Preguntas Frecuentes

Clase 1

¿Las variables se pueden usar una única vez?

No, las variables pueden usarse las veces que sean necesarias y sus valores pueden cambiar a lo largo del programa.

¿Hay alguna forma recomendada para definir el nombre de las variables?

En programación existen reglas para escribir los elementos a los que llamamos convenciones de nombres o nomenclatura. Para las variables en JavaScript recomendamos emplear CameCase. Referencia de interés: http://www.w3bai.com/es/js/js_conventions.html

¿Hay diferencia entre usar comillas dobles o simples para declarar una cadena de caracteres?

No, la notación es equivalente. Más adelante se pueden combinar este estilo de comillas para concatenar cadenas complejas.

¿Cómo agrego un espacio en blanco en la cadena al sumar dos strings?

El espacio en blanco debe incluirse en la cadena como cualquier otro caracter. Ejemplo durante la concatenación:

```
var textoA = "Juan";
var textoB = "Perez";
var resultado = textoA + " " + textoB;
```

¿Se puede escribir código JS directamente sobre el HTML?

No, el código JS siempre debe estar construido entre etiquetas <script></script> o en un archivo .js.

¿Es mejor escribir el código JS en un archivo aparte?

A menos que sea un código pequeño, exclusivo para la página HTML, siempre es recomendable separar el script en uno o más archivos .js ya

que es más fácil de leer y mantener. Referencia: https://www.crockford.com/code.html

¿Cuál es la diferencia entre inicializar y asignar una variable?

Estamos inicializando una variable cuando le asignamos un valor al declararla, ejemplo:

```
var textoA = "Juan";
```

Una vez declarada, cada vez que damos un nuevo valor a la variable, usando el igual (=) decimos que "asignamos un valor", ejemplo:

```
textoA = "Perez";
```

¿Qué son o para qué sirven las sentencias en javascript?

Podemos decir que, en javascript, el término sentencia es otra forma de llamar a las instrucciones con la que construimos el script. Referencia **https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias**

Clase 2

¿Cuál es la diferencia entre =, == y ===?

- El operador (**=**) es para asignar un valor a una variable. (a = "valor")
- El operador (**==**) es para comparar si dos valores son iguales, sin importar su tipo de dato. (1 == "1") es verdadero porque es el mismo valor independientemente de si el dato es numérico o string.
- El operador (**===**) es para comparar si dos elementos son iguales, tanto en valor como en tipo de dato. (1 === 1) es verdadero (1 === "1") es falso.

¿Qué es un if anidado?

Se dice que un if está ***anidado*** cuando se codifica dentro de otro if o bloque. En otras palabras, si dentro de un condicional hay otros condicionales se dice que contiene condicionales anidados.

¿Qué tipo de operadores existen?

En el siguiente enlace podés acceder a distintos tipos de operadores https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions_and_Operators

¿Puedo comparar más de dos valores en un condicional?

Si. Es posible construir una comparación con más de dos valores (Ejemplo (1 > 2 < 4)). Generalmente se emplean operadores lógicos para comparar distintos valores en la misma instrucción. Referencia: https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Operadores/Operadores_l%C3%B3gicos

¿Es necesario usar siempre el else en el if?

No es obligatorio. Sin embargo, es necesario cuando quiero codificar la respuesta al caso “falso” del condicional evaluado.

¿Son relevantes las mayúsculas y minúsculas a la hora de comparar cadenas de caracteres?

Sí ya que el intérprete distingue entre mayúsculas y minúsculas (case sensitive), siendo el valor de una letra en mayúscula distinto a la misma letra en minúscula. Esto quiere decir que “A” es distinto que “a”.

¿Por qué obtengo NaN en una operación?

Al parsear un string de valor no numérico (Ejemplo parseInt("A")) obtengo ***NaN*** y representa que el resultado obtenido “No es un Número”. Se emplea para notificar que el valor resultante de una operación no puede convertirse en número.

Clase 3

¿Qué diferencia hay entre var y let?

