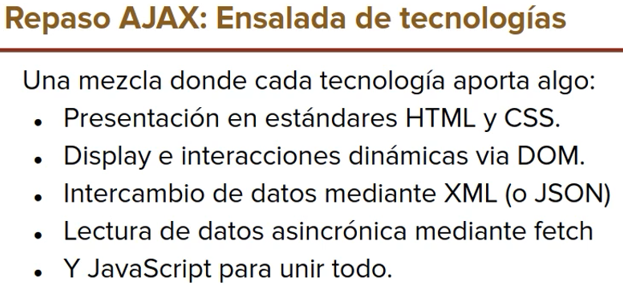
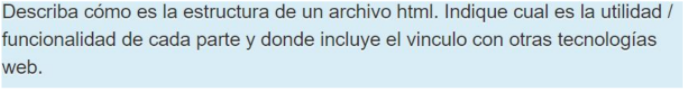
<https://web-unicen.herokuapp.com/api/groups/pia/practica>





Un archivo HTML se compone de 3 partes:

<HTML> Es el elemento raíz, indica el comienzo y el fin del documento. Allí además se indica al navegador que active el soporte de media queries.

<HEAD> contiene la información para el navegador que no es visible. A su vez contiene el tag <TITLE> que lleva el nombre de la pestaña. En esta sección se incluye el vínculo con CSS si lo incluimos como una hoja de estilos externa (visto desde el punto de vista de la cátedra, ya que es un beneficio aplicar un mismo estilo en un sitio sin duplicar código). También, es posible ubicar el script en el head, pero sería necesario realizar una modificación en el archivo js, de manera que permita que se cargue el DOM.

<BODY> solo hay uno por archivo, su contenido es visible en la página web. Contiene todo el desarrollo del código. En esta sección, justo antes de cerrar el body es posible ubicar el script, luego de que ya se cargó el HTML con todos sus elementos.



HTML5 es la última versión de HTML, conocida por sus múltiples beneficios, que hacen que Google Search priorice este tipo de páginas: agrega sentido y estructura al contenido de las páginas mediante etiquetas semánticas nuevas, agrega elementos multimedia y elementos gráficos, está orientado a responsive, posee una mayor accesibilidad y carga más rápido en el navegador.

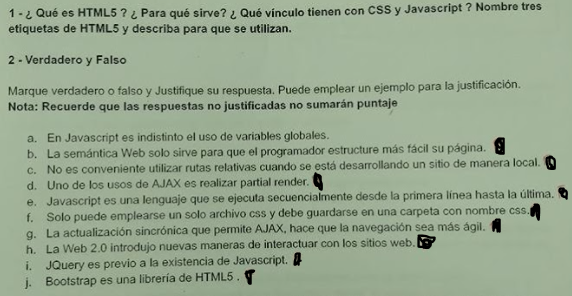
Algunos de los tags agregados son: <header>, <nav>, <article>, <section>, <aside>, <footer>, <time>, <figure>, <figcaption>, <video>, <audio>, <svg>, <canvas>.

<time> se interpreta como la fecha de publicación

<figcaption> se escribe dentro del tag <figure>. Se crea un contenedor para multimedia con su título correspondiente

<section> puede haber más de uno. Agrupa cosas conceptualmente relacionadas. Necesita otras partes para estar completo, es parte de algo más grande.

HTML5 agregó el elemento <button>. Antes un botón de submit se realizaba de la siguiente manera: <input type=”submit” value=”Enviar”/> Ahora se realiza así: <button> Enviar </button> lo que es más amigable, ya que es posible agregarle una imagen o cambiarle el formato al mismo.



-HTML5 es la última versión de HTML, conocida por sus múltiples beneficios, que hacen que Google Search priorice este tipo de páginas: agrega sentido y estructura al contenido de las páginas mediante etiquetas semánticas nuevas, agrega elementos multimedia y elementos gráficos, está orientado a responsive, posee una mayor accesibilidad y carga más rápido en el navegador

-Una página web está compuesta por su base, HTML, encargado de darle estructura y contenido a una página web; CSS, que sirve para dar estilo y apariencia a la misma; y JS, que es lo que le proporciona comportamiento y la posibilidad de interactuar con el usuario. CSS y JS son linkeados desde el HTML.

