

Software y estándares para la Web

P9. APIS DE HTML5 Y SERVICIOS WEB DE CARTOGRAFÍA

Contenido

Temática del proyecto: MotoGP Desktop	3
Ejercicio 1: Introducción al API File.....	4
Tarea 1. Creación de un nuevo documento JavaScript.....	4
Tarea 2. Creación de la clase Circuito	4
Guía para resolver la tarea 2	4
Tarea 3. Lectura del archivo infoCircuito.html	4
Guía para resolver la tarea 3	4
Tarea 4. Representación de la información de infoCircuito.html en circuito.html	5
Guía para resolver la tarea 4	5
Resultado del ejercicio 1.....	5
Ejercicio 2: Lectura y representación gráfica de un archivo SVG.....	6
Tarea 1. Creación de la clase CargadorSVG	6
Guía para resolver la tarea 1	6
Tarea 2. Lectura del archivo altimetria.svg.....	6
Guía para resolver la tarea 2	6
Tarea 3. Representación del gráfico de altimetrías en circuito.html.....	6
Guía para resolver la tarea 3	6
Resultado del ejercicio 2.....	7
Ejercicio 3: Introducción al uso de mapas	8
Tarea 1. Seleccionar el proveedor de cartografía.....	8
Guía para resolver la tarea 1 con Google Maps.....	8
Guía para resolver la tarea 1 con MapBox.....	8
Tarea 2. Enlazar el proveedor de mapas en circuito.html	8
Guía para resolver la tarea 2	8
Tarea 3. Creación de la clase CargadorKML.....	9
Guía para resolver la tarea 3	9
Tarea 4. Lectura del archivo circuito.kml.....	9
Guía para resolver la tarea 4	9
Tarea 5. Representar el circuito en un mapa dinámico	10
Guía para resolver la tarea 5	10
Tarea 6. Visualización de un mapa dinámico.....	10
Guía para resolver la tarea 6	10
Tarea 7. Adaptabilidad del mapa dinámico	10
Guía para resolver la tarea 7	11

Tarea 8. Validación del código estático y el código dinámico de circuito.html	11
Guía para resolver la tarea 8	11
Resultado del ejercicio 3.....	11
Ejercicio Optativo: Menú para dispositivos móviles	12
Recuerda.....	13
Anexo I. Circuitos del mundial de MotoGP 2025	15

Objetivos

En esta práctica se va a realizar:

- El uso de APIs de HTML5 para la realización de diferentes tareas que complementen la funcionalidad de un ejercicio.
- Uso del API File de HTML5 para obtener información almacenada en archivos en la máquina local en formato HTML y en formato SVG
- El consumo de servicios web de cartografía en combinación con archivos KML para la representación de la información sobre un mapa.
- La validación del código HTML estático y el código HTML generado

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

IMPORTANTE: El uso de mapas de proveedores de cartografía externos pueden contener errores de validación del estándar HTML y errores de accesibilidad. Al tratarse de servicios externos no pueden ser corregidos, pero deben documentarse en la documentación del proyecto.

Todos los ejemplos disponibles se pueden consultar en:

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/>

(*) **ACLARACIÓN:** Los ejemplos utilizados en los ejercicios de ECMAScript contienen código ECMAScript incrustado dentro de los archivos HTML. Esto se realiza con fines didácticos para facilitar la explicación en un único archivo. En los documentos HTML no se debe incrustar código ECMAScript, salvo: referencias a archivos ECMAScript, creación de instancias de las clases e invocación de métodos.

Temática del proyecto: MotoGP Desktop

El proyecto MotoGP Desktop es evolutivo y será creado, completado y modificado en las diferentes prácticas de la asignatura.



Ejercicio 1: Introducción al API File

Las APIs de HTML5 incluidas en el estándar proporcionan funcionalidades muy variadas: acceso a la información de archivos locales, uso de pantalla completa, geolocalización, almacenamiento de la sesión del usuario, entre otras.

En este ejercicio se utilizará API File de HTML5 para leer un archivo desde la máquina cliente. Posteriormente se mostrará la información en un documento HTML.

Se va a cargar la información del archivo `infoCircuito.html` creado en las sesiones de prácticas de XML con la información del circuito asignado.

Tarea 1. Creación de un nuevo documento JavaScript

Dentro de la carpeta `js` del directorio del proyecto `MotoGP Desktop` crea un nuevo documento llamado **`circuito.js`**.

Añade una referencia a este nuevo archivo en el documento **`circuito.html`**.

