(1) tutte le funzioni sono closure

	f ARROW ⇒	g NON ARROW
• prototype	• ∅, non posso aggiungere metodi o proprietà a f	• Object, posso aggiungere metodi e proprietà a g.
• this	• context definition • non sovrascrivibile f.call(x, ...), f.bind(x)	• context execution • sovrascrivibile g.call(o, ...), g.bind(o)
• scope	• binding definition (*)	• binding definition (*)
• hoisted	• dipende const, let no function sì	• dipende const, let no function sì
		g.call(o) eseguo g nel contesto di o let g_2 = g.bind(o) monto g su o. mi torna g montata su o.

Esempi utili:

1) uniqueInt.count = 0; (la 4) è meglio)
function uniqueInt() { return uniqueInt.count++; }

2) function factorial(n) { ... }
factorial[1] = 1 // initialize cache
factorial(6) // = 720 comput
factorial[5] // = 120 from cache

3) Namespace
function chunkNamespace { // cose } chunkNamespace();
(function() { // cose }());
// eseguito subito.

4) Closure (vedi 1)
let uniqueInt = (function() { ← Closure, inacessibile da esterno
let counter = 0;
return function() { return counter++; };
}());

uniqueInt() // 0
uniqueInt() // 1