<figcaption> se escribe dentro del tag <figure>. Se crea un contenedor para multimedia con su título correspondiente

<section> puede haber más de uno. Agrupa cosas conceptualmente relacionadas. Necesita otras partes para estar completo, es parte de algo más grande.

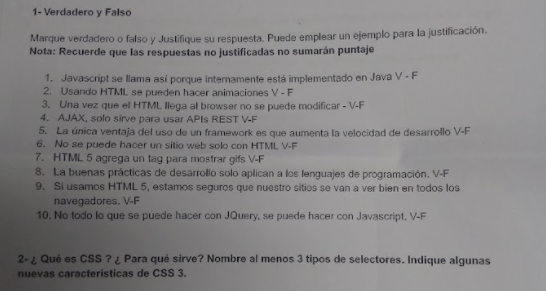
<button> más amigable que el <input type=”submit”>, se le puede agregar una imagen o cambiarle el formato al mismo.



Si bien HTML5 incorpora semántica en su mayor parte, no es sinónimo de la misma.

La semántica es la práctica de darle sentido y estructura al contenido de una página web.

Por otra parte, HTML5 es la última versión de HTML que además de agregar nuevas etiquetas estructurales, agrega elementos multimedia y elementos gráficos, está orientado a responsive, posee una mayor accesibilidad y carga más rápido en el navegador.



CSS es un lenguaje que se encarga de dar apariencia y estilo al contenido. Su estructura está basada en Selectores (determinan a qué elemento se le aplicará cierto estilo y pueden ser de tipo, clase o id), Propiedades (tales como color, tamaño de fuente, tipo de fuente, etc) y Valores (que definen el valor que toma cada propiedad).

CSS3 incorpora la posibilidad de animar mediante @keyframes estableciendo su evolución y cambios en las propiedades. Además, expande la funcionalidad para bordes a través de por ejemplo la propiedad border-radius que agrega un redondeado a los mismos. También agrega efectos de texto tales como sombreado con la propiedad text-shadow. Luego agrega la posibilidad de colocar imágenes de fondo, nuevas fuentes de texto, transparencias, colores y degradados.



La herencia es el mecanismo por el cual algunas propiedades de un elemento se transmiten a sus hijos. Esta va de padres a hijos. El color, todo lo relacionado a las fuentes se hereda, pero el tamaño de los elementos no.

La cascada en cambio, es el mecanismo que, cuando hay diferentes clases, resuelve cuál de esas gana. Se basa en 3 criterios básicos:

1- Importancia: refiere a !important , que sobrescribe toda cascada (mala práctica)

2- Especificidad: más específico es el ID, luego la clase, luego el tipo.

3- Orden en el código fuente: si empata especificidad gl último que aparece en el css

Ejemplos

Herencia: suponiendo que tengo un div, al cual le asigno un color, todo lo que esté dentro de ese div, si no le pongo color, lo heredo del div.

<div class= "colorear">

<div>

<h1> titulo 1 </h1>

</div>

<p> párrafo 1 </p>

</div>

.colorear{

color: red;

}

Cascada: con el estilo a p.colorear se pisa el color que le puse al p, por cascada, especificid

<p>parrafo 1</p>

<p class="colorear">parrafo 2</p>

p {

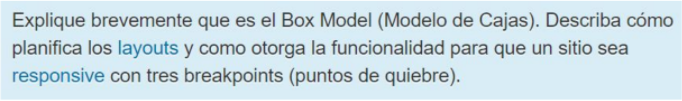
color: pink;

}

p.colorear {

color: yellow;

}



El Box Model es el comportamiento de CSS en el que cada elemento de una página se representa mediante cajas rectangulares/cuadrados. Cada caja presenta un alto (height) y un ancho (width) por lo que es posible cambiar el tamaño de estos contenedores. El contenido está rodeado por espaciados/márgenes transparentes: un padding (espacio interior al borde), un border y un margin (espacio por fuera del borde).