Tarea 2. Creación de la clase Circuito

En el documento **`circuito.js`** crea la clase **`Circuito`**

La clase debe tener un método denominado **`comprobarApiFile`** que verifique si el navegador soporta el uso de la API File. Este método debe invocarse en el constructor de la clase **`circuito.js`**

Si el navegador no soporta el API File se debe mostrar un mensaje al usuario en el documento `circuito.html`.

Guía para resolver la tarea 2

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

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/120-atributos-privados.html>

Consulta el ejemplo incluido a continuación para entender de qué forma se puede comprobar el soporte que ofrece el navegador a determinadas características del objeto `Window`.

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/62API-FILE-LeerArchivoTexto.html>

Tarea 3. Lectura del archivo `infoCircuito.html`

Crea en la clase **`Circuito`** un método llamado **`leerArchivoHTML`** que realice la lectura del archivo **`infoCircuito.html`** y cargue su contenido que se procesará posteriormente.

Se debe utilizar API File para leer el archivo.

Guía para resolver la tarea 3

Consulta la documentación de estándar API File de HTML5 del W3C
<https://www.w3.org/TR/FileAPI/>

Consulta el código del método del ejemplo leerArchivoTexto, que es el encargado de leer el contenido del archivo.

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/62API-FILE-LeerArchivoTexto.html>

Consulta el Objeto predefinido FileReader

<https://developer.mozilla.org/es/docs/Web/API/FileReader>

Tarea 4. Representación de la información de infoCircuito.html en circuito.html

Se debe enriquecer el árbol DOM del documento **circuito.html** para incorporar la información leída del archivo infoCircuito.html.

Guía para resolver la tarea 4

Consulta la información sobre el objeto predefinido DOMParser (es una tecnología experimental implementada en todos los navegadores)
<https://developer.mozilla.org/es/docs/Web/API/DOMParser>

Utiliza los métodos de la librería jQuery o los métodos de la clase document para crear en el html el marcado que permita representar la información leída desde el archivo.

Consulta el siguiente enlace para conocer los métodos existentes en jQuery que permiten crear los diferentes elementos en el documento HTML para representar la información.

<https://api.jquery.com/add/#post-8>

Consulta los siguientes ejemplos para ver formas en las que añadir, modificar y eliminar elementos y propiedades de elementos:

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/80-jQuery-DOM-set-val-text-html.html>

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/81-jQuery-DOM-set-attr.html>

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/82-jQuery-DOM-add-elementos.html>

Resultado del ejercicio 1

Una vez completado el ejercicio deberían haberse añadido los siguientes archivos al proyecto del MotoGP Desktop:

- Archivo circuito.js en el directorio js del proyecto

Además, se debe haber modificado el archivo circuito.html para contener la información sobre el circuito.

Recuerda actualizar la información sobre el documento circuito.html que se incluye en el documento de ayuda de la web en base a las modificaciones realizadas en este ejercicio.

Ejercicio 2: Lectura y representación gráfica de un archivo SVG

En este ejercicio se utilizará API File de HTML5 para leer un archivo en formato SVG desde la máquina cliente y, posteriormente, mostrar la representación gráfica del SVG dentro de un archivo HTML.

El archivo que se va a leer es el archivo **altimetria.svg** creado en las sesiones de prácticas de XML con el perfil de altimetría del circuito asignado.

Tarea 1. Creación de la clase CargadorSVG

En el documento **circuito.js** crea la clase **CargadorSVG**, con al menos dos métodos, **leerArchivoSVG** que cargue un archivo SVG desde la máquina cliente utilizando API File e **insertarSVG** que muestre el contenido del archivo SVG en un elemento HTML.

Guía para resolver la tarea 1

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

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/120-atributos-privados.html>

Consulta el ejercicio:

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/125-API-File-cargadorSVG.html>

Tarea 2. Lectura del archivo altimetria.svg

Modifica el archivo **circuito.html** para incluir un elemento `<input>` que realice la invocación al método **leerArchivoSVG**.

Guía para resolver la tarea 2

Consulta el ejemplo incluido a continuación para comprobar cómo se debe configurar un input de html para que reciba un único archivo desde la máquina cliente e invoque a un método cuando haya un cambio en el input.

Además, consulta el código del método del código leerArchivoTexto, que es el encargado de leer el contenido del archivo.

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/62API-FILE-LeerArchivoTexto.html>

Tarea 3. Representación del gráfico de altimetrías en circuito.html

Añade al contenido de circuito.html la información leída desde el archivo **altimetria.svg**

Guía para resolver la tarea 3

Utiliza los métodos de la librería jQuery o los métodos de la clase document para crear en el html el marcado que permita representar el gráfico de altimetrías.

