Prácticas de Laboratorio Tecnologías Audiovisuales en la Web

Grado Ing. Sistemas Audiovisuales y Multimedia
Universidad Rey Juan Carlos, Curso 2015-2016
versión 2.4

Índice

Introducción

Este documento presenta el enunciado de las prácticas de la asignatura Laboratorio de Tecnologías Audiovisuales en la Web. Las prácticas no se entregan, aunque es muy recomendable realizarlas todas. Son preparatorias para el *examen de prácticas*, que es el que os evaluamos. En él os pediremos que hagais alguna aplicación web, con alguna variante respecto a lo que habeis hecho en estas prácticas preparatorias.

Las prácticas cubren muchos aspectos del temario de teoría:

1. INTRODUCCIÓN
    a) Tecnologías web
    b) Repaso http
2. DATOS Y DOCUMENTOS
    a) Html5 (etiquetas)
    b) JSON
    c) XML
3. TECNOLOGÍAS CLIENTE
    a) JavaScript
    b) DOM y BOM
    c) Html5 (APIS)
4. TECNOLOGÍAS SERVIDOR
    a) Node.js
    b) Django
5. INTERACCIÓN CLIENTE SERVIDOR
    a) Formularios
    b) Cookies
    c) AJAX
    d) WebSockets
    e) Server Side Events
    f) Servicios web
6. WEBRTC

IMPORTANTE: las prácticas puedes prepararlas en tu ordenador, pero han de funcionar *obligatoriamente* en el laboratorio de la ETSIT.

Figura 1: Ejemplo de CV generado con Html5

1. Práctica Html5 básico

Crea tu Curriculum Vitae como una página web estática. Utiliza elementos típicos de html5: tablas, una imagen, etc.. Agrupa varios elementos que tienen sentido juntos en $< div >$ y asóciales un estilo de visualización propio (clase). Maneja hojas de estilos de CSS3, definidos en la sección $< head >$ de la página para mejorar su apariencia y aplica algún efecto de CSS3.

1.1. Ingredientes obligatorios

- Etiquetas semánticas y de estructura de documento que están disponibles en html5 pero no lo estaban en html4.01.
- Estilos CSS3
- Efectos CSS3

- $< div >$ y $< span >$

2. Práctica JavaScript: Juego del Comecocos

El objetivo de esta prueba es que practiques con JavaScript, manejo de DOM y de BOM. Aunque los contenidos de la página web vienen desde un servidor web, típicamente aplicaciones como ésta hacen hincapié en el código que se ejecuta en el navegador, con muchas interacciones locales con el usuario humano, es decir, en el lado cliente de una aplicación web.

Programa el juego del **comecocos** como aplicación web. Se ofrecerá un laberinto con paredes (no es necesario que sean redondeadas las paredes). En el escenario habrá un comecocos que debe ser teleoperable por el humano con el teclado, que lo va guiando pulsando algunas teclas. No puede moverse en diagonal.

Tienes un tutorial interesante sobre cómo manejar elementos de html5 para videojuegos en `http://www.w3schools.com/games/default.asp`

Comecocos elegible

En una parte de la página, fuera del escenario habrá tres comecocos, de tres colores distintos y un recuadro. El usuario humano puede elegir con qué comecocos concreto jugar arrastrando alguno de esos tres comecocos posibles hasta el recuadro. Empieza por defecto con un comecocos amarillo. Cuando el usuario termina de seleccionar uno se restauran las tres opciones posibles. Utiliza el API `drag & drop` para materializarlo.

Fantasmas

Habrá dos **fantasmas**, que se moverán autónomamente de modo continuo en línea recta. Se moveran en los dos grandes pasillos del circuito cambiando hacia la dirección opuesta cuando choquen con una pared. Su comportamiento no varía cuando el comecocos come un punto gordo, siguen exactamente igual.