-Una vez que pensamos en la distribución de nuestra página web podemos hacer uso de la técnica Sketching, es decir, realizar un bosquejo sin mucho detalle que reproduzca nuestra idea de manera sencilla. Luego podemos optar por quedarnos con ese concepto o tirarlo y realizar uno nuevo, para luego sacar provecho del comportamiento de Box Model y así lograr el layout deseado.  
Para poder lograr una página web que esté enfocada a distintos dispositivos debemos recurrir a:

1- La idea de Mobile First, basada en pensar primero qué es lo más importante y así desarrollar primero una pantalla chica con menor cantidad de elementos (el móvil) para luego ir ajustando el contenido a las pantallas más grandes e ir agregando elementos.

2- El uso de las Medias Queries, que son un recurso de css que permite asignar distintas apariencias/estilos para diferentes dispositivos con las cuales podremos agregar breakpoints que nos definan que el diseño se comporte diferente bajo ciertas condiciones (determinada resolución).



Para poder lograr una página web que esté enfocada a distintos dispositivos debemos recurrir a:

1- La idea de Mobile First, basada en pensar primero qué es lo más importante y así desarrollar primero una pantalla chica con menor cantidad de elementos (el móvil) para luego ir ajustando el contenido a las pantallas más grandes e ir agregando elementos.

2- El uso de las Medias Queries, que son un recurso de css que permite asignar distintas apariencias/estilos para diferentes dispositivos con las cuales podremos agregar breakpoints que nos definan que el diseño se comporte diferente bajo ciertas condiciones (determinada resolución).

@media only screen and (min-width: 400px) {

// todo el código escrito acá se aplica al menos hay 400px

}

@media only screen and (max-width: 400px) {

// todo el código escrito acá se aplica cuando como máximo hay 400px

}



La declaración de variables en JavaScript se realiza mediante la palabra reservada Let que posee un ámbito de bloque. Luego de la palabra reservada va el nombre de la variable.

Ejemplo: let numero; // en este caso declaramos numero como una variable.

Una variable global es aquella que se declara al principio del archivo JS o se declara en un nivel más alto respecto a la función en la cual va a ser requerida, de esta manera se tiene una variable que puede ser utilizada en todas las funciones ya que está por sobre ellas. Estas variables sólo deben utilizarse para lo que necesito recordar entre distintas ejecuciones de eventos. El uso de este tipo de variable es inadecuado y se debería evitar a toda costa ya que perjudica la legibilidad del código, y tiene una vida igual a la ejecución del código, siendo que una variable debe ser creada y dejada de usar en la ventana de tiempo más corta posible.

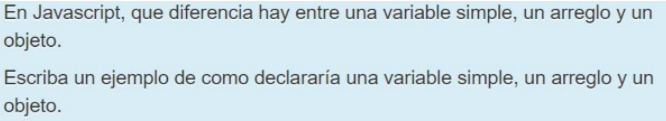


-Una buena práctica en JS es utilizar “use strict” al comenzar un nuevo archivo de este tipo ya que esto hace que el compilador no deje pasar algunos errores (restringe posibles errores de sintaxis y convierte en obligatoria la declaración de variables)

-Al ser JS un lenguaje de tipado dinámico, otra buena práctica es, al declarar una variable, no cambiarle el tipo. Si defino “nombre” como string, que sea siempre de ese tipo.

-En la programación dirigida por eventos, en vez de asignar handlers en HTML (por ejemplo: onclick), buscar el elemento con su id y luego asociarle el handler mediante el addEventListener.

-Una buena práctica es poner las claves de los JSON entre “” para evitar confusiones con palabras reservadas.



-Una variable es un espacio en memoria el cual puede almacenar un dato.

-Un arreglo es un espacio en memoria, una variable que toma muchos valores, los cuales poseen una posición/orden, por las cuales pueden ser accedidos. En el caso de JS, es posible mezclar distintos tipos dentro de un arreglo, pero esto se considera mala práctica, por lo cual siempre lo utilizaremos con tipos de datos iguales.

