Grado en Ingeniería Informática del Software

Software y estándares para la Web

P8. JQUERY Y CONSUMO DE SERVICIOS WEB

Escuela de Ingeniería Informática
Universidad de Oviedo

Contenido

Objetivos

En esta práctica se va a realizar:

- Refactorización del código aplicando buenas prácticas: separación de contenido, presentación y computación y siguiendo el paradigma de orientación a objetos.
- Utilización de la librería o Framework jQuery
- Consumo de servicios web a través de llamadas AJAX.
- Transformación de la información devuelta por un servicio web
- Modificación del árbol DOM utilizando jQuery.
- Validación de código HTML estático y código HTML generado

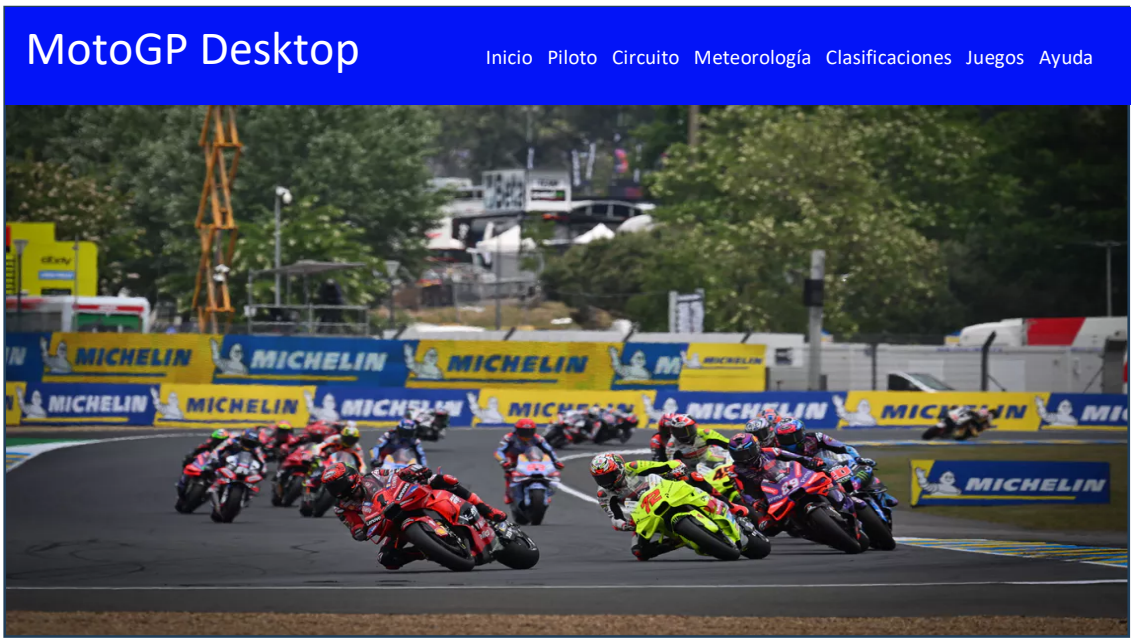
IMPORTANTE: Recuerda las pautas de trabajo establecidas en la primera sesión de prácticas (P0. Pautas de trabajo): valida todos los documentos HTML, valida todas las hojas de estilo CSS, comprueba la adaptabilidad y la accesibilidad con las herramientas proporcionadas.

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

Se va a crear MotoGP Desktop para alojar el proyecto de la asignatura. El proyecto es evolutivo y será creado, completado y modificado en las diferentes prácticas de la asignatura.



Ejercicio 1: Refactorización y buenas prácticas

La refactorización de código es un proceso que se basa en la reestructuración del código existente en un software **sin afectar a su comportamiento**, con el fin de mejorar las siguientes características de dicho código fuente: legibilidad, mantenibilidad y eficiencia.

En este ejercicio se realizará la refactorización del código desarrollado en las sesiones anteriores del proyecto MotoGP Desktop.