Son diferentes palabras reservadas que se utilizan para declarar variables. Podes revisar las diferencias de cada unadesde el siguiente enlace:
https://plataforma.coderhouse.com/foro/development/diferencias-entre-var-let-y-const

¿Puedo tener una función dentro de otra?

Es posible llamar a una función dentro de otra. También es posible declarar funciones anidadas. Referencia: https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Funciones

¿Cuál es la diferencia entre declaración y llamada de la función?

La declaración de la función comprende la codificación de las instrucciones que componen la función, en alguna parte del script. Mientras que la llamada de la función es la utilización de una función **ya declarada** en una instrucción empleando el nombre de la misma.

¿Cuándo debo usar un for y cuándo un while?

- El ***for*** se emplea cuando vamos a repetir un conjunto de instrucciones un número determinado de veces. La cantidad de veces a repetir está sujeta al valor de una variable numérica, generalmente conocida como índice, cuyo valor suele incrementar con cada iteración y cuando llega a un valor específico el bucle termina. Por ejemplo, si necesitamos repetir 5 veces algo usamos un for.
- El ***while*** se emplea cuando vamos a repetir un conjunto de instrucciones hasta que cierta condición deja de ser verdadera. Generalmente se efectúa una comparación sobre una variable y

un valor esperado, repitiendo hasta que la comparación sea falsa (Ejemplo repetir hasta que el usuario presiona “ESC”).

¿Puedo recorrer en sentido inverso con un for o recorrer de 2 en 2?

Sí, con ***for*** puedo hacer variar la sentencia para hacer un recorrido decremental o por iteraciones pares.

¿Cuándo es recomendable usar break o continue?

- Es recomendable usar ***break*** cuando quiero detener la repetición al detectar cierta situación adicional. Por ejemplo, se detecta un error en la conversión de un número al operarlo, entonces interrumpo la ejecución del bucle porque es un valor necesario en la siguiente iteración.
- Es recomendable usar ***continue*** cuando quiero omitir una iteración completa, pero continuar con la ejecución de la siguiente iteración del bucle. Por ejemplo, si detecto un valor que no es necesario procesar, pero en la siguiente repetición este valor va a cambiar.

¿Cómo saber si un número es par o impar?

Puedo emplear el ***operador resto*** (%), teniendo presente que si al dividir un número por 2 el resto es 0 es un número par, en caso contrario será impar. https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Operadores/Aritm%C3%A9ticos#.25_.28modular.29

(4 % 2) es igual a 0 entonces cuatro es par
(5 % 2) es igual a 1 entonces cinco es impar

Clase 4

¿Cuál es la diferencia entre parámetro y variable?

Las variables son elementos que se utilizan para almacenar valores cuyo uso puede darse en todo el programa (global) o en un bloque

específico (local). Mientras que los parámetros (también llamados argumentos) son elementos específicos de las funciones, identificados como las variables o valores que enviamos a la función para ser procesados cuando esta sea llamada.

¿Cuántas veces debo declarar una variable para utilizarla en una función?

Todas las variables deben declararse una única vez, independientemente si son locales o globales. Si dentro de una función se declara una variable local con el mismo nombre que una global, la función utilizará el valor local ante el global de la variable.

¿Cómo interpreta javascript qué valor corresponde a cada parámetro en una llamada a la función?

Los interpreta en ***orden*** de escritura. Por ejemplo, si la función admite dos parámetros:
unaFuncion(parametro1, parametro2) al llamarla pasamos los valores unaFuncion(22,33). 22 es parametro1 y 33 es parametro2.
Si invertimos el orden de los valores unaFuncion(33,22) ahora 33 es parámetro1 y 22 parametro2.

¿Cómo sé cuándo debo ocupar una variable local o una global?

- Debo declarar la variable como ***global*** si quiero emplearla en más de un bloque en el script, es decir, busco poder acceder a esta variable desde cualquier parte del programa.
- Debo declarar la variable como ***local*** si ésta es importante únicamente para cierta función o bloque, es decir, no es necesario usar la variable en otras partes del programa.

¿Por qué el return está en la declaración de la función y no en la llamada?

