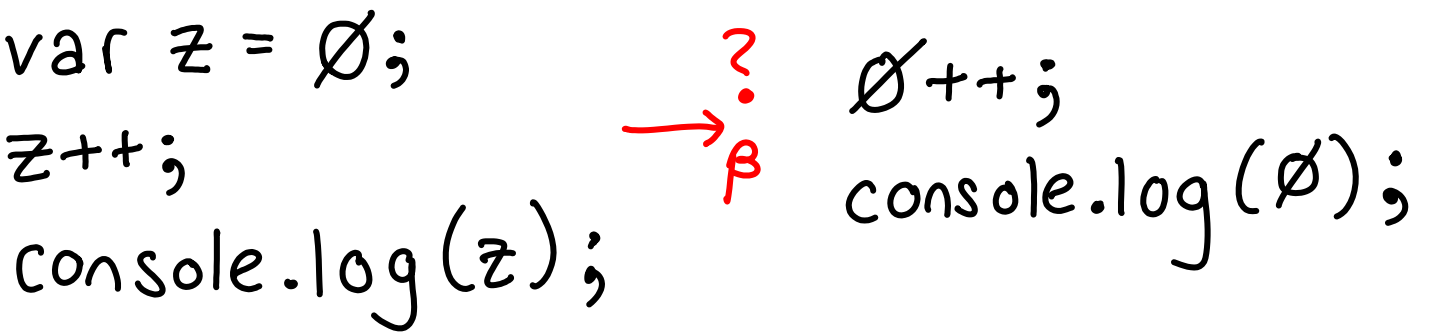
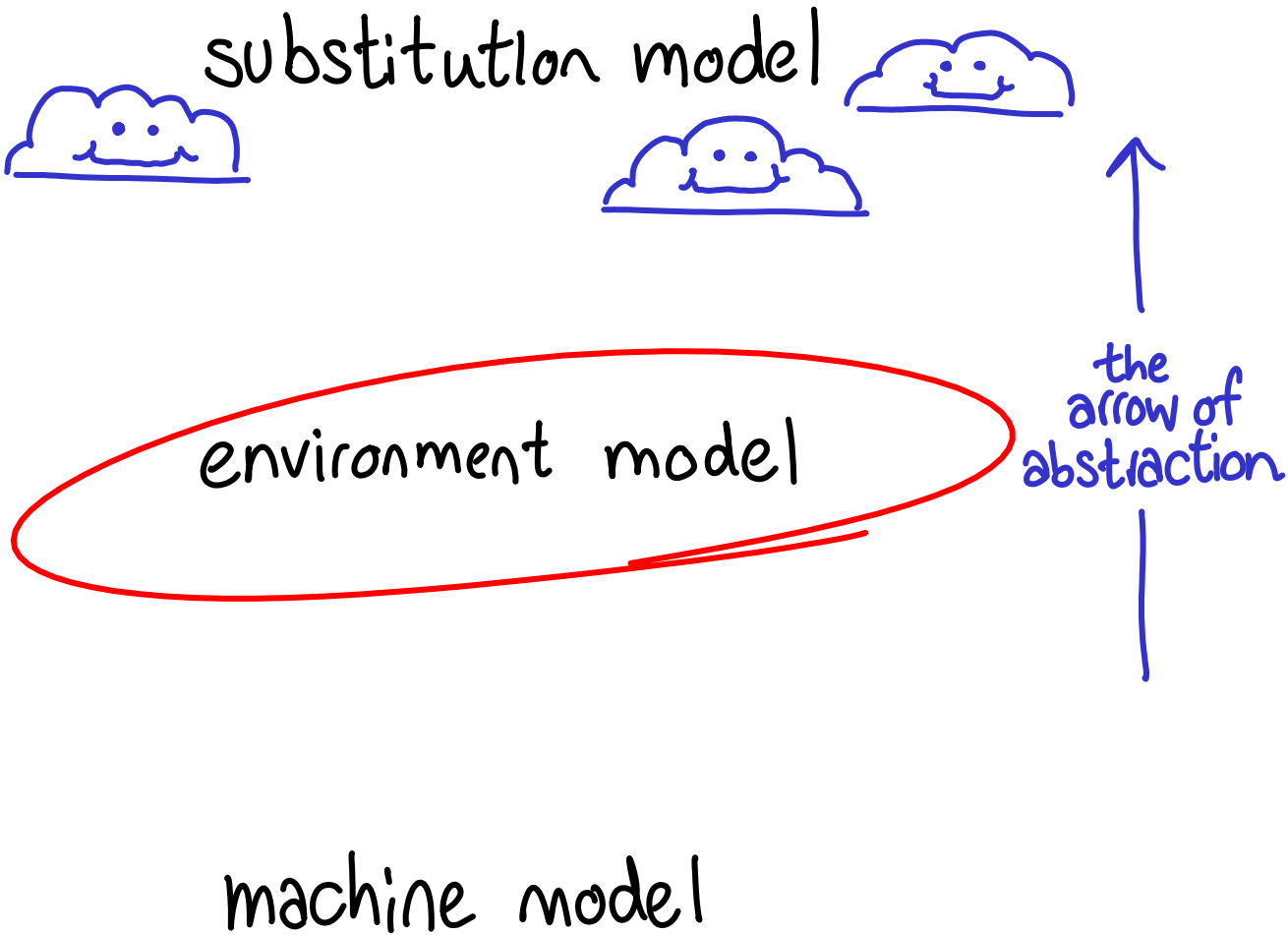

DOSES
Breath Feels Fresh
Up To 5X Longer*
L'haleine reste fraîche
jusqu'à 5 fois plus longtemps*
*vs. Brushing Alone/vs brossage des dents seulement
Scope®
Edward Z. Yang
98890288
750 mL

$$(\lambda x.\, f\, x\, x)\,(\lambda y.\, z)$$

$$\rightarrow_{\beta} \quad f\,(\lambda y.\, z)\,(\lambda y.\, z)$$


var z = Ø;
z++;
console.log(z);
?
β
Ø++;
console.log(Ø);