Tarea 1: Evitar el uso de métodos obsoletos (deprecated)

En el estándar ECMAScript, existen una serie de métodos que sido declarados obsoletos (deprecated). Un método obsoleto (deprecated) es aquel que no se recomienda usar y que no será soportado en futuras versiones de los agentes de usuario (navegadores).

Los métodos **write()** y **writeln()** del objeto predefinido **document** han sido declarados obsoletos (deprecated).

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

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

En el proyecto MotoGP Desktop se tienen que sustituir estos métodos obsoletos (deprecated) por operaciones de creación e inserción de objetos en el árbol DOM.

Guía para resolver la tarea 1

Debes buscar donde se utilizan estos métodos obsoletos (deprecated) dentro de MotoGP Desktop.

Para sustituir estos métodos debes:

- Crea el objeto que representa al elemento HTML necesario mediante document.createElement()
- Insertar el objeto creado en el árbol DOM en la posición adecuada.

Ejemplos:

- Creación e inserción de objetos de encabezado <h1> en el <body> del árbol DOM https://web.dptoinformatica.uniovi.es/cueva/JavaScript/00Hola.html
- Creación e inserción de objetos párrafo <p> en el árbol DOM https://web.dptoinformatica.uniovi.es/cueva/JavaScript/115-Saludo-OO.html
- Creación e inserción de objetos párrafo <p> en el árbol DOM sustituyendo a document.write() https://web.dptoinformatica.uniovi.es/cueva/JavaScript/68-ES6-Ejemplo-uso-de-clases.html

Tarea 2: Separación de contenido, presentación y computación

Uno de los principios fundamentales del desarrollo de software es la separación en capas, de tal manera que la lógica y los contenidos estén completamente separados. De esta forma se consigue mejorar la mantenibilidad, escalabilidad y reutilización del código.

En el caso de las aplicaciones web esto implica reducir la aparición del código ECMAScript dentro de los documentos HTML al mínimo posible, dejando el grueso del código ECMAScript en los documentos js.

Por ejemplo, el manejo de los diferentes tipos de eventos debe realizarse desde el código ECMAScript y evitar el uso de los atributos HTML destinados a tal efecto.

En el proyecto MotoGP Desktop se tienen que sustituir los atributos HTML de manejo de eventos por código ECMAScript que "escuchen" (listener) y manejen los eventos.

Debido a la dependencia que implica entre el código ECMAScript y el código HTML, no es recomendable utilizar en ECMAScript los atributos id y class (#id y .class) para seleccionar elementos del árbol DOM.

Guía para resolver la tarea 2

Busca en el código del proyecto MotoGP Desktop los atributos de manejo de eventos que se estén utilizando en los documentos HTML.

Para sustituir estos atributos debes escribir código ECMAScript que:

- Seleccionar los elementos HTML que generan eventos, por ejemplo mediante querySelector o querySelectorAll https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector utilizando como selectores una cadena válida del Módulo [CSS-SELECTORS] https://www.w3.org/Style/CSS/current-work
- Escuche los eventos registrando un evento sobre cada elemento seleccionado mediante https://developer.mozilla.org/es/docs/Web/API/EventTarget/addEventListener
- Procese los eventos mediante el objeto *event* que implementa la *infertaz Event* https://developer.mozilla.org/es/docs/Web/API/Event

Ejemplos:

- Manejo de eventos: uso no recomendable del atributo *onClick* en HTML https://web.dptoinformatica.uniovi.es/cueva/JavaScript/13Botones.html
- Manejo de eventos: buenas prácticas garantizando la separación de contenido y lógica https://web.dptoinformatica.uniovi.es/cueva/JavaScript/114-evento-botones.html

Tarea 3: Utilización estricta del paradigma de orientación a Objetos

En ECMAScript existen dos modelos de objetos:

- **Modelo basado en delegación** (también conocido como *modelo sin clases*): Este paradigma se fundamenta en el uso de prototipos, donde los objetos pueden heredar directamente de otros objetos mediante encadenamiento de prototipos. No requiere la definición explícita de clases, lo que permite una mayor flexibilidad en la composición y reutilización de comportamientos.
- **Modelo basado en clases:** Introducido formalmente en ECMAScript 2015 (ES6), este modelo proporciona una sintaxis más cercana a los lenguajes orientados a objetos tradicionales, como Java o C++. Permite definir clases, constructores y herencia mediante palabras clave específicas, facilitando la organización estructurada del código y la creación de jerarquías de objetos.

