UD1

Tarea: Selección de arquitectura y herramientas de programación

Tarea 1: Lenguajes Interpretados Vs Lenguajes Compilados Vs Lenguaje

Realice un **resumen** explicando cada una de ellas y mencione las **ventajas y desventajas** de estos 3 tipos/clasificación de los lenguajes de programación. ¿En qué categoría se sitúa JavaScript?, ¿Y Typescript?, ¿Y el lenguaje C++? . Muestre algún ejemplo y trate de que se le queden bien claros y asimilados estos 3 conceptos.

NOTA: Debe de encontrar la respuesta en Internet.

Lenguajes compilados

Los lenguajes compilados son convertidos directamente a código máquina que el procesador puede ejecutar. Como resultado, suelen ser más rápidos y más eficientes al ejecutarse en comparación con los lenguajes interpretados. También le dan al desarrollador más control sobre aspectos del hardware, como la gestión de memoria y el uso del CPU. Algunos ejemplos de lenguajes compilados C, Java y Go

Lenguajes interpretados

Estos lenguajes ejecutan línea por línea el programa y a la vez ejecuta cada comando. Aquí, si el autor decide que quiere usar un distinto aceite de oliva, podría borrar el anterior y agregar el nuevo. Tu amigo traductor puede decirte ese cambio a medida que sucede. Ejemplos comunes de lenguajes interpretados son PHP, Ruby, Python y JavaScript.

Transpilado

Este consiste básicamente en tomar las características de la semántica y sintaxis del lenguaje de origen sobre el código fuente de origen y adaptarlo para otro lenguaje destino sobre su código fuente destino. El proceso de transpilado sólo puede hacerse al mismo nivel de abstracción de los lenguajes. Es decir, un lenguaje que es codificado en bytes, solo admite una transpilación cuyo lenguaje transpilado de destino sea codificado en bytes y no en assembler. Un ejemplo sería TypeScript

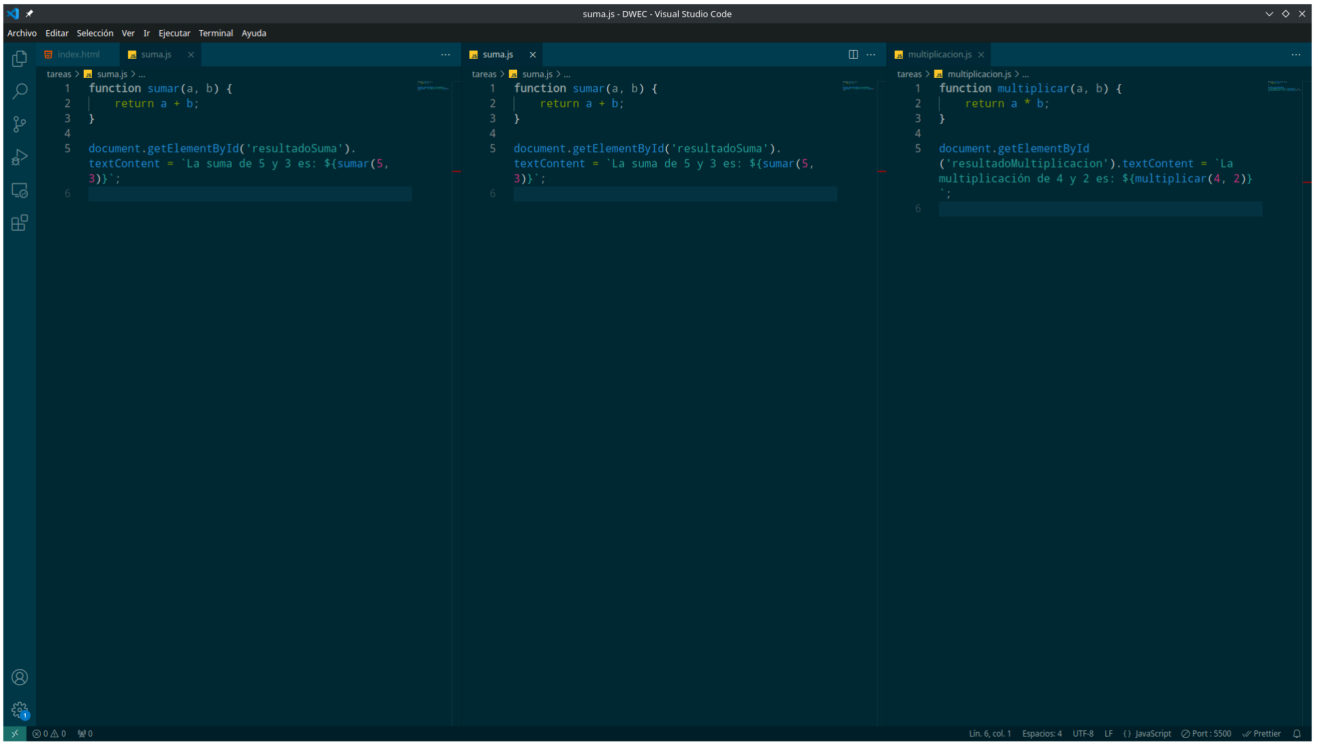
Tarea 2: ¡¡¡ Varios códigos fuentes .js !!!