-Luego un objeto es un formato/notación/forma de escribir, organizado y de fácil acceso compuesto por una clave y un valor donde no importa el orden y que surge la necesidad del mismo por lo explicado anteriormente (para no mezclar distintos tipos en un mismo arreglo).

Declaración de una variable (utilizando Let) → let nombreVariable;

Declaración de un arreglo vacío (utilizando Let) → let nombreArreglo [];

Declaración de un objeto con datos (utilizando Let) →

let nombreObjeto = {

“nombre” : “Jorge”,

“dni” : “2222”,

“telefonos” : [12,2,1,433],

“MetodoSumar” : { sumar(a,b) }

};



Un JSON es un formato/notación/forma de escribir, organizado y de fácil acceso compuesto por una clave y un valor donde no importa el orden. El mismo surge de la necesidad de no mezclar distintos tipos de variables en un mismo arreglo. Una buena práctica es poner las claves de los JSON entre “” para evitar confusiones con palabras reservadas.

Declaración de un objeto vacío → let nombreObjeto {};

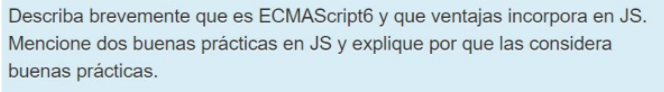
Declaración de un objeto con datos → let nombreObjeto = {

“nombre” : “Jorge”,

“dni” : “2222”,

“telefonos” : [12,2,1,433]

};



ECMAScript6 es una versión del lenguaje de programación JavaScript el cual fue aprobado en el 2015. Cada actualización del lenguaje por lo general no modifica mucho el lenguaje, pero ES6 realmente cambió el juego y podemos decir que es un punto de quiebre en el lenguaje.

Un ejemplo de esto es el cambio en las declaraciones de variables/constantes, donde antes de la ES6 no se declaraban las constantes y las variables eran declaradas con la palabra reservada VAR que maneja un scope por funciones. En la nueva versión se declaran constantes con la palabra CONST e incluyeron la variable LET que maneja un scope de bloque. Otro ejemplo de una ventaja que incorpora ES6 es el foreach, que logra recorrer un arreglo completamente más fácil y sencillamente.

-Una buena práctica en JS es utilizar “use strict” al comenzar un nuevo archivo de este tipo ya que esto hace que el compilador no deje pasar algunos errores (restringe posibles errores de sintaxis y convierte en obligatoria la declaración de variables)

-Al ser JS un lenguaje de tipado dinámico, otra buena práctica es, al declarar una variable, no cambiarle el tipo. Si defino “nombre” como string, que sea siempre de ese tipo.

-En la programación dirigida por eventos, en vez de asignar handlers en HTML (por ejemplo: onclick), buscar el elemento con su id y luego asociarle el handler mediante el addEventListener.

-Una buena práctica es poner las claves de los JSON entre “” para evitar confusiones con palabras reservadas.



La programación dirigida por eventos es en la que se produce un evento (generalmente cuando el usuario interactúa con la página) que es capturado por handlers y en respuesta se ejecute una porción de JS que haga que la apariencia de la página se actualice/modifique de alguna manera. Una consecuencia/desventaja de este tipo de programación es que al no ser de orden secuencial/lineal no se sabe en qué orden sucederán los eventos. En otras palabras, el desafío de este tipo de programación es que los eventos se ejecutan de forma asíncrona en cualquier orden, por lo que el código tiene que funcionar para cualquiera de esos órdenes.

Visto como una buena práctica, desde el JS se trae un elemento (por ejemplo un botón) mediante un id y se le especifica que al ocurrir un evento (por ejemplo click) se ejecute lo que retorna una función. Esto es posible gracias al método addEventListener.

**document.getElementById(“btnPrender”).addEventListener(“click”,prender);**

Otra manera de hacer lo mismo podría ser asignando el handler en el HTML, pero esto es una mala práctica.

