Criterio de Corrección	Descripción	Resultado Esperado
Modularización	- separación de la complejidad en capas - aislamiento de responsabilidades - facilita ra reutilización de código común (frameworks, bibliotecas externas, bibliotecas propias) - facilita la activación/desactivación de código - reduce los conflictos al aplicar cambios (trabajando en equipo)	- mínimo tener algún import y export y un script de tipo "module" en el HTML - idealmente conseguir separar control y vista (lógica en un módulo, renderizado en otro), para respetar el Single Responsibility Principle.
Clean Code	Código al que volveremos en el futuro y sobre el que trabajarán otras personas no para la máquina. De modo que tiene que ser fácil de leer, entener y evolucionar.	- mínimo, que el código sea legible - nombres (de variebles, constantes y funciones) semánticos - funciones que tengan un único propósito, breves y fáciles de leer - idealmente autoformateado (eslint y otro)
Iteradores		- mínimo, que se use for...of al menos una vez - idealmente, buscar oportunidades para abstraer. Ejemplo: en vez de sacar 3 posible respuestas incorrecta al juego, crear un iterador de respuestas (que eliminara respuestas inválidas), y luego capturar 3 con otra funcion tipo .take(3)
Uso de asincronía		- mínimo: usar await al menos una vez en un sitio donde haga falta (promises, fetch etc) - idealmente, linearizar todas las funciones asíncronas para que el flujo en el tiempo sea evidente; marcar todas las funciones asincronas cuando lo sean (en lugar de devolver promises); aprovechar concurrencia de varios fetch a la vez
Presentación		- mínimo que se vea el trabajo ejecutado - idealmente que la audiencia tenga oportunidad de aprendizaje
cobertura de la funcionalidad requerida en la letra		- mínimo, haber intentado cubrir todo lo pedido - idealmente (para 9) cubrir el 100% de la funcionalidad) - para 10: aplicar soluciones creativas para extender la funcionalidad