Grado en Ingeniería Informática del Software

Software y estándares para la Web

P3. COMPUTACIÓN EN EL CLIENTE WEB

Escuela de Ingeniería Informática
Universidad de Oviedo

Contenido

RECUERDA: Revisar las normas y la forma de entrega de la práctica

Objetivos

En esta práctica el objetivo es hacer computación en el cliente Web usando el estándar ECMAScript.

Esta práctica se corresponde con los temas de teoría:

- Computación Web
- Lenguajes de Script
- El lenguaje JavaScript
- Tecnologías y recursos relacionados con JavaScript

Normas

La práctica entregada debe cumplir estrictamente las siguientes normas, **en caso contrario la calificación será cero puntos**:

- Se deben entregar todos los ejercicios y todas las tareas del guion de la práctica.
- Se debe utilizar ECMAScript "puro" sin bibliotecas ni extensiones fuera del estándar ECMAScript, excepto en los ejercicios que especifiquen el uso de una determinada biblioteca o API.
- Se usará el paradigma de orientación a objetos obligatoriamente, con clases o sin clases (basado en prototipos), para todos los ejercicios.
    - No se permitirá código no orientado a objetos (por ejemplo: funciones libres no ligadas a objetos o clases, paradigma procedimental).
    - En el caso de usar el paradigma funcional, por ejemplo con jQuery, debe encapsularse en los métodos de los objetos.
- Los archivos HTML, CSS y ECMAScript deben estar siempre separados y no incrustados en el archivo HTML.
    - El objetivo es tener en archivos separados el contenido, la presentación y la computación.
- TODOS los documentos HTML deben ser HTML5 válidos utilizando el validador de lenguajes de marcado del W3C
- TODAS las hojas de estilo que se utilizan en el sitio web deben ser validas utilizando el validador CSS del W3C
- TODOS los ejercicios deben funcionar en todos los navegadores de referencia, Chrome, Firefox, Microsoft Edge y Opera. Opcionalmente para sistemas MacOS el navegador Safari.
- No se permite hacer computación en el cliente usando las etiquetas HTML <object> y <embed>

Entrega de la práctica

Se deben entregar **todo en un archivo** comprimido denominado "UOXXXXXX-practica3.zip" (tamaño máximo 40MB) donde el UOXXXXXX será el identificador del estudiante en la Universidad de Oviedo

El archivo comprimido debe contener una carpeta “PRACTICA-3” con TODOS los ejercicios de la práctica, una carpeta por ejercicio:

- Ejercicio01
- Ejercicio02
- …
- Ejercicio15

Cada ejercicio puede dividirse en tareas, que se presentará en carpetas separadas: tarea1, tarea2, etc. Según lo especificado en el “Epílogo” de cada ejercicio.

Se considerarán como “no entregadas” las prácticas que no cumplan el formato de entrega, el nombrado de los archivos y que no contengan todos los ejercicios de la práctica.

El plazo de entrega se encuentra publicado en el Campus Virtual y la tarea de entrega se habilitarán con anterioridad suficiente.

Ejercicio 1

Tarea 1. Escribir un archivo en HTML5 denominado **Ejercicio1** (con extensión **.html**) que referencie a un archivo CSS denominado **Ejercicio1** (con extensión **.css**) y también debe referenciar a varios archivos en ECMAScript (extensión **.js**). El archivo en HTML5 deberá contener:

- Incluir en cabeza (head) un archivo ECMAScript denominado **Cabecera.js**
- En el cuerpo (body) incluir un archivo ECMAScript denominado **Titulo1.js**
- En el cuerpo (body) incluir un archivo ECMAScript denominado **Titulo2.js**
- En el cuerpo (body) incluir un archivo ECMAScript denominado **Titulo3.js**
- En el cuerpo (body) incluir un archivo ECMAScript denominado **Titulo4.js**
- En el cuerpo (body) incluir un archivo ECMAScript denominado **Parrafos.js**

Tarea 2. Escribir un archivo CSS para el HTML de la tarea anterior que especifique el estilo de h1, h2, h3, h4, p y el fondo. Se deja libre al estudiante elegir el diseño de la hoja de estilo. El archivo se denominará **Ejercicio1.css**

Tarea 3. Escribir un archivo ECMAScript denominado **Cabecera.js** que contiene un objeto con información del nombre de la asignatura, nombre de la titulación, nombre del centro donde se imparte, nombre de la Universidad, curso actual, nombre del estudiante y e-mail.