Utilizar estrictamente el modelo de objetos en ECMAScript permite una mejor **modularidad**, **reutilización de código y claridad estructural**, especialmente en proyectos complejos o colaborativos. Esto se produce beneficios en el **mantenimiento**, la **escalabilidad** y la **legibilidad** del software.

Los **métodos y atributos privados** en ECMAScript permiten encapsular la lógica interna de una clase, protegiendo su estado y comportamiento frente a accesos no autorizados o modificaciones externas. Esto mejora la **seguridad**, la **mantenibilidad** y la **robustez** del código.

En el proyecto MotoGP Desktop se tiene que revisar y refactorizar el código para que garantizar el uso estricto del paradigma de orientación a objetos, utilizando métodos y atributos privados.

Guía para resolver la tarea 3

Debes revisar la organización del código para decidir el uso estricto y más eficiente del paradigma de orientación a objetos.

Ejemplos:

- Modelo basado en delegación usando el objeto predefinido o nativo *document* https://web.dptoinformatica.uniovi.es/cueva/JavaScript/00Hola.html
- Modelo basado en clases creando la clase *Saludo* con un constructor y un método público https://web.dptoinformatica.uniovi.es/cueva/JavaScript/115-Saludo-OO.html
- Modelo basado en clases utilizando atributos y métodos privados https://web.dptoinformatica.uniovi.es/cueva/JavaScript/120-atributos-privados.html

Resultado del ejercicio 1

Una vez completado el ejercicio se debe haber refactorizado el código del proyecto MotoGP Desktop, por lo que podría haber cambios en varios documentos HTML, CSS y ECMAScript.

IMPORTANTE: Estos criterios de diseño se deben mantener en todo el proyecto MotoGP Desktop

Ejercicio 2: Consumo de servicios web de fotos con jQuery

En este ejercicio se realizará el consumo de un servicio web de imágenes usando la librería (Framework) jQuery. Se utilizará obligatoriamente el servicio web Flickr https://www.flickr.com/.

El servicio web debe devolver la información de las imágenes en formato JSON.

Las imágenes obtenidas serán relacionadas con MotoGP y con el circuito que fue asignado en las sesiones del módulo de Tecnologías de XML. La lista de circuitos puede consultarse en el anexo de este documento.

Las imágenes obtenidas en JSON se deberán mostrar en el documento index.html

Tarea 1. Consultar el API de Flickr y el formato JSON que devuelve

Consulta el funcionamiento de la API de Flickr a través del siguiente enlace https://www.flickr.com/services/api/.

Identifica el método que permite obtener las fotos de una localización concreta, en este caso del circuito de MotoGP asignado.

Guía para resolver la tarea 1

Consulta el formato de respuesta JSON de la API de Flickr.

Puedes consultar en el siguiente ejemplo cómo se obtienen las 20 últimas fotos cargadas en Flickr con la etiqueta “Oviedo”, en formato JSON:

https://web.dptoinformatica.uniovi.es/cueva/JavaScript/44jQueryJSON.html

La API tiene diversos métodos, y pulsando en los nombres, proporciona información sobre su funcionalidad y los parámetros que recibe, junto un ejemplo de respuesta y la lista de posibles códigos de error en la llamada.

Tarea 2. Enlazar la biblioteca (Framework) jQuery en index.html

Se debe incluir en el <head> del documento **index.html** la biblioteca (Framework) jQuery.

