Aplicaciones Telemáticas
Grado en Ingeniería Telemática
Programa del curso 2015/2016

Jesús M. González Barahona, Gregorio Robles Martínez
GSyC, Universidad Rey Juan Carlos

19 de enero de 2016

Índice general

Capítulo 1

Características de la asignatura

1.1. Datos generales

Título:	Aplicaciones Telemáticas
Titulación:	Grado en Ingeniería Telemática
Cuatrimestre:	Cuarto curso, segundo cuatrimestre
Créditos:	6 (3 teóricos, 3 prácticos)
Horas lectivas:	4 horas semanales
Horario:	miércoles, 13:00–15:00 jueves, 13:00–15:00
Profesores:	Jesús M. González Barahona jgb @ gsyc.es Despacho 110, Departamental III, Campus de Fuenlabrada Gregorio Robles Martínez grex @ gsyc.es Despacho 110, Departamental III, Campus de Fuenlabrada
Sede telemática:	`http://campusvirtual.urjc.es/`
Aulas:	Laboratorio 209, Edif. Laboratorios III

1.2. Objetivos

En esta asignatura se pretende que el alumno obtenga conocimientos detallados sobre los servicios y aplicaciones comunes en las redes de ordenadores, y en particular en Internet. Se pretende especialmente que conozcan las tecnologías básicas que los hacen posibles.

1.3. Metodología

La asignatura tiene un enfoque eminentemente práctico. Por ello se realizará en la medido de lo posible en el laboratorio, y las prácticas realizadas (incluyendo especialmente la práctica final) tendrán gran importancia en la evaluación de la asignatura. Los conocimientos teóricos necesarios se intercalarán con los prácticos, en gran media mediante metologías apoyadas en la resulución de problemas. En las clases teóricas se utilizan, en algunos casos, transparencias que sirven de guión. En todos los casos se recomendarán referencias (usualmente documentos disponibles en Internet) para profundizar conocimientos, y complementarias de los detalles necesarios para la resolución de los problemas prácticos. En el desarrollo diario, las sesiones docentes incluirán habitualmente tanto aspectos teóricos como prácticos.

Se usa un sistema de apoyo telemático a la docencia (Moodle) para realizar actividades complementarias a las presenciales, y para organizar la documentación ofrecida a los alumnos.

1.4. Evaluación

Parámetros generales:

- Teoría (obligatorio): 0 a 5.
- Práctica final (obligatorio): 0 a 2.
- Opciones y mejoras de la práctica final: 0 a 3
- Prácticas incrementales: 0 a 1
- Ejercicios en foro: 0 a 1
- Nota final: Suma de notas, moderada por la interpretación del profesor
- Mínimo para aprobar:
  - aprobado en teoría (2.5) y práctica final (1)
  - 5 puntos de nota final

Evaluación teoría: prueba escrita
Evaluación prácticas incrementales (evaluación continua):

- entre 0 y 1 (sobre todo las extensiones)
- es muy recomendable hacerlas

Evaluación práctica final:

- posibilidad de examen presencial para práctica final
- tiene que funcionar en el laboratorio
- enunciado mínimo obligatorio supone 1, se llega a 2 sólo con calidad y cuidado en los detalles
- realización individual de la práctica

Opciones y mejoras práctica final:

- permiten subir la nota mucho

Evaluación ejercicios (evaluación continua):

- preguntas y ejercicios en foro

Evaluación extraordinaria:

- prueba escrita (si no se aprobó la ordinaria)
- nueva práctica final (si no se aprobó la ordinaria)

Capítulo 2

Programa

Programa de las prácticas de la asignatura (tentativo, irá evolucionando según avanza la asignatura).

2.1. Presentación

2.1.1. Sesión del 20 de enero (1 hora)

- **Presentación:** Presentación de la asignatura. Breve introducción y motivación de las aplicaciones web.
- **Material:** Transparencias, tema "Presentación".
- **Ejercicio (discusión en clase):** "Espía a tu navegador" (ejercicio 5.2.1)

2.1.2. Sesión del 27 de enero (0.5 horas)

- **Presentación:** Introducción a la entrega de prácticas en GitHub (seccion 5.1).

2.2. Introducción a HTML

Introducción a algunos conceptos de HTML y tecnologías relacionadas.

2.2.1. Sesión del 20 de enero (1 hora)

- **Ejercicio (discusión en clase):** "Página HTML simple" (ejercicio 5.2.2).
- **Ejercicio (discusión en clase):** "Página HTML con JavaScript" (ejercicio 5.2.4).

- **Presentación:** Introducción a HTML
- **Material:** Transparencias, tema "HTML".
- **Ejercicio libre:** Practica con HTML. Elige una página web y modifícala (trata de hacerlo para todos los elementos que puedas entre los vistos en la presentación).
- **Ejercicio (discusión en clase):** "Manipulación de HTML desde Firebug" (ejercicio 5.2.5)

2.3. Introducción a CSS

Introducción a algunos conceptos de CSS.

2.3.1. Sesión del 27 de enero (1.5 horas)

- **Presentación:** Introducción a CSS
- **Material:** Transparencias, tema "CSS".
- **Ejercicio (discusión en clase):** "Añadir selectores" (ejercicio 5.3.1)
- **Ejercicio (en clase):** "Tipografía" (ejercicio 5.3.6)
- **Ejercicio (entrega en el foro):** "Una caja CSS2" (ejercicio 5.3.11)

2.3.2. Sesión del 3 de febrero (2 horas)

- **Presentación:** CSS3
- **Material:** Transparencias, tema "CSS3".
- **Ejercicio (discusión en clase):** "Una caja CSS2" (ejercicio 5.3.11)
- **Ejercicio (discusión en clase):** "Bordes redondeados" (ejercicio 5.3.12)
- **Ejercicio (discusión en clase):** "Sombra de texto" (ejercicio 5.3.13)
- **Ejercicio (discusión en clase):** "Sombra de borde" (ejercicio 5.3.14)
- **Ejercicio (discusión en clase):** "Fondo semitransparente" (ejercicio 5.3.15)
- **Ejercicio (discusión en clase):** "Fondo en gradiente" (ejercicio 5.3.16)

- **Ejercicio (entrega en el foro):** "Alpha en los bordes" (ejercicio 5.3.17)
- **Ejercicio (discusión en clase):** "Alpha en los bordes" (ejercicio 5.3.17)
- **Ejercicio (discusión en clase):** "Rotación" (ejercicio 5.3.18)
- **Ejercicio (discusión en clase):** "Escalado" (ejercicio 5.3.19)
- **Ejercicio (discusión en clase):** "Rotación en el eje Y" (ejercicio 5.3.20)
- **Ejercicio (discusión en clase):** "Animación" (ejercicio 5.3.21)
- **Ejercicio (entrega en el foro):** "Transiciones" (ejercicio 5.3.22)

2.4. Introducción a Bootstrap

2.4.1. Sesión del 10 de febrero (2 horas)

- **Presentación:** Bootstrap
- **Material:** Transparencias, tema "Bootstrap".
- **Ejercicio (discusión en clase):** "Inspeccionando Bootstrap" (ejercicio 5.4.1)
- **Ejercicio (discusión en clase):** "Una sencilla página con Bootstrap" (ejercicio 5.4.2)
- **Ejercicio (entrega en GitHub):** "Utilizando el Carousel de Bootstrap" (ejercicio 5.4.4)

2.4.2. Sesión del 17 de febrero (2 horas)

- **Presentación:** Bootstrap
- **Material:** Transparencias, tema "Bootstrap".
- **Ejercicio:** "El *grid* de Bootstrap" (ejercicio 5.4.5)
- **Ejercicio:** "Bootstrap responsivo" (ejercicio 5.4.6)
- **Ejercicio (entrega en GitHub):** Concurso: "Tu diseño Bootstrap" (ejercicio 5.4.7)

2.5. Introducción a JavaScript

2.5.1. Sesión del 21 de enero (2 horas)

- **Presentación:** JavaScript: objetos
- **Material:** Transparencias, tema "JavaScript"
- **Ejercicio (discusión en clase):** "Página HTML con JavaScript" (ejercicio 5.2.4).
- **Ejercicios:** Ejercicios varios ejecutados en la consola de Firebug. Exploración de las opciones de depuración de Firebug para JavaScript.

2.5.2. Sesión del 28 de enero (2 horas)

- **Presentación:** JavaScript: tipos, funciones, strings, expresiones regulares.
- **Material:** Transparencias, tema "JavaScript"
- **Ejercicio (discusión en clase):** "Iteración sobre un objeto" (ejercicio 5.5.1)
- **Ejercicio (discusión en clase):** "Función que cambia un elemento HTML" (ejercicio 5.5.4)
- **Ejercicio (discusión en clase):** "Vacía página" (ejercicio 5.5.2)

2.5.3. Sesión del 4 de febrero (2 horas)

- **Presentación:** JavaScript: números, booleanos, vectores (arrays), variables, sentencias de control, prototipos y herencia, constructores, riesgos a evitar.
- **Material:** Transparencias, tema "JavaScript"
- **Ejercicio (discusión en clase):** "De lista a lista ordenada" (ejercicio 5.5.3)
- **Ejercicio (discusión en clase):** "Sumador JavaScript muy simple" (ejercicio 5.5.5)
- **Ejercicio (discusión en clase, entrega en GitHub):** "Sumador JavaScript muy simple con sumas aleatorias" (ejercicio 5.5.6)
  Entrega recomendada: antes del 5 de febrero.
  Repo GitHub: `https://github.com/CursosWeb/X-Nav-JS-Sumador`
- **Ejercicio (discusión en clase):** "JSFIDDLE" (ejercicio 5.5.8)

2.5.4. Sesión del 11 de febrero (2 horas)

- **Ejercicio (discusión de la solucion):** "Sumador JavaScript muy simple con sumas aleatorias" (ejercicio 5.5.6)
- **Ejercicio (discusión en clase, entrega en GitHub):** "Mostrador aleatorio de imágenes" (ejercicio 5.5.7)
  Entrega recomendada: antes del 12 de febrero
  Repo GitHub: `https://github.com/CursosWeb/X-Nav-JS-Fotos`
- **Ejercicio (discusión en clase, entrega en GitHub):** "Calculadora binaria simple" (ejercicio 5.5.10)
  Entrega recomendada: antes del 12 de febrero
  Repo GitHub: `https://github.com/CursosWeb/X-Nav-JS-Calculadora`
- **Ejercicio (discusión en clase):** "Greasemonkey" (ejercicio 5.5.9)

2.5.5. Sesión del 18 de febrero (2 horas)

- **Ejercicio (discusión en clase):** "Prueba de addEventListener para leer contenidos de formularios" (ejercicio 5.5.11)
- **Ejercicio (discusión en clase, entrega en GitHub):** "Colores con addEventListener" (ejercicio 5.5.12)
  Entrega recomendada: antes del 19 de febrero
  Repo GitHub: `https://github.com/CursosWeb/X-Nav-JS-Event`

2.6. Introducción a jQuery

2.6.1. Sesión del 25 de febrero (2 horas)

- **Presentación:** JQuery: introducción
- **Material:** Transparencias, tema "jQuery"
- **Ejercicio (discusión en clase):** "Uso de jQuery" (ejercicio 5.6.1)
- **Ejercicio (discusión en clase, entrega en Git Hub):** "Cambio de colores con jQuery" (ejercicio 5.6.2)
  Entrega recomendada: antes del 26 de febrero.

Repo GitHub: `https://github.com/CursosWeb/X-Nav-JQ-Colores`

2.6.2. Sesión del 3 de marzo (2 horas)

- **Presentación:** jQuery: continuamos
- **Material:** Transparencias, tema "jQuery"
- **Ejercicio (discusión en clase):** "Texto con jQuery" (ejercicio 5.6.3).
- **Ejercicio (discusión en clase):** "Difuminado (fading) con jQuery" (ejercicio 5.6.4).
- **Presentación de práctica de entrega voluntaria:** "Calculadora SPA" (3.1)
  Entrega recomendada: antes del 5 de marzo.
  Repo GitHub: `https://github.com/CursosWeb/X-Nav-Practica-Calculadora`

2.6.3. Sesión del 10 de marzo (2 horas)

- **Presentación:** jQuery: AJAX. Historia, motivación, el objecto XMLHTTPRequest. Uso de AJAX desde jQuery.
- **Material:** Transparencias, tema "jQuery"
- **Ejercicio (discusión en clase, entrega en el foro):** "Ejemplos simples con Ajax" (ejercicio 5.6.5)
  Entrega recomendada: antes del 11 de marzo.
  Repo GitHub: `https://github.com/CursosWeb/X-Nav-JQ-Ajax`

2.6.4. Sesión del 17 de marzo (2 horas)

- **Presentación:** JSON, AJAX con JSON y uso de AJAX con JSON desde jQuery.
- **Material:** Transparencias, tema "jQuery"
- **Ejercicio (discusión en clase):** "Ejemplos simples con Ajax y JSON" (ejercicio 5.6.6).

- **Ejercicio (entrega en el foro):** "Generador de frases aleatorias" (ejercicio 5.6.7)
  Entrega recomendada: antes del 19 de marzo.
- **Ejercicio (entrega en el foro):** "Utilización de JSONP" (ejercicio 5.6.8)
  Entrega recomendada: antes del 12 de marzo.
  Repo GitHub: `https://github.com/CursosWeb/X-Nav-JQ-Flickr`

2.7. Introducción a HTML5

Introducción a algunos conceptos de HTML5.

2.7.1. Sesión del 24 de febrero (2 horas)

- **Presentación:** HTML5: introducción
- **Material:** Transparencias, tema "HTML5"
- **Ejercicio (discusión en clase):** "Un sencillo Paint" (ejercicio 5.7.3)
- **Ejercicio (discusión en clase):** "Un sencillo Paint con brocha" (ejercicio 5.7.4)
- **Ejercicio (discusión en clase):** "Un sencillo juego con canvas" (ejercicio 5.7.5)
- **Ejercicio (entrega en el foro):** "Mejora el juego con canvas" (ejercicio 5.7.6)
  Entrega recomendada: antes del 11 de marzo.

2.7.2. Sesión del 2 de marzo (2 horas)

- **Presentación:** HTML5: Guardar en local; aplicaciones sin conexión
- **Material:** Transparencias, tema "HTML5"
- **Ejercicio (discusión en clase):** "Juego con estado" (ejercicio 5.7.7)
- **Ejercicio (entrega en GitHub):** "Juego sin conexión" (ejercicio 5.7.8)
  Entrega recomendada: antes del 18 de marzo

2.7.3. Sesión del 9 de marzo (2 horas)

- **Presentación:** HTML5: Otras cuestiones HTML5
- **Material:** Transparencias, tema "HTML5"
- **Ejercicio:** "Modernizr: Comprobación de funcionalidad HTML5" (ejercicio 5.7.9)
- **Ejercicio:** "Audio y vídeo con HTML5" (ejercicio 5.7.10)
- **Ejercicio:** "Geolocalización con HTML5" (ejercicio 5.7.11)
  Entrega recomendada: antes del 25 de marzo.

2.7.4. Sesión del 16 de marzo (2 horas)

- **Presentación:** HTML5: Web Workers
- **Material:** Transparencias, tema "HTML5"
- **Ejercicio (entrega en GitHub):** "Cálculo de números primos con Web Workers" (ejercicio 5.7.13)
  Entrega recomendada: antes del 8 de abril.
- **Presentación:** HTML5: WebSocket
- **Material:** Transparencias, tema "HTML5"
- **Ejercicio:** "Cliente de eco con WebSocket" (ejercicio 5.7.14)
- **Ejercicio:** "Cliente y servidor de eco con WebSocket" (ejercicio 5.7.15)
- **Ejercicio:** "Cliente y servidor de chat con WebSocket" (ejercicio 5.7.16)

2.7.5. Sesión del 30 de marzo (2 horas)

- **Presentación:** HTML5: History API
- **Ejercicio (entrega en GitHub):** "History API - Cambiando la historia con HTML5" (ejercicio 5.7.18)
  Entrega recomendada: antes del 15 de abril.

2.8. Otras bibliotecas JavaScript

2.8.1. Sesión del 31 de marzo (2 horas)

- **Presentación:** JQueryUI: introducción
- **Ejercicio (discusión en clase):** "JQueryUI: Instalación y prueba" (ejercicio 5.8.1)
- **Ejercicio (discusión en clase):** "JQueryUI: Uso básico" (ejercicio 5.8.2)
- **Ejercicio (entrega en el foro):** "JQueryUI: Juega con JQueryUI" (ejercicio 5.8.3)
  Entrega recomendada: antes del 19 de marzo.
- **Ejercicio (optativo):** "JQueryUI: Clon de 2048" (ejercicio 5.8.4)
  Entrega recomendada: antes del 19 de marzo.
- **Presentación de práctica de entrega voluntaria:** "Socios" (3.2)
  Entrega recomendada: antes del 26 de marzo.
  Repo GitHub: `https://github.com/CursosWeb/X-Nav-Practica-Socios`

2.9. APIs JavaScript

2.9.1. Sesión del 7 de abril (2 horas)

- **Presentación:** Leaflet: introducción
- **Ejercicio (discusión en clase):** "Leaflet: Instalación y prueba" (ejercicio 5.9.1)
- **Ejercicio (discusión en clase):** "Leaflet: Coordenadas" (ejercicio 5.9.2)
- **Ejercicio (entrega en el foro):** "Leaflet: Aplicación móvil" (ejercicio 5.9.3)
  Entrega recomendada: antes del 23 de marzo. Repo GitHub: `https://github.com/CursosWeb/X-Nav-APIs-Leaflet`

2.9.2. Sesión del 14 de abril (2 horas)

- **Ejercicio (discusión en clase):** "Leaflet: Coordenadas y búsqueda de direcciones" (ejercicio 5.9.5).