Consulta el siguiente enlace para conocer los métodos existentes en jQuery que permiten crear los diferentes elementos en el documento HTML para representar la información.

<https://api.jquery.com/add/#post-8>

Consulta los siguientes ejemplos para ver formas en las que añadir, modificar y eliminar elementos y propiedades de elementos:

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/80-jQuery-DOM-set-val-text-html.html>

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/81-jQuery-DOM-set-attr.html>

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/82-jQuery-DOM-add-elementos.html>

Resultado del ejercicio 2

Una vez completado el ejercicio deberían haberse modificado los siguientes archivos: el archivo `circuito.js` para incluir la clase `CargadorSVG` con el código de lectura y representación del archivo SVG, y el archivo `circuito.html` con los elementos que permiten mostrar el archivo SVG desde la máquina cliente.

Recuerda actualizar la información en el documento ayuda de la web sobre el documento `circuito.html`.

Ejercicio 3: Introducción al uso de mapas

En este ejercicio se utilizará el API File para leer un archivo desde la máquina cliente y representar la información del archivo en un mapa dinámico.

El archivo que se debe leer es el archivo **circuito.kml** creado en las sesiones de prácticas de XML.

Es posible utilizar las siguientes soluciones como proveedor de cartografía (mapas): Google Maps (<https://www.google.es/maps/preview>) y MapBox (<https://www.mapbox.com>). En caso de querer utilizar otra alternativa, se debe consultar con los profesores de la asignatura.

NOTA: Se permite el uso de la biblioteca jQuery para la realización de este ejercicio.

IMPORTANTE: En este ejercicio se debe utilizar el bloque anónimo div para la representación de los mapas dinámicos. Se permite el uso del selector div en los estilos de CSS asociados a los mapas dinámicos.

Tarea 1. Seleccionar el proveedor de cartografía

Seleccionar el proveedor de cartografía a utilizar, teniendo en cuenta sus características:

- Google Maps (<https://developers.google.com/maps>)
- MapBox (<https://docs.mapbox.com/mapbox-gl-js/api/>)

Guía para resolver la tarea 1 con Google Maps

Es obligatorio utilizar una tarjeta de crédito para obtener la API Key personal, aunque Google para un número limitado de consultas no se generan cargos. La tarjeta de crédito es utilizada para identificar a las personas.

IMPORTANTE: Está **prohibido** utilizar el API Key personal de los profesores que aparece en los ejemplos y ejercicios proporcionados.

Se debe consultar el API de Google Maps: <https://developers.google.com/maps>

Guía para resolver la tarea 1 con MapBox

Se debe crear una cuenta gratuita en **MapBox** para obtener un Access Token personal.

Se debe consultar el API de MapBox: <https://docs.mapbox.com/mapbox-gl-js/api/>

Tarea 2. Enlazar el proveedor de mapas en circuito.html

Modifica el documento circuito.html para incluir en él la(s) referencia(s) necesarias para utilizar el proveedor de mapas que has seleccionado.

Guía para resolver la tarea 2

Accede a la documentación proporcionada por el proveedor de mapas seleccionado para conocer la forma de importar el servicio en un documento HTML.

Consulta los siguientes ejemplos para comprobar cómo se importa Google Maps para utilizar sus mapas.

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/106-ClaseMapaEstaticoGoogle.html>

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/107-objetoMapaDinamicoGoogle.html>

Tarea 3. Creación de la clase CargadorKML

En el documento **circuito.js** crea la clase **CargadorKML**, con al menos dos métodos, **leerArchivoKML** (tarea 4) que cargue un archivo KML desde la máquina cliente utilizando API File y el método **insertarCapaKML** (tarea 5) que superponga un archivo KML en un mapa.

Guía para resolver la tarea 3

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

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/120-atributos-privados.html>

Tarea 4. Lectura del archivo circuito.kml

El método **leerArchivoKML** de la clase CargadorKML debe realizar la lectura del archivo **circuito.kml** utilizando API File.

Se deben extraer las coordenadas del punto origen y de los puntos que definen los tramos del circuito.

Modifica el archivo **circuito.html** para incluir un input que permita recibir un archivo desde la máquina cliente. Configura el elemento para que se realice una invocación al método **leerArchivoKML** de la clase CargadorKML cuando el cliente cargue el archivo en el input.

Guía para resolver la tarea 4

Consulta el ejemplo incluido a continuación para comprobar cómo se debe configurar un input de html para que reciba un único archivo desde la máquina cliente e invoque a un método cuando haya un cambio en el input.