Guía para resolver la tarea 2

Consulta el apartado de jQuery del tema 4 de Computación en el Cliente titulado “Tecnologías y recursos relacionados con JavaScript” como enlazar la librería (Framework jQuery).

Utiliza el siguiente enlace para saber cómo es la referencia que debes incluir en el documento index.html para utilizar la versión mínima de jQuery Core 3.7.1:

https://releases.jquery.com/

Tarea 3. Creación del documento JavaScript

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

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

Tarea 4. Creación de la clase Carrusel

En el documento **carrusel.js** crea la clase **Carrusel**.

El constructor debe inicializar los siguientes atributos:

- **busqueda:** string para contener el término de búsqueda en la llamada a la API de Flickr
- **actual:** indicará el número de foto actual que se visualiza en el carrusel. Valor inicial 0.
- **maximo:** indicará el número de fotos del carrusel. Valor inicial 4.

Guía para resolver la tarea 4

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 5. Obtener las imágenes del circuito de MotoGP

En la clase **Carrusel** del archivo **carrusel.js**, hay que crear un método **getFotografias**() que:

- Realice una llamada AJAX a través de jQuery al servicio web de imágenes para obtener un objeto JSON con las imágenes que respondan al término de búsqueda.
- Las imágenes que devuelve el servicio web deben ser del mismo tamaño (640 en el borde más grande)
- El resultado de esta tarea es un objeto JSON

Guía para resolver la tarea 5

Consulta el ejemplo incluido a continuación para conocer la forma en la que debe realizarse una llamada AJAX a través de jQuery para consumir un servicio web y cómo extraer la información devuelta por el servicio en el formato JSON:

https://web.dptoinformatica.uniovi.es/cueva/JavaScript/86-jQuery-JSON-meteo.html

Debes utilizar el sufijo que permita obtener las imágenes de la API de Flickr de unas dimensiones concretas, consulta el siguiente enlace con los sufijos posible:

https://www.flickr.com/services/api/misc.urls.html

Tarea 6. Procesar la información del objeto JSON

En la clase **Carrusel** del archivo **carrusel.js**, hay que crear el método **procesarJSONFotografias**() que extraiga la información de 5 fotografías diferentes del objeto JSON.

Guía para resolver la tarea 6

Consultar el estándar JSON https://json.org/json-es.html

Para comprender la estructura clave:valor del objeto JSON se puede consultar: https://web.dptoinformatica.uniovi.es/cueva/JavaScript/46JSONusuarios.html

Para comprender cómo añadir métodos a un objeto JSON se puede consultar: https://web.dptoinformatica.uniovi.es/cueva/JavaScript/47JSONmetodosObjetos.html

Para procesa un objeto JSON se puede consultar el método verDatos(): https://web.dptoinformatica.uniovi.es/cueva/JavaScript/85-jQuery-JSON-meteo.html

Tarea 7. Añadir la información procesada del objeto JSON en el documento index.html

En la clase **Carrusel** del archivo **carrusel.js**, hay que crear el método **mostrarFotografias()** que muestre la primera de las 5 imágenes obtenidas del servicio web.

Guía para resolver la tarea 7

El método **mostrarFotografias()** debe incluir la imagen a mostrar dentro de un artículo con un encabezado de orden 2 con el texto "Imágenes del circuito de [NOMBRE DEL CIRCUITO]".

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

Tarea 8. Añadir el cambio de imagen con un temporizador

En la clase **Carrusel** del archivo **carrusel.js**, hay que crear el método **cambiarFotografia()** y que este modifique la fotografía que se muestra cada 3 segundos.

Guía para resolver la tarea 8

Dentro del método **mostrarFotografias()** utiliza el método **setInterval** de ECMAScript para programar la ejecución del método **cambiarFotografia()** en función del tiempo indicado. Recuerda hacer uso de la función **bind** de ECMAScript para que no se pierda el contexto de ejecución.