Porque el ***return*** sirve para determinar si la función devuelve o no un valor al ser llamada. Mientras que la llamada es la ejecución de dicha función en el programa, devuelva un valor o no.

¿Cuándo es aconsejable utilizar el return?

Se utiliza ***return*** cuando queremos obtener un valor resultante al llamar a la función. Este valor es el resultado del proceso que ocurre dentro de la función, el cual varía a razón de los parámetros enviados.

¿Es necesario tener los valores y las variables predefinidas para declarar una función que los utilice?

Es posible enviar valores literales como parámetros si la función los admite.(EJ suma(1,2))
Pero si busco enviar variables como parámetro estas tienen que estar declaradas previamente. (EJ suma(unNumero, otroNumero)
Tanto para variables globales como locales, estas tienen que declararse previamente antes de emplearse en las sentencias de la función.

Clase 5

¿Siempre es necesario crear objetos?

Podemos decir que necesitamos un objeto cuando existe la siguiente situación en el proyecto:

- Tengo que definir un elemento cuya información está compuesta por más de un valor (Ej: Necesito el nombre, apellido, DNI y edad de una Persona).
- Tengo más de un elemento de este tipo en el programa (Ej: Tengo almacenadas muchas Personas).
- Existen operaciones comunes (funciones y/o métodos) para todos los elementos de este tipo (Ej: Necesito imprimir los nombres y apellidos de todas las personas almacenadas).

Si un elemento cumple estas características debe definirse como un objeto.

¿Un objeto siempre se guarda en una variable?

Para utilizar un objeto en otras instrucciones del programa se lo asocia a una variable. Si es un único objeto se emplea una única variable y si pertenece a un conjunto de objetos se lo almacena en un array.

¿Cuáles son las diferencias entre propiedades y parámetros?

Las propiedades son elementos que se utilizan para almacenar valores en los objetos. Mientras que los parámetros (también llamados argumentos) son elementos que se utilizan especificamente en las funciones. Si bien los paràmetros pueden ser variables o valores que enviamos a la función constructora de un objeto, es en las propiedades donde estos son almacenados durante el intanciado del objeto. (EJ this.nombre = parametroNombre)

¿Para qué sirven los métodos? ¿En qué se diferencian de otras funciones?

En JavaScript, identificamos a los métodos como funciones que no retornan un valor. Es decir, que un método es un conjunto de instrucciones que efectúan un procesamiento y/o almacenamiento sin retornar un resultado al ser llamado.

¿Los this solo funcionan dentro de la función constructora?

No. This puede emplearse dentro de cualquier bloque o parte del programa para hacer referencia al contexto actual de la instrucción. Referencia (distintos casos de this) https://yeisondaza.com/entendiendo-this-javascript

Clase 6

Cuando ocupo los métodos de string ¿estoy modificando el string original o solamente como se ve?

El valor del string original es inmutable. Esto quiere decir que una vez asignado solo se puede cambiar el valor de la variable, asignando un nuevo string. No se puede cambiar el valor original carácter a carácter.

¿Se pueden aplicar varios métodos a una variable en una misma línea?
Sí. Es posible encadenar métodos en JavaScript.
Referencia:
https://abalozz.es/patron-chaining-o-encadenamiento-de-metodos-en-javascript/

¿Qué tipo de datos puedo incluir en mi array?
Todos los tipos y estructuras de datos. El incluir un tipo de dato en el array no excluye a otros. Es posible contar con un array de datos heterogéneos.

¿Las listas siempre comienzan en 0?
Sí. Las posiciones de los arrays siempre inician en 0.

¿Una lista o array es lo mismo que un objeto?
En JavaScript un array es un tipo de objeto que permite almacenar otros objetos o valores literales. El array se reconoce como una estructura de datos para agrupar elementos.

¿Cómo puedo ordenar un array?
En JavaScript podemos usar el método short() para ordenar una array.
Referencia:
https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array/sort

¿Es necesario inicializar el array para comenzar a trabajar con él?

Como buena práctica, para declarar una variable de tipo array vacía se recomienda emplear la siguiente sintaxis: **let unArray = [];**

Clase 7

¿Qué tipo de datos puedo almacenar en un storage y dónde se almacenan?