Frutas

Dentro del escenario habrá frutas que el comecocos puede comer. Saldrán como mucho cuatro, siempre en la misma posición del laberinto. Saldrán en un intervalo aleatorio entre (0 y 10 segundos) una vez que la anterior fruta ha sido comida.

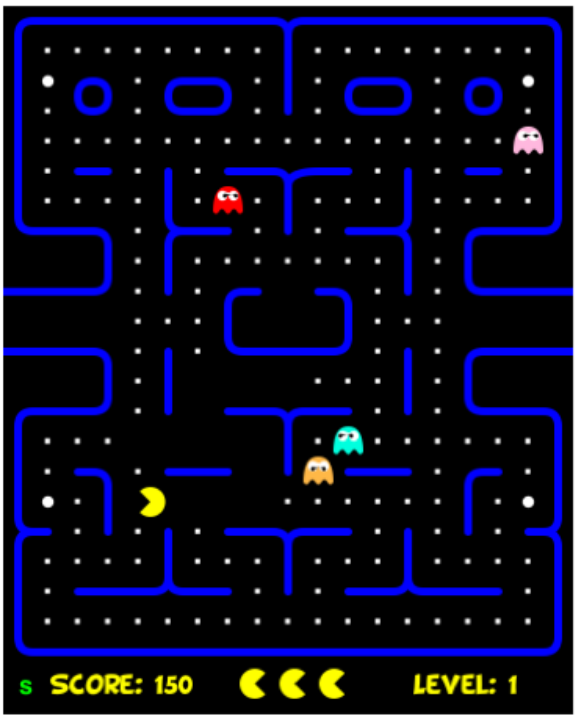

Figura 2: Aplicación web del comecocos

Tiempo

El juego tiene una duración máxima, un Tiempo Máximo Permitido. Habrá un botón de inicio y otro de pausa+continúa (es un único botón). Cuando se pulsa la pausa el juego y el tiempo para la puntuación se detienen. La aplicación medirá el tiempo que el comecocos tarda en completar el juego desde que el usuario pulsa el botón de comienzo. Utiliza para ello algún temporizador propio de JavaScript (`SetInterval`).

Puntuación y records

A medida que el comecocos come los puntos o frutas va acumulando puntos. Cada punto pequeño suma 1 a la puntuación, y cada punto gordo 3. Al final del juego el número de segundos restantes hasta el Tiempo Máximo permitido se suma a los puntos, y esa es la puntuación final.

La puntuación temporal se muestra en pantalla. La calcula un `WebWorker`, al que se le notifican sucesos (pacman se come un punto normal, se come un punto gordo, se come una fruta, ha pasado un segundo...) y entrega la puntuación al hilo principal en un mensaje.

La aplicación web guardará y mostrará en todo momento, fuera del escenario, dos records: el de la sesión y el absoluto de siempre. Utiliza las APIS de `SessionStorage` y de `LocalStorage` para materializar esto.

Sonido y video

Durante el juego habrá un sonido de continuo y cuando pasa encima de una fruta, se oirá brevemente un sonido especial. Utiliza el elemento $< audio >$ para ello.

Cuando se termine el juego, bien porque el comecocos se come todos los puntos, bien porque le atrapa un fantasma, o bien porque expira el tiempo máximo, se reproducirá un video en la página, fuera del escenario. En el primer caso un video con aplausos y en los otros dos, un video de desilusión.

2.1. Ingredientes obligatorios

- Manipulación del DOM
- Canvas 2D
- Motor temporal, `setInterval`
- Teleoperación desde teclas
- `WebWorkers`
- `LocalStorage` y `SessionStorage`
- API `Drag and Drop`
- Audio y video.

3. Práctica Servidor: Node.js web de presentación de una tienda

El objetivo de este ejercicio es que practiques con una tecnología del lado del servidor: Node.js. Este entorno emplea JavaScript para programar la lógica del servidor.