**<button type=”button” id=”btnPrender” onclick=”prender();”> Prender </button>**



técnica frontend que combina un set de tecnologías

AJAX es un conjunto de técnicas y tecnologías utilizadas para hacer las páginas web más rápidas,transparentes y amigables/usables para el usuario (para mejorar su experiencia). Esta técnica de carga asíncrona de contenido dinámico y datos del servidor está o es del lado del cliente. Explicado con un ejemplo: cuando un usuario hace un click, en lugar de ir a otra página se dispara un evento JS, que pide que baje los archivos del servidor. Mientras tanto al usuario se le muestra la página entera, pudiendo éste interactuar con la misma. Cuando el servidor responde, el código JS (AJAX Engine) toma los datos y los ingresa en el DOM, con lo cual la página nunca se refresca, sino que cambia en tiempo real.

Una promesa es un objeto que representa la terminación/fracaso eventual de una operación asíncrona (una respuesta). En otras palabras, se le pide algo al servidor y éste avisa cuando esté listo, mientras tanto el usuario sigue interactuando con la página. Una promesa posee 4 estados: cumplida o fulfilled, rechazada o rejected, pendiente o pending y finalizada o settled.

AJAX es el acrónimo de Asynchronous JavaScript And XML, nos permite realizar páginas web asíncronas permitiéndonos así ejecutar código en “segundo plano”. Por ejemplo si tenemos una sección de código que usa partial render a mitad del js no podemos dejar al usuario con la página congelada hasta que se nos devuelva la promesa con lo que pedimos, entonces al utilizar AJAX logramos que esa petición se lleve al WEB APIs que luego irá a la callback queque, se interpreta que la petición se realiza en segundo plano. Una vez terminado los procesos principales (se vacíe la Stack) pondremos el primero de la callback queque en la stack para que se ejecute, y cuando se vuelva a vaciar la stack se volverá a poner el primero de la callback queque y así. de esta manera logramos realizar llamadas asincrónicas permitiendo que el usuario al realizar pedidos a otra web por ejemplo no se le queda la página congelada (una experiencia no grata para el usuario).

Al momento de realizar funciones asincrónicas u operaciones asincrónicas como lo es fetch, nos devolverán objetos que son del tipo promesa, estos debemos tratarlos como tales y por tanto deberemos esperar hasta que se consiga el objeto y ahí podremos tratarlo, este concepto de esperar hasta que se consiga y ahí hacer algo es el método .then(). además podríamos obviar este.then() si en la declaración de la función utilizamos el async y en las peticiones que devuelven promesas y/o que las manipulen les ponemos el await antes.



Para definir REST primero es necesario entender que son las APIs.

Una API es una interfaz que nos da una aplicación para que hable con otra aplicación, es decir, trata la comunicación entre programas. Un ejemplo de esto es cuando determinada página web te solicita loguearte con Facebook o Instagram, y te redirige para que inicies sesión.

Por otra parte, REST es un protocolo/estándar/tipo de arquitectura utilizado para crear y comunicar APIs, por lo tanto, una API se puede implementar con REST. En el mismo, se asocian URLs (que son obtenidas y modificadas con los métodos del protocolo HTTP) a recursos. Dichos métodos, capaces de manipular los datos son:

-POST: crea un recurso

-GET: obtiene 1 o + recursos

-PUT: actualiza/edita 1 o + recursos

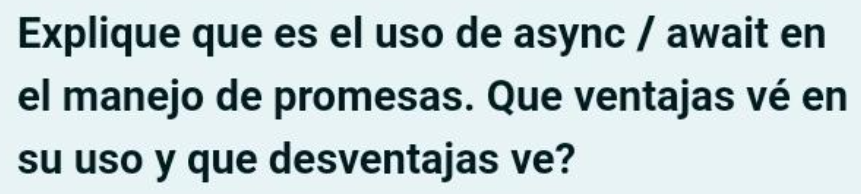
-DELETE: borra un recurso

A diferencia de PARTIAL RENDER, REST no accede a un HTML, sino que baja del servidor un JSON, que luego es procesado del lado del cliente guardándolo/convirtiéndolo en HTML para luego ser insertado y mostrado en el DOM.