Realice una ejemplo de cómo cargar código javascript desde más de un archivo .js en una misma página HTML. Busque por internet códigos javascript “sencillos” que hagan alguna operación sencilla. Al menos 2 códigos, y construya una página web simple que “importe” esos códigos que estarán guardados en archivos diferentes. Lo que se trata es que aprenda a construir una web simple, que haga uso de código javascript, pero que ese código se importe desde diferentes archivos.

index.html

suma.js

multiplicacion.js



Tarea 3: ¿Dónde es más recomendable incluir código javascript y por qué?

Es más recomendable incluir código JavaScript al final del documento HTML, justo antes de la etiqueta de cierre `</body>`. mi

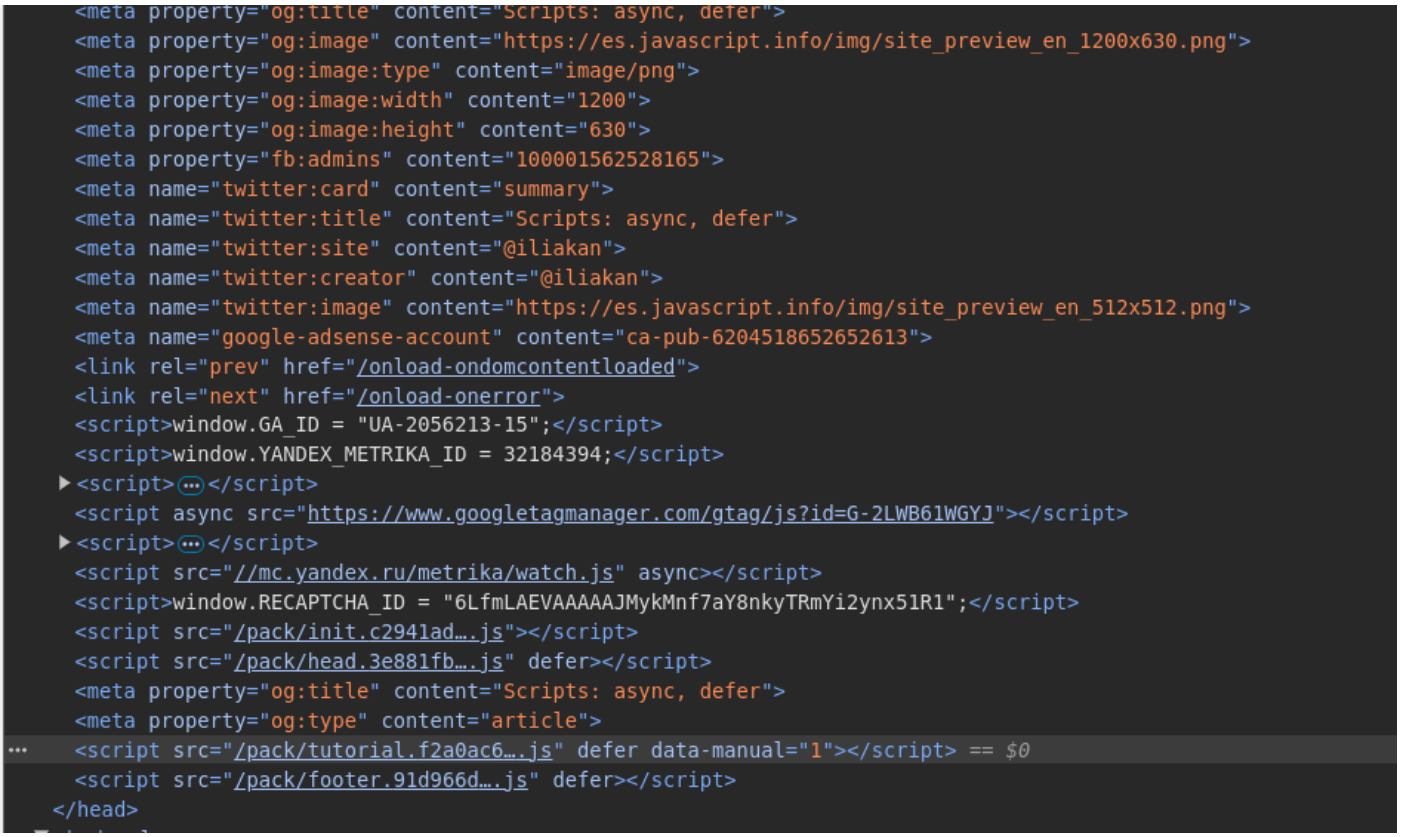
Sin embargo, en algunos casos, como scripts que son necesarios para la inicialización de la página, puede ser útil incluirlos en la sección `<head>`, pero `defer` o `async`.