substitution model
environment model
the arrow of abstraction
machine model

The Environment Model by example

EX1 Anatomy of a scope

EX2 First-order functions

EX3 Free variables

EX4 Higher-order functions

control links

access links

The Environment Model by example

EX1 Anatomy of a scope

EX2 First-order functions

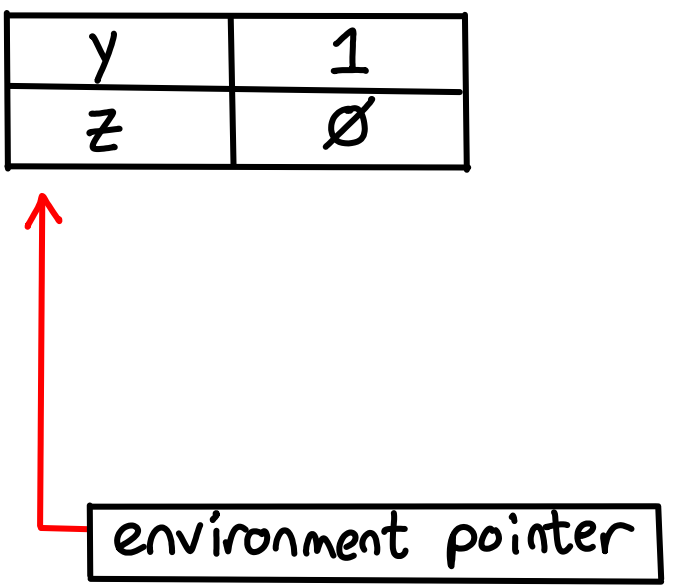
EX3 Free variables

EX4 Higher-order functions

control links

access links

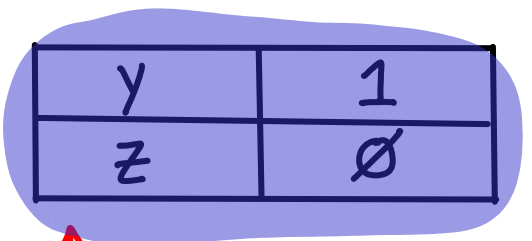
EX1 Anatomy of a scope




```
var y = 1;
var z = Ø;
z++;
console.log(z);
```

EX1 Anatomy of a scope

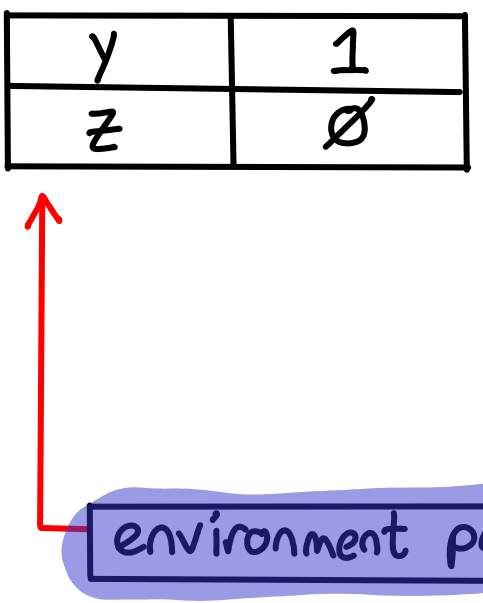
Environment/Activation Record



environment pointer

```
var y = 1;
var z = ø;
z++;
console.log(z);
```

EX1 Anatomy of a scope

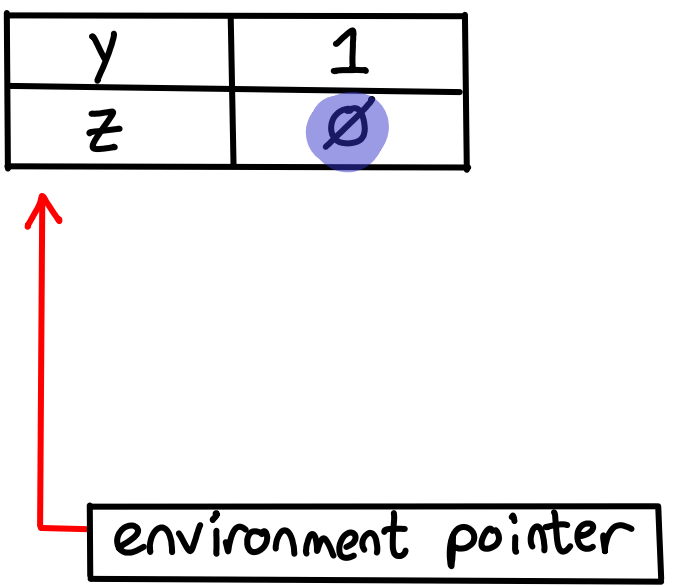



```
var y = 1;
var z = Ø;
z++;
console.log(z);
```

e.g. %esp on x86-32

EX1 Anatomy of a scope

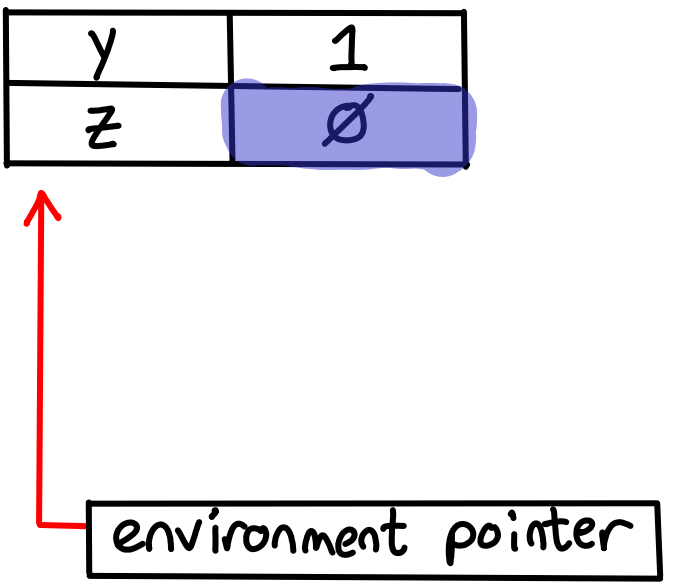
r-value: the "value"




```
var y = 1;
var z = Ø;
z++;
console.log(z);
```