- **Ejercicio (entrega en el foro):** "Leaflet: Fotos de Flickr" (ejercicio 5.9.6)
  Entrega recomendada: antes del 16 de abril.
  . Repo GitHub: `https://github.com/CursosWeb/X-Nav-APIs-Map-Flickr`

2.9.3. Sesión del 21 de abril (2 horas)

- **Ejercicio (discusión en clase):** "GitHub.js: Datos de un repositorio" (ejercicio 5.9.7).
- **Ejercicio (entrega en GitHub):** "GitHub.js: Crea un fichero" (ejercicio 5.9.8)
  Entrega recomendada: antes del 23 de abril.
  . Repo GitHub: `https://github.com/CursosWeb/X-Nav-APIs-GitHub-Fichero`

2.10. OAuth

2.10.1. Sesión del 28 de abril (2 horas)

- **Ejercicio (discusión en clase):** "OAuth con GitHub" (ejercicio 5.12.1).
- **Ejercicio (entrega en GitHub):** "Listado de ficheros en GitHub" (ejercicio 5.12.2)
  Entrega recomendada: antes del 30 de abril.
  Repo GitHub: `https://github.com/CursosWeb/X-Nav-OAuth-GitHub-Fichero`

2.11. APIs de Google

2.11.1. Sesión del 6 de abril (2 horas)

- **Presentación:** La API de los servicios de Google
- **Material:** Transparencias, tema "APIs de Google"
- **Ejercicio:** "Conociendo la Google API Console" (ejercicio 5.10.1)
- **Ejercicio:** "Tu Perfil vía la API de Google+" (ejercicio 5.10.2)
- **Ejercicio (entrega en GitHub):** "Tomando datos de la API de Google+" (ejercicio 5.10.3)
  Entrega recomendada: antes del 22 de abril.

2.12. Firefox OS

2.12.1. Sesión del 13 de abril (2 horas)

- **Presentación:** Firefox OS
- **Material:** Transparencias, tema "Firefox OS"
- **Ejercicio:** "Primera aplicación con Firefox OS" (ejercicio 5.11.1)
- **Ejercicio (entrega en GitHub):** "Open Web Apps: Aplicación para Firefox OS" (ejercicio 5.11.2)
  Entrega recomendada: antes del 29 de abril.

Capítulo 3

Enunciados de prácticas: aplicaciones simples

3.1. Calculadora SPA

Enunciado:

Esta práctica consistirá en la creación de una calculadora que funcione como una SPA (single page application), compuesta por un documento HTML, una hoja de estilo CSS y un fichero JavaScript.

La calculadora deberá realizar al menos las cuatro operaciones aritméticas básicas. Tendrá teclas (o similar) para poder escribir al menos: números (del 0 al 9), la operación a realizar, "=" (para obtener el resultado), "C" (para borrar). Tendrá también una pantalla donde se mostrará lo que se va escribiendo con las teclas de la propia calculadora, pero que permitirá también usar el teclado del ordenador. En esta pantalla se verán también los resultados de las operaciones. Bastará con que la calculadora funcione con enteros positivos, aunque se valorará que lo haga con enteros de cualquier signo, y aún mejor, con números reales.

El documento HTML incluirá elementos con marcadores (y si es caso contenidos) para los distintos elementos de la calculadora. No incluirá ninguna referencia al estilo de los elementos, y tendrá una sola inclusión de la hoja de estilo CSS mencionada. Esta hoja de estilo tendrá toda la información de estilo necesaria, incluyendo, en su caso, la que pueda requerir el programa JavaScript. El documento HTML incluirá, en su cabecera, elementos con referencias a las bibliotecas JavaScript que se usen (en principio, jQuery) y al fichero JavaScript que se realice para implementar la funcionalidad de la calculadora. No habrá más código JavaScript en otros elementos del documento HTML. En caso de querer usar alguna otra biblioteca además de jQuery, consultar con los profesores.

La aplicación, como se ha dicho, tendrá que funcionar como una SPA. Esto

es, una vez descargados los tres elementos (documento HTML, hoja CSS y fichero JavaScript) no hará falta nada más para que funcione dentro de un navegador.

Se valorará que la aplicación tenga un aspecto de calculadora lo más logrado posible, y que la funcionalidad, cumpliendo este enunciado, sea también lo más lograda y completa posible.

Entrega:

La práctica se entregará según se describe en el apartado 5.1. El repositorio contendrá los tres ficheros mencionados (HTML, CSS, JavaScript) más cualquier biblioteca JavaScript que pueda hacer falta (normalmente, sólo jQuery) para que la aplicación funcione. El fichero HTML se llamará "calculadora.html", y el fichero que se entregue estará construido de tal forma que una vez se haya descomprimido, bastará con cargar este fichero HTML en el navegador para que la calculadora funcione.

3.2. Socios

Enunciado:

Vamos a construir parte del interfaz de usuario de la aplicación Socios, una nueva red social. En particular, vamos a representar en el navegador la información que nos va a llegar en varios documentos JSON. Para simular lo suficiente para poder realizar la interfaz de usuario, estos documentos JSON serán ficheros estáticos que se servirán al navegador con el resto de la aplicación, que estará compuesta por un fichero HTML, otro JavaScript y otro CSS.

Los documentos JSON mencionados son los siguientes:

- timeline.json: Mensajes de los socios del usuario, en modo resumen (ver detalle más abajo).
- update.json: Mensajes de los socios que aún no se ha mostrado en el timeline.
- myline.json: Mensajes del usuario, puestos en el pasado.

Para cada mensaje, los documentos JSON tendrán al menos la siguiente información:

- Autor: nombre del autor del mensaje.
- Avatar: url del avatar (imagen) del autor del mensaje.
- Título: título del mensaje.
- Contenido: contenido del mensaje.

- Fecha: fecha en que fue escrito el mensaje.

Opcionalmente, para cada mensaje se podrá ofrecer otra información adicional, como coordenadas de geolocalización, urls de anexos (attachements), etc.

Además de estos documentos JSON con la información de mensajes, se servirán vía HTTP las imágenes (avatares) que se citen en ellos, y los tres documentos básicos de la aplicación: uno HTML, otro CSS y otro JavaScript.

La aplicación mostrará en pestañas (tabs) diferentes la siguiente información:

- Timeline del usuario: mensajes de sus socios, según listado en timeline.json. Además, una vez mostrados estos mensajes, se buscará update.json. Si tiene alguna noticia, se mostrará una nota al principio del timeline indicando el número de mensajes pendientes. Cuando se pulse en esa nota, se desplegarán los mensajes pendientes que estaban en update.json.

- Mensajes enviados por el usuario, según listado en myline.json

En principio, de cada mensaje se mostrará sólo el nombre del autor, su avatar, y el título del mensaje. Se ofrecerá un botón para desplegar todo el mensaje: si se pulsa, se desplegará el resto de la información.

Se podrán realizar otras mejoras a este comportamiento básico.

Entrega:

La práctica se entregará según se describe en el apartado 5.1, utlizando la rama gh-pages para que sea visible directamente como sitio web (ver detalles en dicho apartado).

En ambas ramas (master y gh-pages) del repositorio de entrega habrá al menos:

- Un fichero README.md que resuma las mejoras, si las hay, y explique cualquier peculiaridad de la entrega.

- Los tres ficheros mencionados (HTML, CSS, JavaScript).

- Los ficheros JSON especificados.

- Los ficheros de avatar (imágenes) necesarios.

- Cualquier biblioteca JavaScript que pueda hacer falta (normalmente, sólo jQuery y jQueryUI) para que la aplicación funcione.

El fichero HTML se llamará "index.html", y todo el directorio (repositorio) estará construido de tal forma que bastará con servirlo mediante un servidor HTTP, y cargar en un navegador este fichero HTML, para que la vista de nuestros socios (y todo el interfaz de usuario) funcione. Igualmente, deberá funcionar si se carga el repositorio desde GitHub (que mostrará lo que haya en la rama gh-pages).

Capítulo 4

Enunciados de prácticas: proyectos finales

4.1. Adivina dónde está (junio 2015)

Nota: Enunciado en elaboración. Aún puede cambiar

La práctica consiste en la creación de una aplicación HTML5 que permita jugar a una variante del juego de las adivinanzas. Se tratará de mostrar al usuario pistas, en forma de fotografías, y que éste tenga que adivinar, marcándolo en un mapa, a qué parte del mundo se refieren.

4.1.1. Enunciado

Concretamente, la aplicación mostrará, al arrancar, un panel con:

- Una banda horizontal donde se irán mostrando las fotos una a continuación de otra (ver más abajo).
- Una zona (bajo la banda anterior) donde se mostrará un mapa, para que el jugador pueda marcar un punto, indicando que ese es el sitio a adivinar.
- Una zona con los controles del juego, la puntuación, etc. Los controles incluirán opciones al menos para iniciar un juego, para abortar un juego en curso, para "rebobinar" un juego (volver a mostrar sus fotos desde el principio), y para empezar uno nuevo (cada una, visible sólo en el momento adecuado). Además, habrá forma de elegir un juego entre los disponibles, para ver los juegos que se han jugado, el momento en que se jugaron y su puntuación, y a partir de esa lista volver a jugar cualquiera de ellos.

Cuando empiece el juego, el jugador verá una foto durante un cierto tiempo. Cuando termine ese tiempo, aparecerá a su derecha otra, y así sucesivamente. Cuando el espacio en la banda horizontal para fotos se termine, irá desapareciendo la foto de más a la izquierda para ir añadiendo una nueva foto por la derecha. Cuando el jugador crea que sabe a qué parte del mundo se refiere la foto, pulsará con el ratón sobre esa parte del mundo. En ese momento, la aplicación calculará la distancia entre el punto en que ha pulsado el jugador y el punto al que se refería la adivinanza, y mostrará la distancia entre ambos, y la puntuación obtenida. La puntuación obtenida se calculará multiplicando esta distancia por el tiempo transcurrido desde que comenzó el juego.

Cada juego mantendrá, como base para plantear las adivinanzas, un fichero JSON, compuesto una lista de objetos, cada uno con dos propiedades: "name" y "collection". "name" tendrá como valor el nombre de un juego, y "collection" tendra'como valor una "FeatureCollection" (según se especifica por GeoJSON). En cada "FeatureCollection", cada elemento de la colección corresponderá con un sitio a adivinar. El campo "coordinates" marcará la localización del punto a adivinar, y el campo "Name" de "properties", el nombre del sitio a adivinar (que sólo se mostrará al usuario al terminar cada juego). El nombre del sitio se utilizará como etiqueta (tag) en una búsqueda en Flickr, para obtener fotos relacionadas con ese nombre. Esas fotos son las que se mostrarán al jugador. Cada vez que se juege uno de estos juegos, la aplicación elegirá aleatoriamente uno de los puntos, que será el punto a adivinar.

El fichero JSON se mantendrá en un repositorio en GitHub (que será el de entrega de la práctica), con el nombre "games.json". Por ejemplo, este fichero podrá tener un listado con dos objetos, uno con la propiedad "name" igual a "Capitales" y otro con essa propiedad igual a "Islas". Cada uno tendría en su propiedad "collection" la colección de puntos de su juego ("Capitales" con capitales del mundo o "Islas" con islas).

La historia de juegos pasados se mantendrá en un objeto en almacenamiento persistente local. Este objeto se actulizará cada vez que se termine un juego, y esta información se reflejará en la parte correspondiente de la página mostrada por el navegador. Al ser un objeto en almacenamiento persistente, se conservará si la págian se regarga (y por lo tanto la aplicación, cuando se ejecute, tendrá que comenzar por cargar ese objeto si está disponible).

Para la maquetación de la aplicación se utilizará Bootstrap, haciendo lo posible para que la aplicación sea jugable tanto en un navegador de escritorio (con una ventana utilizable grande) como en un móvil (con una pantalla utilizable pequeña, en la que no se podrán ver todos los elementos del juego a la vez). Se utilizará, en la medida de lo razonable, CSS3 para todo lo relacionado con el aspecto de la aplicación. Se usará Leaflet para mostrar los mapas. Se podrán utiliziar otras

bibliotecas JavaScript en lo que pueda ser conveniente.

4.1.2. Funcionalidad optativa

En general, se podrá añadir cualquier funcionalidad a la aplicación, mientras no perjudique o impida la funcionalidad básica descrita en el apartado anterior. A modo de ejemplos, se proponen las siguientes ideas:

- Permitir seleccionar un repositorio GitHub, del que se leerán los juegos disponibles en él (leyendo el fichero "games.json" correspondiente).
- Poder poner la aplicación "en modo edición de juego", para añadir puntos a un juego cualquiera. Para ello, se mostrarían todos los puntos de un juego. Cada punto se podría eliminar, o modificar el nombre asociado con él. También se podrían añadir nuevos puntos, pulsando sobre el mapa. Cuando la edición esté lista, se producirá un nuevo fichero GeoJSON con los nuevos datos, y se almacenará con el mismo nombre que tenía.
- Poder poner la aplicación "en modo edición de juegos", para poder crear nuevos juegos, eliminar juegos, editar juegos (siguiendo el procedimiento anterior), etc.
- Dar la opción de que cada usuario designe un repositorio GitHub en el que se almacene la puntuación de cada partida en la que juega, y poder elegir también los repositorios de otros jugadores, para poder ver sus partidas jugadas y su puntuación como parte de la aplicación.
- Realiza lo necesario para que la aplicación sea una Open Web App, que funcione en Firefox OS o en el navegador Firefox instalándola como app.

Todas las opciones que necesitan escribir en repositorios GitHub, requerirían autenticación previa por parte del usuario.

4.1.3. Entrega

Fecha límite de entrega de la práctica: 24 de junio de 2015.

Fecha de publicación de notas: viernes, 26 de junio de 2015, en la plataforma Moodle.

Fecha de revisión: martes, 29 de junio de 2014 a las 13:00. Se requerirá a algunos alumnos que asistan a la revisión **en persona**; se informará de ello en el mensaje de publicación de notas.

La entrega de la práctica consiste en rellenar un formulario y en seguir las instrucciones que se describen en el apartado 5.1. El repositorio contendrá todos

los ficheros necesarios para que funcione la aplicación (ver detalle más abajo), tanto en la rama master como en la rama gh-pages. Es muy importante que el alumno haya registrado su cuenta GitHub en el sitio de la asignatura en el campus virtual, porque si no, la práctica no podrá ser identificada.

Para la entrega se utilizará un fork del repositorio siguiente:
`https://github.com/CursosWeb/X-Nav-Practica-Adivina2/`

Los alumnos que no entreguen las práctica de esta forma serán considerados como no presentados en lo que a la entrega de prácticas se refiere. Los que la entreguen podrán ser llamados a realizar también una entrega presencial, que tendrá lugar en la fecha y hora exacta se les comunicará oportunamente. Esta entrega presencial podrá incluir una conversación con el profesor sobre cualquier aspecto de la realización de la práctica.

Se han de entregar los siguientes ficheros:

- Un fichero README.md que resuma las mejoras, si las hay, y explique cualquier peculiaridad de la entrega.
- Los ficheros de la práctica (HTML, CSS, JavaScript). El fichero HTML principal se llamará "index.html", y estará construido de forma que bastará con cargarlo en el navegador para que funcione el programa.
- Cualquier biblioteca JavaScript que pueda hacer falta para que la aplicación funcione, junto con los ficheros auxiliares que utilice, si es que los utiliza.

La aplicación se probará accediendo al servidor web que monta GitHub para los contenidos de la rama gh-pages.

Se incluirán en el fichero README.md los siguientes datos (la mayoría de estos datos se piden también en el formulario que se ha de rellenar para entregar la práctica - se recomienda hacer un corta y pega de estos datos en el formulario):

- Nombre de su cuenta en el laboratorio del alumno.
- Resumen de las peculiaridades que se quieran mencionar sobre lo implementado en la parte obligatoria.
- Lista de funcionalidades opcionales que se hayan implementado, y breve descripción de cada una.
- URL del vídeo demostración de la funcionalidad básica
- URL del vídeo demostración de la funcionalidad optativa, si se ha realizado funcionalidad optativa

Asegúrate de que las URLs incluidas en este fichero están adecuadamente escritas en Markdown, de forma que la versión HTML que genera GitHub los incluya como enlaces "pinchables".

Los vídeos de demostración serán de una duración máxima de 3 minutos (cada uno), y consistirán en una captura de pantalla de un navegador web utilizando la aplicación, y mostrando lo mejor posible la funcionalidad correspondiente (básica u opcional). Siempre que sea posible, el alumno comentará en el audio del vídeo lo que vaya ocurriendo en la captura. Los vídeos se colocarán en algún servicio de subida de vídeos en Internet (por ejemplo, Vimeo o YouTube).