Haciendo una lectura sobre el siguiente documento **"Cargando archivos .js en nuestra Web"** trate de razonar y ver los pros y contras, y/o los diversos casos en los que un programador se puede ver en esta disyuntiva. Debe de escribir una conclusión similar a la que podrá ver en el documento con sus propias palabras. Además responda a las siguientes preguntas:

3.1. ¿Cómo se cargan los scripts en la web "www.google.com" (busque en su código fuente)?

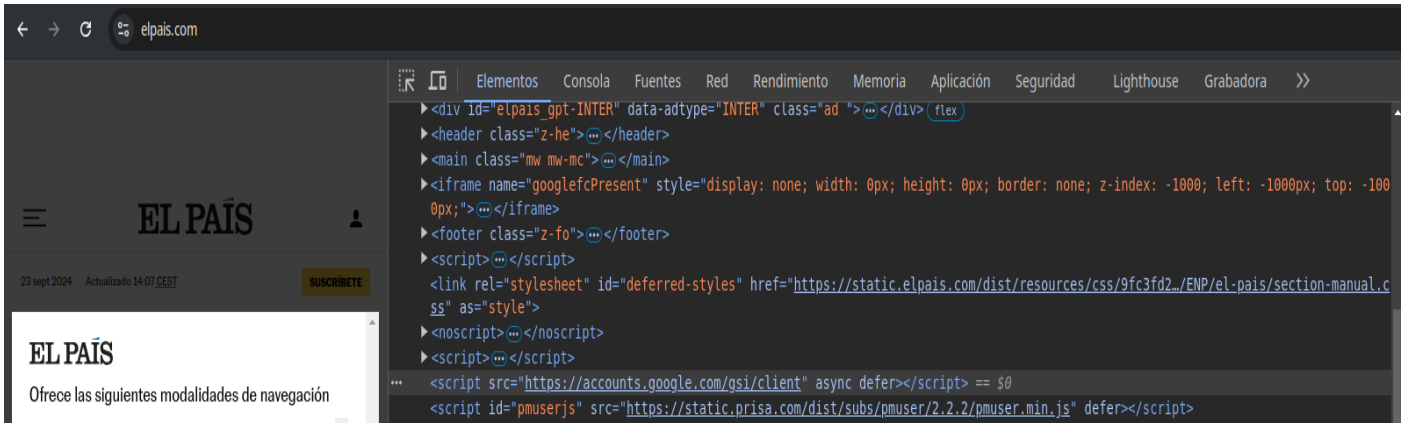
Al final del body hay varios scripts

3.2. ¿Utiliza el atributo "defer" la web "https://es.javascript.info/script-async-defer" en su código fuente? En caso de que sea afirmativo Indique el número de línea y la línea en sí.



3.3. Busca alguna web en la que aparezca el atributo “defer” y/o “sync”. Indica la url, copia la línea de código e indica en qué línea de código aparece.

***NOTA**: En esta web también puede encontrar información valiosa: [The Placement of Script Tags in HTML: Choosing Between Top and Bottom]*



Tarea 4: Introducción a la recursividad.

La recursividad es una técnica de programación en la que una función que realiza cierta operación, se realiza internamente una llamada a sí misma, y de esta forma, el problema a resolver se subdivide en iteraciones que paso a paso resuelven el problema. Es importante resaltar que dentro de esta función siempre tiene que existir un **“caso base”** que determine el final de las iteraciones.

