here's your **print-friendly comparison table** so you can drop it straight into your backend/JS interview prep kit.
This distills *declaration* vs *expression* with all the key differences:

Feature	**Function Declaration**	**Function Expression (var)**	**Function Expression (let/const)**
Hoisting – Name	✅ Yes — identifier is hoisted	✅ Yes — variable name hoisted	✅ Yes — variable name hoisted
Hoisting – Value/Body	✅ Yes — full body hoisted	❌ No — value set at runtime	❌ No — value set at runtime
Pre-declaration Call	✅ Works anywhere in scope	❌ `TypeError: … is not a function` (value is `undefined`)	❌ `ReferenceError` (TDZ)
Temporal Dead Zone (TDZ)	❌ None	❌ None (initialized to `undefined`)	✅ Yes — access before assignment → error
Recursion Support	✅ Yes — name visible in entire scope	✅ Yes if **named expression**; else use variable name	✅ Yes if **named expression**; else use variable name
Typical Usage	Library functions, utility functions	Inline callbacks, assignable logic	Block-scoped inline logic, safer bindings

📝 Quick Rule-of-Thumb

- **Declaration** → "always ready" anywhere in scope.
- **Expression (var)** → "name is ready, but body arrives later."
- **Expression (let/const)** → "nothing is ready until that exact line."