Kahoot - Tema TypeScript

1. El tipo “any”...

A) Solo funciona con strings
B) Impide asignar valores distintos
C) Es recomendable usarlo siempre
D) Permite almacenar cualquier valor y evita comprobaciones de tipo ✅

Explicación: El tipo `any` le dice a TypeScript que no valide el tipo de la variable, permitiendo que contenga cualquier valor, pero sacrificando la seguridad del tipado estático.

---

2. Una clase en TypeScript se declara con

A) template
B) class ✅
C) object
D) struct

Explicación: Al igual que en JavaScript moderno (ES6), TypeScript utiliza la palabra reservada `class` para definir plantillas de objetos con propiedades y métodos.

---

3. ¿Qué palabra reservada permite importar elementos de otros archivos?

A) use
B) include
C) import ✅
D) require

Explicación: TypeScript utiliza el sistema de módulos de ES6, donde la palabra clave `import` es el estándar para traer funciones, clases o variables desde otros archivos.

---

4. ¿Qué método de Number representa “no es un número”?

A) Number.ZERO
B) Number.NONE

C) Number.UNDEFINED
D) Number.NaN ✅

Explicación: `NaN` (Not-a-Number) es una propiedad del objeto global `Number` que se utiliza para indicar que el resultado de una operación aritmética no es un número válido.

---

5. Un namespace sirve para...

A) Evitar colisiones de nombres agrupando código lógico ✅
B) Generar clases automáticamente
C) Crear estilos CSS
D) Declarar variables globales

Explicación: Los namespaces se utilizan para organizar el código en contenedores lógicos, evitando que nombres de variables o funciones iguales en diferentes partes del proyecto entren en conflicto entre sí.

---

6. TypeScript es...

A) Un lenguaje totalmente distinto a JavaScript
B) Una superextensión de JavaScript con tipado estático ✅
C) Un framework de backend
D) Un preprocesador de CSS

Explicación: TypeScript es un "superset" de JavaScript, lo que significa que añade funcionalidades (como tipos y clases avanzadas) sobre el lenguaje base, pero todo código JS válido es código TS válido.

---

7. El tipado en TypeScript es...

A) Estático y comprobado en compilación ✅
B) Dinámico en todo momento
C) Igual que en JavaScript
D) Solo para funciones

Explicación: A diferencia de JavaScript (dinámico), TypeScript comprueba los tipos durante la fase de compilación (antes de ejecutarse), detectando errores de asignación de forma temprana.

---

8. Un tipo union permite...

A) Combinar propiedades de varios tipos
B) Crear tipos genéricos
C) Aceptar uno u otro tipo entre varias opciones ✅
D) Convertir tipos automáticamente

Explicación: Las uniones (usando el símbolo `|`) permiten que una variable pueda contener valores de diferentes tipos específicos (por ejemplo: `string | number`).

---

9. ¿Cuál es un modificador de acceso válido?

A) protected, global, visible
B) public, protected, global
C) public, private, protected ✅
D) public, private, vinner

Explicación: TypeScript utiliza `public` (acceso total), `private` (solo dentro de la clase) y `protected` (clase y herederos) para controlar la visibilidad de los miembros de una clase.

---

10. Las tuplas se caracterizan por...

A) Tener un número fijo de elementos y tipos específicos por posición ✅
B) No tener un tamaño definido
C) Ser idénticas a los arrays
D) Funcionar solo con strings

Explicación: A diferencia de un array común, una tupla permite definir exactamente cuántos elementos contiene y qué tipo de dato debe ir en cada índice específico.

---

11. ¿Cuál de estas NO es una manera válida de declarar variables en TypeScript?

A) fixed ✅
B) let
C) const
D) var

Explicación: TypeScript utiliza `let`, `const` y `var` (heredados de JavaScript). La palabra `fixed` no existe como palabra reservada para declarar variables en el lenguaje.

---

12. Las interfaces en TypeScript...

A) Son equivalentes a las clases
B) Definen estructura de objetos sin implementación ✅
C) Solo sirven para métodos
D) Son opcionales pero solo para módulos

Explicación: Una interfaz es un contrato que define la "forma" o estructura que debe tener un objeto (propiedades y tipos), pero no contiene la lógica ni el código de ejecución de los métodos.

