Actividades Unidad 1

1. Define la siguiente lista de términos. Para cada uno de ellos debes indicar su propósito, si pertenecen al lado servidor o cliente y al menos dos características diferenciadoras:

1. HTML5

1.1. Propósito: Es un lenguaje de marcado utilizado para estructurar todo el contenido de una página web e incluye elementos para texto, imágenes, videos, enlaces y más

1.2. Lado: Cliente

1.3. Características:

1.3.1. Posee elementos semánticos como <header>, <article>, <section>, <footer>, etc, que ayuda a describir mejor la estructura del contenido

1.3.2. Ofrece soporte nativo de elementos multimedia como vídeo o audio.

2. PHP

2.1. Propósito: Es un lenguaje de programación utilizado para crear webs dinámicas. Puede generar contenido HTML y trabajar con BBDD para crear webs interactivas.

2.2. Lado: Servidor

2.3. Características:

2.3.1. Posee una gran compatibilidad con todo tipo de BBDD

2.3.2. Tiene una gran comunidad de desarrolladores y muy buenas bibliotecas y frameworks.

3. CSS

3.1. Propósito: CSS (Cascading Style Sheets) es utilizado para dar estilos y diseño a documentos HTML, como tamaños, colores, márgenes y posicionamiento de los elementos.

3.2. Lado: Cliente

3.3. Características:

3.3.1. Nos permite separa la estructura (HTML) del estilo (CSS), lo que facilita el mantenimiento y legibilidad del código.

3.3.2. Ofrece gran flexibilidad para personalizar la apariencia de la página.

4. **jQuery**
   - **4.1. Propósito:** Es una biblioteca de JavaScript que nos simplifica la manipulación del DOM (Document Object Model) y la interacción con eventos en la página web. El código creado es compatible entre navegadores.
   - **4.2. Lado:** Cliente
   - **4.3. Características:**
     - 4.3.1. Simplifica tareas comunes de JavaScript como manejas eventos, animaciones y realizar solicitudes AJAX
     - 4.3.2. Posee una gran compatibilidad entre navegadores gracias a la abstracción sobre el DOM y las API

5. **Node.js**
   - **5.1. Propósito:** Es un entorno de ejecución de JavaScript del lado del servidor nos permite construir aplicaciones web escalables y en tiempo real. Muy adecuado para aplicaciones basadas en eventos y en tiempo real.
   - **5.2. Lado:** Servidor
   - **5.3. Características:**
     - 5.3.1. Utiliza JavaScript tanto en el Cliente como en el Servidor nos facilita el desarrollo de todo con el mismo lenguaje
     - 5.3.2. Es muy eficiente y escalable debido a su modelo de entrada/salida no bloqueante.

6. **React.js**
   - **6.1. Propósito:** React.js es una biblioteca de JavaScript para elaborar interfaces de usuario interactivas y reactivas. Permite la creación de componentes reutilizables y se utiliza comúnmente para construir aplicaciones de una sola página (SPA)
   - **6.2. Lado**: Cliente
   - **6.3. Características:**
     - 6.3.1. Utiliza un enfoque de componentes para la construcción de interfaces de usuario, lo que facilita la reutilización y el mantenimiento del código.
     - 6.3.2. Implementa el concepto “virtual DOM” para mejorar el rendimiento al actualizar la interfaz de usuario de manera eficiente.Es una representación en memoria del DOM real. Cuando el estado de un componente cambia, React vuelve a renderizar la interfaz.

7. **AJAX**
   - 7.1. **Propósito:** Es una tecnica de desarrollo web que permite cargar y enviar datos enter el cliente y el servidor de forma asíncrona, sin necesidad de recargar toda la página. Su utiliza para crear aplicaciones web más dinámicas y responsivas.
   - 7.2. **Lado:** Cliente
   - 7.3. **Características:**
     - 7.3.1. Permite actualizar parte especificas de la página web sin afectar al resto de la página.
     - 7.3.2. Utiliza XMLHttpRequest o la API Fech para realizar solicitudes asíncronas al servidor.