Tarea 4. Escribir un archivo ECMAScript denominado **Titulo1.js** que llame a un método que escriba en el objeto **document** el nombre de la asignatura en un encabezado de nivel 1 (h1).

Tarea 5. Escribir un archivo ECMAScript denominado **Titulo2.js** que llame a un método que escriba en el objeto **document** el nombre de la titulación en un encabezado de nivel 2 (h2).

Tarea 6. Escribir un archivo ECMAScript denominado **Titulo3.js** que llame a un método que escriba en el objeto **document** el nombre del centro donde se imparte la titulación en un encabezado de nivel 3 (h3).

Tarea 7. Escribir un archivo ECMAScript denominado **Titulo4.js** que llame a un método que escriba en el objeto **document** el nombre de la universidad donde se imparte la titulación en un encabezado de nivel 4 (h4).

Tarea 8. Escribir un archivo ECMAScript denominado **Parrafos.js** que llame a un método que escriba en el objeto **document** el resto de información disponible en **Cabecera.js** en párrafos (p).

Epílogo. Todas las tareas se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-01** que debe contener los archivos:

- Ejercicio1.html
- Ejercicio1.css
- Cabecera.js
- Titulo1.js
- Titulo2.js
- Titulo3.js
- Titulo4.js
- Parrafos.js

Guía para resolver el ejercicio 1

Ejecutar el ejercicio resuelto:

http://di002.edv.uniovi.es/~cueva/JavaScript/73-Asignatura.html

Abrir el código fuente y leerlo detenidamente. La organización en archivos separados (.html, .css y .js) es la estructura que se solicita para todos los ejercicios en esta práctica.

Debe observarse: la inferencia de tipos, las distintas formas de crear y usar objetos y el objeto predefinido **document**.

También debe observarse que aunque el código JavaScript está separado en varios archivos. Una vez cargados en memoria por el navegador (agente de usuario), desde cualquier archivo .js se puede acceder a objetos creados en otro archivo .js previamente.

Ejercicio 2

Tarea 1. Escribir un archivo en HTML5 denominado **Ejercicio2** (con extensión **.html**) que referencie a un archivo CSS denominado **Ejercicio2** (con extensión **.css**) y también debe referenciar a varios archivos en ECMAScript (extensión **.js**). El archivo en HTML5 deberá contener:

- Incluir en cabeza (head) un archivo ECMAScript denominado **InfoNavegador.js**
- Un encabezado de nivel 1 (h1), con llamada en su interior a un archivo ECMAScript denominado **NombreNavegador.js**
- Un encabezado de nivel 2 (h2), con llamada en su interior a un archivo ECMAScript denominado **IdiomaNavegador.js**
- Un párrafo (p), con llamada en su interior a un archivo ECMAScript denominado **MasInfoNavegador.js**

Tarea 2. Escribir un archivo CSS para el HTML de la tarea anterior que especifique el estilo de h1, h2, p y el fondo. Se deja libre al estudiante elegir el diseño de la hoja de estilo. El archivo se denominará **Ejercicio2.css**

Tarea 3. Escribir un archivo ECMAScript denominado **InfoNavegador.js** que contiene un objeto con información del navegador que utiliza el cliente Web.

Tarea 4. Escribir un archivo ECMAScript denominado **NombreNavegador.js** que llame a un método que escriba en el objeto **document** el nombre del navegador.

Tarea 5. Escribir un archivo ECMAScript denominado **IdiomaNavegador.js** que llame a un método que escriba en el objeto **document** el idioma del navegador.

Tarea 6. Escribir un archivo ECMAScript denominado **MasInfoNavegador.js** que llame a un método que escriba en el objeto **document** el resto de información disponible del navegador.