Tarea 4.1. Haciendo uso de **Copilot** u **chatGPT**(cualquier inteligencia artificial es válida) busque cómo se realiza haciendo uso de **recursividad** el cálculo del **“factorial de un número”** (el factorial de un número se calcula multiplicando el mismo número por todos sus precedentes al restar 1 al número).

```
function factorial(n) {
  // Caso base: 1! = 1
  if (n === 1) {
    return 1;
  } else {
    // Caso recursivo: n! = n * (n-1)!
    return n * factorial(n - 1);
  }
}
```

Tarea 4.2. Para esta tarea entregue el código con recursividad de una función que calcule la **potencia de un número**. La función debe llamarse **potencia(b, e)** donde b es la base, y "e" es el exponente. Realice la programación de la misma por sí mismo.

```
function potencia(b, e) {
  // Caso base: cualquier número elevado a la potencia 0 es 1
  if (e === 0) {
    return 1;
  } else if (e < 0) {
    // Caso recursivo para exponentes negativos
    return 1 / potencia(b, -e);
  } else if (e % 2 === 0) {
    // Caso recursivo para exponentes pares
    return potencia(b * b, e / 2);
  } else {
    // Caso recursivo para exponentes impares
    return b * potencia(b * b, (e - 1) / 2);
  }
}
```

Tarea 5: Busque información sobre los distintos y más usados frameworks/librerías utilizados hoy día para desarrollar funcionalidad propia de JavaScript.

Hay mucha información sobre estos frameworks en la web. De lo que trata la tarea es de alcanzar una visión global y observar cuales son tendencia en el mundo actual. Lo discutiremos en clase. Indique las webs desde donde ha conseguido dicha información y trate de averiguar para qué se realiza cada uno de ellos.

Front-end Frameworks:

1. React (https://reactjs.org/)
    - Propósito: Construir componentes de interfaz de usuario reutilizables para aplicaciones web.
    - Descripción: Desarrollado por Facebook, React es una opción popular para construir interfaces de usuario complejas e interactivas.
2. Angular (https://angular.io/)
    - Propósito: Construir aplicaciones web complejas y aplicaciones de página única (SPAs).
    - Descripción: Desarrollado por Google, Angular es un framework completo para construir aplicaciones web robustas y escalables.
3. Vue.js (https://vuejs.org/)
    - Propósito: Construir aplicaciones web progresivas y flexibles.
    - Descripción: Vue.js es un framework en crecimiento rápido que ofrece una curva de aprendizaje más accesible que React y Angular.

Back-end Frameworks:

1. Express.js (https://expressjs.com/)
    - Propósito: Construir aplicaciones servidoras rápidas, escalables y flexibles.
    - Descripción: Express.js es un framework popular de Node.js para construir aplicaciones web y APIs.
2. Koa.js (https://koajs.com/)
    - Propósito: Construir aplicaciones servidoras robustas, escalables y flexibles.
    - Descripción: Koa.js es un framework de Node.js de próxima generación diseñado para ser más ligero y flexible que Express.js.

3. Node.js (https://nodejs.org/)
    - Propósito: Construir aplicaciones servidoras escalables y de alto rendimiento.
    - Descripción: Node.js es un entorno de ejecución de JavaScript construido sobre el motor de JavaScript V8 de Chrome, lo que permite a los desarrolladores ejecutar JavaScript en el servidor.

Librerías:

1. jQuery (https://jquery.com/)
    - Propósito: Simplificar la manipulación del DOM y el manejo de eventos.
    - Descripción: jQuery es una librería popular para simplificar el desarrollo de JavaScript, especialmente para navegadores antiguos.
2. Lodash (https://lodash.com/)
    - Propósito: Proporcionar funciones de utilidad para la manipulación de arrays, objetos y funciones.
    - Descripción: Lodash es una librería de utilidad que ofrece una amplia gama de funciones para trabajar con estructuras de datos.
3. Moment.js (https://momentjs.com/)
    - Propósito: Simplificar la manipulación de fechas y horas.
    - Descripción: Moment.js es una librería popular para trabajar con fechas y horas en JavaScript.

Otros mencionados:

1. Ember.js (https://emberjs.com/)
    - Propósito: Construir aplicaciones web complejas y escalables.
    - Descripción: Ember.js es un framework maduro para construir aplicaciones web robustas.
2. Backbone.js (https://backbonejs.org/)
    - Propósito: Construir aplicaciones web escalables y modulares.
    - Descripción: Backbone.js es un framework ligero para construir aplicaciones web con un enfoque en la modularidad.

Evaluación de la Tarea

Tareas	Puntuación	Criterios
Tarea 1	2.25 puntos	x
Tarea 2	2.25 puntos	x
Tarea 3	2.25 puntos	x
Tarea 4	2.25 puntos	x
Tarea 5	1 punto	x