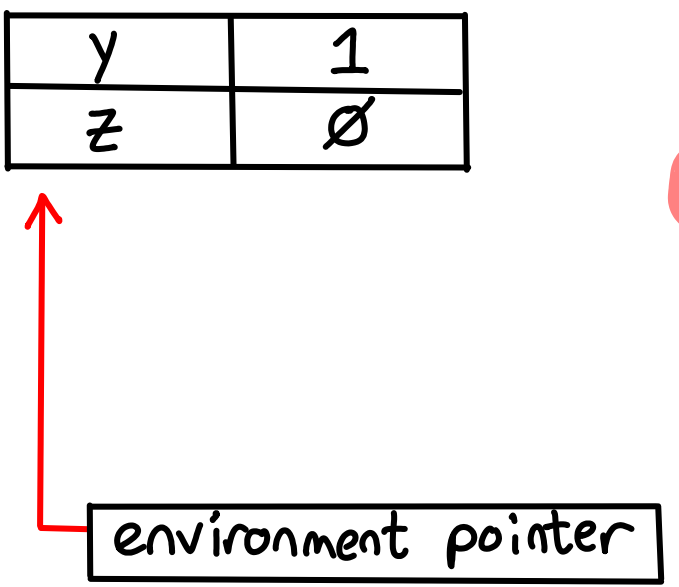
EX1 Anatomy of a scope

l-value: the location



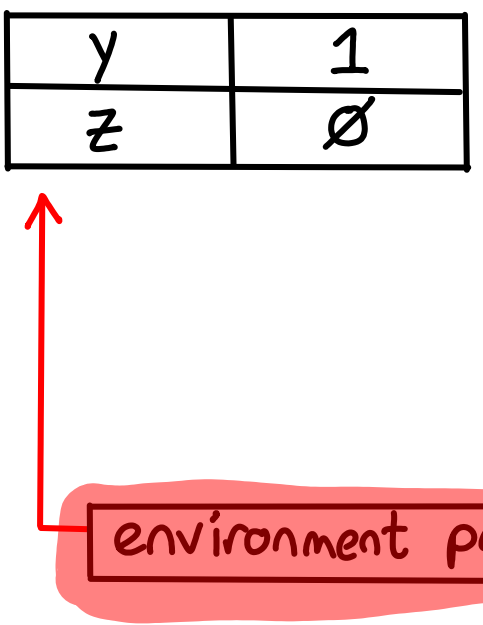

```
var y = 1;
var z = Ø;
z++;
console.log(z);
```

EX1 Anatomy of a scope



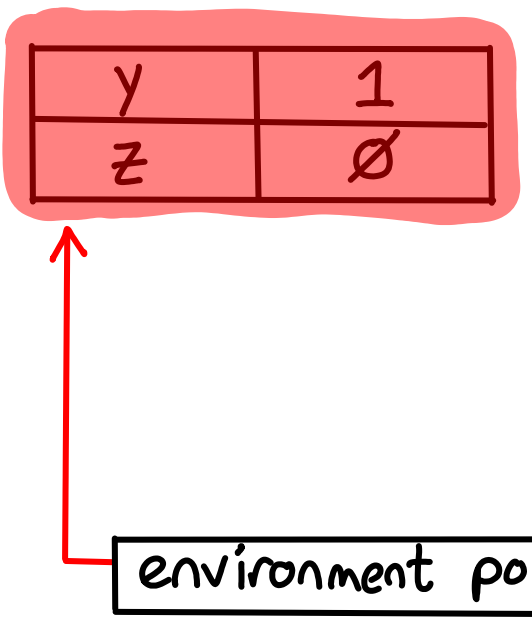

```
var y = 1;
var z = Ø;
z++;
console.log(z);
```

EX1 Anatomy of a scope



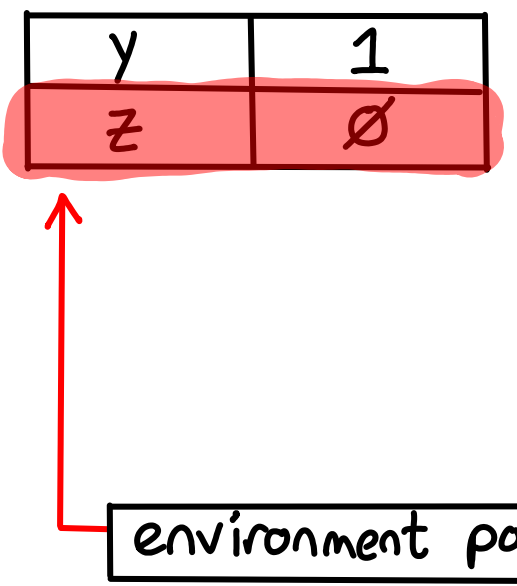

```
var y = 1;
var z = ø;
z++;
console.log(z);
```

EX1 Anatomy of a scope



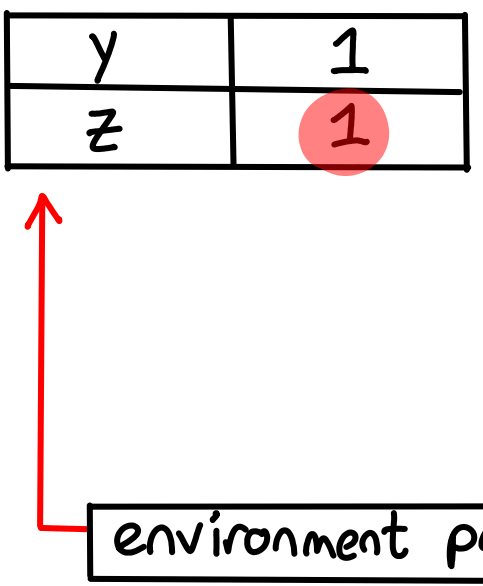

```
var y = 1;
var z = Ø;
z++;
console.log(z);
```

EX1 Anatomy of a scope



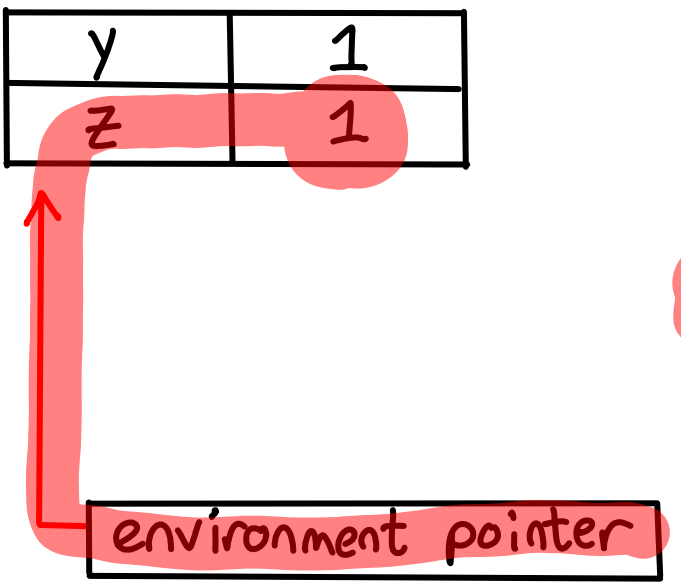

```
var y = 1;
var z = Ø;
z++;
console.log(z);
```