Hay muchas herramientas que permiten realizar la captura de pantalla. Por ejemplo, en GNU/Linux puede usarse Gtk-RecordMyDesktop o Istanbul (ambas disponibles en Ubuntu). Es importante que la captura sea realizada de forma que se distinga razonablemente lo que se grabe en el vídeo.

En caso de que convenga editar el vídeo resultante (por ejemplo, para eliminar tiempos de espera) puede usarse un editor de vídeo, pero siempre deberá ser indicado que se ha hecho tal cosa con un comentario en el audio, o un texto en el vídeo. Hay muchas herramientas que permiten realizar esta edición. Por ejemplo, en GNU/Linux puede usarse OpenShot o PiTiVi.

4.2. Adivina dónde está (mayo 2015)

La práctica consiste en la creación de una aplicación HTML5 que permita jugar a una variante del juego de las adivinanzas. Se tratará de mostrar al usuario pistas, en forma de fotografías, y que éste tenga que adivinar, marcándolo en un mapa, a qué parte del mundo se refieren.

4.2.1. Enunciado

Concretamente, la aplicación mostrará, al arrancar, un panel con:

- Una zona donde se mostrarán las fotos (de una en una).
- Una zona donde se mostrará un mapa, para que el jugador pueda marcar un punto, indicando que ese es el sitio a adivinar.
- Una zona con los controles del juego, la puntuación, etc. Los controles incluirán opciones al menos para iniciar un juego, para abortar un juego en curso, para empezar uno nuevo, para seleccionar la dificultad (cada una, visible sólo en el momento adecuado). Además, habrá forma de elegir un juego entre los disponibles, para ver los juegos que se han jugado, el momento en que se jugaron y su puntuación, y a partir de esa lista volver a jugar cualquiera de ellos.

Cuando empiece el juego, el jugador verá una foto durante un cierto tiempo. Cuando termine ese tiempo, será sustuida por otra. Cuando el jugador crea que sabe a qué parte del mundo se refiere la foto, pulsará con el ratón sobre esa parte del mundo. En ese momento, la aplicación calculará la distancia entre el punto en que ha pulsado el jugador y el punto al que se refería la adivinanza, y mostrará la distancia entre ambos, y la puntuación obtenida. La puntuación obtenida se calculará multiplicando esta distancia por el número de fotos que se han mostrado. Así, una puntuación más pequeña es una puntuación mejor.

La dificultad será un número entero, que controlará el tiempo que estará visible una fotografía antes de ser sutituida por la siguiente: a más dificultad, menos tiempo estará visible cada foto.

Cada juego mantendrá, como base para plantear las adivinanzas, un fichero GeoJSON, compuesto por una "FeatureCollection", en la que cada elemento de la colección corresponderá con un sitio a adivinar. El campo "coordinates" marcará la localización del punto a adivinar, y el campo "Name" de "properties", el nombre del sitio a adivinar (que sólo se mostrará al usuario al terminar cada juego). El nombre del sitio se utilizará como etiqueta (tag) en una búsqueda en Flickr, para obtener fotos relacionadas con ese nombre. Esas fotos son las que se mostrarán al jugador. Cada vez que se juege uno de estos juegos, la aplicación elegirá aleatoriamente uno de los puntos, que será el punto a adivinar.

Los ficheros GeoJSON se mantendrán en un repositorio en GitHub (que será el de entrega de la práctica), en un directorio con el nombre "juegos", cada uno con un nombre de fichero que será el "nombre del juego", cuando la aplicación haya de mostrarlo. Por ejemplo, el fichero "Capitales.json" tendrá los puntos del juego "Capitales" (que podría tener como puntos ciudades que sean capitales de paises).

Para mantener el estado de juegos pasados se utilizará el historial del navegador. Cada vez que se termine un juego, se anotará en el historial el nombre del juego, el momento en que se ha terminado, y la puntuación. Para localizar fácilmente las entradas en el historial que correspondan con el juego, se anotarán empezando por una cierta cadena de texto, por ejemplo "Adivinanzas:". Estos datos se utilizárán para mostrar los juegos jugados. Igualmente, si se selecciona directamente uno de esos juegos desde el historial del navegador, se comenzará a jugar ese juego.

Para la maquetación de la aplicación se utilizará Bootstrap, haciendo lo posible para que la aplicación sea jugable tanto en un navegador de escritorio (con una ventana utilizable grande) como en un móvil (con una pantalla utilizable pequeña, en la que no se podrán ver todos los elementos del juego a la vez). Se utilizará, en la medida de lo razonable, CSS3 para todo lo relacionado con el aspecto de la aplicación. Se usará Leaflet para mostrar los mapas. Se podrán utiliziar otras bibliotecas JavaScript en lo que pueda ser conveniente.

4.2.2. Funcionalidad optativa

En general, se podrá añadir cualquier funcionalidad a la aplicación, mientras no perjudique o impida la funcionalidad básica descrita en el apartado anterior. A modo de ejemplos, se proponen las siguientes ideas:

- Permitir seleccionar un repositorio GitHub, del que se leerán los juegos disponibles en él.
- Poder poner la aplicación "en modo edición de juego", para añadir puntos a un juego cualquiera. Para ello, se mostrarían todos los puntos de un juego. Cada punto se podría eliminar, o modificar el nombre asociado con él. También se podrían añadir nuevos puntos, pulsando sobre el mapa. Cuando la edición esté lista, se producirá un nuevo fichero GeoJSON con los nuevos datos, y se almacenará con el mismo nombre que tenía.
- Poder poner la aplicación "en modo edición de juegos", para poder crear nuevos juegos, eliminar juegos, editar juegos (siguiendo el procedimiento anterior), etc.
- Dar la opción de que cada usuario designe un repositorio GitHub en el que se almacene la puntuación de cada partida en la que juega, y poder elegir también los repositorios de otros jugadores, para poder ver sus partidas jugadas y su puntuación como parte de la aplicación.
- Realiza lo necesario para que la aplicación sea una Open Web App, que funcione en Firefox OS o en el navegador Firefox instalándola como app.

Todas las opciones que necesitan escribir en repositorios GitHub, requerirían autenticación previa por parte del usuario.

4.2.3. Entrega (convocatoria de junio entre paréntesis)

Fecha límite de entrega de la práctica: 24 de mayo de 2015 (24 de junio de 2015).

Fecha de publicación de notas: martes, 26 de mayo de 2015 (26 de junio), en la plataforma Moodle.

Fecha de revisión: viernes, 29 de mayo de 2014 a las 13:00 (30 de junio a las 14:00). Se requerirá a algunos alumnos que asistan a la revisión **en persona**; se informará de ello en el mensaje de publicación de notas.

La entrega de la práctica consiste en rellenar un formulario y en seguir las instrucciones que se describen en el apartado 5.1. El repositorio contendrá todos los ficheros necesarios para que funcione la aplicación (ver detalle más abajo), tanto

en la rama master como en la rama gh-pages. Es muy importante que el alumno haya registrado su cuenta GitHub en el sitio de la asignatura en el campus virtual, porque si no, la práctica no podrá ser identificada.

Para la entrega se utilizará un fork del repositorio siguiente:
`https://github.com/CursosWeb/X-Nav-Practica-Adivina/`

Los alumnos que no entreguen las práctica de esta forma serán considerados como no presentados en lo que a la entrega de prácticas se refiere. Los que la entreguen podrán ser llamados a realizar también una entrega presencial, que tendrá lugar en la fecha y hora exacta se les comunicará oportunamente. Esta entrega presencial podrá incluir una conversación con el profesor sobre cualquier aspecto de la realización de la práctica.

Se han de entregar los siguientes ficheros:

- Un fichero README.md que resuma las mejoras, si las hay, y explique cualquier peculiaridad de la entrega.
- Los ficheros de la práctica (HTML, CSS, JavaScript). El fichero HTML principal se llamará "index.html", y estará construido de forma que bastará con cargarlo en el navegador para que funcione el programa.
- Cualquier biblioteca JavaScript que pueda hacer falta para que la aplicación funcione, junto con los ficheros auxiliares que utilice, si es que los utiliza.

La aplicación se probará accediendo al servidor web que monta GitHub para los contenidos de la rama gh-pages.

Se incluirán en el fichero README.md los siguientes datos (la mayoría de estos datos se piden también en el formulario que se ha de rellenar para entregar la práctica - se recomienda hacer un corta y pega de estos datos en el formulario):

- Nombre de su cuenta en el laboratorio del alumno.
- Resumen de las peculiaridades que se quieran mencionar sobre lo implementado en la parte obligatoria.
- Lista de funcionalidades opcionales que se hayan implementado, y breve descripción de cada una.
- URL del vídeo demostración de la funcionalidad básica
- URL del vídeo demostración de la funcionalidad optativa, si se ha realizado funcionalidad optativa

Asegúrate de que las URLs incluidas en este fichero están adecuadamente escritas en Markdown, de forma que la versión HTML que genera GitHub los incluya como enlaces "pinchables".

Los vídeos de demostración serán de una duración máxima de 3 minutos (cada uno), y consistirán en una captura de pantalla de un navegador web utilizando la aplicación, y mostrando lo mejor posible la funcionalidad correspondiente (básica u opcional). Siempre que sea posible, el alumno comentará en el audio del vídeo lo que vaya ocurriendo en la captura. Los vídeos se colocarán en algún servicio de subida de vídeos en Internet (por ejemplo, Vimeo o YouTube).

Hay muchas herramientas que permiten realizar la captura de pantalla. Por ejemplo, en GNU/Linux puede usarse Gtk-RecordMyDesktop o Istanbul (ambas disponibles en Ubuntu). Es importante que la captura sea realizada de forma que se distinga razonablemente lo que se grabe en el vídeo.

En caso de que convenga editar el vídeo resultante (por ejemplo, para eliminar tiempos de espera) puede usarse un editor de vídeo, pero siempre deberá ser indicado que se ha hecho tal cosa con un comentario en el audio, o un texto en el vídeo. Hay muchas herramientas que permiten realizar esta edición. Por ejemplo, en GNU/Linux puede usarse OpenShot o PiTiVi.

4.3. Mashup de servicios (mayo 2014)

Enunciado:

La práctica va a consistir en la construcción de una aplicación HTML5 que permita al usuario acceder a varios servicios desde su navegador (construyendo, de facto, un mashup de servicios), relacionando la información que éstos proporcionan.

La aplicación ha de utilizar al menos información de Google+, Flickr, OpenStreetMap y Nominatim (estos dos últimos mediante la biblioteca Leaflet). La aplicación ha de funcionar en el navegador, y estará compuesta por documentos HTML, CSS y JavaScript. Como herramienta básica para la maquetación se usará Bootstrap, y la interacción con el usuario y la manipulación genérica del árbol DOM se gestionará usando fundamentalmente jQuery y jQueyUI.

Más en particular, la aplicación proporcionará la siguiente funcionalidad:

- La aplicación permitirá introducir en un formulario identificadores de usuario de Google+ (identificadores numéricos). Este formulario estará normalmente "escondido", representado por un icono, y se desplegará al seleccionar ese icono.

- Para cada identificador de usuario introducido, la aplicación recogerá de Google+ información sobre estos usuarios (al menos su nombre y su foto), los añadirá a la matriz de fotos que compondrá la parte principal de la página

principal de la aplicación, y los almacenará en almacenamiento estable en el navegador. También se proporcionará la posibilidad de eliminar usuarios, seleccionando un icono al efecto asociado a cada foto (este icono se hará visible sólo al colocar el ratón sobre la foto en cuestión).

- Se permitirá seleccionar uno cualquiera de los usuarios de Google+ introducidos, seleccionando su foto de alguna forma. Al hacerlo, se mostrarán en una nueva zona (pestaña o desplegable, por ejemplo) los mensajes que hayan puesto en su línea temporal. Para cada uno de estos mensajes se mostrará al menos su contenido y las coordenadas (latitud y longitud) en que fue puesto, si esta información está disponible.

- Además, en esta nueva zona, se mostrará la localización de cada uno de los mensajes que aparezcan se mostrará (con una marca o similar) en un mapa servido por OpenStreetMap. Al cambiar el usuario seleccionado de Google+, cambiarán también las marcas de localización.

- Si se selecciona una marca en particular, se resaltarán de alguna forma la marca y el mensaje en cuestión, y se mostrarán sus coordenadas y la dirección más cercana (usando Nominatim). Esto se podrá realizar repetidamente, mostrando resaltados todos los mensajes correspondientes. Habrá alguna forma (un botón, por ejemplo) para dejar de resaltar todos los mensajes. Igualmente, se podrá resaltar un mensaje (por ejemplo, arrastrándolo sobre un icono de resalte), en cuyo caso se resaltará también la marca correspondiente, y mostrándose también las coordenadas y la dirección, como se indicó anteriormente.

- Cuando se resalte un mensaje, se mostrarán también en alguna zona de la página fotos de Flickr que tengan como etiqueta el nombre de la ciudad correspondiente con la dirección de ese mensaje. Al resaltar más mensajes, se irán sustituyendo las fotos por otras de las nuevas localizaciones. Al cambiar el usuario seleccionado de Google+, se eliminarán todas las fotos.

Cada una de las acciones que se describen se podrán decorar con transiciones u otros efectos según el interés del alumno. Además, el alumno podrá realizar otras acciones si eso le resulta conveniente (y estas otras acciones se tendrán en cuenta para la nota de la práctica).

El estudiante también podrá usar otros servicios además de los ya descritos, incorporándolos a la práctica según le parezca conveniente.

4.3.1. Funcionalidad optativa

De forma optativa, se podrá incluir cualquier funcionalidad relevante en el contexto de la asignatura. Se valorarán especialmente las funcionalidades que impliquen el uso de técnicas nuevas, o de aspectos de JavaScript, APIs, HTML y CSS3 no utilizados en los ejercicios previos, y que tengan sentido en el contexto de esta práctica y de la asignatura.

Sólo a modo de sugerencia, se incluyen algunas posibles funcionalidades optativas:

- Interfaz CSS3 cuidada
- Utilizar otras APIs
- Hacer que la aplicación (o parte de ella) se pueda usar off-line
- Hacer que la aplicación utilice Local Storage de HTML5
- Que la aplicación utilice las nuevas etiquetas semánticas de HTML5
- Que la aplicación utilice las nuevas funcionalidesde de formularios de HTML5 (no visto en clase)
- Utilizar el protocolo de autorización OAuth 2.0 en alguna API

Entrega:
Fecha límite de entrega de la práctica: 14 de mayo de 2014.

La práctica se entregará subiéndola al recurso habilitado a tal fin en el sitio Moodle de la asignatura. Los alumnos que no entreguen las práctica de esta forma serán considerados como no presentados en lo que a la entrega de prácticas se refiere. Los que la entreguen podrán ser llamados a realizar también una entrega presencial, que tendrá lugar en la fecha y hora exacta se les comunicará oportunamente. Esta entrega presencial podrá incluir una conversación con el profesor sobre cualquier aspecto de la realización de la práctica.

Para entregar la práctica en el Moodle, cada alumno subirá al recurso habilitado a tal fin un fichero tar.gz con todo el código fuente de la práctica. El fichero se habrá de llamar practica-user.tar.gz, siendo "user" el nombre de la cuenta del alumno en el laboratorio. Se han de entregar los siguientes ficheros:

- Un fichero README que resuma las mejoras, si las hay, y explique cualquier peculiaridad de la entrega.
- Los ficheros de la práctica (HTML, CSS, JavaScript).

- Cualquier biblioteca JavaScript que pueda hacer falta (normalmente, sólo jQuery, jQueryUI y OpenLayers) para que la aplicación funcione, junto con los ficheros auxiliares que utilice, si es que los utiliza.

El fichero HTML se llamará "redes.html", y el archivo que se entregue estará construido de tal forma que una vez se haya descomprimido, bastará con servirlo mediante un servidor HTTP, y cargar en un navegador este fichero HTML, para que la vista de la aplicación, y todo su interfaz de usuario, funcione.

Se incluirá en el fichero README los siguientes datos:

- Nombre de la asignatura.
- Nombre completo del alumno.
- Nombre de su cuenta en el laboratorio.
- Resumen de las peculiaridades que se quieran mencionar sobre lo implementado en la parte obligatoria.
- Lista de funcionalidades opcionales que se hayan implementado, y breve descripción de cada una.
- URL del vídeo demostración de la funcionalidad básica
- URL del vídeo demostración de la funcionalidad optativa, si se ha realizado funcionalidad optativa

El fichero README se incluirá también como comentario en el recurso de subida de la práctica, asegurándose de que las URLs incluidas en él son enlaces "pinchables".

Los vídeos de demostración serán de una duración máxima de 3 minutos (cada uno), y consistirán en una captura de pantalla de un navegador web utilizando la aplicación, y mostrando lo mejor posible la funcionalidad correspondiente (básica u opcional). Siempre que sea posible, el alumno comentará en el audio del vídeo lo que vaya ocurriendo en la captura. Los vídeos se colocarán en algún servicio de subida de vídeos en Internet (por ejemplo, Vimeo o YouTube).