Además de la ventaja que conlleva que se trate de una carga asincrónica (no quedamos esperando una respuesta, el usuario puede seguir interactuando con la página), otra ventaja no menos importante es la persistencia, ya que con REST se guardan los datos, por lo que al refrescar la página o si se corta la luz los mismos no se perderán.

Los servicios REST son aquellos servicios que nos permiten pedir información a un servidor y que este nos devuelva información en distintos formatos dependiendo de nuestros requerimientos. Estos son utilizados mediante, por ejemplo un fetch (con la información adecuada para lograr establecer conexión con el servidor). Las ventajas de las API REST es que podemos conectar las páginas web y cruzar información en tiempo real. Una posible aplicación es que el banco central tiene un servidor con el precio del dólar a tiempo real (Hace uso de las APIs REST) y nosotros en vez de tener que ir calculando las cotizaciones manualmente podemos directamente utilizar las APIs REST para hacer una petición al servidor del banco central y que nos devuelva el precio del dólar en tiempo real

La consecuencia de esto es que al realizar pedidos a servidores tiene una alta demora que es perceptible al ser humano y por tanto si debiéramos tener que esperar o congelar la página hasta que se devuelva o no la información estaríamos dando una mala experiencia al usuario de nuestra página web.



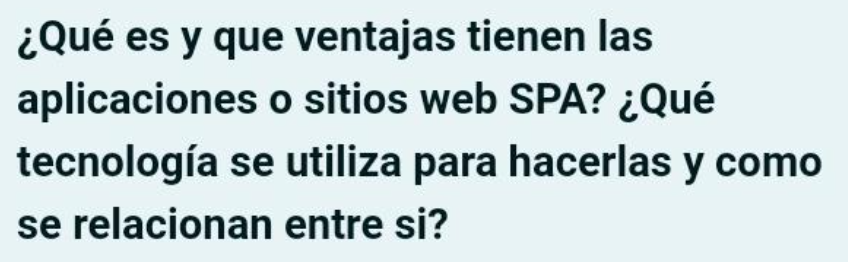
Async y await son 2 palabras reservadas que facilitan la escritura de código con promesas (evitan escribir explícitamente cadenas de promesas) y permiten el asincronismo:

Async se escribe antes que la palabra function y hace que ésta devuelva una promesa. El return se encapsula en la promesa automáticamente. No sirve para funciones anónimas.

Await desencapsula el contenido de una promesa. Se reescribe como el .then de la promesa. Solo puede ser usado dentro de funciones Async. Gracias al await es posible escribir el código como si no hubieran promesas.

Ventajas: es más fácil de seguir y más legible ya que este código asincrónico parece/se escribe como si fuera sincrónico, al escribir código “planchado” evitamos escribir funciones dentro de funciones. Menos código.

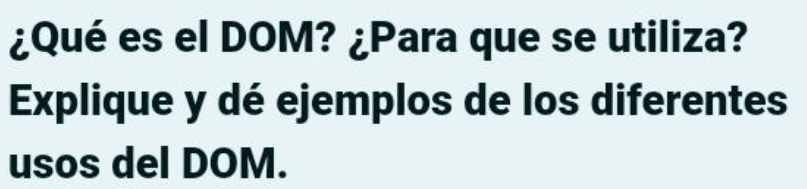
Desventajas: no se puede usar solo funciones anónimas. Si o si las funciones tienen que tener el async.



Una Single Page Application es un tipo de aplicación web en la que nunca se recarga completamente el navegador, sino que los recursos se cargan parcial y dinámicamente cuando la página lo requiere. Toda la aplicación es en realidad una única página. Una Single-Page Application se genera toda en el lado del cliente y ésta va llamando a diferentes API a medida que lo va necesitando. Un ejemplo de esto es cuando en Gmail creamos un nuevo mail y se abre una pestañita abajo.