Epílogo. Todas las tareas se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-02** que debe contener los archivos:

- Ejercicio2.html
- Ejercicio2.css
- InfoNavegador.js
- NombreNavegador.js
- IdiomaNavegador.js
- MasInfoNavegador.js

Guía para resolver el ejercicio 2

Ejecutar el ejercicio resuelto:

http://di002.edv.uniovi.es/~cueva/JavaScript/74-InfoNavegador.html

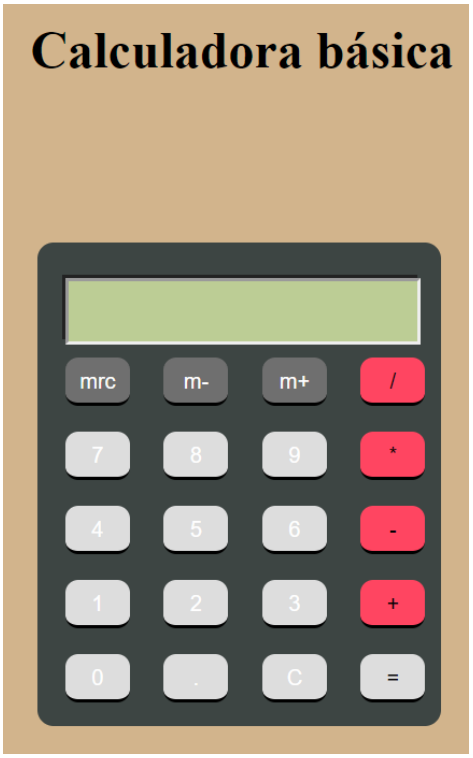
Abrir el código fuente y leerlo detenidamente. La organización en archivos separados (.html, .css y .js) es la estructura que se solicita para todos los ejercicios en esta práctica.

Debe observarse: la inferencia de tipos, las distintas formas de crear y usar objetos y el objeto predefinido **navigator**.

Probar con distintos navegadores y buscar los motivos de la información mostrada para cada tipo de navegador.

Ejercicio 3

Tarea 1. Escribir un archivo en HTML5 denominado **CalculadoraBasica** (con extensión **.html**) que referencie a un archivo CSS denominado **CalculadoraBasica** (con extensión **.css**) y también debe referenciar a un archivo en ECMAScript denominado **CalculadoraBasica** (extensión .**js**). El archivo en HTML5 deberá usar la metáfora de una calculadora para contener la interfaz web de una calculadora básica. La interfaz puede ser de la forma siguiente:



Tarea 2. Escribir un archivo CSS denominado **CalculadoraBasica** (con extensión **.css**) con la hoja de estilo de la interfaz web de la calculadora básica.

Tarea 3. Escribir un archivo ECMAScript denominado **CalculadoraBasica** (con extensión **.js**) con una clase denominada Calculadora con los atributos y métodos necesarios para realizar las operaciones de la calculadora.

Epílogo. Todas las tareas se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-03** que debe contener los archivos:

- CalculadoraBasica.html
- CalculadoraBasica.css
- CalculadoraBasica.js

Guía para resolver el ejercicio 3

El objetivo es crear objetos a partir de clases. Para ver como se crea una clase en ECMAScript puede consultarse el ejercicio:

http://di002.edv.uniovi.es/~cueva/JavaScript/68-ES6-Ejemplo-uso-de-clases.html

Crear una clase Calculadora. El constructor puede no tener parámetros. Escribir los métodos: dígitos, punto, suma, resta, multiplicación, mrc, mMenos, mMas, borrar e igual.

Definir el atributo pantalla, que es un String donde se acumulan las teclas pulsadas.

En el método igual se puede llamar a la función eval(). Pueden consultarse los ejercicios:

http://di002.edv.uniovi.es/~cueva/JavaScript/48eval-calculadora.html

http://di002.edv.uniovi.es/~cueva/JavaScript/66-Try-Catch.html

Observar el manejo del evento onClick al pulsar un botón.

Ejercicio 4

Tarea 1. Escribir un archivo en HTML5 denominado **CalculadoraCientifica** (con extensión **.html**) que referencie a un archivo CSS denominado **CalculadoraCientifica** (con extensión **.css**) y también referencia a un archivo en ECMAScript denominado **CalculadoraCientifica** (extensión **.js**).

Debe utilizarse la herencia de la clase calculadora básica del ejercicio anterior.

El archivo en HTML5 deberá usar la metáfora de una calculadora para contener la interfaz web de una calculadora científica. La interfaz puede emular a la calculadora científica de Windows 10 (aunque no es obligatorio):

Tarea 2. Escribir un archivo CSS denominado **CalculadoraCientifica** (con extensión **.css**) con la hoja de estilo de la interfaz web de la calculadora científica.

