

























JS Concepts

 Concept	 Plain English Explanation
Closure	A function that remembers variables from its outer scope 
Hoisting	JS moves declarations to the top of the file 
Event Loop	Handles async tasks behind the scenes (like setTimeout) 
Callback	A function passed into another function to be called later 
Promise	A value that will be available later (async placeholder) 
async/await	Cleaner way to write async code instead of chaining .then() 
Currying	Break a function into smaller, chained functions 
IIFE	Function that runs immediately after it's defined 
Prototype	JS's way of sharing features across objects (object inheritance) 
This	Refers to the object currently calling the function 
Debounce	Delay a function until user stops typing or clicking 
Throttle	Limit how often a function can run in a time frame 
Lexical Scope	Inner functions have access to outer function variables 

Garbage Collection	JS automatically frees up unused memory 🧹
Shadowing	A variable in a smaller scope overwrites one in a larger scope 🌑
Callback Hell	Nesting many callbacks leads to messy code 😵
Promise Chaining	Using .then() repeatedly to handle multiple async steps 📌
Microtask Queue	Where promises get queued (after main code, before rendering) 🧪
Execution Context	The environment in which JS runs each piece of code 🧠
Call Stack	A stack where function calls are managed 🧱
Temporal Dead Zone	Time between variable declaration and initialization with let/const 🚫
Type Coercion	JS automatically converts types (e.g., "5" + 1 → "51") 🔄
Falsy Values	Values treated as false (0, "", null, undefined, NaN, false) 🚫
Truthy Values	Values treated as true ("a", [], 1, {}) ✅
Short-circuiting	JS skips the rest if result is already known (`true`)
Optional Chaining	?. safely accesses deep properties without crashing ✨
Nullish Coalescing	?? gives first non-null/undefined value 🧰

Set & Map	Set = unique values, Map = key-value pairs 
Memory Leak	When unused data stays in memory and slows the app 
Event Delegation	One event listener handles many elements efficiently 
Immutability	Avoid changing existing values — return new ones instead 
Spread Operator	... to copy/expand arrays or objects 
Destructuring	Quickly extract values from arrays/objects 
Rest Parameters	...args collects all remaining arguments into one 
typeof	Tells you the data type of a variable 
Instanceof	Checks if an object belongs to a certain class/type 
Reference vs Value	Objects/arrays are referenced, primitives are copied 