Utiliza las variables **actual** y **maximo** dentro del método **cambiarFotografia()** para ir avanzando por las 5 fotos obtenidas y cambiando la imagen que se ve en pantalla. Cuando se llegue a ver la quinta foto se volverá a empezar con la primera.

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

Tarea 9. Adaptabilidad del carrusel de imágenes

Se debe garantizar la adaptabilidad del carrusel de imágenes para dispositivos móviles

Guía para resolver la tarea 9

Dentro de la regla @media de la hoja de estilo **layout.css** añade el código CSS necesario para que el carrusel de imágenes se muestre correctamente en dispositivos móviles (por ejemplo, ocupando todo el ancho de la pantalla del dispositivo). Para ello se puede utilizar el selector **article h2 + img**

Resultado del ejercicio 2

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

- Fichero carrusel.js en el directorio js del proyecto

Además, se debe haber modificado el fichero index.html para incluir la referencia al fichero mencionado anteriormente y para incluir la llamada que permita crear el carrusel y ponerlo en funcionamiento.

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

Ejercicio 3: Consumo de servicios web de meteorología con jQuery

En este ejercicio se realizará el consumo de un servicio web de meteorología usando la librería (Framework) jQuery. Se utilizará obligatoriamente el servicio web Open-Meteo https://open-meteo.com/.

El servicio web debe devolver la información meteorológica en formato JSON.

La información meteorológica será de la zona del circuito de MotoGP en las fechas de la competición (entrenamientos, clasificación y carrera).

La información obtenida en JSON se deberá mostrar en el documento meteorología.html

Tarea 1. Consultar el API de Open-Meteo y el formato JSON que devuelve

Consulta el funcionamiento de la API de datos históricos del tiempo que ofrece Open-Meteo a través del siguiente enlace: https://open-meteo.com/en/docs/historical-weather-api

Identifica los datos que se devuelven en fracciones horarias y los datos que se devuelven para un día completo.

Guía para resolver la tarea 1

Puedes utilizar el interfaz disponible en el enlace https://open-meteo.com/en/docs/historical-weather-api para realizar una llamada de muestra a la API, marcando los elementos del tiempo que deseas que se incluyan en la respuesta e indicando las coordenadas para las que se consulta la información meteorológica. Además, en la parte inferior del documento enlazado verás un ejemplo de respuesta.

Tarea 2. Enlazar la biblioteca (Framework) jQuery en meteorologia.html

Se debe incluir en el <head> del documento **meteorologia.html** la biblioteca (Framework) jQuery.

Guía para resolver la tarea 2

Consulta el apartado de jQuery del tema 4 de Computación en el Cliente titulado “Tecnologías y recursos relacionados con JavaScript” como enlazar la librería (Framework jQuery).

Utiliza el siguiente enlace para saber cómo es la referencia que debes incluir en el documento meteorologia.html para utilizar la versión mínima de jQuery Core 3.7.1:

https://releases.jquery.com/

Tarea 3. Obtener los datos meteorológicos del circuito de MotoGP el día de la carrera

En la clase **Ciudad** del archivo **ciudad.js**, hay que crear un método **getMeteorologiaCarrera**() que:

- Realice una llamada AJAX a través de jQuery al servicio web de meteorología para obtener un objeto JSON con la información meteorológica en el **circuito** de las franjas horarias del día de la **carrera**.
- La información meteorológica mínima que se debe obtener es: temperatura a 2 metros del suelo, sensación térmica, lluvia, humedad relativa a 2 metros del suelo, velocidad del viento a 10 metros del suelo, dirección del viento a 10 metros del suelo, hora de salida del sol y hora de puesta del sol.
- Se deben identificar los datos que son devueltos en días completos y los datos en fracciones horarias.
- El resultado de esta tarea es un objeto JSON

Guía para resolver la tarea 3

Consulta el ejemplo incluido a continuación para conocer la forma en la que debe realizarse una llamada AJAX a través de jQuery para consumir un servicio web y cómo extraer la información devuelta por el servicio en el formato JSON:

https://web.dptoinformatica.uniovi.es/cueva/JavaScript/86-jQuery-JSON-meteo.html

Tarea 4. Procesar la información del objeto JSON

En la clase **Ciudad** del archivo **ciudad.js**, hay que crear el método **procesarJSONCarrera**() que extraiga del objeto JSON la información.