8. **Fetch API**
   - 8.1. **Propósito:** Proporciona una interfaz JavaScript para realizar solicitudes HTTP y trabajar con respuestas de manera más moderna y versátil.
   - 8.2. **Lado:** Cliente
   - 8.3. **Características**
     - 8.3.1. Promesas y sintaxis moderna ya que utiliza promesas en lugar de callbacks, lo que hace que el código sea mas limpio y fácil de leer.
     - 8.3.2. Soporte nativo de JSON simplifica el manejo de este formato permitiendo una conversión automática de la respuesta a un objeto JavaScript.

9. **WebSocket**
   - 9.1. **Propósito:** Proporciona una comunicación bidireccional en tiempo real entre el cliente y el servidor, lo que lo hace adecuado para aplicaciones en tiempo real como chats.
   - 9.2. **Lado:** Cliente y Servidor
   - 9.3. **Características**
     - 9.3.1. Comunicación bidireccional en tiempo real.
     - 9.3.2. Baja sobrecarga de datos ya que utiliza un protocolo eficiente que reduce la sobrecarga de datos y la latencia .

10. **Web Worker**
    - 10.1. **Propósito:** Permite la ejecución de scripts en segundo plano en hilos separados para mejorar la capacidad de respuesta de las aplicaciones web sin bloquear la interfaz de usuario
    - 10.2. **Lado:** Cliente
    - 10.3. **Características**
      - 10.3.1. Procesamiento en segundo plano permite tareas en segundo plano sin bloquear la interfaz de usuario
      - 10.3.2. Utiliza hilos de ejecución separados lo que evita conflictos y permite que múltiples tareas se ejecuten de manera independiente y paralela para un rendimiento más eficiente.

11. **ECMAScript**
    - **11.1. Propósito:** Es el estándar en el que se basa JavaScript, especificando la sintaxis y las características del lenguaje.
    - **11.2. Lado:** Cliente y Servidor (ya que es la especificación base de JavaScript)
    - **11.3. Características:**
        - 11.3.1. Es el estándar en el que se basa JavaScript asegurando la consistencia y portabilidad del lenguaje en diferentes entornos y navegadores.
        - 11.3.2. Se actualiza regularmente para incluir nuevas características y mejorar el lenguaje para aprovechar las últimas capacidades de JavaScript

ECMAScript → Define las reglas y el comportamiento del lenguaje de programación JavaScript, Pertenece al Cliente y Algunas de sus características son: Está en continua actualización para sacar mejorar y aumentar el rendimiento de JavaScript en la Web.

2. Responde y Justifica tu respuestas

2.1. ¿Podemos ocultar el código escrito en JavaScript de una página web?

No ya que JavaScript se ejecuta en el Cliente por lo tanto el Cliente puede ver todo el código pero sí se puede **ofuscar** para hacer el código totalmente ilegible e imposible de copiar o de saber lo que hace

2.2. Ventajas de incluir el código JavaScript en archivos en lugar del propio documento.

Crear el código en diferentes archivos en vez de añadirlo al HTML tiene varias ventajas.

1. **Mantenibilidad:** Al dividir el código en archivos diferentes facilita su mantenimiento ya que cada fichero se destina a realizar una función exacta.
2. **Reutilización:** Al tener el código en archivos diferentes podemos reutilizar las funciones y scripts para múltiples páginas webs
3. **Cache**: Los archivos pueden ser cacheados en el navegador del cliente y al volver a visitar la web tardará menos en cargar ya que tiene varios archivos ya cargados localmente.
4. **Colaboración:** Al trabajar en equipos es más sencillo si se trabaja con archivos separados ya que pueden trabajar en diferentes archivos sin interferirse.
5. **Mejora la legibilidad HTML:** Se mantiene el HTLM más limpio y centrado en la estructura en lugar de mezclarlo con JavaScript , hace que el HTML sea más legible y mantenible.

2.3. ¿Para qué sirve y cómo funciona la función console.log()?

Es una función de JavaScript que se utiliza para imprimir mensajes o datos en la consola del navegador. Es principalmente usada para depurar el código y verificar que los valores de las variables sean los deseados durante la ejecución del código.

El uso es sencillo se escribe:
console.log()
Y dentro de los paréntesis se añade el nombre de las variables, un string concatenado con “+” o también objetos. Todo lo que deseemos para verificar su contenido. Se usa para identificar errores.