Programa la web estática de presentación de una tienda. Integra en ella textos, fotografías y videos estáticos. El sitio web de la tienda consistirá en varias páginas enlazadas, compartiendo hojas de estilo.

```
var http = require('http');

http.createServer(function(request, response) {
  var headers = request.headers;
  var method = request.method;
  var url = request.url;
  var body = [];
  request.on('error', function(err) {
    console.error(err);
  }).on('data', function(chunk) {
    body.push(chunk);
  }).on('end', function() {
    body = Buffer.concat(body).toString();
    // BEGINNING OF NEW STUFF

    response.on('error', function(err) {
      console.error(err);
    });

    response.statusCode = 200;
    response.setHeader('Content-Type', 'application/json');
    // Note: the 2 lines above could be replaced with this next one:
    // response.writeHead(200, {'Content-Type': 'application/json'})

    var responseBody = {
      headers: headers,
      method: method,
      url: url,
      body: body
    };

    response.write(JSON.stringify(responseBody));
    response.end();
    // Note: the 2 lines above could be replaced with this next one:
    // response.end(JSON.stringify(responseBody))

    // END OF NEW STUFF
  });
}).listen(8080);
```

Figura 3: Ejemplo de servidor Node.js

Este servidor será la base para muchas de las siguientes prácticas de interacción cliente-servidor: AJAX, Cookies, etc.

Opcionalmente puedes extender la práctica para que maneje una base de datos MySQL o SQLite y el servidor entregue respuestas dinámicas a consultas que le lleguen vía formulario de búsqueda.

3.1. Ingredientes obligatorios

- Usa módulo `http` de Node.js
- Sirva ficheros estáticos: html, hojas de estilo, fotografías, videos, audios e imágenes.

4. Práctica Servidor: Django web (dinámica) de una tienda

El objetivo de esta práctica es que practiques con una tecnología del lado del servidor: Django. Esta plataforma emplea Python para programar la lógica del servidor.

Programa la web de una tienda utilizando tecnología Django para el lado del servidor. Es una tienda de bicicletas, discos y libros. Asociado a cada bicicleta debe ofrecer un video descriptivo, a cada CD un audio y a cada libro una foto. El usuario podrá preguntar al servidor, vía formulario de búsqueda, por los productos con ciertas características. Esto se traducirá en una búsqueda en la base de datos de productos de la tienda y la entrega al navegador de una página con la respuesta concreta.

Las hojas de estilo, fotografías, videos, audios e imágenes serán exactamente los mismos que los de la práctica anterior.

4.1. Ingredientes obligatorios

- Caja de búsqueda para que el cliente humano pueda buscar en el catálogo (las bases de datos de productos) de la tienda. Esto desencadenará búsquedas en el lado del servidor, que consultará mediante filtros de Python la información pedida por el usuario.
- El sitio web debe ofrecer varias *vistas*: la general, sólo-bicis, sólo-libros, sólo-discos, los resultados de una búsqueda, etc..
- El servidor debe utilizar *plantillas* para generar contenidos dinámicos concretos.
- Incorporará diseño de URLs
- La tienda debe tener algún ingrediente para que el usuario humano provoque la escritura en una base de datos en el servidor. Lo típico es que el cliente rellene un formulario con un pedido concreto, ese pedido se reciba en servidor, que lo inserta en la base de datos respectiva, con un identificador único.

Diseña los modelos de datos necesarios y la aplicación de la tienda. Cuida el aspecto de la aplicación en el lado del cliente usando CSS y html5.

Si le quieres incluir mejoras opcionales, se valorarán positivamente.

5. Práctica Interacción Cliente-Servidor: Formularios en Node.js

Programa una aplicación web con un formulario utilizando tecnología de servidor Node.js. La aplicación:

1. envía una página que incluye un formulario vacío al cliente.