EX1 Anatomy of a scope



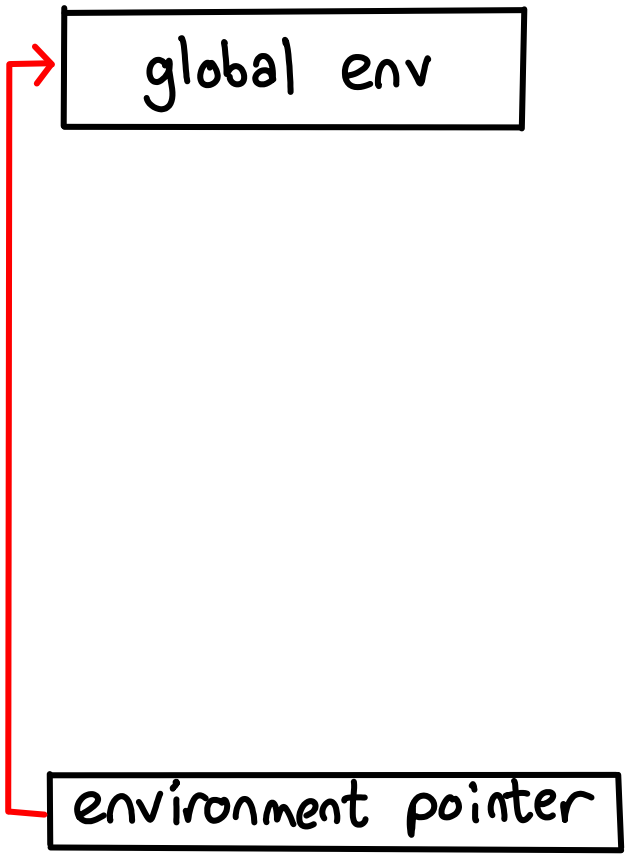

```
var y = 1;
var z = 0;
z++;
console.log(z);
```

EX1 Anatomy of a scope



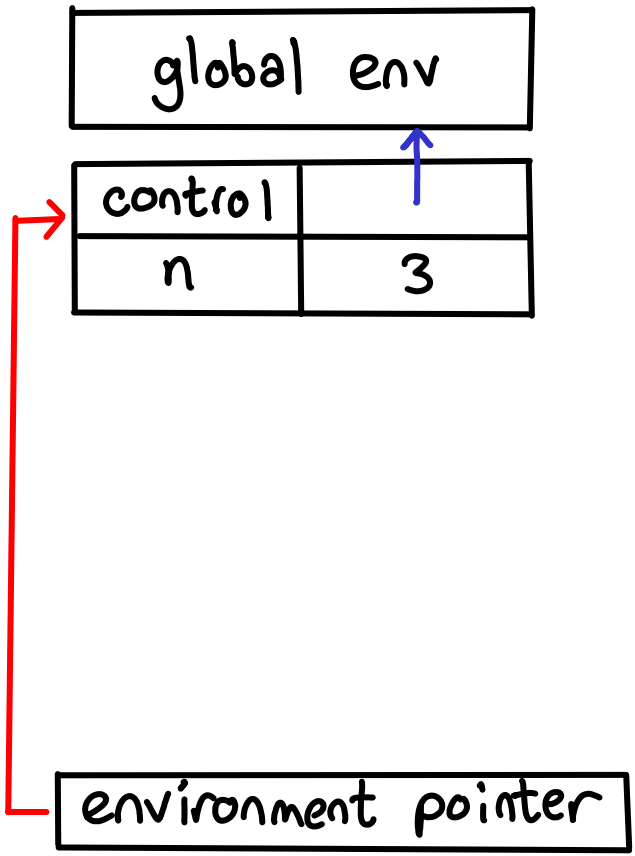

```
var y = 1;
var z = 0;
z++;
console.log(z);
```

EX2 First-order functions



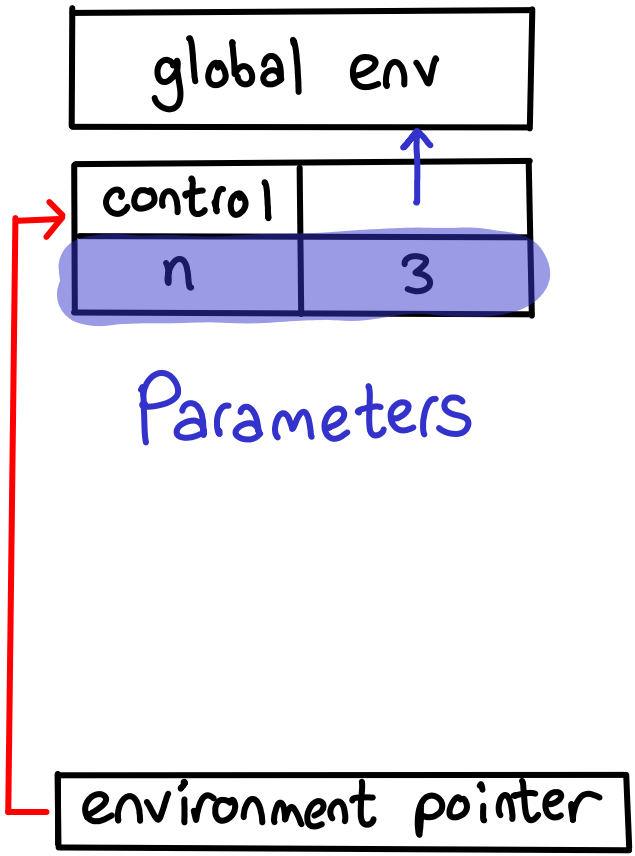

```
function fact(n) {
    if (n ≤ 1) {
        return 1;
    } else {
        return n * fact(n-1)
    }
}
fact(3);
```

EX2 First-order functions



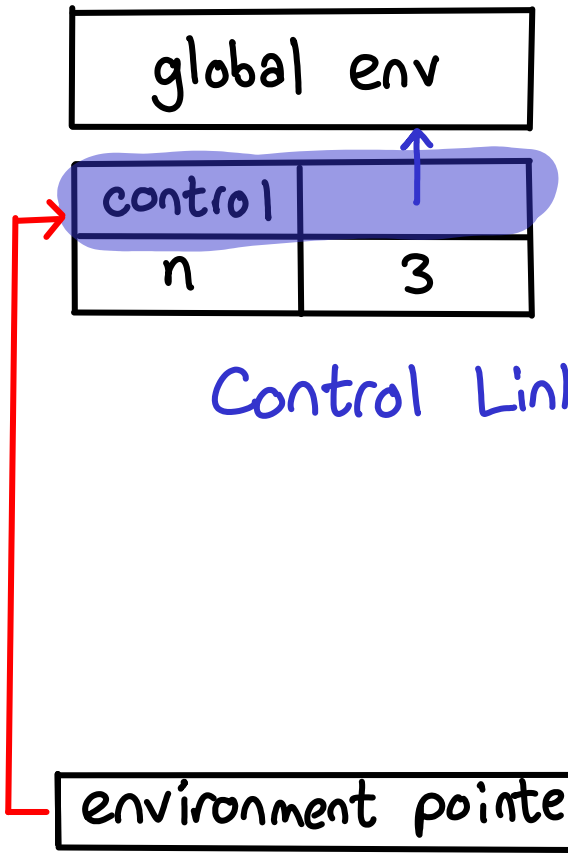

```
function fact(n) {
    if (n ≤ 1) {
        return 1;
    } else {
        return n * fact(n-1)
    }
}
fact(3);
```