Tarea 3. Escribir un archivo ECMAScript denominado **CalculadoraCientifica** (con extensión **.js**) con una clase denominada Calculadora con los atributos y métodos necesarios para realizar las operaciones de la calculadora.

Epílogo. Todas las tareas se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-04** que debe contener los archivos:

- CalculadoraCientifica.html
- CalculadoraCientifica.css

- CalculadoraCientifica.js

Guía para resolver el ejercicio 4

Para el manejo de funciones matemáticas puede consultarse el ejercicio

http://di002.edv.uniovi.es/~cueva/JavaScript/18Math.html

Se puede utilizar la herencia de clases. Se puede crear una clase CalculadoraCientífica que hereda de la clase Calculadora.

Sobre el uso de la herencia consultar el ejercicio:

http://di002.edv.uniovi.es/~cueva/JavaScript/70-ES6-Ejemplo-herencia-de-clases-con-extends.html

Ejercicio 5

Tarea 1. Escribir un archivo en HTML5 denominado **CalculadoraRPN** (con extensión **.html**) que referencie a un archivo CSS denominado **CalculadoraRPN** (con extensión **.css**) y también referencia a un archivo en ECMAScript denominado **CalculadoraRPN** (extensión .**js**).

Las calculadoras RPN utilizan la notación postfija o notación polaca inversa (reverse polish notation). Evalúan las expresiones utilizando una pila. No tienen el botón igual.

La calculadora debe tener además de las funciones básicas (+,-,*,/), funciones científicas (sin, cos, tan,…). Se deja libre al estudiante el diseño de las funciones incorporadas.

Epílogo. Todas las tareas se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-05** que debe contener los archivos:

- CalculadoraRPN.html
- Calculadora.css
- Calculadora.js

Guía para resolver el ejercicio 5

En este ejercicio **no se debe usar eval()**, que puede dar problemas de seguridad

La notación postfija o RPN se puede estudiar en

https://es.wikipedia.org/wiki/Notaci%C3%B3n_polaca_inversa

Para implementar la calculadora es necesario usar una estructura de datos Pila.

Puede verse una implementación de una pila en el ejercicio:

http://di002.edv.uniovi.es/~cueva/JavaScript/98-EC6-Ejemplo-clase-Pila.html

Ejercicio 6

Tarea 1. Realización de calculadoras especializadas en función de la terminación del UOXXXXXX de cada estudiante.

- UOXXXXXX terminados en 0: Calculadora financiera. Por ejemplo dado un préstamo, un tipo de interés y un tiempo. Determina los pagos mensuales.
- UOXXXXXX terminados en 1: Calculadora de fechas. Por ejemplo dado un año determina la fecha del martes de carnaval, del domingo de Pascua, del jueves de la ascensión, del martes de campo (fiesta en Oviedo), etc.
- UOXXXXXX terminados en 2: Calculadora de coordenadas. Dadas unas coordenadas geográficas determina distancia entre dos puntos. También puede transformar coordenadas entre distintos sistemas por ejemplo coordenadas geográficas a UTM (Universal Transversal Mercator).
- UOXXXXXX terminados en 3: Calculadora matemática avanzada. Por ejemplo puede calcular limites, derivadas, integrales, etc.
- UOXXXXXX terminados en 4: Calculadora de cambio de unidades. Por ejemplo puede cambiar de unidades entre el sistema métrico y el imperial. Puede trabajar con unidades de longitud, superficie, volumen, potencia, energía, etc.
- UOXXXXXX terminados en 5: Calculadora dietética. Por ejemplo determina calorías de los alimentos y el gasto de calorías por realización de distintas actividades físicas. Maneja el índice de masa corporal.
- UOXXXXXX terminados en 6: Calculadora de cambio de base. Por ejemplo dado un número puede obtener su representación en base 2 (binario), base 8 (octal), base 10, base 16 (hexadecimal), etc.
- UOXXXXXX terminados en 7: Calculadora estadística. Por ejemplo a partir de un conjunto de valores determina la media, la desviación típica, el coeficiente de correlación, etc.
- UOXXXXXX terminados en 8: Calculadora energética. Por ejemplo realiza cálculos de consumo de energía y costes en una vivienda, en un edificio o en una fábrica. También puede tener en cuenta las perdidas de energía por mal aislamiento. También puede tener en cuenta las condiciones ambientales como la temperatura exterior.
- UOXXXXXX terminados en 9: Calculadora de consumos de agua. Por ejemplo realiza cálculos de consumo de agua y costes en una vivienda, en un edificio o en una fábrica. Puede tener en cuenta el número de personas y sus consumos diarios. También puede tener en cuenta las perdidas de agua por fugas.