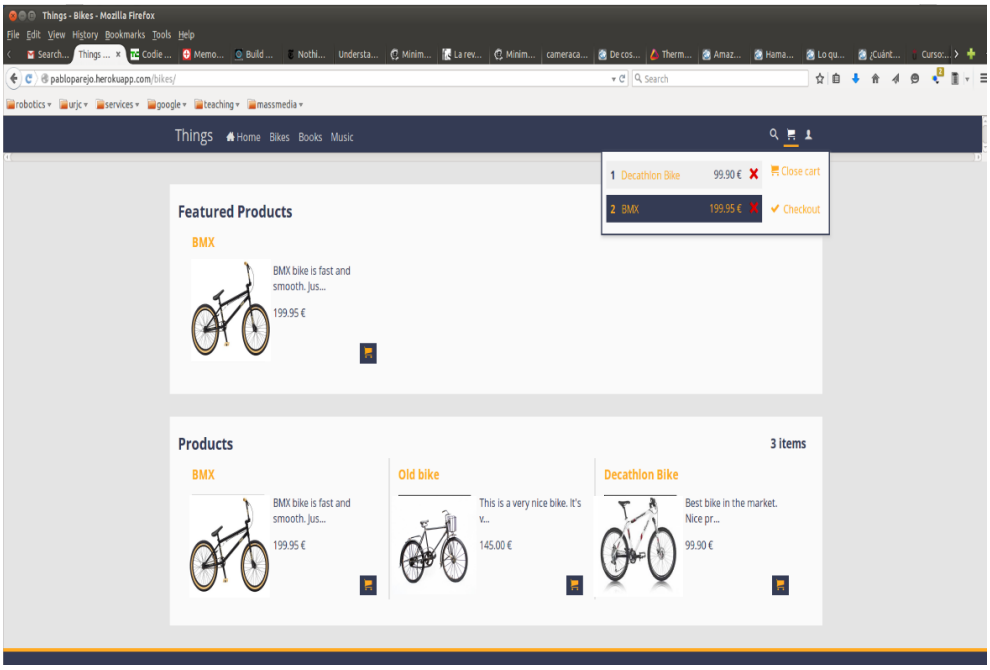

Figura 4: Ejemplo de vista de Django, con bicicletas y carrito de la compra

2. el usuario humano lo rellena en el navegador.
3. el navegador lo envía relleno de vuelta al servidor.
4. el servidor analiza el contenido del formulario relleno y responde al cliente con una página con texto donde explicita el contenido de cada uno de los campos del formulario y valor

Activa trazas en el servidor para que vayas viendo cada uno de los pasos del proceso.

5.1. Ingredientes obligatorios

- Envío vía POST. Opcionalmente se valorará que lo materialices también enviando el formulario relleno vía GET.
- Se repasarán las etiquetas relacionadas con formularios, especialmente las nuevas entradas que permite Html5.Se deben elegir en el formulario al menos: un color, una fechas, dos diales, un campo numérico y dos campos de texto.

6. Práctica Interacción Cliente-Servidor: Cookies en Node.js

Incorpora el Carrito de la compra a la *práctica del servidor web de la tienda con Node.js*. Retoca la aplicación web para que incorpore cookies. La aplicación:

1. envía al cliente la página inicial adosando una cookie con un identificador de usuario generado al azar

2. por cada elemento mostrado (bicicleta, libro, CD) ofrecerá un botón para añadir al carrito. Esa adición incorporará en las cookies, en cliente, un nuevo producto.

3. apaga el navegador y vuelvelo a arrancar para continuar con la compra, añadiendo elementos al carro.

4. habrá un botón de "compra" que cierra el pedido, envía un formulario vacío al servidor pero que lleva adjunta la cookie con todo el carrito en el estado actual.

5. el servidor analizará la cookie y responde al cliente con una página con texto donde explicita el contenido de cada uno de los productos del carro de la compra

Activa trazas en el cliente y el servidor para que vayas viendo cada uno de los pasos del proceso.