EX2 First-order functions



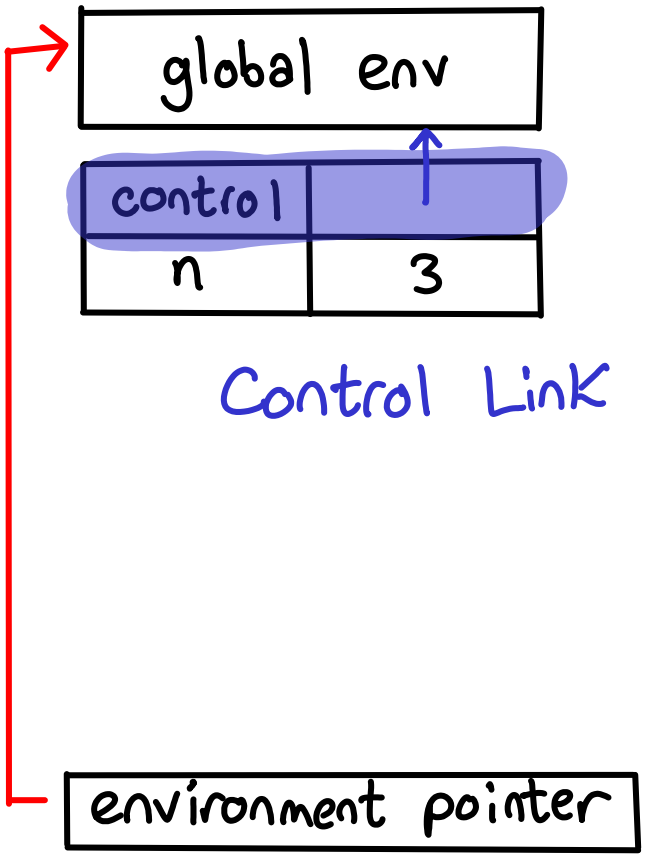

```
function fact(n) {
  if (n ≤ 1) {
    return 1;
  } else {
    return n * fact(n-1)
  }
}
fact(3);
```

EX2 First-order functions



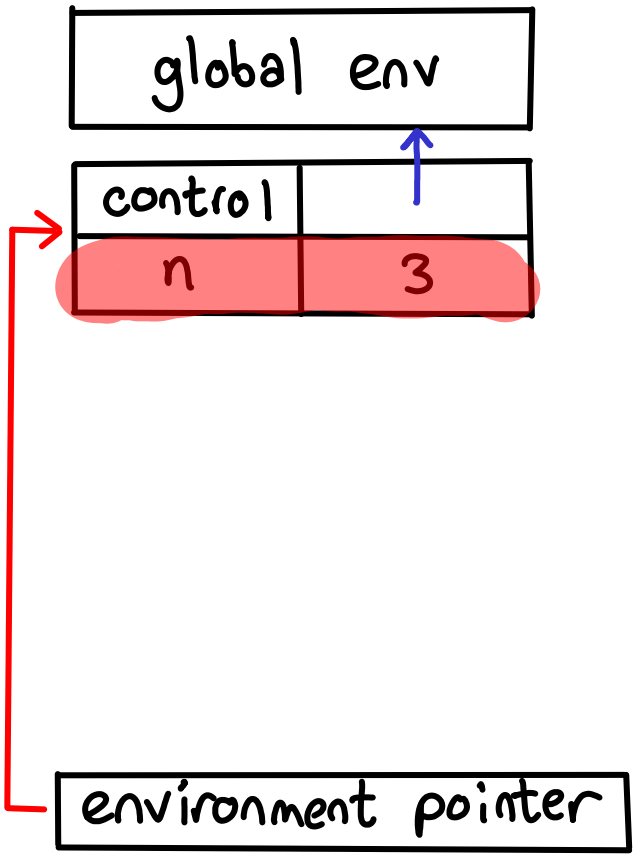

```
function fact(n) {
  if (n ≤ 1) {
    return 1;
  } else {
    return n * fact(n-1)
  }
}
fact(3);
```

EX2 First-order functions



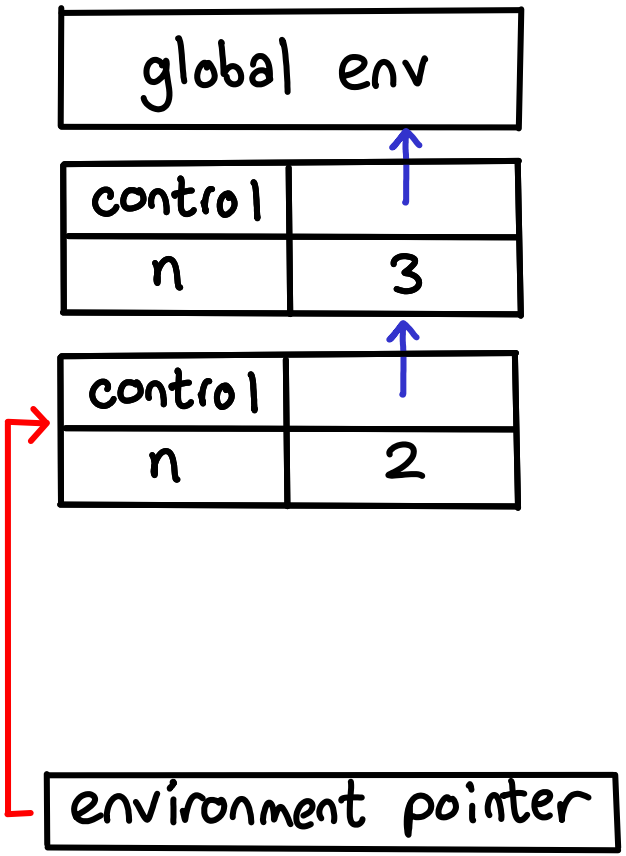

```
function fact(n) {
  if (n ≤ 1) {
    return 1;
  } else {
    return n * fact(n-1)
  }
}
fact(3);
```