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/62API-FILE-LeerArchivoTexto.html>

Para procesar KML al ser un lenguaje derivado de XML, se pueden utilizar varias opciones:

- Utilizando el objeto predefinido DOMParser (es una tecnología experimental implementada en todos los navegadores) Consulta la información en <https://developer.mozilla.org/es/docs/Web/API/DOMParser>
- Utilizando jQuery para procesar XML, consulta el ejercicio <https://web.dptoinformatica.uniovi.es/cueva/JavaScript/87-jQuery-AJAX-XML-meteo.html>

Tarea 5. Representar el circuito en un mapa dinámico

Utiliza el proveedor de mapas escogido para obtener un mapa dinámico que represente el circuito asignado.

El método **insertarCapaKML** debe incluir la información leída del fichero **circuito.kml** sobre el mapa dinámico creado.

Incluye el mapa dinámico obtenido en el documento circuito.html en un elemento <div>

Guía para resolver la tarea 5

Se debe colocar/fijar el punto origen del circuito sobre el mapa dinámico usando un marcador. Consultar el siguiente ejercicio:

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/55MarcadoresEnMapasGoogleDinamicos.html>

Se deben representar los tramos del circuito con una poli-línea sobre el mapa dinámico. Consulta el siguiente ejercicio:

<https://web.dptoinformatica.uniovi.es/cueva/JavaScript/119-Google-maps-polilinea.html>

El uso del div para la representación del mapa dinámico viene dado por la falta de una etiqueta adecuada para el caso, siguiendo la recomendación recogida en el estándar de HTML:

*Authors are strongly encouraged to view the div element as an element of last resort, for when no other element is suitable. Use of more appropriate elements instead of the div element leads to **better accessibility** for readers and **easier maintainability** for authors.*

Se puede consultar la información sobre el bloque anónimo div en el siguiente enlace: <https://html.spec.whatwg.org/multipage/grouping-content.html#the-div-element>

Tarea 6. Visualización de un mapa dinámico

Añade el código CSS necesario al proyecto MotoGP Desktop para que el mapa dinámico se visualice correctamente.

Guía para resolver la tarea 6

Dentro de la hoja de estilo **layout.css** añade el código CSS necesario para que el mapa dinámico se muestre correctamente. Para ello se puede utilizar el selector **body > div**

Tarea 7. Adaptabilidad del mapa dinámico

Se debe garantizar la adaptabilidad del mapa dinámico para dispositivos móviles

Guía para resolver la tarea 7

Dentro de la regla `@media` de la hoja de estilo **layout.css** añade el código CSS necesario para que el mapa dinámico se muestre correctamente en dispositivos móviles (por ejemplo, ocupando todo el ancho de la pantalla del dispositivo). Para ello se puede utilizar el selector **div**

Tarea 8. Validación del código estático y el código dinámico de circuito.html

Se debe validar el contenido del documento `circuito.html`

Guía para resolver la tarea 8

Valida el **código estático** del documento `circuito.html` usando la opción “Validate by File Upload” del validador de HTML del W3C.

Valida el **código dinámico** del documento `circuito.html` usando la opción “Direct Input” del validador de HTML del W3C. Obtén el código dinámico utilizando las herramientas de desarrollador del navegador.

Recuerda que aquellos documentos en los que se incluyan mapas de los diferentes proveedores de cartografía pueden tener errores de marcado HTML y de accesibilidad por la propia estructura de los mapas. Estos errores están permitidos, pero deben justificarse en la entrega de los ejercicios.

Resultado del ejercicio 3

Una vez completado el ejercicio deberían haberse modificado los siguientes archivos: el archivo `circuito.js` para incluir el código de lectura del archivo KML, el archivo `circuito.html` para incluir en él los diferentes elementos que permitan leer el archivo KML desde la máquina cliente y el archivo `layout.css` para incluir los estilos necesarios para mostrar el mapa dinámico.

Recuerda actualizar la información sobre el documento `circuito.html` que se incluye en el documento de ayuda de la web en base a las modificaciones realizadas en este ejercicio.

Ejercicio Optativo: Menú para dispositivos móviles

Realizar las modificaciones necesarias en el proyecto MotoGP Desktop para construir el menú de navegación para dispositivos móviles. El menú debe desplegarse y plegarse cuando el usuario lo requiera, accionando para ello un elemento en el HTML.

NOTA: Este ejercicio puede resolverse utilizando la biblioteca JQuery o utilizando únicamente ECMAScript puro o ‘vanilla’.