Se utilizará HTML5, CSS y ECMAScript.

Se valorará la presentación, la complejidad de la aplicación, la originalidad, la creatividad y los elementos usados de ECMAScript.

En este ejercicio **no** se pueden utilizarse bibliotecas como *jQuery* u otras. Debe usarse ECMAScript "puro".

Epílogo. Todos los archivos se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-06**

Guía para resolver el ejercicio 6

El estudiante debe documentarse sobre la temática de la aplicación antes de empezar a diseñarla. Se aconseja utilizar fuentes fiables y contrastadas.

Ejercicio 7

Tarea 1. Escribir un archivo en HTML5 denominado **Ejercicio7** (con extensión **.html**) que referencie a un archivo CSS denominado **Ejercicio7** (con extensión **.css**) y también debe referenciar a un archivo en JavaScript (extensión **.js**). El HTML debe contener elementos h1, h2, h3, p , tablas y otros de libre diseño por el estudiante.

Se debe escribir un código JavaScript usando **jQuery** que permita:

- Ocultar y mostrar algunos de los elementos del HTML

Tarea 2. Se debe ampliar el código JavaScript usando **jQuery** para que permita:

- Modificar algunos de los elementos HTML

Tarea 3. Se debe ampliar el código JavaScript usando **jQuery** para que permita:

- Añadir nuevos elementos HTML

Tarea 4. Se debe ampliar el código JavaScript usando **jQuery** para que permita:

- Eliminar algunos de los elementos HTML

Tarea 5. Se debe ampliar el código JavaScript usando **jQuery** para que permita:

- Recorrer todos los elementos HTML y mostrar de cada uno de ellos: quien es su elemento padre y que tipo de elemento es.

Tarea 6. Se debe ampliar el código JavaScript usando **jQuery** para que permita:

- Sumar las filas y columnas de la tabla

Epílogo. Todas las tareas se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-07** que debe contener los archivos:

- Ejercicio7.html
- Ejercicio7.css
- Ejercicio7.js

Guía para resolver el ejercicio 7

Se recomienda ejecutar y comprender los ejercicios:

http://di002.edv.uniovi.es/~cueva/JavaScript/26jQueryOcultaParrafos.html

http://di002.edv.uniovi.es/~cueva/JavaScript/27jQueryOcultaMuestraParrafos.html

http://di002.edv.uniovi.es/~cueva/JavaScript/76-jQuery-DOM-get-val.html

http://di002.edv.uniovi.es/~cueva/JavaScript/77-jQuery-DOM-get-text.html

http://di002.edv.uniovi.es/~cueva/JavaScript/78-jQuery-DOM-get-html.html

http://di002.edv.uniovi.es/~cueva/JavaScript/79-jQuery-DOM-get-attr.html

http://di002.edv.uniovi.es/~cueva/JavaScript/80-jQuery-DOM-set-val-text-html.html

http://di002.edv.uniovi.es/~cueva/JavaScript/81-jQuery-DOM-set-attr.html

http://di002.edv.uniovi.es/~cueva/JavaScript/82-jQuery-DOM-add-elementos.html

http://di002.edv.uniovi.es/~cueva/JavaScript/83-jQuery-%20recorrer-DOM.html

http://di002.edv.uniovi.es/~cueva/JavaScript/71-jQuery-Ocultar-filas-tabla.html

Ejemplo de cómo encapsular jQuery en una clase

http://di002.edv.uniovi.es/~cueva/JavaScript/84-jQuery-clase-Bombilla.html

Ejercicio 8

Tarea 1.

Escribir un archivo en HTML5 denominado **Ejercicio8** (con extensión **.html**) que referencie a un archivo CSS denominado **Ejercicio8** (con extensión **.css**) y también debe referenciar a un archivo en JavaScript (extensión .**js**).

