

keyword  
const  
let  
var

değişken adı  
sayıTopla = function (a, b) {  
değişken  
return a + b

}

<lg(sayıTopla(5, 7)) → invok.

<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>
const	topla =	(a, b)	⇒	a + b	
expression ✓		parameter listesi	fonk. gövdesi	(b'ny' alanı)	
		2 ↓ 6	↓	2 + 6	8
			fonk. gövdesi'nin baş b'diği yer		

topla(2, 6)

8

ARROW

16

4

const kareAl = (num) => num \* num ✓

kareAl(4)

4 \* 4

16

const yaz = () => dg ("Hello")

II yaz()

suche, return

const hesapla = (x, y) => {

const toplam = x + y → 1.

const carpim = x \* y → 2.

const sonuc = x \* y / (x + y) → 3.

return sonuc

}

Recursion



const fonk = () => {

divide

fonk()

→ kontrol

}

Sonsuz dongü

→ fonk()

$$n! = n \cdot (n-1)!$$

$\downarrow$ 
 $\downarrow$

$$f(n) \Rightarrow n \cdot f(n-1)$$

$$n \cdot (n-1) \cdot f(n-2)$$

Özyinelemeli bir matematiksel  
 işlemimizi kodlamak doğru yerine (kötü)  
 rekursif kullanmak daha (kırk veya kırk)

1 1 2 3 5 8 13 21

*(Arrows indicate recursive steps: 2 from 1 and 1; 3 from 2 and 1; 5 from 3 and 2; 8 from 5 and 3; 13 from 8 and 5; 21 from 13 and 8)*

$$f(n) = f(n-1) + f(n-2)$$

$\downarrow$ 
 $\downarrow$

$$f(2) = 1$$

$$f(3) = 2$$

$$f(n-2) + f(n-3)$$