IMPORTANTE: La solución aportada para este ejercicio debe garantizar el cumplimiento de los estándares y las normas de la asignatura. Además, el menú de navegación debe ser adaptable y accesible.

La realización de este ejercicio optativo de forma correcta, eficiente y cumpliendo todos los requisitos generales de la asignatura (véase el apartado “Recuerda” del guion), permite obtener un máximo de 2 puntos adicionales en el proyecto de prácticas.

Recuerda

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

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

Solamente se permite el paradigma de orientación de objetos y todo debe estar organizado con clases y objetos. Además, no se permite el uso de ningún tipo de bibliotecas externas y debe usarse ECMAScript puro o “vanilla”, salvo en aquellos ejercicios cuyo enunciado indique que se use una biblioteca externa (por ejemplo, jQuery).

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

- **TODOS** los documentos HTML que componen el proyecto deben ser HTML5 válidos y sin advertencias utilizando el validador de lenguajes de marcado del W3C.
 - Recuerda que el contenido de los documentos HTML puede cambiar después de la ejecución del código ECMAScript. En ese caso, se debe comprobar la validez del código HTML en todos los diferentes estados por los que pase el documento.
- Se deben utilizar las etiquetas semánticas de HTML5 (section, article, etc.) y no está permitido el uso de bloques anónimos (**div**). En aquellos ejercicios en los que se permita el uso de bloques anónimos (**div**) estará indicado en el enunciado del ejercicio; **fuera de esos ejercicios, el uso de bloques anónimos no está permitido.**
- Se deben utilizar selectores de CSS específicos, el uso de selectores id y class no está permitido. **En aquellos ejercicios en los que se permita el uso de los selectores class o id estará indicado expresamente en el enunciado del ejercicio.**
- **TODAS** las reglas de todas las hojas de estilo deben estar precedidas por un comentario donde se indique la especificidad del (o los) selectores de la regla.
- **TODAS** las hojas de estilo que se utilizan en el sitio web deben ser validas utilizando el validador CSS del W3C
- **TODAS** las hojas de estilo deben tener 0 advertencias.
 - Recuerda seleccionar en “Más opciones” el informe de “Todas las advertencias” dentro de las opciones del validador CSS del W3C.
 - Excepcionalmente se permite las advertencias referidas a la verificación de los colores (color y background-color). **OBLIGATORIAMENTE** se debe indicar mediante un comentario en la regla de la hoja de estilo afectada la herencia de colores garantizando que la advertencia ha sido comprobada, verificada y garantizando que no provoca efectos laterales no deseados.
 - Excepcionalmente se permiten las advertencias referidas a la redefinición de propiedades derivadas del uso de @media-queries. **OBLIGATORIAMENTE** se debe indicar mediante un comentario en las reglas de la hoja de estilo afectada que propiedades se están redefiniendo.
- Se debe garantizar la adaptabilidad y realizar su verificación para todos los documentos que componen el proyecto.
 - Se deben utilizar medidas relativas en las hojas de estilo.

- Se debe garantizar la accesibilidad del proyecto mediante los test de las herramientas de accesibilidad para el nivel AAA de las WCAG 2.0 con 0 errores de modo automático en todos los documentos que lo componen.

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

Anexo I. Circuitos del mundial de MotoGP 2025

El listado contiene los 22 circuitos del mundial de MotoGP 2025 y las ciudades asociadas.

CODIGO	CIRCUITO	CIUDAD
1	Chang International Circuit	Buriram
2	Termas de Río Hondo	Termas de Río Hondo
3	Circuit of the Americas	Austin
4	Lusail International Circuit	Lusail
5	Circuito de Jerez – Angel Nieto	Jerez de la Frontera
6	Le Mans	Le Mans
7	Silverstone	Towcester
8	MotorLand Aragon	Teruel
9	Autodromo Internazionale del Mugello	Scarperia
10	TT Circuit Assen	Assen
11	Sachsenring	Oberlungwitz
12	Automotodrom Brno	Brno
13	Red Bull Ring - Spielberg	Spielberg (Austria)
14	Balaton Park	Balaton (Hungria)
15	Circuit de Barcelona-Catalunya	Barcelona
16	Misano World Circuit Marco Simoncelli	Misano Adriático
17	Mobility Resort Motegi	Motegi
18	Pertamina Mandalika Circuit	Baturiti (Indonesia)
19	Phillip Island	Cowes (Australia)
20	Petronas Sepang International Circuit	Kuala Lumpur
21	Autódromo Internacional do Algarve	Portimao
22	Circuit Ricardo Tormo	Valencia