Se debe escribir una aplicación en **jQuery** que **consuma servicios Web** en formato **JSON** con datos meteorológicos de **tres ciudades o pueblos** de la ruta turística de la práctica de XML Pueden ser sitios próximos a la ruta, no es obligatorio que la ruta pase por ellos.

Deben consumirse servicios Web de meteorología. Se deja libre al estudiante la elección del proveedor.

El diseño de la presentación se deja libre al estudiante, pero debe contener la máxima información meteorológica posible y con un icono que indique la situación del tiempo atmosférico. El icono se obtiene automáticamente del proveedor de servicios web (no se tiene que diseñar el icono)

Epílogo. Todas las tareas se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-08** que debe contener los archivos:

- Ejercicio8.html
- Ejercicio8.css
- Ejercicio8.js

Guía para resolver el ejercicio 8

Consultar los ejercicios
http://di002.edv.uniovi.es/~cueva/JavaScript/85-jQuery-JSON-meteo.html

http://di002.edv.uniovi.es/~cueva/JavaScript/86-jQuery-JSON-meteo.html

Ejercicio 9

Tarea 1. Escribir un archivo en HTML5 denominado **Ejercicio9** (con extensión **.html**) que referencie a un archivo CSS denominado **Ejercicio9** (con extensión **.css**) y también debe referenciar a un archivo en JavaScript (extensión **.js**).

Se debe escribir una aplicación en **jQuery** que **consuma servicios Web** en formato **XML** con datos meteorológicos de **cinco ciudades o pueblos** de la ruta turística de la práctica de XML Pueden ser sitios próximos a la ruta, no es obligatorio que la ruta pase por ellos.

Deben consumirse servicios Web de meteorología. Se deja libre al estudiante la elección del proveedor.

El diseño de la presentación se deja libre al estudiante, pero debe contener la máxima información meteorológica posible y con un icono que indique la situación del tiempo atmosférico. El icono se obtiene automáticamente del proveedor de servicios web (no se tiene que diseñar el icono)

Epílogo. Todas las tareas se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-09** que debe contener los archivos:

- Ejercicio9.html
- Ejercicio9.css
- Ejercicio9.js

Guía para resolver el ejercicio 9

Consultar el ejercicio:

http://di002.edv.uniovi.es/~cueva/JavaScript/87-jQuery-AJAX-XML-meteo.html

Ejercicio 10

Tarea 1. Realización de una aplicación web que consuma un tipo servicios Web en función del de la terminación del UOXXXXXX de cada estudiante.

- UOXXXXXX terminados en 0 o en 5: Consumo de servicios Web de noticias
- UOXXXXXX terminados en 1 o en 6: Consumo de servicios Web de traducción
- UOXXXXXX terminados en 2 o en 7: Consumo de servicios Web de cotizaciones de bolsa
- UOXXXXXX terminados en 3 o en 8: Consumo de servicios Web de cambio de moneda entre el Euro y un mínimo de cinco monedas
- UOXXXXXX terminados en 4 o en 9: Consumo de servicios Web de imágenes y fotografías

Se utilizará HTML5, CSS y JavaScript. La aplicación web debe usar **jQuery** y **consumo de servicios Web** obligatoriamente.

Se valorará la presentación, la complejidad de la aplicación, la originalidad, la creatividad y los elementos usados de **jQuery** y de **consumo de servicios Web**.

Epílogo. Todas las tareas se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio10** que debe contener los archivos:

- Ejercicio10.html
- Ejercicio10.css
- Ejercicio10.js

Guía para resolver el ejercicio 10

El estudiante deberá buscar un proveedor de servicios Web de la temática que le ha tocado. Es importante elegir un proveedor con una buena documentación de la API y si es posible con buenos ejemplos.

Ejemplos:

http://di002.edv.uniovi.es/~cueva/JavaScript/44jQueryJSON.html

https://fixer.io/

https://currencylayer.com/

https://yandex.com/dev/translate/

https://www.quandl.com/docs-and-help

https://newsapi.org/

Ejercicio 11

Tarea 1. Utilizando el API de HTML5 denominado ***Geolocalization***, escribir una clase denominada ***GeoLocalizacion*** que nos informe de la posición del usuario.