7. Práctica Interacción Cliente-Servidor: AJAX

Programa una aplicación web con autocompletado empleando para ello peticiones ligeras AJAX. La aplicación debe ofrecer una caja de búsqueda, y cuando el usuario humano teclea 3 o más caracteres, aunque no haya terminado de escribir la palabra entera, el cliente enviará una petición AJAX al servidor que contenga justo ese prefijo (por ejemplo 'man'). Esa petición será recibida por el servidor y atendida. En su respuesta el servidor incluirá al menos 4 palabras que empiezan por el prefijo, que son las que sugiere al cliente para terminar la frase (por ejemplo 'manzana', 'manto', 'manicura', 'manso'...). El cliente digerirá la respuesta AJAX y modificará el DOM de la página que se está visualizando para que aparezca un menú desplegable (elemento de HTMl5) bajo la caja de búsqueda, tal y como se muestra en la figura 5. El usuario humano podrá elegir la opción de autocompletado que prefiera, que quedará como contenido de ese campo del formulario. El formulario, de un solo campo, se enviará al servidor relleno (como en una práctica anterior).

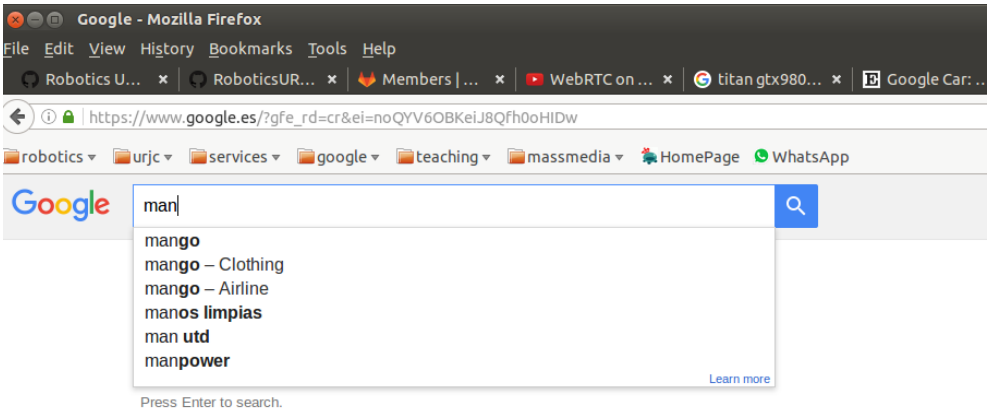

Figura 5: Caja de búsqueda con sugerencias

7.1. Ingredientes obligatorios

El servidor en node.js y enviando las respuestas AJAX con JSON, que serán procesadas en cliente.

El servidor puede buscar las sugerencias en su base de datos (MySQL o MongoDB por ejemplo), que es lo habitual en aplicaciones comerciales. Si preferís, podeis meter en el código fuente del servidor unas cuantas sugerencias para varios prefijos de antemano y que vuestra práctica sólo funcione para ellos. No es lo más elegante, ni escala, pero se acepta en esta práctica porque el énfasis lo ponemos en la interacción AJAX, no tanto en los datos que acarrea.

Opcionalmente se recomienda también que las programes en tu servidor Django. Se sugiere también que practiques a enviar las respuestas en XML, tanto para el servidor en NodeJs como el servidor en Django.

8. Práctica Interacción Cliente-Servidor: chat con websockets

El objetivo de esta práctica es que juegues con una tecnología dentro de html5 que permite iniciativa desde el lado del servidor (PUSH).