Guía para resolver la tarea 4

Consultar el estándar JSON https://json.org/json-es.html

Para comprender la estructura clave:valor del objeto JSON se puede consultar: https://web.dptoinformatica.uniovi.es/cueva/JavaScript/46JSONusuarios.html

Para comprender cómo añadir métodos a un objeto JSON se puede consultar: https://web.dptoinformatica.uniovi.es/cueva/JavaScript/47JSONmetodosObjetos.html

Para procesa un objeto JSON se puede consultar el método verDatos(): https://web.dptoinformatica.uniovi.es/cueva/JavaScript/85-jQuery-JSON-meteo.html

Tarea 5. Añadir la información procesada del objeto JSON en el documento meteorologia.html

En el documento **meteorologia.html** se debe añadir elemento HTML con la información procesada obtenida del objeto JSON.

Guía para resolver la tarea 5

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

Tarea 6. Obtener los datos meteorológicos del circuito de MotoGP los días de entrenamientos

En la clase **Ciudad** del archivo **ciudad.js**, hay que crear un método **getMeteorologiaEntrenos**() que:

- Realice una llamada AJAX a través de jQuery al servicio web de meteorología para obtener un objeto JSON con la información meteorológica en el **circuito** para las franjas horarias de los **3 días de entrenamientos previos a la carrera**.
- La información meteorológica mínima que se debe obtener es: temperatura a 2 metros del suelo, lluvia, velocidad del viento a 10 metros del suelo y humedad relativa a 2 metros del suelo.
- El resultado de esta tarea es un objeto JSON

Guía para resolver la tarea 6

Consulta el ejemplo incluido a continuación para conocer la forma en la que debe realizarse una llamada AJAX a través de jQuery para consumir un servicio web y cómo extraer la información devuelta por el servicio en el formato JSON:

https://web.dptoinformatica.uniovi.es/cueva/JavaScript/86-jQuery-JSON-meteo.html

Tarea 7. Procesar la información del objeto JSON

En la clase **Ciudad** del archivo **ciudad.js**, hay que crear el método **procesarJSONEntrenos**() que extraiga del objeto JSON la información.

Una vez extraída la información, se debe calcular la media de cada dato obtenido para cada uno de los 3 días de entrenamientos. Las medias obtenidas deben ajustarse a dos decimales.

Guía para resolver la tarea 7

Consultar el estándar JSON https://json.org/json-es.html

Para comprender la estructura clave:valor del objeto JSON se puede consultar: https://web.dptoinformatica.uniovi.es/cueva/JavaScript/46JSONusuarios.html

Para comprender cómo añadir métodos a un objeto JSON se puede consultar: https://web.dptoinformatica.uniovi.es/cueva/JavaScript/47JSONmetodosObjetos.html

Para procesa un objeto JSON se puede consultar el método verDatos(): https://web.dptoinformatica.uniovi.es/cueva/JavaScript/85-jQuery-JSON-meteo.html

Tarea 8. Añadir la información procesada del objeto JSON en el documento meteorologia.html

En el documento **meteorologia.html** se deben añadir elementos HTML con la información procesada (media de los datos del tiempo) obtenida del objeto JSON.

Guía para resolver la tarea 8

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 3

Una vez completado el ejercicio deberían haberse modificado los siguientes ficheros: el fichero ciudad.js para incluir el código de las llamadas al servicio de meteorología y el fichero meteorologia.html para incluir en él la información meteorológica.

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

Ejercicio 4: Consumo de servicios web de noticias usando fetch()

En este ejercicio se realizará el consumo de un servicio web de noticias usando fetch(). Se utilizará obligatoriamente el servicio web de noticias TheNewsApi: https://www.thenewsapi.com/

El servicio web debe devolver la información de las noticias en formato JSON.