EX2 First-order functions



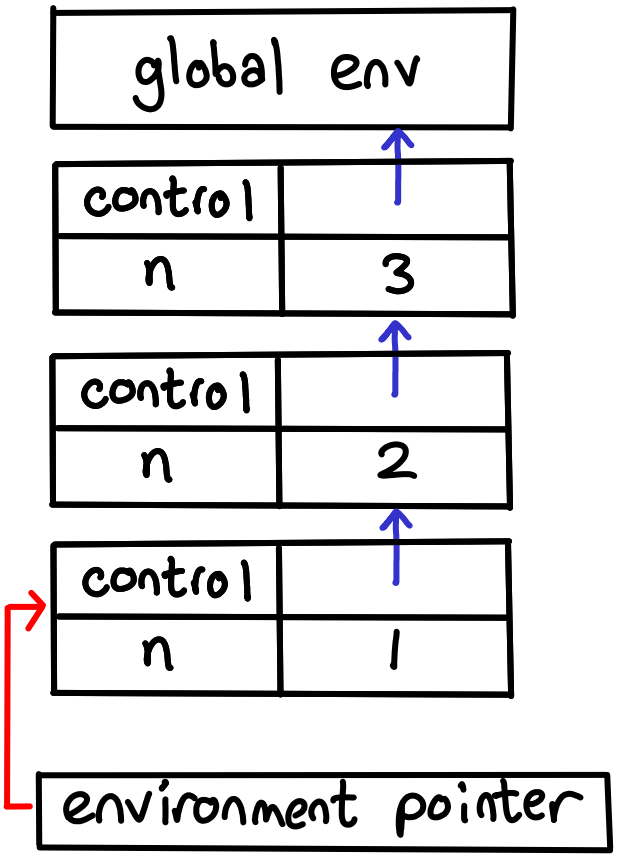

```
function fact(n) {
    if (n ≤ 1) {
        return 1;
    } else {
        return n * fact(n-1)
    }
}
fact(3);
```

EX2 First-order functions



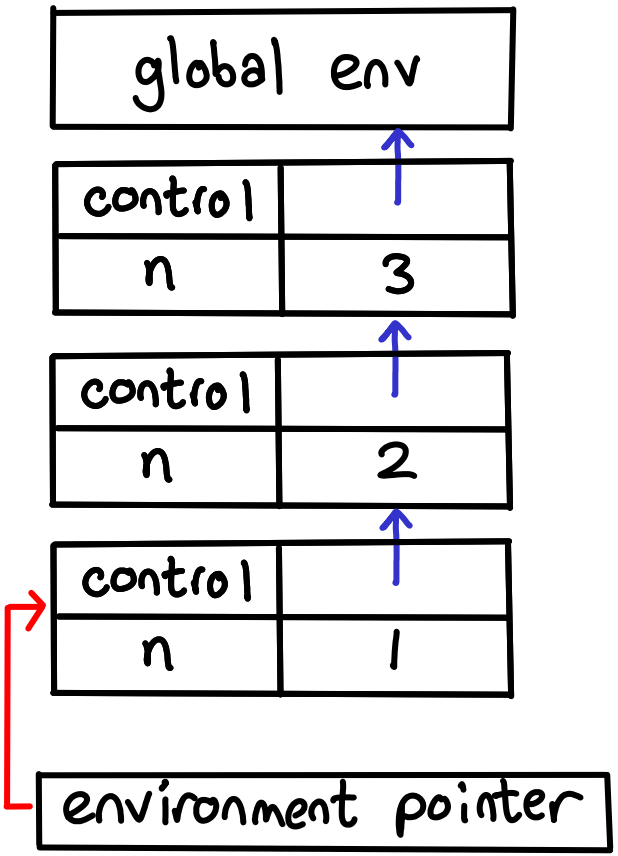

```
function fact(n) {
    if (n ≤ 1) {
        return 1;
    } else {
        return n * fact(n-1)
    }
}
fact(3);
```

EX2 First-order functions



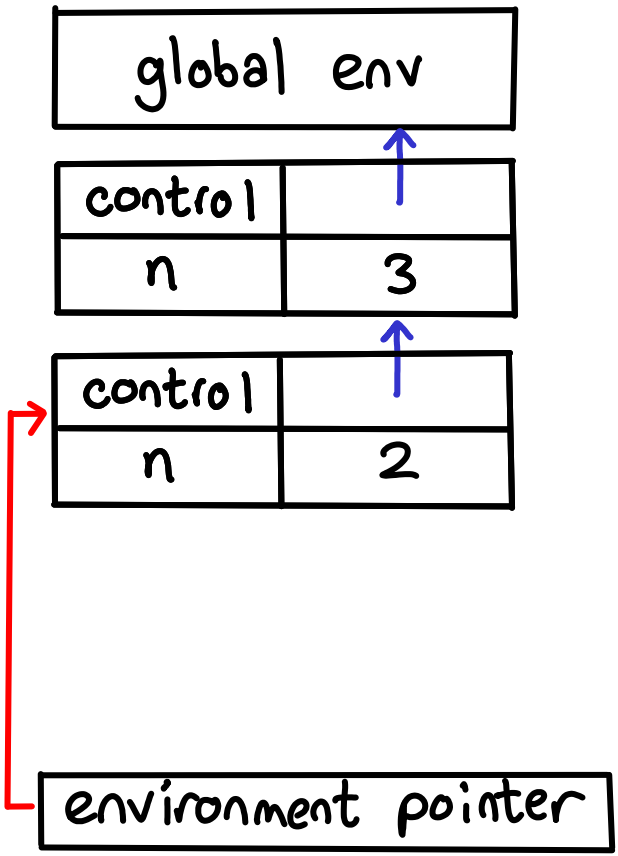

```
function fact(n) {
    if (n ≤ 1) {
        return 1;
    } else {
        return n * fact(n-1)
    }
}
fact(3);
```

EX2 First-order functions




```
function fact(n) {
    if (n ≤ 1) {
        return 1;
    } else {
        return n * fact(n-1)
    }
}
fact(3);
```