Puedo almacenar datos en formato clave/valor, ambos se almacenan como texto plano (https://developer.mozilla.org/es/docs/Web/API/DOMString). Esta información se almacena en el propio navegador del usuario.
Referencia: **https://developer.mozilla.org/es/docs/Web/API/Storage**

¿Cuándo necesito usar un localStorage?

Cuando necesito conservar un dato para consultarlo cada vez que abro el simulador desde el navegador. También si quiero compartir esta información entre pestañas del mismo navegador. Por ejemplo: Nombre del usuario que emplea el simulador en ese navegador".

¿Cuándo necesito usar un sessionStorage?

Cuando necesito conservar un dato de manera temporal mientras cierto proceso ocurre en el simulador y no quiero que sea una información compartida entre pestañas o se conserve más allá de la pestaña actual. Por ejemplo: Productos visualizados antes de confirmación de agregar al carrito, para sugerirlos durante el proceso de compra en la pestaña actual.

¿Es lo mismo storage que cookies?

No, existen diferencias destacables entre los storage y cookies:

- LocalStorage: La información perdura hasta que limpias caché e información local. Puedes almacenar más información que una cookie. La seguridad de lo almacenado depende del propio navegador.
- SessionStorage: La información es almacenada hasta que se cierra el navegador o la pestaña. Puede almacenar más

información que una cookie. La seguridad de lo almacenado depende del propio navegador.

- Cookies: Almacenan poca información en comparación al storage pero perduran el tiempo que se define al crearlas. Una de las ventajas es que las cookies se pueden emplear fácilmente en el front-end y en el back-end (Sin tener que emplear AJAX ya que la información del LocalStorage o SessionStorage esta almacena en el front-end/cliente) La seguridad es limitada.

¿En qué se diferencia JSON de un objeto o de un array?

JSON es un lenguaje de marcado extendido de la forma de escribir objetos en JavaScript. Existen similitudes entre cómo se accede a un JSON y a un Objeto en JavaScript por lo cual a veces se los confunde.
Hay que tener presente que JSON es un formato para definir información en texto plano, es decir, es una cadena de caracteres con cierto formato que contiene datos.
En cambio, cuando se declara, en el código JS, una estructura clave/valor pasa a ser un Objeto Literal. Si bien se le puede asignar métodos, es recomendable separar los datos (clave/valor) del comportamiento/funcionalidad (Clase/Objeto prototipo).
Los datos contenidos en el JSON son agregados a nuestros objetos para operar sobre ellos.

¿Qué utilidad nos podría brindar utilizar el formato JSON en el simulador?

La idea es definir una estructura para determinar con qué información trabaja el simulador. Es común que aplicaciones web modernas obtengan (consuman) información del propio servidor de aplicación o de algún servicio de terceros (webservices) para funcionar. Estos datos generalmente están en formato JSON.

Para determinar qué información necesito en el simulador podemos preguntarnos:

- ¿Tengo que crear objetos a partir de información previa? (no generada por el usuario, obtenida previamente al cargar el simulador). De ser así, definir los datos que se utilizarán para instanciar los objetos parseando un JSON.
- ¿Necesito datos fijos de entre los cuales el usuario puede optar en la interfaz?. Por ejemplo, seleccionar entre un listado de países, generado a partir de la información del JSON, modificando el DOM.

¿Puedo utilizar los datos de un JSON directamente en javascript?