Programa un aplicación de chat entre múltiples usuarios usando Websockets. Como servidor puedes utilizar un servidor ws en Node.Js. Se sugiere el uso de la biblioteca socket.io. En particular se recomienda seguir el tutorial en Socketio [1]

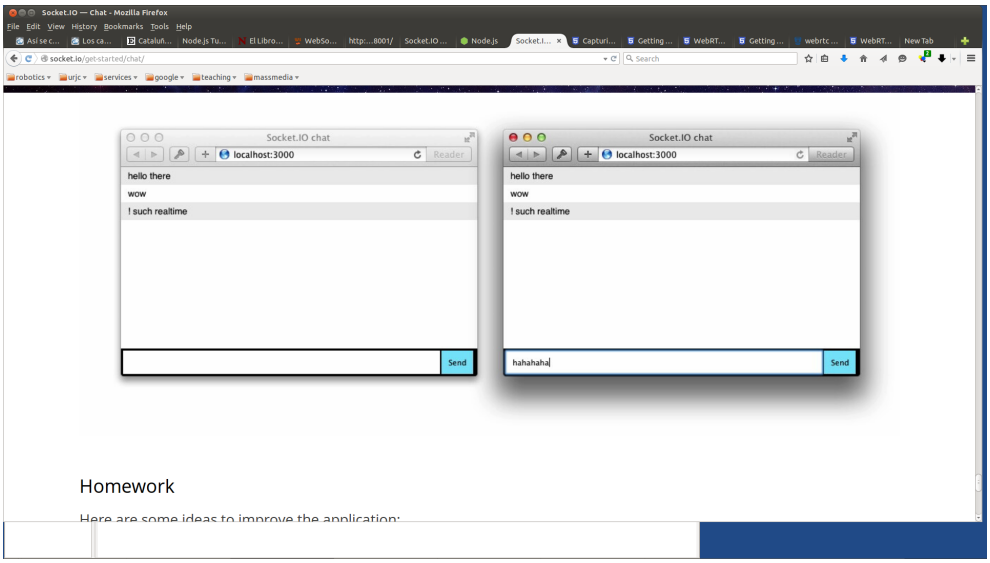

Figura 6: Dos usuarios chateando con esta aplicación

9. Práctica WebRTC: videoconferencia de navegador a navegador

El objetivo de esta práctica es que juegues con las tres interfaces de WebRTC: GetUserMedia, RTCPeerConnection y RTC-DataChannel.

Programa una aplicación web que ofrezca una videoconferencia entre dos navegadores. Se utilizará un mecanismo de sincronización a elegir por el alumno. En particular se recomienda seguir el tutorial

[1] `http://socket.io/get-started/chat/`

en WebRTC-CodeLab [2]. En él se emplea un servidor en Node.js y con socket.io como intermediario de señalización.

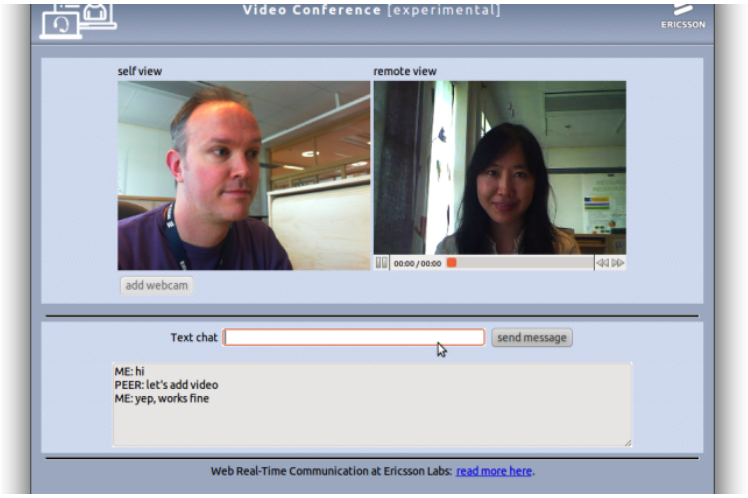

Figura 7: Aplicación de videoconferencia con chat

Además, se pide reimplementar una aplicación de chat dentro de la misma página de videoconferencia donde se ven los dos flujos multimedia. A diferencia de la práctica 8 aquí se consigue esa funcionalidad utilizando otra tecnología distinta: RTC-DataChannel.

[2] `https://bitbucket.org/webrtc/codelab/`