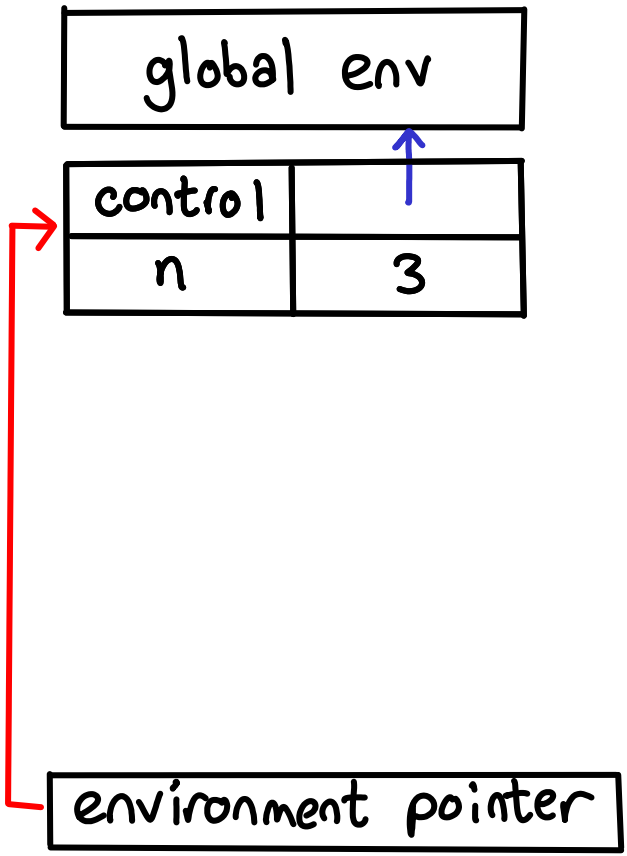
EX2 First-order functions




```
function fact(n) {
  if (n ≤ 1) {
    return 1;
  } else {
    return n * fact(n-1)
  }
}
fact(3);
```

1

EX2 First-order functions



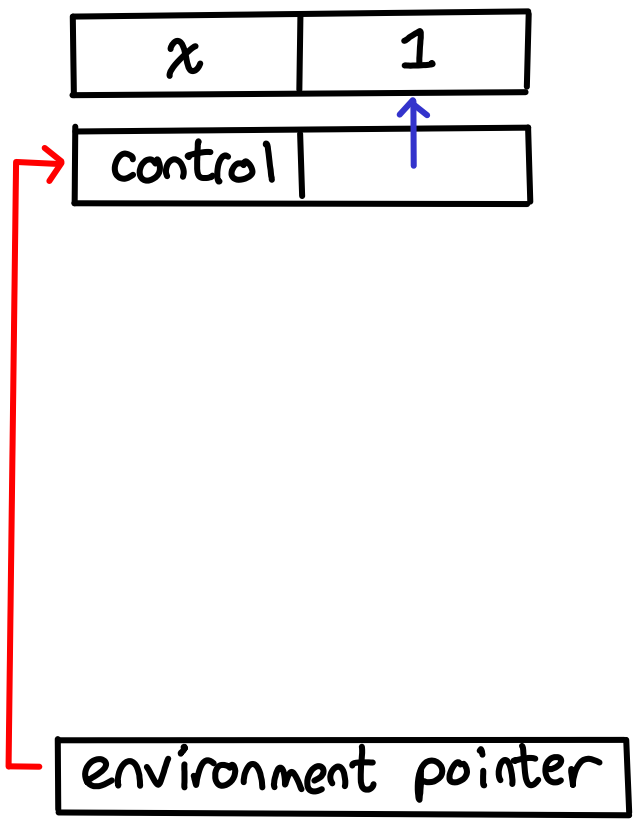

```
function fact(n) {
  if (n ≤ 1) {
    return 1;
  } else {
    return n*fact(n-1)
  }
}
fact(3);
```

2

Ex 3 Free variables

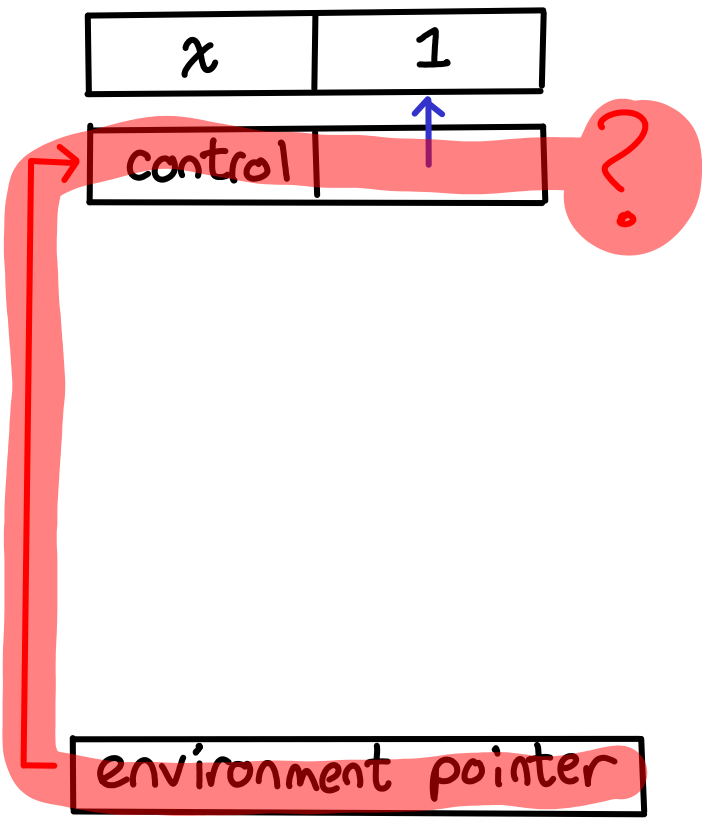

```
var x = 1;
function f() {
    console.log(x);
}
f();
```

Ex 3 Free variables



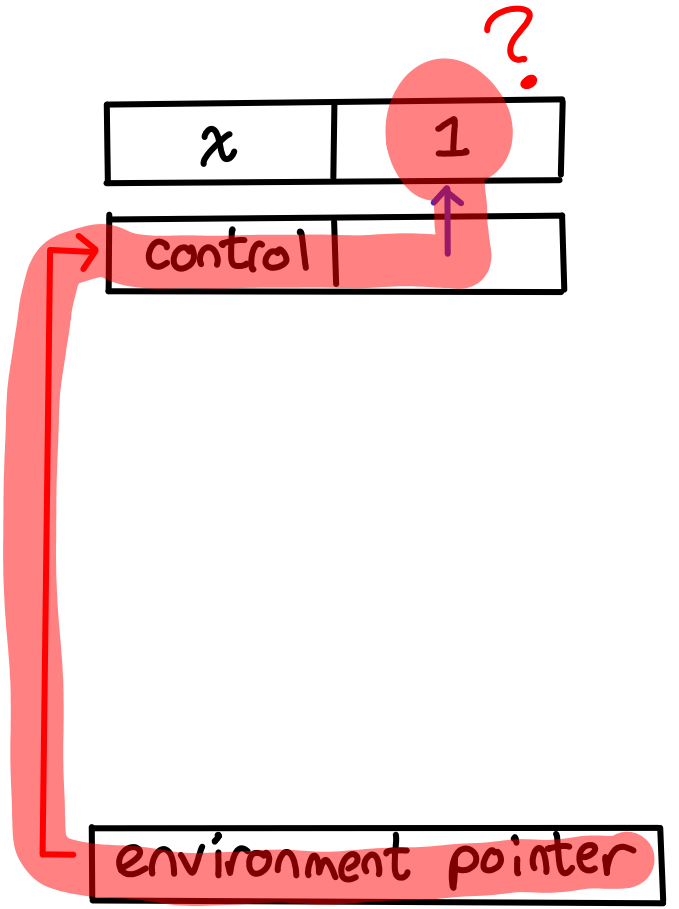

```
var x = 1;
function f() {
    console.log(x);
}
f();
```