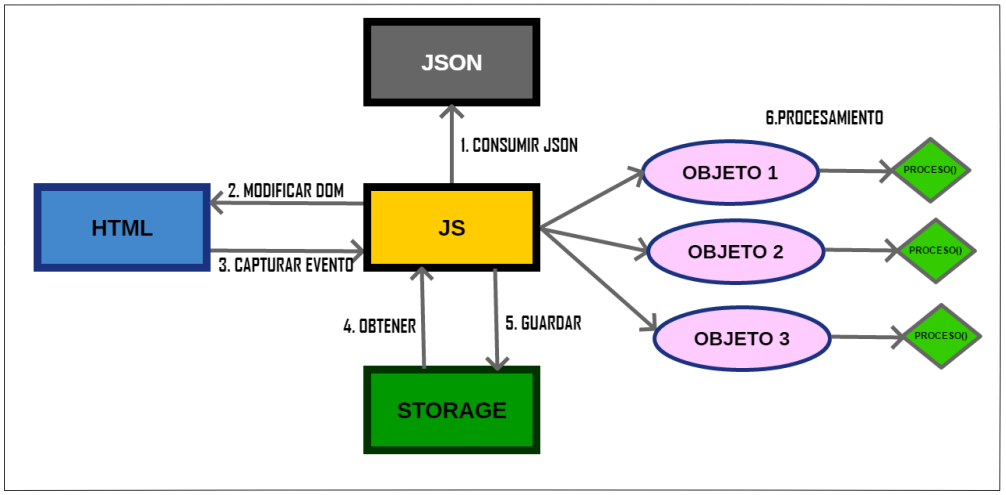
No. Si contamos con un string en formato JSON es necesario parsear a objeto usando JSON.parse.
https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/JSON/parse

Clase 8

¿Para qué sirve modificar el HTML desde JavaScript?

Para incluir salidas visuales, resultado del procesamiento, en el HTML y responder a las acciones del usuario. Podemos visualizar la importancia del script para determinar la interactividad del simulador en el siguiente esquema:

Mi simulador



Este es un diagrama simplificado, que si bien no refleja la arquitectura real de un aplicación web moderna, nos permite tener una idea aproximada de los componentes e interacciones, entre las cuales se detallan:

- JSON: Elemento que contiene datos necesarios para que nuestra aplicación funcione (información de productos, datos de usuario, datos de servicios, etc). Es el repositorio de datos que nuestro simulador emplea. Mediante JavaScript obtenemos (1.Consumir JSON) la información para 2.Modificar el DOM o crear objetos.
- HTML: Representa la interfaz del simulador, es la parte con la que el usuario interactúa. Con JavaScript podemos 2.Modificar el DOM para que el usuario perciba la información resultado del 6.Procesamiento o detectar que está haciendo el usuario en el simulador (3. Capturar Evento) para ejecutar funciones o métodos en consecuencia.
- STORAGE: Recurso del navegador que nos permite guardar información en él. Esos datos pueden ser compartidos entre todas las pestañas y permanentes al cierre del navegador(LocalStorage) o temporales a una pestaña particular (SessionStorage). Con JavaScript podemos 4.Obtener y 5.Guardar en cada Storage para utilizar esa información en el simulador.
- JavaScript: Representa el componente principal del simulador. Nosotros programamos el comportamiento de este componente

para resolver el problema a simular. Usamos variables, estructuras de control, funciones, métodos y objetos con el objetivo de construir una solución. En este sentido, es el código JS el encargado de realizar todo el procesamiento, modificar el HTML para mostrar datos, reaccionar a los eventos, guardar y obtener información si se requiere.

¿Cómo se cuál es la jerarquía de un dom?

El DOM es una interpretación del documento HTML actual generada por el navegador. En JavaScript poseemos un objeto (document) para acceder a los elementos (nodos) de esta jerarquía, la cual varía en función de la disposición de las etiquetas HTML que forman parte del documento y su anidamiento. Referencia
https://developer.mozilla.org/es/docs/Referencia_DOM_de_Gecko/Introducci%C3%B3n

¿Cómo capturo los datos de un formulario para poder trabajar con ellos?

Escuchando el evento submit sobre el formulario, previniendo el comportamiento por defecto del evento (e.preventDefault) y obteniendo el valor de cada input.
Tutorial: **https://es.javascript.info/forms-submit**

¿Cómo capturo los datos con input de tipo select?

Escuchando el evento change sobre el select y obteniendo el valor actual.
Tutorial: **https://es.javascript.info/events-change-input**

¿Se puede obtener elementos del DOM por data attributes?

