero que represente el número de columnas del sudoku (9)
- Una variable que represente el tablero en forma de array; por el momento se puede dejar sin inicializar.

Añade a la clase Sudoku un método constructor que inicialice el tablero a un array bidimensional del tamaño que le corresponde.

Guía para resolver la tarea 2

Inicializa el array bidimensional a partir de las variables con el número de filas y el número de columnas del sudoku.

La cadena que representa el tablero del juego se puede inicializar, dentro o fuera del constructor, con uno de los siguientes valores (incluyendo las comillas):

- "3.4.69.5....27...49.2..4....2..85.198.9...2.551.39..6....8..5.32...46....4.75.9.6"
- "23.94.67.8..3259149..76.32.1.....7925.321.4864..68.5317..1....96598721433...9...7"

- "8.4.71.9.976.3....5.196....3.7495...692183...4.5726..92483591..169847...753612984"

Los valores numéricos de las cadenas anteriores representan celdas que ya se dan rellenas al comenzar el juego mientras que los valores "." representan celdas que estarán vacías y que el usuario deberá rellenar para completar satisfactoriamente el juego.

Tarea 3. Inicializando el array del Sudoku

Crea un método auxiliar de nombre *start* dentro de la clase Sudoku que ponga valores dentro de las celdas del array bidimensional que se ha inicializado dentro del constructor.

Guía para resolver la tarea 3

El valor de cada posición del array se debe tomar de la cadena que representa el tablero, que se debe ir recorriendo carácter a carácter:

- Si el valor en dicha cadena es numérico se vuelca el mismo número al array
- Si el valor en la cadena es un "." se debe introducir el valor 0 (cero) en dicha posición del array.

Tarea 4. Creando la estructura del sudoku

Crea dos métodos auxiliares de nombre *createStructure* y *paintSudoku* respectivamente dentro de la clase Sudoku.

El primero de los métodos será el encargado de crear en el documento HTML, a través de ECMAScript, los párrafos que representarán las celdas del sudoku.

El segundo método será el encargado de poner, dentro de cada párrafo, el valor que corresponda.

Guía para resolver la tarea 4

Los párrafos se deben crear dentro del elemento main del documento HTML.

Según el contenido del array que representa el estado del sudoku, los párrafos creados deberán tener el siguiente contenido:

- Si en el array hay un 0, el párrafo no tendrá contenido y se añadirá al mismo en manejador del evento click para que se pueda seleccionar con el ratón.
    - **NOTA:** Al hacer click en una celda se debe modificar el valor del atributo *data-state* al valor *clicked*.
- Si en el array hay un valor distinto de cero, se escribirá dicho valor en el párrafo y se establecerá el valor del atributo *data-state* a *blocked* para dicho párrafo para que no se pueda seleccionar a través del click de ratón.

Añade como primera línea del método *paintSudoku* la llamada al método *createStructure*.

En el documento sudoku.html añade una etiqueta script fuera del body y antes del cierre de la etiqueta html e incluye una llamada al método *paintSudoku*.

Tarea 5. Pintando la cuadrícula del sudoku

Crea una estructura de cuadrícula que permita representar las celdas del Sudoku. Al tratarse de una estructura de 9 filas y 9 columnas, esta deberá estar dividida en 9 columnas iguales y 9 filas iguales.

Para simplificar el proceso de asignación del tamaño de las celdas se puede establecer un tamaño fijo para el elemento main, de tal forma que las celdas se repartan el tamaño de forma igualitaria. Por ejemplo, se puede dar al main un tamaño de 500px de ancho y 500px de alto.

El resultado final debe ser similar al que se observa en la siguiente imagen.

3		4		6	9		5	
			2	7				4
9		2			4			
	2			8	5		1	9
8		9				2		5
5	1		3	9			6	
			8			5		3
2				4	6			
	4		7	5		9		6

Guía para resolver la tarea 5

Aplica GRID Layout para conseguir la disposición de las celdas del Sudoku que se observa en la imagen de ejemplo. Consulta las características del módulo [CSS-GRIDLAYOUT] a través de los siguientes enlaces:

- Referencia: https://www.w3.org/Style/CSS/current-work
- Más información: https://css-tricks.com/snippets/css/complete-guide-grid/ y https://jsfiddle.net/solisjaime/snwzmrpu/5/

Tarea 6. Añadiendo el evento de teclado

Dentro de la etiqueta script creada en el documento sudoku.html añade el código que permita el tratamiento del evento de teclado.