Para utilizar la geolocalización del usuario:

- Debe usarse el protocolo https.
- Los servidores desde donde se ejecuta la geolocalización pueden dar avisos en los distintos navegadores de que sus certificados no están validados. Debe hacerse una excepción y permitir cargar el sitio Web
- El usuario debe aceptar que sea geo-localizado, debido a que su situación es un dato de carácter personal.

Tarea 2. Se debe ampliar el código JavaScript de la clase ***GeoLocalizacion*** para que permita:

- Manejo de errores de geo-localización.

Tarea 3. Se debe ampliar el código JavaScript de la clase ***GeoLocalizacion*** para que permita:

- Mostrar un mapa estático en Google Maps con un marcador con la posición del usuario.

Tarea 4. Se debe crear un objeto en JavaScript usando **el API de Google Maps** para que permita:

- Crear un mapa dinámico con un determinado centro. El centro debe señalarse con un marcador

Tarea 5. Se debe crear un objeto en JavaScript usando el **API de Google Maps** para que permita:

- Crear un mapa dinámico con Google Maps indicando la posición del usuario

Tarea 6. Se debe crear una aplicación de temática libre en JavaScript usando **el API de Google Maps.**

- Puede combinarse con el uso de otras APIs.

Epílogo. Todas las tareas se presentarán en distintas sub-carpetas de **PRACTICA-3** denominadas **Tarea-1**, **Tarea-2**, etc. Dentro de la carpeta denominada **Ejercicio-11.**

Guía para resolver el ejercicio 11

Consultar los ejercicios:

https://di002.edv.uniovi.es/~cueva/JavaScript/104-ClaseGeolocalizacion.html

https://di002.edv.uniovi.es/~cueva/JavaScript/105-ClaseGeolocalizacionManejoErrores.html

https://di002.edv.uniovi.es/~cueva/JavaScript/106-ClaseMapaEstaticoGoogle.html

https://di002.edv.uniovi.es/~cueva/JavaScript/107-objetoMapaDinamicoGoogle.html

https://di002.edv.uniovi.es/~cueva/JavaScript/108-GeolocalizacionMapaDinamicoGoogle.html

Ejercicio 12

Tarea 1.

Escribir un archivo en HTML5 denominado **Ejercicio12** (con extensión **.html**) que referencie a un archivo CSS denominado **Ejercicio12** (con extensión **.css**) y también debe referenciar a un archivo en JavaScript (extensión .**js**).

Se debe escribir una aplicación que use el **API FILE de HTML5** que **cargue archivos desde la máquina cliente y que visualice sus propiedades (tamaño, tipo, etc.).**

En el caso de que se seleccionen archivos de tipo:

- **Texto**
- **JSON**
- **XML**

Deben mostrarse íntegramente por pantalla.

El diseño de la presentación se deja libre al estudiante.

Epílogo. Todas las tareas se presentan en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-12** que debe contener los archivos:

- Ejercicio12.html
- Ejercicio12.css
- Ejercicio12.js
- Ejemplo.txt
- Ejemplo.json
- Ejemplo.xml

Guía para resolver el ejercicio 12

Consultar los ejercicios:

http://di002.edv.uniovi.es/~cueva/JavaScript/62API-FILE-LeerArchivoTexto.html

http://di002.edv.uniovi.es/~cueva/JavaScript/63API-FILE-ArchivosMultiples.html

Ejercicio 13

Tarea 1. Escribir un archivo en HTML5 denominado **MapaKML** (con extensión **.html**) que referencie a un archivo CSS denominado **MapaKML** (con extensión **.css**) y también debe referenciar a uno o varios archivos en JavaScript (extensión .**js**).

Se debe escribir una aplicación usando el **API de Google Maps** que **cargue desde la máquina cliente el archivo KML de rutas turísticas creado en la práctica de XML**.

Debe incluirse el archivo KML de rutas turísticas utilizado en la práctica de XML.

El diseño de la presentación se deja libre al estudiante.

Tarea 2. Escribir un archivo en HTML5 denominado **MapaGeoJSON** (con extensión **.html**) que referencie a un archivo CSS denominado **MapaGeoJSON** (con extensión **.css**) y también debe referenciar a uno o varios archivos en JavaScript (extensión .**js**).

Se debe escribir una aplicación usando el **API de Google Maps** que **cargue desde la máquina cliente un archivo en formato GeoJSON** con la situación del inicio y de los hitos de las rutas turísticas construidas en la práctica de XML.