La tecnología que se utiliza para estas páginas es AJAX, que hace que las páginas web sean más rápidas,transparentes y amigables/usables para el usuario (para mejorar su experiencia). Ventajas:

* Experiencia del usuario más rápida y fluida
* Multitud de frameworks y librerías JS para desarrollar SPA.
* Menor coste de nuestro servidor, si lo tenemos, ya que tiene que soportar menos carga al ejecutarse la mayoría de la lógica en el lado del cliente.



El DOM es un modelo/interfaz que posee la estructura del HTML (todo el HTML) y su relación con JS y CSS. Es una API que representa todo el documento HTML y nos permite manipularlo desde Javascript. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina DOM. A través de él es posible acceder, recorrer, “traer”, modificar, reemplazar, eliminar y crear los elementos HTML de nuestra página.

A través de JS se puede manipular el DOM para crear,remover y renderizar elementos en el DOM, agregar o quitar clases para cambiar el estilo, y demás.



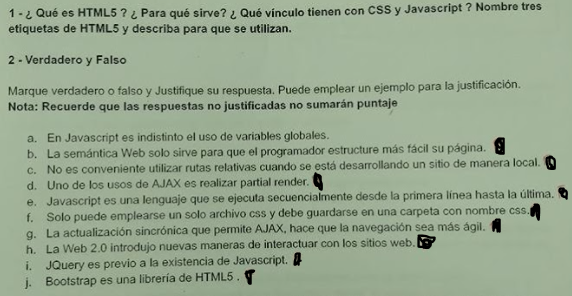
Las funciones de flecha son una forma abreviada de escribir funciones:

function nombre(parámetros) {} pasa a ser (parámetros) => {}

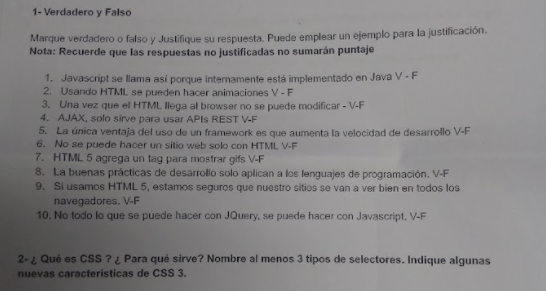
* Si el parámetro es uno solo los paréntesis puede ser omitidos
* Si la función contiene una sola declaración/sentencia se pueden omitir los corchetes y la palabra return
* En las funciones regulares **this** representa el objeto que llama a la función, que puede ser la ventana, el documento, un botón o lo que sea.
* Con las funciones de flecha, **this** siempre representa el objeto que definió la función de flecha.

Como ventajas las arrow function son más cortitas, prácticas para escribir, lo cual hace el código más legible

Por otra parte, una desventaja de las mismas es que el **this** no hace referencia a la misma cosa que en las funciones normales, trabaja distinto.



1. F - estas variables sólo deben utilizarse para lo que necesito recordar entre distintas ejecuciones de eventos. El uso de este tipo de variable es inadecuado y se debería evitar a toda costa ya que perjudica la legibilidad del código, y tiene una vida igual a la ejecución del código, siendo que una variable debe ser creada y dejada de usar en la ventana de tiempo más corta posible.
2. F - además sirve para que Google Search priorice este tipo de páginas, ya que le da. a computadoras, buscadores, y otros dispositivos la capacidad de leer y entender páginas web. Otras ventajas del código semántico son: Fácil de manejar - Fácil de trabajar con él - Comprender para qué está cada porción de contenido.
3. F - Las absolutas no son recomendables. Las rutas deben ser siempre relativas, ya que van a funcionar siempre y cuando el proyecto mantenga la misma estructura de las carpetas/archivos. Si se utilizaran rutas absolutas el proyecto solo funcionaría en donde fue creado.
4. V
5. V - Esto es porque javascript tiene modelo de ejecución de código donde utiliza un ciclo de eventos (event loop)
6. F - Se pueden emplear tantos css como se requiera, deben guardarse en la carpeta css.
7. V
8. V - se incorporó JS
9. JQUERY
10. F - Es un framework de CSS muy utilizado de 12 columnas, pensado mobile first y responsive que está orientado al diseño de layout y tiene soporte de múltiples navegadores