Guía para resolver la tarea 6

Consultar el evento keydown, que se produce cuando se pulsa una tecla

https://developer.mozilla.org/es/docs/Web/API/Document/keydown_event

Al tratarse de un sudoku, las únicas teclas que se deben permitir son las numéricas del 1 al 9 por lo que el resto de teclas que pulse el usuario se pueden ignorar.

Al detectarse una pulsación de teclado se deben realizar las siguientes comprobaciones:

1. Debe estar seleccionada, con anterioridad a la pulsación de teclado, una celda del Sudoku como receptora del número pulsado.
2. Si la tecla pulsada es un número y hay una celda seleccionada se debe llamar al método *introduceNumber* de la clase Sudoku, pasándole como parámetro a dicho método la tecla que se ha pulsado.
3. Si la tecla pulsada es un número pero no hay una celda del sudoku seleccionada, se debe informar al usuario que debe seleccionar una celda antes de pulsar un número.

Tarea 7. Introduciendo números y comprobando su validez

Crea el método *introduceNumber* en la clase Sudoku. Dicho método recibirá como parámetro el número que se ha pulsado en el teclado. Este método debe comprobar si el número pulsado es válido para la casilla que está seleccionada.

Guía para resolver la tarea 7

Se considera que un número es válido para una casilla si:

1. No existe un número igual en la misma fila de la casilla seleccionada
2. No existe un número igual en la misma columna de la casilla seleccionada
3. No existe un número igual en la sub-cuadrícula de 3 x 3 en la que se encuentra la celda seleccionada.

Si el número introducido pasa las tres comprobaciones anteriores se considera que es válido para la casilla seleccionada, momento en el que se debe:

- Deshabilitar la opción de hacer click en la casilla seleccionada quitando el manejador del evento
- Modificar el valor del atributo *data-state* a *correct* a la casilla seleccionada

Una vez comprobado que el número introducido es válido para la casilla seleccionada, se debe comprobar si ya están rellenas todas las cuadrículas del sudoku. En caso afirmativo, el sudoku se ha completado.

Para realizar la comprobación del número con respecto a la columna en la que se encuentra se recomienda consultar el método map de ECMAScript en el siguiente enlace:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

Resultado del ejercicio 3

Una vez completado el ejercicio deberían haberse añadido los siguientes ficheros al proyecto del Escritorio Virtual:

- Fichero sudoku.js en el directorio js del proyecto
- Fichero sudoku.html en el directorio raíz del proyecto
- Fichero sudoku.css en el directorio estilo del proyecto

Además, se debe haber modificado la sección de menú del fichero juegos.html para incluir el enlace de acceso al juego del Sudoku.

Recuerda

Se requiere el uso correcto de los elementos HTML5 establecidos en los ejercicios y se valorará positivamente el uso correcto y adecuado al contexto y funcionalidad, de elementos adicionales HTML5.

Se requiere el uso correcto de las propiedades de los módulos CSS establecidos en los ejercicios y se valorará positivamente el uso correcto y adecuado al contexto, de propiedades y módulos adicionales CSS.

Solamente se permite el paradigma de orientación de objetos y todo debe estar organizado con clases y objetos. Además, no se permite el uso de ningún tipo de bibliotecas externas y debe usarse ECMAScript puro o "vanilla".

Las siguientes condiciones son de obligado cumplimiento en todo el proyecto:

- **TODOS** los documentos HTML que componen el proyecto deben ser HTML5 válidos y sin advertencias utilizando el validador de lenguajes de marcado del W3C.
    - Recuerda que el contenido de los documentos HTML puede cambiar después de la ejecución del código ECMAScript. En ese caso, se debe comprobar la validez del código HTML en todos los diferentes estados por los que pase el documento.
- No está permitido el uso indiscriminado de bloques anónimos (**div**) en los documentos HTML5, se debe priorizar el uso de contenedores semánticos HTML5.
- Se deben utilizar selectores específicos, el uso de selectores id y class deberá estar justificado a través de comentarios en las hojas de estilo que expliquen la necesidad de su utilización y para que se utilizan.
- **TODAS** las reglas de todas las hojas de estilo deben estar precedidas por un comentario donde se indique la especificidad del (o los) selectores de la regla.
- **TODAS** las hojas de estilo que se utilizan en el sitio web deben ser validas utilizando el validador CSS del W3C
- **TODAS** las hojas de estilo deben tener 0 advertencias.
    - Recuerda seleccionar en "Más opciones" el informe de "Todas las advertencias" dentro de las opciones del validador CSS del W3C.
    - Excepcionalmente se permite las advertencias referidas a la verificación de los colores (color y background-color). **OBLIGATORIAMENTE** se debe indicar mediante un comentario en la regla de la hoja de estilo afectada la herencia de colores garantizando que la advertencia ha sido comprobada, verificada y garantizando que no provoca efectos laterales no deseados.
    - Excepcionalmente se permiten las advertencias referidas a la redefinición de propiedades derivadas del uso de @media-queries. **OBLIGATORIAMENTE** se debe indicar mediante un comentario en las reglas de la hoja de estilo afectada que propiedades se están redefiniendo.
- Se debe garantizar la adaptabilidad y realizar su verificación para todos los documentos que componen el proyecto.
    - Se deben utilizar medidas relativas en las hojas de estilo.
- Se debe garantizar la accesibilidad del proyecto mediante los test de las herramientas de accesibilidad para el nivel AAA de las WCAG 2.0 con 0 errores de modo automático en todos los documentos que lo componen.