Es posible acceder a los atributos de datos de un elemento empleado getAtributte(). Pero para obtener un nodo del DOM se emplean lo métodos `getElementById`, `getElementsByClassName`,

getElementsByName, getElementsByTagName o getElementsByTagNameNS.

Clase 9

¿Para qué puedo utilizar los eventos en mi proyecto?

Para detectar las acciones que realiza el usuario en el navegador y/o en los elementos de la aplicación. Referencia: **https://developer.mozilla.org/es/docs/Learn/JavaScript/Building_blocks/Eventos**

¿Puedo combinar más de una sintaxis de captura de eventos?

Es posible escuchar más de un evento por elemento. Pero al menos que se tenga una respuesta idéntica para ambos eventos, se recomienda programar sus respuestas por separado.

¿Un evento realizado por un teclado es lo mismo que el del mouse?

Ambos son eventos, pero varían en tipo de evento y en sus propiedades.
Mouse: https://developer.mozilla.org/es/docs/Web/API/MouseEvent
Teclado: https://developer.mozilla.org/es/docs/Web/API/KeyboardEvent

¿Los eventos solamente me sirven para interactuar con los botones y formularios?

No, además es posible controlar eventos sobre el propio navegador, la carga de elementos (Interfaces e imágenes), teclado, mouse y errores. Referencia: **https://developer.mozilla.org/es/docs/Web/Events**

¿Cuál es el uso del evento onload?

Para determinar si todos los elementos del HTML o un recurso en específico concluyeron su carga. Referencia: **https://developer.mozilla.org/es/docs/Web/Events/load**

¿Todos los navegadores trabajan de la misma forma con los eventos?

Los llamados eventos estándar son soportados por todos los navegadores. Pero hay navegadores que ofrecen eventos adicionales, no compatibles con otros navegadores.
Referencia:
https://developer.mozilla.org/es/docs/Web/Events#Eventos_Est%C3%A1ndar

Clase 11

¿Necesito emplear siempre librerías en mi proyecto?

Las librerías se emplean porque nos permiten contar con métodos y recursos pre programados y optimizados. No son obligatorias, pero son herramientas que acotan el tiempo de trabajo y nos brindan una base para programar múltiples funcionalidades.

¿Cuál es la diferencia entre librería y framework?

Una librería normalmente proporciona una serie de funciones/métodos concretos para simplificar tareas complejas.
Un framework aporta una estructura completa bajo la cual nosotros desarrollamos el proyecto, implementando la lógica concreta de la aplicación.
Generalmente un framework establece una forma de trabajo que adoptamos para construir una solución entera, mientras que las librerías simplemente nos brindan funciones para incluir en nuestro script como creamos conveniente.

¿Cuál de las 2 formas de cargar jquery es mejor?

Tanto si cargamos jQuery en nuestro proyecto de forma local o al hacerlo desde un servidor CDN tenemos distintas ventajas y desventajas a analizar, siendo en algunos casos una forma preferible

sobre la otra. Podes consultar una explicación en detalle sobre el uso de CDN aquí: https://raiolanetworks.es/blog/cdn/

¿Cuándo es útil utilizar librerías con formato minificado?

Es útil emplear librerías en formato minificado cuando terminado de programar todo la aplicación (o un conjunto de nuevas funcionalidades) y vamos a publicarla. Es preferible emplear código minificado en librerías y scripts propios en producción, ya que permite disminuir los tiempos de carga de la aplicación y el tamaño de los archivos que componen el sitio.

¿documet.ready es lo mismo que window.onload?

No, con document.ready() estamos escuchando el evento de carga del DOM. Mientras que con window.onload debemos esperar a la carga de todos los elementos en la ventana, incluyendo imágenes y elementos embebidos. Por ejemplo, si buscamos trabajar con los elementos que componen la web (divs, inputs, …) empleamos *$(document).ready()*. Pero, si por el contrario, vamos a trabajar con imágenes para alinearlas o comprobar sus dimensiones tendremos que usar *window.onload*.

¿Qué diferencia hay entre window y document?