Hay muchas herramientas que permiten realizar la captura de pantalla. Por ejemplo, en GNU/Linux puede usarse Gtk-RecordMyDesktop o Istanbul (ambas disponibles en Ubuntu). Es importante que la captura sea realizada de forma que se distinga razonablemente lo que se grabe en el vídeo.

En caso de que convenga editar el vídeo resultante (por ejemplo, para eliminar tiempos de espera) puede usarse un editor de vídeo, pero siempre deberá ser indicado que se ha hecho tal cosa con un comentario en el audio, o un texto en el vídeo. Hay muchas herramientas que permiten realizar esta edición. Por ejemplo, en GNU/Linux puede usarse OpenShot o PiTiVi.

4.4. Mashup de servicios (mayo 2013)

Enunciado:

La práctica va a consistir en la construcción de una aplicación HTML5 que permita al usuario acceder a varios servicios desde su navegador (construyendo, de facto, un mashup de servicios), relacionando la información que éstos proporcionan.

La aplicación ha de utilizar al menos información de Google+, Flickr, OpenStreetMap y Nominatim (estos dos últimos mediante la biblioteca OpenLayers). La aplicación ha de funcionar en el navegador, y estará compuesta por documentos HTML, CSS y JavaScript.

Más en particular, la aplicación proporcionará la siguiente funcionalidad:

- En una pestaña, la aplicación permitirá introducir en un formulario identificadores de usuario de Google+ (identificadores numéricos). La aplicación recogerá de Google+ información sobre estos usuarios (al menos su nombre y su foto), los mostrará, y los almacenará en almacenamiento estable en el navegador. También debe proporcionarse algún mecanismo para eliminar usuarios.

- En otra pestaña se proporcionará la funcionalidad descrita en los siguientes apartados.

- Se permitirá seleccionar uno de los usuarios de Google+ introducidos (preferiblemente, seleccionando su foto de alguna forma), y mostrará los mensajes que hayan puesto en su línea temporal. Para cada uno de estos mensajes se mostrará al menos su contenido, y en un elemento desplegable, las coordenadas (latitud y longitud) en que fue puesto, si esta información está disponible.

- La localización de cada uno de los mensajes que aparezcan se mostrará (con una marca o similar) en un mapa servido por OpenStreetMap. Al cambiar el usuario seleccionado de Google+, cambiarán también las marcas de localización.

- Si se selecciona una marca en particular, se resaltarán de alguna forma la marca y el mensaje en cuestión, y se mostrarán sus coordenadas y la dirección más cercana (usando Nominatim). Esto se podrá realizar repetidamente, mostrando resaltados todos los mensajes correspondientes. Habrá alguna forma (un botón, por ejemplo) para dejar de resaltar todos los mensajes. Igualmente, se podrá resaltar un mensaje (por ejemplo, arrastrándolo sobre un icono de resalte), en cuyo caso se resaltará también la marca correspondiente, y mostrándose también las coordenadas y la dirección, como se indicó anteriormente.

- Cuando se resalte un mensaje, se mostrarán también en alguna zona de la página fotos de Flickr que tengan como etiqueta el nombre de la ciudad correspondiente con la dirección de ese mensaje. Al resaltar más mensajes, se irán sustituyendo las fotos por otras de las nuevas localizaciones. Al cambiar el usuario seleccionado de Google+, se eliminarán todas las fotos.

Cada una de las acciones que se describen se podrán decorar con transiciones u otros efectos según el interés del alumno. Además, el alumno podrá realizar otras acciones si eso le resulta conveniente (y estas otras acciones se tendrán en cuenta para la nota de la práctica).

El estudiante también podrá usar otros servicios además de los ya descritos, incorporándolos a la práctica según le parezca conveniente.

4.4.1. Funcionalidad optativa

De forma optativa, se podrá incluir cualquier funcionalidad relevante en el contexto de la asignatura. Se valorarán especialmente las funcionalidades que impliquen el uso de técnicas nuevas, o de aspectos de JavaScript, APIs, HTML y CSS3 no utilizados en los ejercicios previos, y que tengan sentido en el contexto de esta práctica y de la asignatura.

Sólo a modo de sugerencia, se incluyen algunas posibles funcionalidades optativas:

- Interfaz CSS3 cuidada
- Utilizar otras APIs
- Hacer que la aplicación (o parte de ella) se pueda usar off-line
- Hacer que la aplicación utilice Local Storage de HTML5
- Que la aplicación utilice las nuevas etiquetas semánticas de HTML5
- Que la aplicación utilice las nuevas funcionalidesde de formularios de HTML5 (no visto en clase)
- Utilizar el protocolo de autorización OAuth 2.0 en alguna API

Entrega:

La práctica se entregará subiéndola al recurso habilitado a tal fin en el sitio Moodle de la asignatura. Los alumnos que no entreguen las práctica de esta forma serán considerados como no presentados en lo que a la entrega de prácticas se refiere. Los que la entreguen podrán ser llamados a realizar también una entrega

presencial, que tendrá lugar en la fecha y hora exacta se les comunicará oportunamente. Esta entrega presencial podrá incluir una conversación con el profesor sobre cualquier aspecto de la realización de la práctica.

Para entregar la práctica en el Moodle, cada alumno subirá al recurso habilitado a tal fin un fichero tar.gz con todo el código fuente de la práctica. El fichero se habrá de llamar practica-user.tar.gz, siendo "user" el nombre de la cuenta del alumno en el laboratorio. Se han de entregar los siguientes ficheros:

- Un fichero README que resuma las mejoras, si las hay, y explique cualquier peculiaridad de la entrega.
- Los ficheros de la práctica (HTML, CSS, JavaScript).
- Cualquier biblioteca JavaScript que pueda hacer falta (normalmente, sólo jQuery, jQueryUI y OpenLayers) para que la aplicación funcione, junto con los ficheros auxiliares que utilice, si es que los utiliza.

El fichero HTML se llamará "redes.html", y el archivo que se entregue estará construido de tal forma que una vez se haya descomprimido, bastará con servirlo mediante un servidor HTTP, y cargar en un navegador este fichero HTML, para que la vista de la aplicación, y todo su interfaz de usuario, funcione.

Se incluirá en el fichero README los siguientes datos:

- Nombre de la asignatura.
- Nombre completo del alumno.
- Nombre de su cuenta en el laboratorio.
- Resumen de las peculiaridades que se quieran mencionar sobre lo implementado en la parte obligatoria.
- Lista de funcionalidades opcionales que se hayan implementado, y breve descripción de cada una.
- URL del vídeo demostración de la funcionalidad básica
- URL del vídeo demostración de la funcionalidad optativa, si se ha realizado funcionalidad optativa

El fichero README se incluirá también como comentario en el recurso de subida de la práctica, asegurándose de que las URLs incluidas en él son enlaces "pinchables".

Los vídeos de demostración serán de una duración máxima de 3 minutos (cada uno), y consistirán en una captura de pantalla de un navegador web utilizando la aplicación, y mostrando lo mejor posible la funcionalidad correspondiente (básica u opcional). Siempre que sea posible, el alumno comentará en el audio del vídeo lo que vaya ocurriendo en la captura. Los vídeos se colocarán en algún servicio de subida de vídeos en Internet (por ejemplo, Vimeo o YouTube).

Hay muchas herramientas que permiten realizar la captura de pantalla. Por ejemplo, en GNU/Linux puede usarse Gtk-RecordMyDesktop o Istanbul (ambas disponibles en Ubuntu). Es importante que la captura sea realizada de forma que se distinga razonablemente lo que se grabe en el vídeo.

En caso de que convenga editar el vídeo resultante (por ejemplo, para eliminar tiempos de espera) puede usarse un editor de vídeo, pero siempre deberá ser indicado que se ha hecho tal cosa con un comentario en el audio, o un texto en el vídeo. Hay muchas herramientas que permiten realizar esta edición. Por ejemplo, en GNU/Linux puede usarse OpenShot o PiTiVi.

Capítulo 5

Ejercicios

5.1. Entrega de prácticas incrementales

Para la entrega de prácticas incrementales se utilizarán repositorios git públicos alojados en GitHub. Para cada práctica entregable los profesores abrirán un repositorio público en el proyecto CursosWeb [1], con un nombre que comenzará por "X–Nav–", seguirá con el nombre del tema en el que se inscribe la práctica (por ejemplo, "JS" para el tema de introducción a JavaScript) y el identificador del ejercicio (por ejemplo, "Sumador"). Este repositorio incluirá un fichero README.md, con el enunciado de la práctica, y cualquier otro material que los profesores estimen conveniente.

Cada alumno dispondrá de una cuenta en GitHub, que usará a efectos de entrega de prácticas. Esta cuenta deberá ser apuntada en una lista, en el sitio de la asignatura en el campus virtual, cuando los profesores se lo soliciten. Si el alumno desea que no sea fácil trazar su identidad a partir de esta cuenta, puede elegir abrir una cuenta no ligada a sus datos personales: a efectos de valoración, los profesores utilizará la lista anterior. Si el alumno lo desea, puede usar la misma cuenta en GitHub para otros fines, además de para la entrega de prácticas.

Para trabajar en una práctica, los alumnos comenzarán por realizar una copia (fork) de cada uno de estos repositorios. Esto se realiza en GitHub, visitando (tras haberse autenticado con su cuenta de usuario de GitHub para entrega de prácticas) el repositorio con la práctica, y pulsando sobre la opción de realizar un fork. Una vez esto se haya hecho, el alumno tendrá un fork del repositorio en su cuenta, con los mismos contenidos que el repositorio original de la práctica. Visitando este nuevo repositorio, el alumno podrá conocer la url para clonarlo, con lo que podrá realizar su clon (copia) local, usando la orden `git clone`.

A partir de este momento, el alumno creará los ficheros que necesite en su

[1] `https://github.com/CursosWeb`

copia local, los irá marcando como cambios con `git commit` (usando previamente `git add`, si es preciso, para añadirlos a los ficheros considerados por git), y cuando lo estime conveniente, los subirá a su repositorio en GitHub usando `git push`.

Por lo tanto, el flujo normal de trabajo de un alumno con una nueva práctica será:

```
[En GitHub: visita el repositorio de la práctica en CursosWeb,
y le hace un fork, creando su propia copia del repositorio]

git clone url_copia_propia

[Se cera el directorio copia_propia, copia local del repositorio propio]

cd copia_propia
git add ... [ficheros de la práctica]
git commit .
git push
```

Conviene visitar el repositorio propio en GitHub, para comprobar que efectivamente los cambios realizados en la copia local se han propagado adecuadamente a él, tras haber invocado `git push`.

5.1.1. Uso de la rama gh-pages

La rama gh-pages en cualquier repositorio de GitHub tiene un significado especial: GitHub entenderá que ha de servir, como sitio web (esto es, como recursos HTTP) cualquier contenido que tenga esa rama. Por ejemplo, el contenido de la rama gh-pages del repositorio prueba del usuario jgbarah se podrá consultar con la siguiente url:

`http://jgbarah.github.io/prueba`

Es importante entender que GitHub no sirve directorios. Si en el repositorio prueba anterior no hay un fichero index.html, GitHub indicará que no se puede servir. Si lo hay, se mostrará su contenido al pedir el recurso /. En otras palabras, GitHub servirá los recursos que no terminen en /. En caso de que se le pida uno que termine en / (esto es, normalmente, los que corresponden con directorios), buscará un fichero index.html en él, y si lo hay lo servirá. Si no lo hay, mostrará una página de error. Por este motivo, normalmente para las prácticas entregadas, el fichero HTML principal se deberá denominar index.html.

Para crear la rama gh-pages, se pueden ejecutar las siguientes órdenes en la consola (se supone que ya se está en un repositorio de entrega de prácitcas, en el que se está trabajando en la rama master):

```
[Primero, nos aseguramos de que estamos en la rama master,
   que aparecera con asterisco]
> git branch
* master
[Creamos la rama gh-pages, que tendrá lo mismo que hay
   ahora en master]
> git checkout -b gh-pages
[Subimos la rama a GitHub]
> git push origin gh-pages
```

A partir de este momento, deberíamos poder ver la nueva rama en GitHub, y las páginas HTML que haya en la rama gh-pages deberían poder verse directamente vía la url indicada más arriba.

Si a partir de este momento se quiere trabajar en master, primero habrá que cambiar a la rama master de nuevo:

```
[Volvemos a la rama master]
> git checkout master
[Ahora seguimos trabajando en la rama master]
```

Cuando queramos actualizar la rama gh-pages con los cambios que hemos hecho en master, tendremos que usar "rebase" o "merge". Usando "merge":

```
[Cambiamos a la rama gh-pages]
> git checkout gh-pages
[Pasamos los cambios de master a la rama actual (gh-pages)]
> git merge master
[Subimos los cambios a GitHub]
> git push origin gh-pages
[Volvemos a la rama master para seguir trabajando]
> git checkout master
```

Usando "rebase":

```
[Cambiamos a la rama gh-pages]
> git checkout gh-pages
[Pasamos los cambios de master a la rama actual (gh-pages)]
> git rebase master
[Subimos los cambios a GitHub]
> git push origin gh-pages
[Volvemos a la rama master para seguir trabajando]
> git checkout master
```

Cuando hayas realizado la entrega de las prácticas en la rama gh-pages (y por favor, también en la rama master), no olvides comprobar en GitHub que los cambios aparecen en la rama gh-master, y en el sitio web correspondiente a tu repositorio.

5.1.2. Uso de la rama gh-pages con dos clones

Una forma de organizarse cuando se use la rama gh-pages es usando dos clones del repositorio de entrega: uno para la rama master, y otro para la rama gh-pages. Si se decide utilizar esta forma de funcionamiento, se comenará normalmente, clonando el fork del repositorio de entrega en GitHub. Se trabajará en ese clon normalmente, en la rama master. Cuando se quiera tener una rama gh-pages, se hará lo siguiente:

```
[En el clon normal, trabajando sobre master, se envían los cambios
    a GitHub, después de haber hecho los commits convenientes.]
> git push
[Se crea un nuevo clon del mismo repositorio, en el directorio
    practica-gh-pages]
> cd ..
> git clone url_repo practica-gh-pages
[Se crea en este nuevo clon la rama gh-pages]
> cd practica-gh-pages
> git checkout -b gh-pages
[Se sube ahora la nueva rama a GitHub]
> git push origin gh-pages
```

A partir de este momento, en el repositorio en GitHub están las dos ramas (master y gh-pages), ambas con el mismo contenido. Localmente, habrán quedado dos directorios, uno con la rama master, y otro con la rama gh-pages.

Si se quiere seguir trabajando en la rama master, se hará normalmente en el directorio local correspondiente, subiendo los cambios a GitHub (con push) cuando sea conveniente. Cuando se quiera sincronizar en la rama gh-pages con el estado actual de master, se podrá hacer sobre el directorio con la rama gh-pages:

```
> cd practica-gh-pages
> git fetch
> git rebase master
> git push origin
```

Alternativamente a rebase, se puede realizar un merge, como se indicó más arriba.

5.1.3. Más información sobre GitHub Pages

Hay más información sobre GitHub Pages y el uso de la rama gh-pages en:

- GitHub Pages:
  `http://pages.github.com/`
- User, organization and project pages (el uso que estamos haciendo es el de "project pages"):
  `http://help.github.com/articles/user-organization-and-project-pages/`

5.2. HTML

5.2.1. Espía a tu navegador

Enunciado:

El navegador hace una gran cantidad de tareas interesantes para esta asignatura. Es muy útil poder ver cómo lo hace, y aprender de los detalles que veamos. De hecho, también, en ciertos casos, se puede modificar su comportamiento. Para todo esto, se pueden usar herramientas específicas. En nuestro caso, vamos a usar el módulo "Firebug" de Firefox (también disponible para otros navegadores).

El ejercicio consiste en:

- Instalar el módulo Firebug en tu navegador
- Utilizarlo para ver la interacción HTTP al descargar una página web real.
- Utilizarlo para ver el árbol DOM de una página HTML real.

Más adelante, lo utilizaremos para otras cosas, así que si quieres jugar un rato con lo que permite hacer Firebug, mucho mejor.

Referencias

Sitio web de Firebug: `https://getfirebug.com/`

5.2.2. HTML simple

Enunciado:

Carga en el navegador la página HTML que utilizamos como ejemplo en este ejercicio, `html.html`. Observa su código fuente, y cómo se representa (renderiza) cada elemento en pantalla.

Modifica las características del CSS embebido de la página, para que tenga otro aspecto.

Añade metadatos para definir “description”, “keywords” y “author”.

Modifica la página para que utilice un CSS externo, y prueba con varios CSS diferentes.

Modifica el contenido de la página para añadir una frase tonta más.

Modifica el contenido para añadir una imagen en cualquier parte de la página.

Explora la página con Firebug.

Materiales:

- Ejemplo HTML simple: `html.html`

Comentarios adicionales:

Si quieres, puedes lanzar un servidor web muy simple, de una línea en Python, para ver desde el navegador (con Firebug) cómo se usa HTTP para acceder al fichero. Uan vez que estés en el directorio con el fichero `html.html`, ejecuta la siguiente línea

```
python -m SimpleHTTPServer
```

A continuación, puedes apuntar tu navegador a `localhost:8000/` y verás los contenidos de ese directorio, vía HTTP.