El no cumplimiento de las características anteriores derivará en la invalidación del proyecto.

Anexo 1. Países del mundo

El listado contiene los 194 países del mundo

NÚMERO	CONTINENTE	PAÍS	CAPITAL
1	ÁFRICA	ANGOLA	LUANDA
2	ÁFRICA	ARGELIA	ARGEL
3	ÁFRICA	BENIN	PORTO-NOVO
4	ÁFRICA	BOTSUANA	GABERONES
5	ÁFRICA	BURKINA FASO	UAGADUGÚ
6	ÁFRICA	BURUNDI	BUYUMBURA
7	ÁFRICA	CABO VERDE	PRAIA
8	ÁFRICA	CAMERÚN	YAUNDÉ
9	ÁFRICA	CHAD	YAMENA
10	ÁFRICA	COMORAS	MORONI
11	ÁFRICA	COSTA DE MARFIL	YAMUSUKRO, ABIYÁN
12	ÁFRICA	EGIPTO	EL CAIRO
13	ÁFRICA	ERITREA	ASMARA
14	ÁFRICA	ETIOPÍA	ADÍS ABEBA
15	ÁFRICA	GABÓN	LIBREVILLE
16	ÁFRICA	GAMBIA	BANJUL
17	ÁFRICA	GHANA	ACCRA
18	ÁFRICA	GUINEA	CONAKRY
19	ÁFRICA	GUINEA ECUATORIAL	MALABO
20	ÁFRICA	GUINEA-BISSAU	BISSAU
21	ÁFRICA	KENIA	NAIROBI
22	ÁFRICA	LESOTO	MASERU
23	ÁFRICA	LIBERIA	MONROVIA
24	ÁFRICA	LIBIA	TRÍPOLI
25	ÁFRICA	MADAGASCAR	ANTANANARIVO
26	ÁFRICA	MALAUI	LILONGÜE
27	ÁFRICA	MALI	BAMAKO
28	ÁFRICA	MARRUECOS	RABAT
29	ÁFRICA	MAURICIO	PORT LOUIS
30	ÁFRICA	MAURITANIA	NUAKCHOT
31	ÁFRICA	MOZAMBIQUE	MAPUTO
32	ÁFRICA	NAMIBIA	WINDHOEK
33	ÁFRICA	NÍGER	NIAMEY
34	ÁFRICA	NIGERIA	ABUYA
35	ÁFRICA	REPÚBLICA CENTROAFRICANA	BANGUI
36	ÁFRICA	REPÚBLICA DEL CONGO	BRAZZAVILLE
37	ÁFRICA	REPÚBLICA DEMOCRÁTICA DEL CONGO	KINSHASA
38	ÁFRICA	RUANDA	KIGALI
39	ÁFRICA	SANTO TOMÉ Y PRÍNCIPE	SANTO TOMÉ
40	ÁFRICA	SENEGAL	DAKAR