Debe buscarse el estándar GeoJSON y crear un archivo denominado **rutas.GeoJSON**. Este archivo es el que se cargará. Debe de incluirse obligatoriamente.

El diseño de la presentación se deja libre al estudiante.

Epílogo. Cada tarea se presenta en la sub-carpeta de **PRACTICA-3** edenominada **Ejercicio-13.** Debe haber dos sub-carpetas denominadas Tarea-1 y Tarea-2 dentro de la carpeta **Ejercicio-13**.

Guía para resolver el ejercicio 13

Para cargar los archivos desde la máquina cliente se debe utilizar el API FILE de HTML5 de forma similar a como se hace en el ejercicio:

http://di002.edv.uniovi.es/~cueva/JavaScript/62API-FILE-LeerArchivoTexto.html

Sobre KML se puede consultar los siguientes tutoriales

https://developers.google.com/kml/documentation/kml_tut?hl=es

https://developers.google.com/maps/documentation/javascript/kml

https://developers.google.com/maps/documentation/javascript/reference/kml?hl=es

https://developer.here.com/documentation/examples/maps-js/maps/display-kml-on-map

Sobre GeoJSON se puede consultar:

https://es.wikipedia.org/wiki/GeoJSON

https://www.developer.here.com/blog/an-introduction-to-geojson

https://developer.here.com/documentation/examples/maps-js/data/display-geojson-on-map

Ejercicio 14

Tarea 1. Realización de una aplicación web, utilizando HTML5, CSS y JavaScript. La aplicación web es de **temática libre**, pero debe usar **como mínimo 3 API de HTML5**.

Se valorará la presentación, la complejidad de la aplicación, la originalidad, la creatividad y las API de HTM5 utilizadas.

Epílogo. Todo se presenta en la misma sub-carpeta de **PRACTICA-3** denominada **Ejercicio-14**

Guía para resolver el ejercicio 14

Algunos ejemplos de API de HTML5

http://di002.edv.uniovi.es/~cueva/JavaScript/22Canvas.html

https://di002.edv.uniovi.es/~cueva/JavaScript/24Geolocalizacion.html

http://di002.edv.uniovi.es/~cueva/JavaScript/62API-FILE-LeerArchivoTexto.html

Listado de APIS de HTML5

- **API TextTrack**. Acceso a las pistas de elementos multimedia (video o audio)
- **API Fullscreen**. Manejo de pantalla completa.
- **API Stream**. Acceso a contenido multimedia de flujo continuo o en streaming.
- **API Canvas**. Permite dibujar, animar y procesar imágenes.
- **API WebGL**. Permite crear gráficos 3D para la Web
    - Suele acompañarse de la biblioteca Three.js (www.threejs.org)
- **API Pointer Lock**. Facilita la interacción con el puntero del ratón.
- **API Drag and Drop**. Facilita arrastrar y soltar en la Web
- **API Web Storage**. Es básicamente una mejora de las cookies. Permite almacenar datos en el disco duro del usuario y utilizarlos posteriormente.
- **API IndexedDB**. Permite crear una pequeña base de datos indexada en el ordenador del usuario.
- **API File**. Permite el manejo de archivos.
- **API Geolocation**. Permiten determinar la ubicación física real del usuario.

- **API History**. Permite gestionar el registro histórico de navegación.
- **API Offline**. Facilita la navegación cuando está el usuario fuera de línea.
- **API Page Visibility**. Informa a la aplicación sobre el estado de visibilidad del documento. Por ejemplo cuando se minimiza una ventana.
- **API Web Messagin**. Permite que aplicaciones de orígenes diferentes se comuniquen entre sí.
- **API WebSocket**. Permite la conexión del cliente con el servidor de una forma más rápida y eficaz a través de TCP sin enviar cabeceras HTTP.
- **API WebRTC**. Permite la comunicación en tiempo real.
- **API Web Audio**. Permite el procesamiento de audio.
- **API Web Workers**. Permite ejecutar tareas en segundo plano.
- **API Clipboard**. Permite el manejo del portapapeles.
- **API Device Orientation**. Permiten determinar la orientación y el movimiento del dispositivo.
- **API Quota Management**. Permite comprobar el espacio de almacenamiento disponible.
- **API SVG**. Permite generar gráficos vectoriales según el estándar SVG