---

13. Una función con parámetros opcionales usa...

A) ? ✅
B) !
C) %
D) &

Explicación: Se añade el símbolo `?` después del nombre del parámetro en la declaración de la función para indicar que ese argumento no es obligatorio al invocarla.

---

14. Los template strings permiten...

A) Crear objetos JSON
B) Incluir expresiones con ${} dentro de cadenas entre backticks ✅
C) Insertar HTML automáticamente
D) Crear estilos dinámicos

Explicación: Los template strings (usando comillas invertidas `` ` ``) permiten la interpolación de variables y expresiones directamente dentro de la cadena de texto de forma sencilla.

---

15. Los métodos estáticos...

A) No pueden tener parámetros
B) Se acceden a través de la clase sin crear instancias ✅
C) Son obligatorios en todas las clases
D) Solo funcionan dentro de objetos instanciados

Explicación: Los métodos marcados con `static` pertenecen a la clase en sí y no a los objetos creados a partir de ella, por lo que se llaman usando `NombreClase.metodo()`.

---

16. ¿Cuál es un tipo primitivo estándar en TypeScript?

A) array
B) number ✅
C) object
D) interface

Explicación: Los tipos primitivos son los más básicos. `number` es uno de ellos, mientras que `array` y `object` son tipos de referencia e `interface` es una estructura de diseño.

---

17. ¿Qué permite el polimorfismo en TypeScript?

A) Compilar más rápido
B) Crear archivos JSON automáticamente
C) Tratar objetos distintos mediante una interfaz o superclase común ✅
D) Duplicar clases

Explicación: El polimorfismo permite que diferentes objetos respondan al mismo mensaje o método de formas distintas, siempre que compartan la misma base o interfaz.

---

18. ¿Qué tipo representa funciones que nunca retornan valor?

A) none
B) void
C) empty
D) never ✅

Explicación: Aunque `void` se usa cuando una función termina pero no devuelve nada, `never` se usa específicamente para funciones que nunca terminan (como un bucle infinito o una función que siempre lanza un error).

---

19. Los parámetros rest permiten...

A) Limitar los tipos permitidos
B) Aceptar un número variable de argumentos en forma de array ✅
C) Evitar usar arrays
D) Devolver múltiples valores

Explicación: Usando la sintaxis `...args`, puedes pasarle a una función todos los argumentos que quieras y esta los recibirá agrupados automáticamente dentro de un solo array.

---

20. ¿Qué archivo contiene la configuración del compilador de TypeScript?

A) tsoptions.js
B) tsconfig.json ✅
C) types.json
D) config.ts

Explicación: El archivo `tsconfig.json` es el corazón de un proyecto TS; allí se definen las reglas de compilación, la versión de salida y qué archivos incluir.

---

21. ¿Cuál de estas palabras permite exportar algo desde un módulo?

A) provide
B) export ✅
C) send
D) expose

Explicación: En el sistema de módulos de ES6 y TypeScript, la palabra clave `export` se utiliza para hacer que funciones, objetos o variables sean accesibles desde otros archivos.

---

22. ¿Cuál es una función de flecha correcta?

A) `=> (x) x`
B) `(x) => x` ✅
C) `(x) -> return x`
D) `function => (x) { return x }`

Explicación: La sintaxis correcta de una función de flecha (*arrow function*) en TypeScript y JavaScript es definir primero los parámetros entre paréntesis `(x)`, seguidos de la flecha `=>` y finalmente el cuerpo o retorno de la función `x`.

---

23. Un enum sirve para ...

A) Definir un conjunto de constantes con nombres descriptivos ✅
B) Crear arrays tipados
C) Declarar variables globales
D) Ejecutar funciones automáticamente

Explicación: Los `enums` (enumeraciones) permiten definir un nuevo tipo de dato que agrupa un conjunto de valores constantes relacionados (como los días de

la semana o estados de un pedido), haciendo el código más legible y fácil de mantener.

---

24. ¿Qué palabra clave indica herencia entre clases?

A) clone
B) inherit
C) extends ✅
D) from

Explicación: En TypeScript (y JavaScript moderno), se utiliza la palabra clave `extends` para crear una subclase que hereda todas las propiedades y métodos de una clase base.