41	ÁFRICA	SEYCHELLES	VICTORIA
42	ÁFRICA	SIERRA LEONA	FREETOWN
43	ÁFRICA	SOMALIA	MOGADISCIO
44	ÁFRICA	SUAZILANDIA	MBABANE
45	ÁFRICA	SUDÁFRICA	PRETORIA, CIUDAD DEL CABO, BLOEMFONTEIN
46	ÁFRICA	SUDÁN	JARTUM
47	ÁFRICA	SUDÁN DEL SUR	YUBA
48	ÁFRICA	TANZANIA	DODOMA
49	ÁFRICA	TOGO	LOMÉ
50	ÁFRICA	TÚNEZ	TÚNEZ
51	ÁFRICA	UGANDA	KAMPALA
52	ÁFRICA	YIBUTI	YIBUTI
53	ÁFRICA	ZAMBIA	LUSAKA
54	ÁFRICA	ZIMBABUE	HARARE
55	AMÉRICA	ANTIGUA Y BARBUDA	SAINT JOHN'S
56	AMÉRICA	ARGENTINA	BUENOS AIRES
57	AMÉRICA	BAHAMAS	NASSAU
58	AMÉRICA	BARBADOS	BRIDGETOWN
59	AMÉRICA	BELICE	BELMOPÁN
60	AMÉRICA	BOLIVIA	SUCRE, LA PAZ
61	AMÉRICA	BRASIL	BRASILIA
62	AMÉRICA	CANADÁ	OTTAWA
63	AMÉRICA	CHILE	SANTIAGO DE CHILE
64	AMÉRICA	COLOMBIA	BOGOTÁ
65	AMÉRICA	COSTA RICA	SAN JOSÉ
66	AMÉRICA	CUBA	LA HABANA
67	AMÉRICA	DOMINICA	ROSEAU
68	AMÉRICA	ECUADOR	QUITO
69	AMÉRICA	EL SALVADOR	SAN SALVADOR
70	AMÉRICA	ESTADOS UNIDOS	WASHINGTON D. C.
71	AMÉRICA	GRANADA	SAINT GEORGE'S
72	AMÉRICA	GUATEMALA	CIUDAD DE GUATEMALA
73	AMÉRICA	GUYANA	GEORGETOWN
74	AMÉRICA	HAITÍ	PUERTO PRÍNCIPE
75	AMÉRICA	HONDURAS	TEGUCIGALPA
76	AMÉRICA	JAMAICA	KINGSTON
77	AMÉRICA	MÉXICO	MÉXICO D. F.
78	AMÉRICA	NICARAGUA	MANAGUA
79	AMÉRICA	PANAMÁ	CIUDAD DE PANAMÁ
80	AMÉRICA	PARAGUAY	ASUNCIÓN
81	AMÉRICA	PERÚ	LIMA
82	AMÉRICA	PUERTO RICO	SAN JUAN
83	AMÉRICA	REPÚBLICA DOMINICANA	SANTO DOMINGO
84	AMÉRICA	SAN CRISTÓBAL Y NIEVES	BASSETERRE

85	AMÉRICA	SAN VICENTE Y LAS GRANADINAS	KINGSTOWN
86	AMÉRICA	SANTA LUCÍA	CASTRIES
87	AMÉRICA	SURINAM	PARAMARIBO
88	AMÉRICA	TRINIDAD Y TOBAGO	PUERTO ESPAÑA
89	AMÉRICA	URUGUAY	MONTEVIDEO
90	AMÉRICA	VENEZUELA	CARACAS
91	ASIA	AFGANISTÁN	KABUL
92	ASIA	ARABIA SAUDITA	RIAD
93	ASIA	BANGLADÉS	DACA
94	ASIA	BARÉIN	MANAMÁ
95	ASIA	BRUNEI	BANDAR SERI BEGAWAN
96	ASIA	BUTÁN	TIMBU
97	ASIA	CAMBOYA	PNON PEHN
98	ASIA	CATAR	DOHA
99	ASIA	CHINA	PEKÍN
100	ASIA	CHIPRE	NICOSIA
101	ASIA	COREA DEL NORTE	PYONGYANG
102	ASIA	COREA DEL SUR	SEÚL
103	ASIA	EMIRATOS ARABES UNIDOS	ABU DABI
104	ASIA	FILIPINAS	MANILA
105	ASIA	INDIA	NUEVA DELHI
106	ASIA	INDONESIA	YAKARTA
107	ASIA	IRÁN	TEHERÁN
108	ASIA	IRAQ	BAGDAD
109	ASIA	ISRAEL	JERUSALÉN
110	ASIA	JAPÓN	TOKIO
111	ASIA	JORDANIA	AMMÁN
112	ASIA	KAZAJISTÁN	ASTANÁ
113	ASIA	KIRGUISTÁN	BISKEK
114	ASIA	KUWAIT	CIUDAD DE KUWAIT
115	ASIA	LAOS	VIENTIÁN
116	ASIA	LÍBANO	BEIRUT
117	ASIA	MALASIA	KUALA LUMPUR
118	ASIA	MALDIVAS	MALÉ
119	ASIA	MONGOLIA	ULAN BATOR
120	ASIA	MYANMAR (BIRMANIA)	NAYPYIDAW
121	ASIA	NEPAL	KATMANDÚ
122	ASIA	OMÁN	MASCATE
123	ASIA	PAKISTÁN	ISLAMABAD
124	ASIA	PALESTINA	RAMALA
125	ASIA	SIRIA	DAMASCO
126	ASIA	SRI LANKA	COLOMBO
127	ASIA	TAILANDIA	BANGKOK
128	ASIA	TAYIKISTÁN	DUSAMBÉ
129	ASIA	TIMOR ORIENTAL	DILI