Las noticias estarán relacionadas con el campeonato del mundo de MotoGP.

La información obtenida en JSON se deberá mostrar en el documento **index.html**, debajo del carrusel de imágenes.

Tarea 1. Consultar el API de TheNewsApi y el formato JSON que devuelve

Consulta el funcionamiento de la API de noticias que ofrece TheNewsApi a través del siguiente enlace: https://www.thenewsapi.com/documentation

La API TheNewsApi requiere del uso de una api_key para realizar las peticiones.

Identifica el método que permita obtener las noticias en base a un término de búsqueda proporcionado.

Guía para resolver la tarea 1

Crea una cuenta gratuita en TheNewsApi y obtén una api_key propia para realizar la consulta.

> **NOTA:** Es recomendable utilizar una cuenta de correo diferente de la de estudiante (UOXXXXXX@uniovi.es) para evitar problemas con el correo electrónico de activación que envía la plataforma.

Puedes consultar la información referente a cada método y los parámetros que recibe pulsando en los nombres de cada uno de ellos. Además, en la parte derecha del documento enlazado verás un ejemplo de respuesta a la llamada.

Tarea 2. Creación del documento JavaScript

Dentro de la carpeta js del directorio del proyecto MotoGP Desktop crea un nuevo fichero llamado **noticias.js**.

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

Tarea 3. Creación de la clase Noticias

En el documento **noticias.js** crea la clase Noticias.

El constructor debe inicializar los siguientes atributos:

- **busqueda:** string para contener el término de búsqueda en la llamada a la API TheNewsAPI
- **url:** string con la parte fija de la URL de la API de TheNewsApi.

Tarea 4. Obtener las noticias sobre MotoGP

En la clase **Noticias** del archivo **noticias.js**, hay que crear un método **buscar**() que:

- Realice una llamada al servicio web de noticias utilizando fetch()
- El resultado de esta tarea es un objeto Promise (promesa) del que se obtiene la información en formato JSON

Guía para resolver la tarea 3

Consulta el ejemplo incluido a continuación para conocer la forma en la que debe realizarse una llamada utilizando fetch() para consumir un servicio web y cómo extraer la información devuelta por el servicio en el formato JSON utilizando el objeto Promise (promesa):

https://web.dptoinformatica.uniovi.es/cueva/JavaScript/126-fetch-clase-climaAPI.html

Tarea 4. Procesar la información del objeto JSON

En la clase **Noticias** del archivo **noticias.js**, hay que crear el método **procesarInformacion**() que extraiga del objeto JSON las noticias obtenidas.

Guía para resolver la tarea 4

Consultar el estándar JSON https://json.org/json-es.html

Para comprender la estructura clave:valor del objeto JSON se puede consultar: https://web.dptoinformatica.uniovi.es/cueva/JavaScript/46JSONusuarios.html

Para comprender cómo añadir métodos a un objeto JSON se puede consultar: https://web.dptoinformatica.uniovi.es/cueva/JavaScript/47JSONmetodosObjetos.html

Para procesa un objeto JSON se puede consultar el método verDatos(): https://web.dptoinformatica.uniovi.es/cueva/JavaScript/85-jQuery-JSON-meteo.html

Tarea 5. Añadir la información procesada del objeto JSON en el documento index.html

En el documento index.html se deben añadir elementos HTML con la información de las noticias obtenida del objeto JSON.

Guía para resolver la tarea 5

Se debe crear una sección debajo del carrusel de imágenes para meter en ella todas las noticias consultadas. Cada noticia debe tener al menos la siguiente información: titular, entradilla, enlace y fuente de la noticia.

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 4

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

- Fichero noticias.js en el directorio js del proyecto

Además, se debe haber modificado el fichero index.html para incluir en él la información de las noticias.

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

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 (Hungría)
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