El objeto window representa la ventana actual, es decir al propio navegador
https://developer.mozilla.org/es/docs/Web/API/Window
Mientras que el objeto document representa el documento HTML cargado en la ventana.
https://developer.mozilla.org/es/docs/Web/API/Document

Clase 12

¿Cuál es la función de un selector JQuery?

Los selectores de jQuery nos permite seleccionar diferentes elementos del DOM, ya sea por su identificador, clase, etiqueta o su posición en la

jerarquía DOM. La ventaja de los selectores es la posibilidad de combinar estos criterios para acceder a elementos específicos.

¿Existe alguna diferencia entre manejar los eventos del DOM desde JQuery?

No. El método on que empleamos para manejar los eventos con jQuery funcionan de forma similar a addeventlistener(). Dado que buscamos determinar qué eventos vamos a escuchar sobre los elementos, obtenidos con selectores jQuery, utilizamos *on* o el método abreviado (*ejemplo .click() en el caso de .on("click", handler)*) para definir la asociación evento-funcion de respuesta.

¿Los selectores de jquery aplican para cualquier elemento html?

Si, podemos emplear los selectores para acceder a cualquier elemento del DOM. En el caso de no existir un elemento que coincida con el selector, no obtenemos elemento alguno.

¿Cuál es la diferencia de hacer hover y focus en js?

La pseudo-clase hover de CSS se emplea cuando el usuario interactúa con un elemento, pero no necesariamente lo activa. Generalmente hover ocurre cuando el usuario se desplaza sobre un elemento con el cursor (puntero del mouse).
Mientras que el evento focus se dispara cuando se activa un input (presionamos sobre el mismo) y tomamos el foco. Podemos capturar el evento focus en input de tipo text, textarea, button, checkbox, file, password, radio, reset y submit.

¿Puedo usar el parámetro event en jQuery?

Si, es posible emplear el parámetro de información del evento en jQuery. Ejemplo

```
$( "body" ).on( "click", "a", function( event ) {
  event.preventDefault();
```

});

Clase 13

¿Para qué me serviría ocultar o mostrar un elemento?

Usar animaciones para ocultar y mostrar elementos de la interfaz nos permite controlar la disponibilidad y acceso a componentes de la aplicación en función de las acciones del usuario. La posibilidad de contar con menús y secciones emergentes (por ejemplo carritos de compras en tiendas de pagos) determinan el grado de interactividad que nuestro software ofrece al usuario.

¿Cuál es la función de callback?

Callback es la función que pasamos por parámetro a otra función y se ejecutará cuando se detecta cierta situación en la aplicación. Por ejemplo, al terminar la animación correspondiente o al detectar el evento escuchado.
Referencia:
https://developer.mozilla.org/es/docs/Glossary/Callback_function

¿Puedo modificar el CSS desde mi JQuery?¿Por qué?

Es posible modificar el estilo de un elemento con jQuery utilizando el método css(). Generalmente se emplea cuando queremos modificar el estilo en respuesta a un evento de usuario. Por ejemplo, focus o blur sobre un input), como parte de la respuesta de una función callback.

¿Las animaciones funcionan con cualquier elemento del HTML?

Si, con jQuery podemos animar cualquier elemento del DOM, accedido previamente con un selector.

¿Se pueden combinar dos animaciones al mismo tiempo?

Si, podemos combinar animaciones encadenado métodos en jQuery
Referencia: http://www.w3bai.com/es/jquery/jquery_chaining.html

Clase 14

¿Qué es un servidor web?

El servidor es una aplicación cuya tecnología nos permite alojar nuestro proyecto web para que otros usuarios puedan utilizarlo. En un servidor web incluimos los ficheros que comprenden nuestro software, pudiendo funcionar esta en forma local o accediendo a través de internet.
Video explicativo: https://youtu.be/4Hcugh7z0cI

¿Qué es HTTP y HTTPS?

HTTP, de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre del protocolo que nos permite realizar una petición de datos y recursos al servidor. Mientras que HTTPS es similar a HTTP, solo que agrega tecnología para proteger las peticiones (encriptado SSL). Con HTTP y HTTPS utilizamos métodos para solicitar y/o enviar información a un servidor.
Video explicativo de Métodos HTTP: https://youtu.be/C4kwGzCIV80

¿Qué es un proceso asíncrono y para qué sirve en simulador?