5.2.3. HTML de un sitio web

Enunciado:

Copia el documento HTML correspondiente a un servidor web, modifícalo, y sírverlo con el servidor Python de una lína. Para copiarlo, puedes usar wget, curl o la opción “guardar enlace” del navegador.

Comentarios adicionales:

Puedes empezar por copiar el documento HTML que corresponde con la página de la asignatura en Twitter.

No todos los documentos que decidas copiar se verán igual, dependiendo fundamentalmente de que las urls de los documentos relacionados (imágenes, JavaScript, CSS) sean relativas o absolutas. Teniendo esto en cuenta, explica las diferencias que puedas encontrar.

5.2.4. HTML con JavaScript

Enunciado:

Carga en el navegador la hoja HTML que utilizamos como ejemplo en este ejercicio, `html-javascript.html`. Observa su código fuente, y cómo se representa (renderiza) cada elemento en pantalla. Estudia el código JavaScript, y cómo se

ejecuta. En particular, observa dónde se obtiene una referencia al nodo del árbol DOM donde se quiere colocar la frase, y cómo se manipula éste árbol para colocarla ahí, una vez está generada.

Una vez lo hayas entendido, modifícalo para que en lugar de usar tres fragmentos para cada frase, use cuatro, cogiendo cada uno, aleatoriamente, de una lista de fragmentos.

Materiales:

- Ejemplo HTML con JavaScript: `html-javascript.html`

5.2.5. Manipulación de HTML desde Firebug

Enunciado:

Utliza los ficheros que se indican en "Materiales", más abajo, cargándolos con el navegador, para manipularlos de distintas formas con Firebug:

- Desde el panel HTML, modifica elementos HTML de la página. Añade etiquetas, añade y modifica atributos, observa a qué parte de lo que se ven en pantalla corresponde cada elemento de la página.
- Desde el panel CSS, modifica elementos del documento CSS que se usa. Añade por ejemplo `"text-align:center;"` a la etiqueta body, cambia algunas propiedades, observa los resultados.
- Desde el panel "Consola" ejecuta JavaScript para modificar elmentos. Por ejemplo, utiliza el siguiente:

```
console.log("Comenzando...")
var sentence = document.getElementById("sentence");
sentence.innerHTML = "Esta es una nueva frase"
alert(sentence.innerHTML)
```

Materiales:

- Ejemplos HTML, CSS, imagen: `html2.html`, `html2.css` y `gsyc-bg.png`

5.3. CSS

5.3.1. Añadir selectores

Enunciado:

A partir del código HTML y CSS proporcionado en el Moodle (css-ejercicio1.html), añadir los selectores CSS que faltan para aplicar los estilos deseados. Cada regla CSS incluye un comentario en el que se explica los elementos a los que debe aplicarse.

5.3.2. Añadir reglas CSS

Enunciado:

A partir del código HTML proporcionado (sin estilos) en el Moodle (css-ejercicio2.html), añadir las reglas CSS necesarias para que la página resultante tenga el mismo aspecto que el de la imagen que se muestra en el Moodle (css-ejercicio2.png).

En este ejercicio solamente es preciso conocer que la propiedad se llama color. Además, se ha de tener en cuenta que como valor se puede indicar directamente el nombre del color. En este ejercicio, se deben utilizar los colores: teal, red, blue, orange, purple, olive, fuchsia y green.

5.3.3. Márgenes y rellenos

Enunciado:

A partir de los documentos HTML y CSS proporcionados en Moodle (css-ejercicio3.html y css-ejercicio3.css), determinar las reglas CSS necesarias para añadir los siguientes márgenes y rellenos:

- El elemento #cabecera debe tener un relleno de 1em en todos los lados.
- El elemento #menu debe tener un relleno de 0.5em en todos los lados y un margen inferior de 0.5em.
- El resto de elementos (#noticias, #publicidad, #principal, #secundario) deben tener 0.5em de relleno en todos sus lados, salvo el elemento #pie, que sólo debe tener relleno en la zona superior e inferior.
- Los elementos .articulo deben mostrar una separación entre ellos de 1em.
- Las imágenes de los artículos muestran un margen de 0.5em en todos sus lados.
- El elemento #publicidad está separado 1em de su elemento superior.
- El elemento #pie debe tener un margen superior de 1em.

(También se puede encontrar en Moodle una imagen con cómo debería ser el resultado, véase el fichero css-ejercicio3-despues.gif).

5.3.4. Bordes

Enunciado:

A partir de los documentos HTML y CSS proporcionados en Moodle (css-ejercicio4.html y css-ejercicio4.css), determinar las reglas CSS necesarias para añadir los siguientes bordes:

1. Eliminar el borde gris que muestran por defecto todos los elementos.
2. El elemento #menu debe tener un borde inferior de 1 píxel y azul (#004C99).
3. El elemento #noticias muestra un borde de 1 píxel y gris claro (#C5C5C5).
4. El elemento #publicidad debe mostrar un borde discontinuo de 1 píxel y de color #CC6600.
5. El lateral formado por el elemento #secundario muestra un borde de 1 píxel y de color #CC6600.
6. El elemento #pie debe mostrar un borde superior y otro inferior de 1 píxel y color gris claro #C5C5C5.

(También se puede encontrar en Moodle una imagen con cómo debería ser el resultado, véase el fichero css-ejercicio4-despues.gif).

5.3.5. Colores e imágenes de fondo

Enunciado:

A partir de los documentos HTML y CSS proporcionados en Moodle (css-ejercicio5.html, css-ejercicio5.css, css-ejercicio5-fondo.gif y css-ejercicio5-logo.gif), determinar las reglas CSS necesarias para añadir los siguientes colores e imágenes de fondo:

1. Los elementos #noticias y #pie tiene un color de fondo gris claro (#F8F8F8).
2. El elemento #publicidad muestra un color de fondo amarillo claro (#FFF6CD).
3. Los elementos $< h2 >$ del lateral #secundario muestran un color de fondo #DB905C y un pequeño padding de .2em.
4. El fondo del elemento #menu se construye mediante una pequeña imagen llamada css-ejercicio5-fondo.gif.

5. El logotipo del sitio se muestra mediante una imagen de fondo del elemento $< h1 >$ contenido en el elemento #cabecera (la imagen se llama css-ejercicio5-logo.gif).

(También se puede encontrar en Moodle una imagen con cómo debería ser el resultado, véase el fichero css-ejercicio5-despues.gif).

5.3.6. Tipografía

Enunciado:

A partir del código HTML y CSS proporcionados en el Moodle (css-ejercicio6.html, css-ejercicio6.css), determinar las reglas CSS necesarias para añadir las siguientes propiedades a la tipografía de la página:

1. La fuente base de la página debe ser: color negro, tipo Arial, tamaño 0.9em, interlineado 1.4.

2. Los elementos $< h2 >$ de .articulo se muestran en color #CC6600, con un tamaño de letra de 1.6em, un interlineado de 1.2 y un margen inferior de 0.3em.

3. Los elementos del #menu deben mostrar un margen a su derecha de 1em y los enlaces deben ser de color blanco y tamaño de letra 1.3em.

4. El tamaño del texto de todos los contenidos de #lateral debe ser de 0.9em.

5. La fecha de cada noticia debe ocupar el espacio de toda su línea y mostrarse en color gris claro #999. El elemento $< h3 >$ de #noticias debe mostrarse de color #003366.

6. El texto del elemento #publicidad es de color gris oscuro #555 y todos los enlaces de color #CC6600.

7. Los enlaces contenidos dentro de .articulo son de color #CC6600 y todos los párrafos muestran un margen superior e inferior de 0.3em.

8. Añadir las reglas necesarias para que el contenido de #secundario se vea como en la imagen que se muestra.

9. Añadir las reglas necesarias para que el contenido de #pie se vea como en la imagen que se muestra.

(También se puede encontrar en Moodle una imagen con cómo debería ser el resultado, véase el fichero css-ejercicio6-despues.gif).

5.3.7. Creación de una cabecera de página

Enunciado:
Crea una cabecera de una página web con las siguientes características:

- Que cubra todo el ancho de la pantalla
- Que de fondo tenga la imagen css-ejercicio7.jpg (sin repetición)
- Que tenga la misma altura que la imagen
- Con el siguiente color de fondo: #233C9B
- Sin márgenes ni padding
- El color de letra ha de ser blanco, centrado, negrita y de tamaño grande (p.ej. 3em) y algo de padding superior.

¿Qué habría que cambiar para que el texto estuviera centrado también horizontalmente?

5.3.8. Creación de un pie de página

Enunciado:

- Que cubra todo el ancho de la pantalla
- Que de fondo tenga el color #192666
- Que tenga una altura de 40px
- Que tenga algo de padding superior
- Sin padding, pero con un margen inferior de 50px
- Que cubra todo el texto (nota: usar clear)
- El color de letra ha de ser #6685CC y el texto ha de estar centrado

5.3.9. Menú de navegación

Enunciado:
A partir de un listado, crear un menú de navegación horizontal. Los elementos del menú son:

- Principal (enlace a la propia página)
- URJC (enlace a la página principal de la URJC)
- ETSIT (enlace a la página de la ETSIT - URJC)
- GSyC (enlace a la página del GSyC)
- DAT (enlace a la página Moodle de la asignatura)

El listado no ha de tener viñeta y cubrir el 100 % de la pantalla.
Habrá un identificador menú con las siguientes características:

- Color de fondo: #192666
- Ningún margen, pero algo de padding superior

Cada enlace del menú tendrá, entre otras, las siguientes propiedades:

- Color de fondo: #253575
- El texto ha de tener el color #B5C4E3 y estar en negrita

5.3.10. Diseño a 2 columnas con cabecera y pie de página

Enunciado:
A la cabecera, menú y pie del ejemplo anterior, añádele un nuevo div contenedor de igual tamaño horizontal que la imagen de cabecera. Dentro de este div se ubicarán la cabecera, el menú, el contenido y el pie.

El contenido contará con dos columnas. La primera columna ha de tener las siguientes características:

- float: left;
- Ancho de 530px
- Margen superior e inferior de 15px, algo de padding

La segunda columna, por su parte, será de la siguiente manera:

- float: left;
- Ancho de 200px
- Algo de margen superior, pero sin padding
- Color de fondo #CEDBF9 e imagen css-ejercicio10-fondo-columna.gif sin repetición

El listado de la segunda columna ha de tener:

- Un margen superior e inferior de 15px
- Sin padding

Cada elemento de la lista, tendrá un borde inferior de 1px sólido de color #E0E8FA.

Cada enlace, será de la siguiente manera:

- Padding: 3px 0 3px 22px;
- Imagen de fondo: css-ejercicio10-icono-elemento.gif, 8px, no-repeat;
- Sin decoración de texto

Cuando se pase por encima con el ratón, el color de fondo cambiará a #E0E8FA y el del texto a color #192666.

Finalmente, toda la página debería estar centrada, con un margen superior de 30 px.

El resultado final ha de ser similar al que se muestra en el fichero css-ejercicio10-resultado.jpg.

5.3.11. Una caja CSS2

Enunciado:

Crea una página web con un elemento div con identificador "sandbox", que contenga una cabecera con el texto "Prueba CSS3" (siendo CSS3 una abreviatura) y un párrafo con el texto "Los cambios se realizan sobre este elemento." Indica que se usará una hoja de estilo externa y guarda la página.

Crea una hoja de estilo llamada css3_1.css para que el fondo de página sea azul. Además, los elementos han de tener las siguientes propiedades:

- div con identificador "sandbox":

- Fuente: “goudy-bookletter-1911-1”, “goudy-bookletter-1911-2”, “Baskerville”, “Georgia”, serif;
- Borde sólido de 1 píxel y rgb(21,11,11)
- Con overflow oculto
- Márgenes 40px 0 0 30px
- Padding: 20px
- Width: 440px
- z-index: 2
- Color negro y tamaño de fuente 16px

- cabecera
  - Márgenes: 0 0 20px 0
  - Tamaño de fuente: font-size: 56px
  - Espacio entre palabras: 2px

- abbr
  - Cursor como “help”
  - Tamaño de fuente 30 % más grande que el normal
  - Color #ff2166;
  - subrayado de 1px discontinuo y de color #888

- p
  - Tamaño de la fuente 30px
  - Altura de la línea: 50px
  - Sin márgenes y con ancho automático

Visualmente, el resultado ha de ser similar al del fichero css3_1.png.

5.3.12. Bordes redondeados

Enunciado:
Modifica la hoja de estilo del ejercicio anterior (la puedes llamar css3_2.css) para incluir bordes redondeados con un radio de 15px. Incluye la notación específica para navegadores, si fuera conveniente. Aumenta el tamaño del borde para que el efecto sea más evidente.

Visualmente, el resultado ha de ser similar al del fichero css3_2.png.

5.3.13. Sombra de texto

Enunciado:

Modifica la hoja de estilo del ejercicio anterior (la puedes llamar css3_3.css) para añadir sombra de 1px de blur y de color #333 a la fuente de la abreviatura. No hace falta que el elemento tenga sombra horizontal ni vertical.

Visualmente, el resultado ha de ser similar al del fichero css3_3.png.

5.3.14. Sombra de borde

Enunciado:

Modifica la hoja de estilo del ejercicio anterior (la puedes llamar css3_4.css) para tener sombra de la caja. La sombra ha de tener 20px de difuminado y blanca.

Visualmente, el resultado ha de ser similar al del fichero css3_4.png.

5.3.15. Fondo semitransparente

Enunciado:

Modifica la hoja de estilo del ejercicio anterior (la puedes llamar css3_5.css) para que el fondo de la caja sea del color rgb(180, 180, 144). Indica que el canal "alpha" sea de 0.6. Juega con el canal "alpha" para ver qué efectos visuales se pueden conseguir.

Visualmente, el resultado ha de ser similar al del fichero css3_5.png.

5.3.16. Fondo en gradiente

Enunciado:

Modifica la hoja de estilo css3_4.css (la puedes llamar css3_6.css) para conseguir un fondo de caja con gradiente. El gradiente ha de ser lineal, de arriba a abajo y del color #444444 al color #999999.

Visualmente, el resultado ha de ser similar al del fichero css3_6.png.

5.3.17. Alpha en los bordes

Enunciado:

Modifica la hoja de estilo del ejercicio anterior (la puedes llamar css3_7.css) para que el borde del "sandbox" tenga un "alpha" de 0.2 y el color de letra en "sandbox" tenga un alpha de 0.6.

Visualmente, el resultado ha de ser similar al del fichero css3_7.png.

5.3.18. Rotación

Enunciado:

Modifica la hoja de estilo del ejercicio anterior (la puedes llamar css3_8.css) para que la caja se encuentre rotada 7.5 grados. Deberás añadir un margen apropiado para que el resultado quede visualmente aceptable.

Visualmente, el resultado ha de ser similar al del fichero css3_8.png.

5.3.19. Escalado

Enunciado:

Modifica la hoja de estilo del ejercicio anterior (la puedes llamar css3_9.css) para que la caja tenga un 80 % del tamaño original.

Visualmente, el resultado ha de ser similar al del fichero css3_9.png.

5.3.20. Rotación en el eje de las Y

Enunciado:

Modifica la hoja de estilo del ejercicio "Alpha con bordes" que habíamos guardado como css3_7.css (la nueva la puedes llamar css3_10.css) para que el contenido de la caja rote 180 grados en el eje de las Y.

Visualmente, el resultado ha de ser similar al del fichero css3_10.png.

5.3.21. Animación

Enunciado:

Modifica la hoja de estilo del ejercicio "Alpha con bordes" que habíamos guardado como css3_7.css (la nueva la puedes llamar css3_11.css) para incluir una animación sencilla. La animación tendrá un valor de opacidad 0 a los 0 segundos, de 0.5 a los 2.5 segundos y de 1 a los 5 segundos. Para realizar la animación, hace falta definir un elemento @keyframes.

Visualmente, el resultado ha de ser similar al del fichero css3_7.png, pero al principio al página web estará vacía y a los 2.5 segundos se mostrará la caja con una opacidad de 0.5.

5.3.22. Transiciones

Enunciado:

Replicar el ejemplo "Example E: On a roll" con transiciones CSS3 que se puede encontrar en: http://www.css3.info/preview/css3-transitions/. Modificar el ejem-

plo para que la caja empiece en la derecha y sea de color amarillo, y termine en la izquierda siendo de color verde.

5.3.23. Tu hoja de estilo CSS3

Enunciado: Toma la página urjc.html que podrás encontrar en la carpeta CSS3 dentro de P2_CSS y añade las instrucciones CSS (incluidas CSS3) que creas conveniente (puedes añadir imágenes también).

5.4. Bootstrap

5.4.1. Inspeccionando Bootstrap

Enunciado: Descárgate la última versión de Bootstrap en tu ordenador (tendrás que ir para ello a `http://getboostrap.com`. Inspecciona qué elementos se han descargado en las carpetas. Deberás reconocer hojas de estilo CSS, Javascript y ficheros con fuentes e iconos.

5.4.2. Una sencilla página con Bootstrap

Enunciado: Descárgate la plantilla básica de Bootstrap y crea un página web simple añadiendo algunos elementos que te ofrece la plataforma, como por ejemplo, un *navbar*, un *jumbotron* y unos cuantos botones.

