Práctica 1: Introducción a JavaScript

Guillermo Jiménez Díaz, Pedro A. González Calero

Entrega: 6 de marzo de 2016, 23.55h

Introducción

Esta primera práctica servirá como toma de contacto con el entorno de desarrollo y el lenguaje JavaScript. Para ello realizaremos un sencillo juego de memoria (en solitario). Puedes ver un vídeo de *gameplay* junto con el resto de material de la asignatura.

Entorno de desarrollo

Para desarrollar las prácticas solo es necesario:

- **Un editor de texto**. Aunque cualquier editor es válido se recomienda utilizar alguno que tenga resaltado de sintaxis (como Brackets, Atom, Eclipse...). En particular, en los laboratorios se ha instalado Sublime Text 2, un editor muy utilizado para el desarrollo web debido a los múltiples plugins que se pueden instalar con el fin de facilitar el desarrollo en HTML5 y JavaScript.
- **Un navegador web**. Se recomienda utilizar Google Chrome ya que es uno de los navegadores con mejor interfaz de desarrollo.
- **Un servidor web**. Veremos que en ocasiones nuestros juegos necesitarán cargar archivos (ficheros de datos, imágenes, sonidos, etc.) que el protocolo `file:` no permitirá realizar. Por este motivo es necesario instalar un servidor web en el que probar los juegos que estamos desarrollando. En los laboratorios está instalado XAMPP.

Desarrollo en Sublime Text 2

Aunque podemos utilizar Sublime Text 2 tal cual se descarga de la web es recomendable incluir algunos paquetes adicionales que ayudarán en el desarrollo en JavaScript:

- Package Control. Ayuda en la instalación de otros paquetes.
- Live Reload o Browser Refresh. Sirven para refrescar automáticamente el navegador a medida que guardamos el código fuente desde Sublime Text. En el caso de Live Reload, es necesario también instalar el plugin en Chrome
- JSHintGutter. Posibilita la ejecución de un proceso de *Linting*, que ayuda a detectar algunos posibles errores en el código en JavaScript. Este paquete es opcional.
- SublimeCodeIntel. Proporciona algunas acciones de autocompletado y de navegación por el código. Este paquete es opcional.

Si utilizamos XAMPP, aseguraos de que los archivos sobre los que trabajemos han de estar en la carpeta `htdocs` del servidor web y éste ha de estar arrancado para poder desarrollar la práctica.

Si tenemos instalado Python en nuestro equipo entonces podemos crear fácilmente un servidor web en cualquier directorio. Para ello solo tendremos que abrir un terminal de consola y, en el directorio deseado, ejecutar la siguiente instrucción:

```
# Servidor en http://localhost:8080
python -m SimpleHTTPServer 8080

# Si tenemos instalado python 3 entonces...
python2.7 -m SimpleHTTPServer 8080
```

Consola de desarrollo en Chrome

Google Chrome proporciona una completa consola de desarrollo en JavaScript. Esta consola se muestra en Windows en *Herramientas >> Consola de JavaScript* o con el atajo de teclado `Shift+Ctrl+J`. Esta consola muestra mensajes de error, la pila de llamadas, permite crear *watches* sobre variables, añadir puntos de ruptura, etc. Todo lo que estamos acostumbrados a utilizar en cualquier otro entorno de desarrollo.

Aunque durante la clase se darán más detalles es importante activar una opción de la consola para evitar comportamientos anómalos durante el desarrollo: con la consola abierta, pulsar en el botón de *Configuración* (con el icono de una rueda dentada que aparece a la derecha en la imagen) y, en el apartado General, activar la opción *Disable cache (while DevTools is open)*.

Desarrollo de la práctica

Para la realización de la práctica se proporciona un ZIP con los siguientes archivos y carpetas:

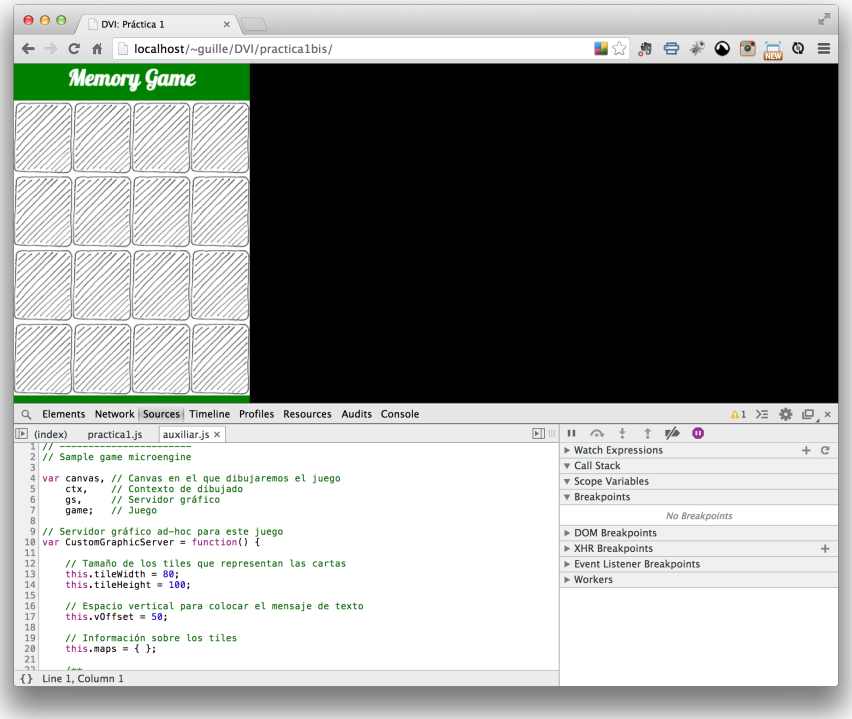

Figure 1: Consola de desarrollo de Google Chrome

- Carpeta `img`: Contiene los archivos de imágenes y de datos necesarios para el juego.
- Carpeta `css`: Contiene la CSS que se empleará en el juego.
- `index.html`: Documento html que contendrá el juego. **No se puede modificar** este archivo por ninguna circunstancia.
- `src/auxiliar.js`: Archivo JavaScript con funciones auxiliares del juego. Será responsable de configurar el servidor gráfico, el de *input* o entrada y de crear e inicializar el juego. **No se puede modificar** este archivo por ninguna circunstancia. Es recomendable leer detenidamente el código y los comentarios de esta clase antes de ponernos a desarrollar.
- `src/practica1.js`: El archivo en el que has de implementar tu juego. En particular, en este archivo crearemos las dos clases necesarias para nuestra práctica y que deberemos implementar: `MemoryGame` y `MemoryGame.Card`

`MemoryGame`

Esta clase guarda un array con las cartas y el estado en el que se encuentra el juego en cada momento, por lo que es la responsable de decidir cuándo ha terminado el juego. También guarda el mensaje que aparece en pantalla y que se irá cambiando a medida que interactuamos con el juego. También ha de tener una referencia al servidor gráfico para poder dibujar el estado del juego. Esta clase ha de implementar al menos los siguientes métodos:

- `MemoryGame(gs)`: La constructora recibe como parámetro el servidor gráfico, usado posteriormente para dibujar.
- `initGame()`: Inicializa el juego creando las cartas (recuerda que son 2 de cada tipo de carta), desordenándolas y comenzando el bucle de juego.
- `draw()`: Dibuja el juego, esto es: (1) escribe el mensaje con el estado actual del juego y (2) pide a cada una de las cartas del tablero que se dibujen.
- `loop()`: Es el *bucle del juego*. En este caso es muy sencillo: llamamos al método `draw` cada 16ms (equivalente a unos 60fps). Esto se realizará con la función `setInterval` de Javascript.
- `onClick(cardId)`: Este método se llama cada vez que el jugador pulsa sobre alguna de las cartas (identificada por el número que ocupan en el array de cartas del juego). Es el responsable de voltear la carta y, si hay dos volteadas, comprobar si son la misma (en cuyo caso las marcará como encontradas). En caso de no ser la misma las volverá a poner boca abajo[1].

[1] Para realizar la animación que aparece en el vídeo se puede utilizar la función `setTimeout`, haciendo que pasados unos cuantos milisegundos las cartas se pongan boca abajo. ¡Cuidado!: para evitar comportamientos extraños tendrás que ignorar los eventos de ratón mientras que la carta está siendo volteada.

`MemoryGame.Card`

Esta clase representa la cartas del juego. Una carta se identifica por el nombre del sprite que la dibuja[2] y puede estar en tres posibles estados: boca abajo, boca arriba o encontrada. Esta clase ha de implementar al menos los siguientes métodos:

- `MemoryGame.Card(sprite)`: Constructora que recibe el nombre del sprite que representa la carta. Las cartas han de crearse boca abajo.
- `flip()`: Da la vuelta a la carta, cambiando el estado de la misma.
- `found()`: Marca una carta como encontrada, cambiando el estado de la misma.
- `compareTo(otherCard)`: Compara dos cartas, devolviendo `true` si ambas representan la misma carta.
- `draw(gs, pos)`: Dibuja la carta de acuerdo al estado en el que se encuentra. Recibe como parámetros el servidor gráfico y la posición en la que se encuentra en el array de cartas del juego (necesario para dibujar una carta).

Entrega

La entrega consistirá en un archivo ZIP que contenga una carpeta con todos los archivos necesarios para ejecutar el juego. El nombre del archivo ha de ser: *Apellido1Apellido2Nombre.zip*

[2] Los nombres de los sprites que se proporcionan con el juego son: *8-ball, potato, dinosaur, kronos, rocket, unicorn, guy, zeppelin.* Hay un sprite adicional llamado *back* que se utiliza para mostrar las cartas que están boca abajo.