El diseño de software asíncrono amplía el concepto al construir código que permita a un programa solicitar que una tarea se realice al mismo tiempo que la tarea (o tareas) original, sin detenerse a esperar a que la primera se haya completado. Cuando la tarea secundaria se completa, la original es notificada usando un mecanismo acordado, de tal forma que sepa que el trabajo se ha completado y que el resultado, si es que existe, está disponible.
Utilizamos peticiones asíncronas con jQuery para efectuar procesamientos en segundo plano en nuestro simulador (por ejemplo enviar datos a una url mientras esperamos por su confirmación en la misma interfaz).

¿Puedo utilizar cualquier API con AJAX?

Si. Empleamos los métodos $.get, $.post y $.ajax de jQuery para hacer peticiones asíncronas a cualquier servidor. No obstante, hay que tener en cuenta que cada API puede tener sus propias reglas de acceso: Las urls varían e incluso se pueden solicitar claves que el desarrollador debe solicitar previamente para su uso.

¿En qué formato se devuelven los valores solicitados a una API? ¿Puedo operar directamente sobre ellos?

Al solicitar datos a un API, la respuesta puede recibirse en formato JSON o XML. Puedo operar sobre los datos recibidos en la función callback, empleando el estado de la petición para determinar si se obtuvieron o enviaron los datos correctamente.

Clase 15

¿Cualquier aplicación que construya puede ser SPA?

Puedo aplicar la arquitectura de aplicaciones de una sola página (SPA) a cualquier solución construida. No obstante, hay proyectos donde se recomienda su utilización más que en otros, como por ejemplo para e-commerces y web apps de altas prestaciones.
Referencia:
https://anexsoft.com/que-son-las-single-page-application-spa-ventajas-y-desventajas

¿Qué función cumple el router?

Un router en SPA es un elemento que procesa las peticiones de rutas del usuario de la siguiente manera:

1. Cargar la ruta. Identificar dónde nos encontramos en el sitio. Se realiza a través de una carga inicial de la ruta.
2. Comparar la URL con una de nuestras rutas. La URL a la que nos queremos mover, se debe comparar con las rutas que tenemos ya

que la ruta solicitada debe de estar entre nuestras rutas definidas para poder ser cargada.

3. Actualizar la ruta en la barra de navegación. Para esto podemos utilizar un método de HTML conocido como pushState. Este método forma parte del objeto window de nuestro navegador windows.history.pushState. Este método nos agrega un estado a nuestra historia de navegación y al agregar un estado nuevo se refleja en la barra de navegación.
4. Actualizar el DOM con el nuevo contenido. El nuevo contenido se puede mandar a través de innerHTML.

https://dev.to/alexcamachogz/creando-un-router-con-vanilla-javascript-27pl

¿Por qué es recomendable emplear MVC?

La arquitectura MVC propone, independientemente de las tecnologías o entornos en los que se base el sistema a desarrollar, la **separación de los componentes de una aplicación en tres grupos** (o capas) principales: **el modelo, la vista, y el controlador**, y describe **cómo se relacionarán entre ellos** para mantener una estructura organizada, limpia y con un acoplamiento mínimo entre las distintas capas. Es decir, que esta separación contribuye no solo a mantener el software, si no a mejorar cada parte por separado, sin comprometer el funcionamiento de la otra.

¿Puedo usar MVC con JS plano?

Es posible crear nuestra propia implementación del patrón MVC en javascript empleando estructura de clases. No obstante, si se determina el empleo de este patrón es aconsejable utilizar un framework que facilite la implementación de esta arquitectura. Como ser vue.js o react.js

¿Qué otro uso tienen las expresiones regulares?

Las expresiones regulares son patrones que se utilizan para determinar si una cadena coincide con cierta combinación de caracteres. Su uso más común es para determinar si un nombre de usuario o contraseña tiene cierto formato.

Referencia:

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_Expressions