5.4.3. Utilizando el Jumbotron de Bootstrap

Enunciado: Parte del esquema "Jumbotron" que ofrece Bootstrap y modifica la página para crear una página que se parezca visualmente a la página de la asignatura, pero que contenga información sobre uno de tus *hobbies*.

5.4.4. Utilizando el Carousel de Bootstrap

Enunciado: Utilizando componentes Bootstrap, realiza una página para la ETSIT URJC. La página web ha de usar al menos:

- Un carousel (con su barra de navegación)
- Al menos dos componentes más de Boostrap (a elección del alumno)
- Al menos un componente de Bootsnipp

Para realizar el ejercicio, debes crear un *fork* del repositorio del ejercicio 5.4.4, donde se incluyen ya algunos elemenos que te servirán de punto de partida (un index.html, el CSS y JavaScript de `carousel`, así como algunas imágenes de la Escuela y del Campus de Fuenlabrada, que quizás tengas que redimensionar). Puedes añadir tantos archivos CSS, Javascript e imágenes como desees, pero sólo puede haber una página web (`index.html`).

5.4.5. El grid de Bootstrap

Enunciado: Modifica la página web de la asignatura para que las tres columnas se conviertan en cuatro. Para ello puedes hacer un *clone* o un *fork* del repositorio `https://github.com/CursosWeb/CursosWeb.github.io`.

Al final del ejercicio, pásale `Bootlint` para comprobar que tu diseño está libre de errores.

5.4.6. Bootstrap responsivo

Enunciado: Modifica la página web de la asignatura para que sea *responsiva*. Antes de nada, añade imágenes a los apartados de programas, código, transparencias y recursos, dejando la imagen en una columna y el texto en otra. Luego, haz la página responsiva, de manera que según se vaya haciendo la pantalla más pequeña:

- Las columnas de arriba deberían pasar de cuatro a dos.
- Las columnas con las imágenes no se verán en dispositivos móviles.

Para realizar este ejercicio, puedes hacer un *clone* o un *fork* del repositorio `https://github.com/CursosWeb/CursosWeb.github.io`.

5.4.7. Concurso: Tu diseño Bootstrap

Enunciado: Realiza una página web para un equipo de investigación. Para eso, toma los elementos gráficos que hay en el repositorio de GitHub y crea una página web utilizando componentes de Bootstrap. La única limitación es que todos los componentes utilizados tengan una licencia libre.

Se hará un concurso en clase, donde los estudiantes podrán votar los mejores diseños. De estos, se escogerán los cinco más votados -habrá premios para todos ellos- y los integrantes del equipo de investigación elegirán al ganador -que, lógicamente tendrá un premio algo mejor.

5.5. JavaScript

5.5.1. Iteración sobre un objecto

Enunciado:
Define el siguiente objeto:

```
movie={title: 'The Godfather',
    'releaseInfo': {'year': 1972, rating: 'PG'}
}
```

Escribe un iterador que muestre el nombre y valor de sus propiedades.
Comentarios:
Puedes empezar a trabajar sobre el siguiente código:

```
for (i in movie) {
    console.log(i);
    console.log(movie[i]);
    for (ii in i) {
        console.log(ii)
        console.log(movie[i][ii])
    }
}
```

5.5.2. Vacía página

Enunciado:
Escribe una función JavaScript que vacíe el contenido de una página HTML cualquiera, dejando sólo un elmento $< body >$ vacío.

A continuación, puedes ser más selectivo, escribiendo una función que vacíe todos los elementos con una etiqueta dada.
Comentarios:
Una posible solución a la primera parte es la siguiente:

```
p = document.getElementsByTagName('body');
p[0].innerHTML='';
```

Para la segunda parte, se puede utilizar una función como la siguiente. Pero ojo, esta función podría no funcionar bien si hay elementos con el tag a borrar anidados dentro de otros elementos con el mismo tag: podría borrar antes los nodos más altos del árbol, de forma que cuando va a borrar los anidados (más bajos), ya no existen. Se anima al alumno a explorar soluciones para estos casos.

```
function Empty (tag) {
    elements = document.getElementsByTagName(tag);
    for (element in elements) {
        elements[element].innerHTML='';
    }
}
```

5.5.3. De lista a lista ordenada

Enunciado:

Utilizando codigo JavaScript, modifica una página HTML convirtiendo todos los elementos $< ul >$ en elementos $< ol >$.

Comentarios:

Se pueden buscar soluciones a partir del próximo código:

```
document.getElementsByTagName('body')
for ... {
    element.innerHTML.replace(/<ul>/, '<ol>')
    element.innerHTML.replace(/</ul>/, '</ol>')
}
```

Pero serían mejores soluciones (y se anima al alumno a buscarlas) las que sustituyan los nodos $< ul >$ por nodos $< ol >$ con el mismo contenido.

5.5.4. Función que cambia un elemento HTML

Enunciado:

Escribe una función JavaScript que acepte dos argumentos: un identificador de elemento HTML, y una cadena de texto HTML. La función, cuando sea ejecutada, buscará ese elemento en la página HTML que está cargada en el navegador, y cambiará su contenido HTML por el que indique el segundo argumento.

Comentarios:

Una posible solución es la siguiente:

```
function changer (id, newValue) {
    var element = document.getElementById(id);
    element.innerHTML = newValue;
}
```

Suponiendo que haya un elemento con identificador "sentence" en la página HTML cargada en el navegador, puedes probarla ejecutando, por ejemplo, las siguientes sentencias (ejecuta primero una y luego otra, para ver lo que hace cada una:

```
changer("sentence", "Hasta luego")
changer("sentence", "<ul><li>Lo que hay que...</li>
  <li>...ver</li></ul>")
```

5.5.5. Sumador JavaScript muy simple

Enunciado:

Utilizando la función creada en el ejercicio "Función que cambia un elemento HTML" (ejercicio 5.5.4), escribir una función que realice una suma. Para ello, partimos de una página HTML que tenga el siguiente código:

```
<p>Adder...<span id='op'>5+3</span>
<span id="res"></span> <a href="">Add!</a></p>
```

El objetivo es que cuando se pulse sobre "Add!" aparezca a continuación de "5+3" la cadena "=8" (siendo 8 la suma de los dos operandos). Para ello la función a escribir aceptará como parámetros dos cadena de texto, que serán el identificador del elemento HTML donde está la operación, y el del elemento donde se escribirá el resultado.

Se considera que la operación estará siempre en formato "primer número, +, segundo número". La función tendrá que buscar la operación, identificar los operandos en ella (por ejemplo, usando "split"), realizar la suma y, utilizando la función de cambio de un elemento HTML, escribir el contenido en el elemento HTML del resultado.

Resolver el problema incluyendo todo lo necesario en una página HTML

Materiales:

- Ejemplo de solución: fichero `calculator.html`.

5.5.6. Sumador JavaScript muy simple con sumas aleatorias

Enunciado:

Realiza un sumador como el descrito en el ejercicio "Sumador JavaScript muy simple" (ejercicio 5.5.5), pero de forma que se incluya un texto en el que, si se pica con el ratón, se genere una nueva suma con dos sumandos aleatorios.

Materiales:

- Ejemplo de solución: fichero `calculator-random.html`.

5.5.7. Mostrador aleatorio de imágenes

Enunciado:

Escribe una página HTML con el código JavaScript asociado necesario, de forma que cuando se pica con el ratón sobre un texto, se muestre una imagen aleatoria entre una lista de urls de imágenes que se encuentra en la propia página.

Materiales:

- Ejemplo de solución: fichero `photos-random.html`.

5.5.8. JSFiddle

Enunciado:

Utiliza el servicio gratuido ofrecido por JSFiddle para ejecutar alguna de las prácticas anteriores que incluyen código HTML y JavaScript.

Cuando te funcione, sálvalo (consiguiendo así una url única), y pásale la url resultante a otro compañero, para que lo vea. Haz tú lo mismo con la url que te pase un compañero.

Materiales:

Servicio JSFiddle: `http://jsfiddle.net/`

Comentarios:

En JSFIDDLE se puede probar código JavaScript actuando sobre HTML y CSS de forma cómoda. Luego, ese mismo código puede componerse en una única página HTML, o en varios documentos (por ejemplo, un documento HTML que referencie un documento CSS y otro JavaScript.

5.5.9. Greasemonkey

Enunciado:

Instala el módulo "Greasemonkey" para Firefox, e instálale un script que escriba un texto sobre el logo de la página principal de google.com cuando ésta se cargue.

A continuación, instala un script de UserScripts.org, pruébalo, mira (y trata de entender) su código.

Realiza un script JavaScript que se pueda ejecutar con GreaseMonkey (recuerda ponerla la extensión user.js). Indica cuál es el sitio (o los sitios) con el que funciona. Incluye en el propio script una o varias reglas ïnclude"para que se ejecute sólo cuando se cargue una página relevante.

Materiales:

- Módulo Greasemonkey:
  `https://addons.mozilla.org/firefox/addon/greasemonkey/`

- Introducción a GreaseMoneky:
  `http://wiki.greasespot.net/Greasemonkey_Manual`
- Scripts para GreaseMoneky:
  `http://greasyfork.org/en`
  `http://userscripts-mirror.org/`

Comentarios:

Una vez instalado Greasemonkey, puedes crear un fichero google.user.js con un contenido como el que ves a continuación. Al cargarlo desde Firefox, si Greasemonkey está activado te preguntará si quieres instalarlo. Una vez instalado, se ejecutará automáticamente, escribiendo sobre el logo cuando se cargue la página principal de google.com.

```
function changer (id, newValue) {
    var element = document.getElementById(id);
    element.innerHTML = newValue;
}

changer ("hplogo", "<H1>Hola</H1>");
```

5.5.10. Calculadora binaria simple

Enunciado:

Escribe una aplicacion JavaScript que implemente una calculadora binaria simple. Esta calculadora sólo realizará sumas de números binarios de una cifra (esto es, 0 ó 1). La interfaz de usuario estará compuesta de los siguientes elementos:

- Un "1", que será un enlace. Cuando se pulse sobre él, aparecerá un "1" en el "display" (ver más abajo).
- Un "0", que será un enlace. Cuando se pulse sobre él, aparecerá un "0" en el "display" (ver más abajo).
- Un "+", que será un enlace. Cuando se pulse sobre él, se almacenará lo que haya en el "display" (un "0" o un "1") como primer sumando, y se borrará a continuación el contenido del "display".
- Un "=", que será un enlace. Cuando se pulse sobre él, se utlizará lo que haya en el "display" (un "0" o un "1") como segundo sumando, se sumará al primer sumando (que debería estar almacenado) y se mostrará el resultado en el "display".

- Un “display”, que mostrará lo indicado en los apartados anteriores.

Comentarios

Para empezar, puede hacerse la calculadora sin tener en cuenta las condiciones de error (por ejemplo, que se pulse “+” sin que haya nada en el “display”. Más adelante, se puede mejorar el código para que gestione adecuadamente estas condiciones de error.

Es importante darse cuenta de que cuando se almacene el primer sumando, ha de ser en una variable que esté disponible cuando posteriormente se realice la suma. En particular, es conveniente recordar que las variables locales a una función desaparecen cuando termina la ejecución de la función.

Al terminar la funcionalidad del ejercicio, se puede utilizar CSS (por ejemplo, propiedades como el color de fondo) para dar una apariencia un poco más adecuada a la calculadora.

5.5.11. Prueba de addEventListener para leer contenidos de formularios

Enunciado:

En esta práctica se va a utlizar el método addEventListener para leer los cambios que se realicen en un campo de texto de un formulario, de forma que a continuación puedan escribirse en otra parte de la página.

Para ello, se escribirá en HTML un formulario con un campo de texto, y un código JavaScript que consiga que según un usuario vaya escribiendo en el campo de texto, lo mismo que escribe vaya mostrándose en otra parte de la página.

Materiales:

- element.addEventListener en la documentación de Mozilla:
  `https://developer.mozilla.org/en/docs/DOM/element.addEventListener`
- Advanced event registration models:
  `http://www.quirksmode.org/js/events_advanced.html`
  (modelo “W3C”)
- Ejemplo de solución (con función anónima):
  Fichero `form.html`
- Ejemplo de solución (con función con nombre):
  Fichero `form-2.html`
- Ejemplo de solución (usando “onload”):
  Fichero `form-3.html`

- Ejemplo de solución (evitanto el uso de "this"):
  Fichero `form-4.html`

- Ejemplo de solución (poniendo el código JavaScript en un fichero):
  Ficheros `form-5.html`, `form-5.js`

Comentarios:

addEventListener permite asociar funciones que atiendan a eventos a los elementos que lanzan estos eventos. En este caso, lo usaremos para asociar el elemento "input" del campo de texto del formulario a una función que escriba, cuando sea invocada, el contenido (valor) en ese momento de ese campo de texto. Como el evento "input" se dispara cada vez que haya un cambio en el campo de texto, el resultado será que todo lo que se escriba en el campo de texto se repetirá en otra parte de la página.

Se recomienda hacerlo primero con una función anónima como argumento a addEventListener, y a continuación con una función con nombre.

5.5.12. Colores con addEventListener

Enunciado:

En esta práctica se va a utlizar el método addEventListener para visualizar colores según se escribe su código en el campo de texto de un formulario.

Materiales:

- Ejemplo de solución: Fichero `form-background.html`

5.6. JQuery

5.6.1. Uso de jQuery

Enunciado:

Utiliza jQuery, según se indica en el apartado "How jQuery Works" del manual "About jQuery" para realizar una página HTML tal que cuando se pulse cualquiera de sus enlaces se muestre una ventana emergente que diga "Hola".

Comentarios:

Normalmente, la versión de jQuery a usar por ahora será la de "producción", que tiene el código "minimizado". Por ejemplo, `jquery-1.11.0.min.js`

Materiales:

- Sitio de jQuery: `http://jquery.com`

- Documentación sobre jQuery: `http://jquery.com`

- Descarga de jQuery: `http://jquery.com/download/`
- Learn jQuery: `http://learn.jquery.com/`
- About jQuery: `http://learn.jquery.com/about-jquery/`
- Ejemplo de solución: Fichero `hello.html`, `hello.js`
- Ejemplo de solución (usando jQuery en una CDN): Fichero `hello-2.html`
- Ejemplo de solución (con varios elementos "a"): Fichero `hello-3.html`
- Ejemplo de solución (no siguiendo los enlaces): Fichero `hello-4.html`, `hello-4.js`

5.6.2. Cambio de colores con jQuery

Enunciado:

Utiliza jQuery para cambiar los colores de fondo de todos los elementos HTML con un cierto identificador, utilizando una clase CSS.

Igualmente, para todos los elementos "li" del identificador anterior, que cambien el texto a otro color.

Igualmente, para el último de los elementos "li" anteriores, que al pasar el ratón por encima cambie a color verde, y deje de tenerlo al dejar de estar el ratón encima (hover).

Comentarios:

Puede usarse "starterkit.html", que ya tiene clases "red", "blue" y "green" que pone el color de fondo a rojo y el color de texto a azul y verde, respectivamente, y el identificador "#orderedlist".

Materiales:

- `starterkit.html`, `starterkit.js`, `starterkit-custom.js`
- Ejemplo de solución: Fichero `colors.js` (para sustituir al "custom.js" del StarterKit)

5.6.3. Texto con jQuery

Enunciado:

Vamos a seguir usando el `starterkit.html` (ver "Cambio de colores con jQuery", ejercicio 5.6.2).

Utiliza JQuery para añadir texto a cada elemento "li" que estén dentro del elemento con identificador "#orderedlist".

Ahora, consigue que cuando se entre con el ratón sobre los elementos "li" bajo "#orderedlist2", se escriba un mensaje a su lado, y otro distinto al salir.

Por último, cambia el texto de los elementos "dd" cuando se entre con el ratón sobre ellos, volviendo a su texto original al salir.

Comentarios:

Para añadir texto a varios "li" bajo "#orderedlist" puedes usar la función "each", aplicada a cada elemento encontrado usando "find".

Materiales:

- Ejemplo de solución: Fichero `add-text-3.js` (para sustituir al "starterkit-custom.js" de `starterkit.html`)

5.6.4. Difuminado (fading) con JQuery

Enunciado:

Vamos a seguir usando el `starterkit.html` (ver "Cambio de colores con jQuery", ejercicio 5.6.2).

Utiliza JQuery para difuminar el texto de los elementos "dt" cuando se pulse el ratón sobre ellos.

A continuación, escribe el texto difuminado, añadiéndolo al elemento "h3" (además de difuminarlo) cuando se pulse el ratón sobre los mismos elementos.

Por último, haz reaparecer el texto difuminado, con un difuminado inversio, al pulsar sobre el elemento "h3"

Materiales:

- Ejemplo de solución: Fichero `fading-3.js` (para sustituir al "starterkit-custom.js" de `starterkit.html`)

5.6.5. Ejemplos simples con Ajax

Enunciado:

Escribe un programa que, una vez está cargado el árbol DOM de una página HTML, realice la petición HTTP de un documento con un texto, y lo incluya en el elemento con un cierto identificador de dicha página HTML. Para realizar la petición del documento, utiliza la función ajax() de jQuery.