1. F - JS no está relacionado de ninguna manera con Java.
2. F - HTML se encarga solo de la estructura y el contenido de una página. Se pueden hacer animaciones desde CSS mediante @keyframes estableciendo su evolución y cambios en las propiedades.
3. F - Se puede modificar con JS.
4. F - AJAX también sirve para implementar Partial Render.
5. F - Además de acelerar el tiempo de desarrollo, un framework tiene otras ventajas tales como: pocos problemas de compatibilidad - si tiene una comunidad grande atrás puede ser fácil del debug - modular, trabajo en grupo.
6. V (ambigua).
7. F - los gifs se muestran con el tag <img> y basta con cambiar la extensión de la ruta por .gif. Sería: <img src="/images/panda.**gif**" alt="Panda">
8. F - ya que HTML y CSS no son específicamente lenguajes de programación y poseen buenas prácticas, por ejemplo:

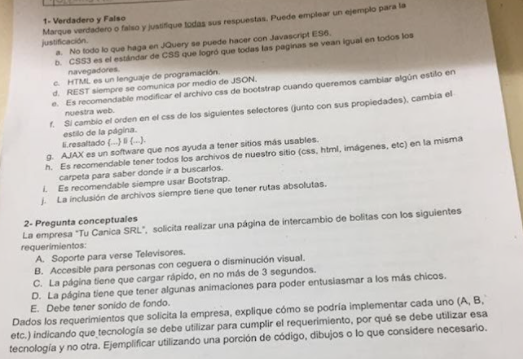
HTML:

* escribir tags en minúsculas para mayor legibilidad
* utilizar el atributo alt en imágenes para gente no vidente y buscadores
* utilizar <label> antes/después de un input ya que proporciona estructura, significado y mejora la accesibilidad de la página
* en el tag <video> siempre incluir el atributo controls

CSS:

* incluir un CSS mediante una hoja de estilos externa, ya que permite aplicar un mismo estilo en un mismo sitio (varios HTMLs) sin duplicar código.
* utilizar la propiedad text-decoration: none para eliminar el subrayado al texto que no sea link
* utilizar unidades de medidas relativas (%) ya que poseen mayor flexibilidad y permiten ajustes con cambios de tamaño de pantallas
* sitio responsive, mismo sitio para todos los dispositivos
* utilizar grupo de selectores para evitar la duplicación de estilos

1. F - La página escrita en HTML5 no va a andar en cualquier navegador, no tiene full support, pero las versiones más recientes de los navegadores más populares del mercado soportan HTML5.
2. JQUERY



1. JQUERY
2. F - CSS3 es el estándar de CSS que aún tiene diferencias entre navegadores.
3. HTML es un lenguaje de marcado que se utiliza para la creación de páginas web.
4. V
5. F - si se quiere cambiar algún estilo hay que cambiar las clases en el html o en caso de querer agregar algo nuevo hay que hacerlo “a mano” en otro css y tener en cuenta que no altere lo demás.
6. F - en cualquier orden lo que sucede es que se le aplica las propiedades a los <li> con clase resaltado (por ser más específico) y por otro lado a los demás <li>
7. V
8. F (ambigua) - es recomendable tener un carpeta que se componga por subcarpetas de css, js e images, con los archivos correspondientes en cada una. Los archivos html quedarían “sueltos” en la carpeta principal, junto a las subcarpetas.
9. F - dependiendo la necesidad y el diseño que se requiera, ya que si un cliente solicita una página con un diseño muy específico quizás en bootstrap no se encuentre ese diseño particular. Además, habría que adaptar el diseño a una grid de 12 columnas.
10. F - Las absolutas no son recomendables. Las rutas deben ser siempre relativas, ya que van a funcionar siempre y cuando el proyecto mantenga la misma estructura de las carpetas/archivos. Si se utilizaran rutas absolutas el proyecto solo funcionaría en donde fue creado.