130	ASIA	TURKMENISTÁN	ASJABAD
131	ASIA	TURQUÍA	ANKARA
132	ASIA	UZBEKISTÁN	TASHKENT
133	ASIA	VIETNAM	HANOI
134	ASIA	YEMEN	SANÁ
135	EUROPA	ALBANIA	TIRANA
136	EUROPA	ALEMANIA	BERLÍN
137	EUROPA	ANDORRA	ANDORRA LA VIEJA
138	EUROPA	ARMENIA	EREVÁN
139	EUROPA	AUSTRIA	VIENA
140	EUROPA	AZERBAIYÁN	BAKÚ
141	EUROPA	BÉLGICA	BRUSELAS
142	EUROPA	BIELORRUSIA	MINSK
143	EUROPA	BOSNIA Y HERZEGOVINA	SARAJEVO
144	EUROPA	BULGARIA	SOFÍA
145	EUROPA	CROACIA	ZAGREB
146	EUROPA	DINAMARCA	COPENHAGUE
147	EUROPA	ESLOVAQUIA	BRATISLAVA
148	EUROPA	ESLOVENIA	LUBLIJANA
149	EUROPA	ESPAÑA	MADRID
150	EUROPA	ESTONIA	TALLÍN
151	EUROPA	FINLANDIA	HELSINKI
152	EUROPA	FRANCIA	PARÍS
153	EUROPA	GEORGIA	TIFLIS
154	EUROPA	GRECIA	ATENAS
155	EUROPA	HUNGRÍA	BUDAPEST
156	EUROPA	IRLANDA	DUBLÍN
157	EUROPA	ISLANDIA	REIKIAVIK
158	EUROPA	ITALIA	ROMA
159	EUROPA	LETONIA	RIGA
160	EUROPA	LIECHTENSTEIN	VADUZ
161	EUROPA	LITUANIA	VILNA
162	EUROPA	LUXEMBURGO	LUXEMBURGO
163	EUROPA	MALTA	LA VALETA
164	EUROPA	MOLDAVIA	CHISINAU
165	EUROPA	MÓNACO	MÓNACO
166	EUROPA	MONTENEGRO	PODGORICA
167	EUROPA	NORUEGA	OSLO
168	EUROPA	PAÍSES BAJOS	AMSTERDAM
169	EUROPA	POLONIA	VARSOVIA
170	EUROPA	PORTUGAL	LISBOA
171	EUROPA	REINO UNIDO	LONDRES
172	EUROPA	REPÚBLICA CHECA	PRAGA
173	EUROPA	REPÚBLICA DE MACEDONIA	SKOPJE
174	EUROPA	RUMANIA	BUCAREST
175	EUROPA	RUSIA	MOSCÚ

176	EUROPA	SAN MARINO	CIUDAD DE SAN MARINO
177	EUROPA	SERBIA	BELGRADO
178	EUROPA	SUECIA	ESTOCOLMO
179	EUROPA	SUIZA	BERNA
180	EUROPA	UCRANIA	KIEV
181	OCEANÍA	AUSTRALIA	CANBERRA
182	OCEANÍA	FIYI	SUVA
183	OCEANÍA	ISLAS MARSHALL	MAJURO
184	OCEANÍA	ISLAS SALOMÓN	HONIARA
185	OCEANÍA	KIRIBATI	TARAWA
186	OCEANÍA	MICRONESIA	PALIKIR
187	OCEANÍA	NAURU	YAREN
188	OCEANÍA	NUEVA ZELANDA	WELLINGTON
189	OCEANÍA	PALAOS	MELEKEOK
190	OCEANÍA	PAPÚA NUEVA GUINEA	PORT MORESBY
191	OCEANÍA	SAMOA	APIA
192	OCEANÍA	TONGA	NUKU'ALOFA
193	OCEANÍA	TUVALU	FUNAFUTI
194	OCEANÍA	VANUATU	PORT VILA