Realiza lo mismo, pero ahora de forma que la petición del documento se realice cuando el usaurio pulsa el ratón sobre otro elemento.

Comentarios:

Puede utlizarse para realizar las pruebas el servidor HTTP Python de una línea que se describe en el ejercicio "HTML simple" (5.2.2).

Materiales:

- jQuery API Documentation: jQuery.ajax():
  `http://api.jquery.com/jQuery.ajax/`
- Ejemplo de solución: Fichero `ajax.tar.gz`

5.6.6. Ejemplo simple con Ajax y JSON

Enunciado:

Escribe un programa que, cuando se pulse el ratón sobre un cierto elemento de una página HTML, realice la petición HTTP de un documento JSON, y lo incluya a continuación del elemento con un cierto identificador de dicha página HTML. Para realizar la petición del documento JSON, utiliza la función getJSON() de jQuery.

Materiales:

- jQuery API Documentation: jQuery.getJSON():
  `http://api.jquery.com/jQuery.getJSON/`
- Ejemplo de solución: Fichero `json-data.tar.gz`

5.6.7. Generador de frases aleatorias

Enunciado:

Escribe un programa que al arrancar muestre una frase en una página HTML. Cuando se pulse sobre la frase, muestre otra, y así sucesivamente. Para generar las frases, cuando se cargue la página HTML se pedirá un documento JSON que tendrá tres listas de "partes" de frases. La primera lista tendrá posibles comienzos de frases, la segunda lista tendrá posibles partes medias de una frase, y la tercera tendrá posibles finales. Cada vez que haya que mostrar una frase, se elegirá aleatoriamente un elemento de cada una de las tres listas, y se generará la frase con ellas. Para realizar la petición del documento JSON, utiliza la función getJSON() de jQuery.

Materiales:

- jQuery API Documentation: jQuery.getJSON():
  `http://api.jquery.com/jQuery.getJSON/`
- Ejemplo de solución: Fichero `json-data.tar.gz`

5.6.8. Utilización de JSONP

Enunciado:

Escribe un programa que, usando la API de Flickr, y en particular el documento JSONP que incluye información sobre las últimas fotos que se han etiquetado con la etiqueta "fuenlabrada", muestre estas fotos cuando se pica en pantalla sobre un cierto elemento.

Mejora tu programa para que admita que el usuario escriba en una caja la eitqueta, o lista de etiquetas que quiera, y el programa muestre las últimas fotos con esas etiquetas.

Comentarios:

Puedes ver un ejemplo muy similar en la página de documentación de jQuery sobre getJSON.

Materiales:

- JSONP en Wikipedia:
  `http://en.wikipedia.org/wiki/JSONP`
- jQuery API Documentation: jQuery.getJSON():
  `http://api.jquery.com/jQuery.getJSON/`
- Documentación de la API de Flickr:
  `http://www.flickr.com/services/api/`
- Acceso a fichero JSONP con las últimas fotos de Flickr con etiqueta "fuenlabrada":
  `http://api.flickr.com/services/feeds/photos_public.gne?tags=fuenlabrada&tagmode=any&format=json&jsoncallback=?`

5.7. HTML5

5.7.1. La misma página, pero en HTML5

Enunciado: Modifica la página html5-original.html que sigue el estándar HTML4 (con su correspondiente hoja de estilo CSS en el archivo html5-style-original.css) para que incluya las nuevas etiquetas de HTML5. El resultado se puede ver en los ficheros html5-final.html y html5-style-final.css. Todos los ficheros necesarios se pueden encontrar en Moodle.

5.7.2. Diagrama de coordenadas con canvas

Enunciado: Crea un diagrama de coordenadas como el que se muestra en la imagen diagramacoordenadas.png (disponible en Moodle) mediante el uso de HTML5 canvas.

5.7.3. Un Paint sencillo

Enunciado: Utilizando el elemento canvas, crea un Paint sencillo que pueda pintar en el canvas mientras se mantiene el botón del ratón pulsado en los colores rojo, verde, azul, negro y blanco. Además, añade un botón para borrar el canvas.

5.7.4. Un Paint con brocha

Enunciado: Modifica el ejercicio anterior ("Un Paint sencillo", ejercicio 5.7.3) para que el usuario puede elegir también el tamaño de la brocha con la que pintar. Además, el título "Un Paint sencillo" ha de modificarse a "Un Paint algo más complicado" y ha de situarse encima del canvas.

5.7.5. Un sencillo juego con canvas

Enunciado: Estudia con detenimiento el juego sencillo realizado con canvas y javascript *simple_canvas_game* que puedes encontrar en GitHub (repositorio X-Nav-5.7.6-JuegoCanvas). Pon especial atención en las siguientes cuestiones:

- Cómo se cargan las imágenes
- Cómo se modelan los elementos (los objetos del juego)
- Cómo se utilizan las pulsaciones de teclas para cambiar el estado
- Cómo es el *loop* principal del juego

5.7.6. Mejora el juego con canvas

Enunciado: Partiendo del juego sencillo anterior, realiza las siguientes modificaciones:

- Impide que la princesa si situe entre los árboles
- Impide que el héroe salga del recinto arbolado
- Añade piedras, que el héroe ha de sortear para llegar a la princesa

- Ten en cuenta de que no puede haber una piedra suficientemente cerca de la posición de la princesa, ni en la posición inicial del príncipe
- Añade monstruos que si tocan al héroe, lo matan
- Añade lógica para que cada 10 princesas cogidas, se "pase" al siguiente nivel (con más piedras y/o monstruos más veloces)

5.7.7. Juego con estado

Enunciado: Partiendo del juego sencillo realizado con el canvas HTML5 (ejercicio 5.7.6), realiza las modificaciones oportunas para que guarde el estado de una partida y un jugador pueda volver a ese estado posteriormente.

5.7.8. Juego sin conexión

Enunciado: Partiendo del juego con estado (ejercicio 5.7.7), realiza las modificaciones pertinentes para que el juego pueda jugarse sin conexión a Internet.

Recuerda que para ello, puedes hacer uso de que el servidor Apache sirve todo aquéllo que coloquemos en el directorio /public_html de nuestro home, como http://watson.gsyc.es/ login/ (donde login es tu nombre de usuario en los laboratorios). Para que funcione correctamente, recuerda que tanto tu home como el public_html y todos los ficheros dentro de public_html deben tener permisos para que Apache pueda leerlos.

5.7.9. Modernizr: Comprobación de funcionalidad HTML5

Enunciado: Bájate e instálate la biblioteca de Javascript `Modernizr`, que se utiliza para comprobar si la funcionalidad de HTML5 está soportada por el navegador (prueba con Firefox y con Konqueror, el navegador de KDE últimamente con menos desarrollo). Comprueba si el navegador tiene la siguiente funcionalidad:

- canvas
- video
- video en formato ogg
- almacenamiento local
- apliaciones sin conexión
- geolocalización

5.7.10. Audio y vídeo con HTML5

Enunciado: Utiliza Modernizr para ver si el navegador soporta vídeo y vídeo en los diferentes formatos. Descárgate un vídeo en formato WebM o otro en formato OGG (extensión ogv) de la web y crea una página que lo muestre. Finalmente, estudia el código utilizado en YouTube para empotrar un vídeo con HTML5 en varios formatos en `http://diveintohtml5.info/video.html#example`, incluyendo el código para que se use Flash en el caso de que no estén soportados. Compara esta aproximación con la que hemos utilizado con Modernizr.

5.7.11. Geolocalización con HTML5

Enunciado: Utiliza la API de HTML5 para geolocalización para determinar tu posición. Comprueba que tu navegador tiene implementada la funcionalidad de geolocalización con Modernizr. Muestra tu localización geográfica actual en un mapa utilizando los mapas de OpenStreetMap. En caso de que el navegador no tenga implementada esta funcionalidad (utiliza `Modernizr` para comprobarlo), utiliza un `polyfill`.

5.7.12. Las antípodas

Enunciado: Crea una página web que a partir de tu geolocalización, muestre en la ventana del navegador las siguientes dos imágenes: tu localización y la de las antípodas. Utiliza `Modernizr` para comprobar si el navegador tiene implementada la funcionalidad de geolocalización y un `polyfill` en el caso de que no sea así (p.ej., en Konqueror).

5.7.13. Cálculo de números primos con Web Workers

Enunciado: Estudia el fichero webworkers.html que encontrarás en el repositorio `https://github.com/CursosWeb/X-Nav-5.7.13-WebWorkers` de GitHub. Comprueba cómo al introducir un número muy grande, el interfaz del navegador deja de ser responsivo. Implementa un `webworker` para que el cálculo de los primos se realice en *background* y el interfaz no pierda responsividad.

5.7.14. Cliente de eco con WebSocket

Enunciado: Inspecciona y ejecuta el archivo `websocket_echo.html` que encontrarás en el repositorio `https://github.com/CursosWeb/X-Nav-5.7.14-WebSocket-Echo` de GitHub para ver el funcionamiento de los `WebSockets`. Comprueba la `URI` del

servidor de `WebSockets`, los eventos que se lanzan, así como el funcionamiento general del cliente de eco.

5.7.15. Cliente y servidor de eco con WebSocket

Enunciado: En el repositorio `https://github.com/CursosWeb/X-Nav-5.7.15-WebSocket-EchoServer` de GitHub encontrarás el script en Python `SimpleWebSocketServer.py` que implementa un servidor de `WebSockets` sencillo (también tiene una variante segura).

En `SimpleExampleServer.py` se hereda del servidor WebSocket para crear servidores específicos. El que vamos a tratar en este ejercicio es el servidor de eco. Para ello, ejecuta `SimpleExampleServer.py` en modo *eco* y pruébalo con la página HTML `SimpleWebSocketClient.html`.

Modifica el servidor (en `SimpleExampleServer.py`) para que imprima por pantalla lo que recibe y envía. Asimismo, modifica el cliente (en `SimpleWebSocketClient.html`) para que no sólo muestre en la página lo que recibe, sino también lo que envía.

5.7.16. Cliente y servidor de chat con WebSocket

Enunciado: Basándote en el script `SimpleExampleServer.py`, modifica `SimpleWebSocketClien` para tener un chat que utilice el protocolo `WebSocket`. Comprueba el tráfico intercambiado entre cliente y servidor con las herramientas para desarrolladores de los navegadores.

Parte del código que encontrarás en el repositorio GitHub `https://github.com/CursosWeb/X-Nav-5.7.16-WebSocket-Chat`.

5.7.17. Canal con obsesión horaria

Enunciado: Basándote en la implementación de chat de `SimpleExampleServer.py` del ejercicio anterior, modifica el servidor para que sea un servicio de *chat* que además:

- Cada minuto en punto, envíe un mensaje a los conectados dando la hora.
- Responda a los conectados con la hora, cuando éstos se lo pidan mediante un mensaje *getTime*.

Recuerda que puedes hacer uso de la biblioteca *time* de Python.

Para realizar (y entregar) este ejercicio, haz un *fork* del repositorio GitHub `https://github.com/CursosWeb/X-Nav-5.7.17-WebSocket-TimeChat`.

5.7.18. History API - Cambiando la historia con HTML5

Enunciado: Basándote en los ficheros que puedes encontrar en el repositorio:

- Añade las páginas HTML parciales en `gallery` para cada una de las imágenes. Puedes utilizar `plantilla.html` como plantilla.
- Añade las páginas HTML completas en el directorio principal. Puedes copiar y pegar la parte correspondiente de las páginas HTML parciales.
- Modifica `gallery.js` para que utilice la biblioteca de detección de funcionalidad `Modernizr`
- Modifica el repositorio convenientemente para que se sirva como web desde `GitHub`.
- Modifica las direcciones pertinentes en `gallery` para que funcione la aplicación web.

Para realizar (y entregar) este ejercicio, haz un *fork* del repositorio GitHub `https://github.com/CursosWeb/X-Nav-5.7.18-HistoryAPI`.

5.8. Otras bibliotecas JavaScript

5.8.1. JQueryUI: Instalación y prueba

Enunciado:
Instala la biblioteca JQueryUI personalizada (customized), incluyendo todas las opciones. Sirve el directorio donde se ha instalado con el servidor Python de una línea, y carga el documento index.html en el navegador, para ver una muestra de los widgets que proporciona la biblioteca.
Materiales:

- Learn jQuery, capítulo sobre JQuery UI:
  `http://learn.jquery.com/jquery-ui/`
- jQuery: `http://jqueryui.com/`
- jQuery UI API Documentation: `http://api.jqueryui.com/`

5.8.2. JQueryUI: Uso básico

Enunciado:

Comenzaremos por crear un "selector de fechas" (date-picker), que despliega un calendario. Para introducir datos en el sector, se podrá desplegar el calendario, y la fecha elegida en él serla la que se tendrá en el selector.

A continuación, pondremos un menú con un submenú, jugando un poco con las distintas opciones que nos proporciona el elemento correspondiente de jQueryUI.

Por último, pondremos un objeto "dropable" y dos "dragable". Cuando cualquiera de los elementos "dragable" se suelte sobre el "dropable", cambiará el color y el texto de este último. Cuando se arrastra el "dragable" sacándolo del "dropable" volverá a cambiar su color y texto, para quedar como estaban. Haz que uno de los "dragable" vuelva automáticamente a su sitio cuando lo sueltes (poniéndole la propiedad adecuada).

Materiales:

- Learn jQuery, capítulo sobre JQuery UI: `http://learn.jquery.com/jquery-ui/`
- jQuery: `http://jqueryui.com/`
- jQuery UI API Documentation: `http://api.jqueryui.com/`

5.8.3. JQueryUI: Juega con JQueryUI

Enunciado:

Prepara una interfaz de usuario que muestre la potencia de JQueryUI. En la medida de lo posible, trata de que sea similar a una interfaz de usuario de una aplicación real: un escritorio, un sitio web para comprar viajes, una aplicación para interaccionar con una red social, etc. Procura experimentar con los elementos que proporciona JQueryUI, y trata de que se muestren de forma adecuada sus capacidades.

5.8.4. JQueryUI: Clon de 2048

Enunciado:

2048 es un juego que se puede implementar con relativa facilidad como una SPA (single page application). Realiza tu propia versión de ese juego (o de uno parecido) usando jQuery y jQueryUI.

Materiales:

- Versión "original" del juego 2048: `http://gabrielecirulli.github.io/2048/`

5.8.5. Elige un plugin de jQuery

Enunciado:

Elige un plugin de jQuery, y has una aplicación que lo use. Puedes elegir el que quieras, pero será más fácil si no requiere de nada específico en el lado del servidor, y si encuentas que la biblioteca tiene un cierto nivel de madurez, disponibilidad de documentación, demos, etc.

Materiales:

- Plugins de jQuery:
  `http://plugins.jquery.com/`

Comentarios:

Puedes empezar por probar las demos de la biblioteca, y hacer una aplicación mínima que corresponda con una de ellas. Luego, puedes integrar la biblioteca en cualquier otra de las prácticas que has hecho hasta el momento.

5.9. APIs JavaScript

5.9.1. Leaflet: Instalación y prueba

Enunciado:

Instala la biblioteca Leaflet, tratando de tener sólo los ficheros y directorios que hacen falta para que funcionen las aplicaciones que la usen. Utiliza la información en la "Leaflet Quick Start Guide" para mostrar un mapa en el navegador de la zona alrededor del campus de la URJC en Fuenlabrada, con un marcador con popup sobre el Aulario III.

Materiales:

- Leaflet: `http://leafletjs.com`
- Leaflet Quick Start Guide:
  `http://leafletjs.com/examples/quick-start.html`
- Documentación sobre Leaflet (tutorials):
  `http://leafletjs.com/examples.html`
- Documentación sobre Leaflet (API):
  `http://leafletjs.com/reference.html`
- Coordenadas del Aulario III: latitud 40.2838, longitud -3.8215.

- OpenStreetMap tile usage policy:
  `http://wiki.openstreetmap.org/wiki/Tile_usage_policy`
- MapQuest-OSM Tiles + MapQuest Open Aerial Tiles:
  `http://developer.mapquest.com/web/products/open/map`

5.9.2. Leaflet: Coordenadas

Enunciado:

Mejora tu solución del ejercicio "Leaflet: Instalación y prueba" (5.9.1) añadiendo lo necesario para que cuando el usuario pulse en un punto del mapa, se muestre un popup con las coordenadas (latitud y longitud) de ese punto.

Comentarios:

Puedes empezar mostrando una alerta, y luego trabajar con el popup.

5.9.3. Leaflet: Aplicación móvil

Enunciado:

Mejora tu solución del ejercicio "Leaflet: Coordenadas" (5.9.2), haciendo que el mapa se muestre a pantalla completa, colocado sobre tu situación actual (que se muestre como un círculo con un tamaño squivalente a la exactitud de la localización), y de forma que funcione en móviles.

Materiales:

- Leaflet on Mobile:
  `http://leafletjs.com/examples/mobile.html`

5.9.4. Leaflet: GeoJSON

Enunciado:

Sobre tu solución para alguno de los ejercicios anteriores (si es posible, "Leaflet: Aplicación móvil" 5.9.3, si no "Leaflet: Coordenadas" 5.9.2), coloca una capa con marcadores procedentes de un fichero GeoJSON. El mapa deberá quedar enfocado de forma que se vean los marcadores.

Materiales:

- Using GeoJSON with Leaflet:
  `http://leafletjs.com/examples/geojson.html`
- Fichero GeoJSON con localización de edificios de la URJC:
  `buildings-urjc.json`

Comentarios:

Atención al orden de las coordenadas en el objecto LatLong de Leaflet (primero latitud, luego longitud) y en la propiedad "coordinates" de GeoJSON (primero longitud, luego latitud).

5.9.5. Leaflet: Coordenadas y búsqueda de direcciones

Enunciado:

Realiza una aplicación que muestre un mapa de OpenStreetMap, usando Leaflet, con un formulario en el que se pueda introducir el nombre de un lugar. Cuando se introduzca uno, se mostrará un listado de los cinco primeros lugares que ofrezca Nominatim para ese nombre de lugar, de forma que al pulsar sobre cualquiera de ellos, el mapa se centre en ese lugar.

Materiales:

Los mismos que el ejercicio "Leaflet: Instalación y prueba" (5.9.1), y además:

- Nominatim:
  `http://nominatim.openstreetmap.org/`
- Address lookups with Leaflet and Nominatim:
  `http://derickrethans.nl/leaflet-and-nominatim.html`

5.9.6. Leaflet: Fotos de Flickr

Enunciado:

Realizar una aplicación que muestre un mapa de OpenStreetMap, usando Leaflet, con un formulario en el que se pueda introducir el nombre de un lugar. Cuando se introduzca uno, se mostrará un listado de los cinco primeros lugares que ofrezca Nominatim para ese nombre de lugar, de forma que al pulsar sobre cualquiera de ellos, el mapa se centre en ese lugar. Además, en ese momento se abrirá un panel donde se mostrarán varias fotos de Flickr que tengan como etiqueta el nombre de lugar.

Para hacer este ejercicio se usará la interfaz JSONP de Flickr.

Materiales:

- Llamadas para obtener canales (feeds) de Flickr (incluye cómo obtener JSONP como respuesta):
  `http://www.flickr.com/services/feeds/`
- Ejemplo de llamada JSONP con jQuery:
  `http://api.jquery.com/jQuery.getJSON/`

5.9.7. GitHub.js: Datos de un repositorio

Enunciado:

Utiliza la biblioteca JavaScript github.js para obtener los datos básicos de un repositorio de GitHub, dados su nombre y el nombre del usuario que es dueño de él.

El funcionamiento concreto será el siguiente:

- Al cargar la aplicación, se verá un formulario en el que se podrá introducir el token de autenticación (ver más abajo).
- Al introducir el token, la aplicación lo utilizará para crear un nuevo objeto GitHub, y mostrará un nuevo formulario en el que se podrá escribir un nombre de usuario de GitHub y el de uno de sus repositorios.
- Al introducir estos datos (usuario, nombre de repositorio), la aplicación lo utilizará para crear un nuevo objeto para acceder al repositorio.
- Cuando la aplicación haya conseguido los datos del repositorio, mostrará alguno de ellos (por ejemplo la descripción del repositorio, la fecha de creación, su nombre completo).

Comentarios:

Para que la biblioteca pueda acceder a la API HTTP de GitHub, hace falta autenticarse ante esta API. Para ello, puedes generar un token (ver más abajo enlace a documentación al respecto), y utilizarlo con el mecanismo de autenticación OAuth, que es uno de los soportados por la biblioteca. Cuando generes este token, ten la precaución de especificar el ámbito "public repositories", para que de acceso a leer y escribir en repositorios públicos. Y ten en cuenta, una vez generado, que ese token da acceso a escribir y leer los repositorios sobre los que tengas permisos en GitHub, por lo que en gran medida deberías custodiarlo como tu contraseña. En particular, no lo incluyas en el código que subas a ningún sitio público, ya que sería una acción muy similar a publicar tu contraseña.

Materiales:

- Biblioteca GitHub.js:
  `https://github.com/michael/github`
- Dirección para obtener la biblioteca GitHub.js lista para su uso:
  `https://raw.githubusercontent.com/michael/github/master/github.js`
- Instrucciones para crear un token de autenticación en GitHub:
  `https://help.github.com/articles/creating-an-access-token-for-command-line-us`

- GitHUB API v3:
  `https://developer.github.com/v3`

5.9.8. GitHub.js: Crea un fichero

Enunciado:

Amplía el ejercicio "GitHub.js: Datos de un repositorio" (5.9.7) para que además de dar los datos de un repositorio, cree en él un fichero. Para ello, incluye un nuevo formulario en el que escribirás el contenido que tendrá ese fichero, y su nombre. Cuando la aplicación los reciba, creará un nuevo fichero en la rama "master" del repositorio especificado, con ese contenido.

5.9.9. OpenLayers: Instalación y prueba

Enunciado:

Instala la biblioteca OpenLayers, tratando de tener sólo los ficheros y directorios que hacen falta para que funcionen las aplicaciones que la usen. Coloca en el mismo directorio los ficheros de la demo openlayers.html, y sirve este directorio con el servidor Python de una línea. Carga el documento openlayers.html en el navegador, para ver la demo (se mostrará un mapa, con una capa formada por teselas procedentes de OpenStreetMap).

Materiales:

- OpenLayers: `http://openlayers.org`
- OpenLayers examples:
  `http://openlayers.org/dev/examples`
- Documentación de la OpenLayers API:
  `http://dev.openlayers.org/releases/OpenLayers-2.7/doc/apidocs`
- Demo: Ficheros `openlayers.html`, `openlayers.css` y `openlayers.js`

5.9.10. OpenLayers: Capas y marcadores

Enunciado:

Modifica el resultado del ejercicio "OpenLayers: Instalación y prueba" (ejercicio 5.9.9) para:

- Añadir una capa de teselas de Bing (por ejemplo, del mapa de satélite), de forma que se pueda seleccionar ésta o la de OpenStreetMap para verla en el mapa.

- Añadir un marcador en una posición del mapa.
- Opcionalmente, incluir un formulario en la página que permita indicar unas coordenadas, de forma que se añada un marcador en el mapa en esas coordenadas.
- Opcionalmente, incluir una zona en la página HTML que muestre las coordenadas de un punto que se seleccione en el mapa.

Materiales:
Los mismos que el ejercicio "OpenLayers: Instalación y prueba" (5.9.9), y además:

- Solución (parcial): Ficheros `openlayers-marker.html`, `openlayers-marker.js`, `openlayers-bing.html` y `openlayers-bing.js`

5.9.11. OpenLayers: Coordenadas y búsqueda de direcciones

Enunciado:
Realiza una aplicación que muestre un mapa de OpenStreetMap, tal que cada vez que se pique sobre él con el ratón, muestre en una leyenda exterior al mapa:

- Las coordenadas (latitud y longitud) en grados.
- La dirección correspondiente a esas corrdenadas (utlizando el servicio proporcionado por Nominatim)

Materiales:
Los mismos que el ejercicio "OpenLayers: Instalación y prueba" (5.9.9), y además:

- Nominatim: `http://nominatim.openstreetmap.org/`
- Solución: Ficheros `openlayers-find.html`, `openlayers-find.js` y `openlayers.css`

5.9.12. OpenLayers: Fotos de Flickr

Enunciado:
Realizar una aplicación que realice una búsqueda en el servicio Flickr, de forma que muestre las fotos que tengan la etiqueta especificada. Esta etiqueta será parte del propio código JavaScript (pero se podría implementar también con un formulario, de forma que lo pueda elegir el usuario). Para hacer este ejercicio se usará la interfaz JSONP de Flickr.
Materiales:

- Llamadas para obtener canales (feeds) de Flickr (incluye cómo obtener JSONP como respuesta):
  `http://www.flickr.com/services/feeds/`
- Ejemplo de llamada JSONP con jQuery:
  `http://api.jquery.com/jQuery.getJSON/`
- Solución: Ficheros `flickr.html`, `flickr.js` y `flickr.css`

5.10. APIs de Google

5.10.1. Conociendo la Google API Console

Enunciado: Crea un proyecto para utilizar la Google API Console.

- Selecciona los servicios de Google+ API y URL Shortener API
- Obtén tu *project key*
- Añade como *referer* http://watson.gsyc.es/
- Incluye información de autorización vía OAuth2.0, también para http://watson.gsyc.es/
- Échale un ojo a los menús de informes y cuotas que te da la API Console.

Nota: Para realizar este ejercicio (y los siguientes), necesitarás tener una cuenta en Google.

5.10.2. Tu Perfil vía la API de Google+

Enunciado: Crea una página web que interatúe con la API de Google+ para obtener tu nombre y tu avatar. Puedes partir del fichero en el repositorio de GitHub `https://github.com/CursosWeb/X-Nav-5.10.2-Perfil-Google-Plus`. Al menos, habrás de realizar los siguientes cambios:

- Introduce tu llave para las APIs de Google
- Introduce tu nombre de usuario (o identificador) en Google+

5.10.3. Tomando datos de la API de Google+

Enunciado: Crea una página web que interatúe con la API de Google+ para obtener la lista de actividades de un usuario de Google+. Puedes partir del fichero en el repositorio de GitHub `https://github.com/CursosWeb/X-Nav-5.10.3-Datos-Google-Plus`. Al menos, habrás de realizar los siguientes cambios:

- Introduce tu llave para las APIs de Google
- Introduce tu nombre de usuario (o identificador) en Google+ de alguien con actividad. Puedes utilizar los siguientes identificadores en el caso de que tu usuario de Google+ no sea uno con actividad:
  - 108086881826934773478
  - 103846222472267112072
- Muestra las actividades en el HTML (anidando una lista en la página web)
- Muetra la información de localización geográfica, si ésta existiera

5.11. Firefox OS

5.11.1. Primera aplicación con FirefoxOS

Enunciado:
Partiendo de una aplicación web, crea una aplicación para FirefoxOS. Para tal fin utiliza la plantilla para una aplicación que proporciona el proyecto Mozilla, que te puedes descargar del siguiente repositorio en GitHub:
`https://github.com/mdn/battery-quickstart-starter-template`.

Realiza las siguientes actividades, teniendo en cuenta que este ejercicio sigue el *tutorial* disponible en `https://developer.mozilla.org/en-US/Apps/Build/Building_apps_for_Firefox_OS/Firefox_OS_app_beginners_tutorial`:

1. Identifica la estructura de directorios y archivos de la aplicación.
2. Añade un manifiesto, que tenga en cuenta lo siguiente:
   - Bajo permisos, has de añadir uno que sea "desktop-notification" cuya descripción sea "Needed for creating system notifications.".
   - Los caminos de los ficheros en el manifiesto han de ser relativos al servidor.
3. Lee e incluye el código de `scripts/battery.js`

4. Lee e incluye el código de `scripts/battery.js`

5. Prueba la aplicación en tu navegador

    - En el navegador Firefox de escritorio
    - En el simulador de Firefox OS

5.11.2. Open Web Apps: Aplicación para Firefox OS

Enunciado:

Crea una Open Web App para Firefox OS basándote en una de las prácticas realizadas con Leaflet. Prueba a instalarla en el simulador de Firefox OS de Firefox y en un dispositivo Android (mediante instalación desde el navegador Firefox).

Repositorio para realizar la entrega del ejercicio:
`https://github.com/CursosWeb/X-Nav-5.11.2-OpenWebApps`

Materiales:

- Uso del FirefoxOS App Manager:
  `https://developer.mozilla.org/en-US/Firefox_OS/Using_the_App_Manager`
- Para entrar al App Manager en Firefox:
  `about:app-manager`
- Instalación del FirefoxOS simulator:
  `https://ftp.mozilla.org/pub/mozilla.org/labs/fxos-simulator/`
- Icon archive:
  `http://www.iconarchive.com`
- Ejemplo de Open Web App lista para instalar:
  `http://gsyc.es/~jgb/ffxos-apps/index.html`

Comentarios:

Para usar el simulador de FirefoxOS de la forma que se indica en los materiales, es necesario tener una versión de Firefox igual o mayor que la 26, e instalar el plugin correspondiente (Firefox simulator). Se recomienda instalar la versión 1.4 de este simulador.

Para probar la app en un dispositivo Android, es necesario tener instalada en él una versión de Firefox igual o mayor que la 28.

Para convertir la aplicación en una Open Web App, no hace falta mucho más que crearle un fichero manifest.webapp adecuado. Ejemplo de manifest.webapp:

```
{
  "name": "My_Locator",
  "description": "My damn simple locator",
  "launch_path": "/map.html",
  "icons": {
    "128": "/map-icon.png"
  },
  "developer": {
    "name": "Jesus M. Gonzalez-Barahona",
    "url": "http://gsyc.es/~jgb"
  },
  "default_locale": "en",
  "permissions": {
    "geolocation": {
      "description": "Access to your position"
      }
  },
  "chrome": { "navigation": true }
}
```

Una vez la aplicación ha sido probada en el simulador, puede colocarse en un directorio para su instalación directa desde Firefox, creando un fichero .zip con todo lo que incluye, y los ficheros package.manifest e index.html adecuados.

Ejemplo de package.manifest:

```
{
    "name": "My_Locator",
    "package_path" : "http://gsyc.es/~jgb/ffxos-apps/map.zip",
    "version": "1",
    "developer": {
        "name": "Jesus M. Gonzalez-Barahona",
        "url": "http://gsyc.es/~jgb"
    }
}
```

Ejemplo de index.html:

```
<html>
  <body>
    <p>My Open Web Apps</p>
    <script>
```

```
      // This URL must be a full url.
      var manifestUrl = 'http://gsyc.es/~jgb/ffxos-apps/package.manifest';
      var req = navigator.mozApps.installPackage(manifestUrl);
      req.onsuccess = function() {
        alert(this.result.origin);
      };
      req.onerror = function() {
        alert(this.error.name);
      };
    </script>
  </body>
</html>
```

5.12. OAuth

5.12.1. OAuth con GitHub

Enunciado:

Realiza una aplicación que escriba un fichero en un repositorio de GitHub, como hacíamos en "GitHub.js: Crea un fichero" (ejercicio 5.9.8), pero usando OAuth para conseguir el token de acceso a la API de GitHub.

Comentarios:

Para acceder a la API de GitHub mediante OAuth, es preciso usar OAuth con una proxy de autenticación. Puedes usar alguna de las proxies de autenticación de uso gratuito que hay disponibles en Internet, o construir una pequeña aplicación con Python o Django para que sirva como tal. En tu aplicación, puedes usar la biblioteca "Hellojs", que permite este tipo de autenticación sobre GitHub y otros servicios, recubriendo los detalles de OAuth.

Antes de poder usar OAuth, tendrás que generar un identificador (client_id) y un secreto (client_secret) para tu aplicación, en GitHub (menú de configuración, opción aplicaciones). A esta operación se le llama también "registro de la aplicación con GitHub".

Necesitarás también una url para indicar a GitHub cuál será el "callback" al que tendrá que redirigir una vez terminado el procedimiento de autenticación.

Materiales:

- Biblioteca Hello.js:
  `http://adodson.com/hello.js/`
- Proxy OAuth para Hello.js:
  `https://auth-server.herokuapp.com/`

- OAuth en la API de GitHub:
  `https://developer.github.com/v3/oauth/`
- OAuth community site:
  `http://oauth.net/`
- RFC 6749: The OAuth 2.0 Authorization Framework
  `https://tools.ietf.org/html/rfc6749`
- OAuth 2.0 Tutorial, por Jakob Jenkov
  `http://tutorials.jenkov.com/oauth2/`

5.12.2. Listado de ficheros en GitHub

Enunciado:

Modifica la aplicación "OAuth con GitHub" (ejercicio 5.12.1) para que, antes de escribit un fichero en GitHub, muestre los ficheros disponibles en el repositorio en cuestión (rama "master"), y permita indicar un nuevo nombre de fichero y un contenido para ese fichero, y lo escriba a continuación en esa rama "master".

5.13. Ejercicios finales

5.13.1. Juego de las parejas

Enunciado:

Realiza una aplicación HTML5 que sirva para jugar a las parejas. Al empezar el juego, aparecerán en pantalla una matriz de cuatro por cuatro con 16 palabras (una en cada posición de la matriz), iguales dos a dos. Esto es, en realidad habrá 8 palabras repetidas: cada una aparecerá dos veces.

Cuando el usuario pulse la opción de comenzar, se ocultarán todas. El usuario seleccionará una posición de la matriz, que se revelará. Luego, pulsará otra, que se revelará también. Luego el jugador tendrá que pulsar en "Listo". Si ambas palabras eran iguales, ambas quedarán reveladas hasta que termine el juego. Si eran distintas, quedarán ocultadas. A continuación el usuari oseleccionará una nueva posición, y seguirá jugando hasta que todas las palabras queden reveladas, o hasta que pulse "Nuevo juego", en cuyo caso volverá a emprezar.

- Se puede usar un plazo, en lugar de un botón, antes de pasar a seleccionar un nuevo par de palabras.
- Se pueden utlizar imágenes, en lugar de palabras.

- Se pueden arrastrar las casillas una sobre otra, en lugar de seleccionar las dos.

Comentarios:
Se recomienda usar, en lo posible, jQuery y JQueryUI.