Ex 3 Free variables



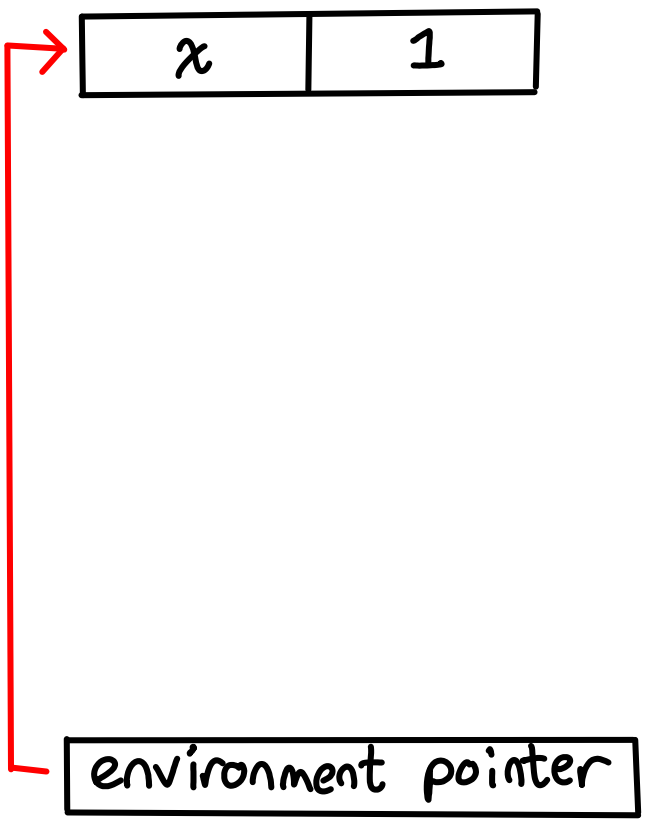

```
var x = 1;
function f() {
    console.log(x);
}
f();
```

Ex 3 Free variables



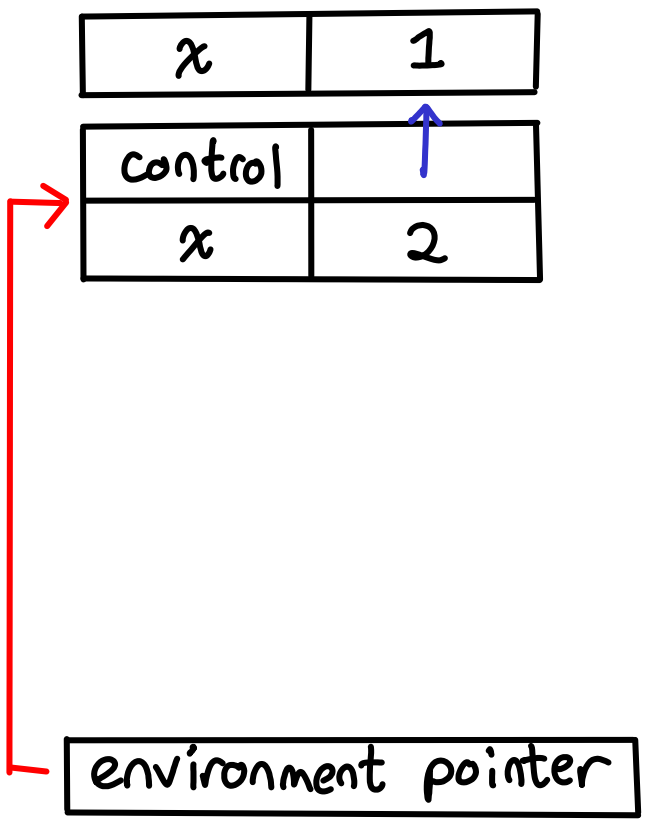

```
var x = 1;
function f() {
    console.log(x);
}
f();
```

Ex 3 Free variables



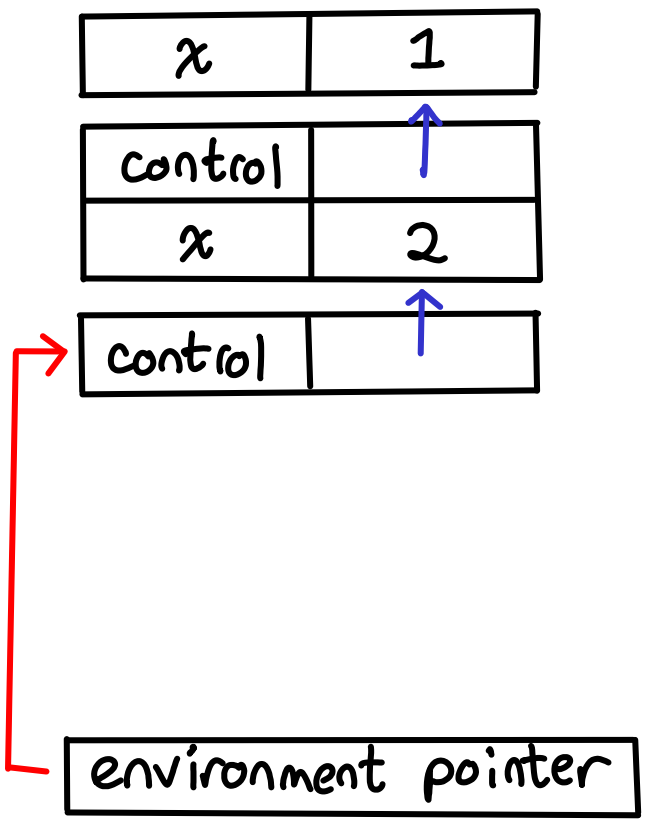

```
var x = 1;
function f() {
    console.log(x);
}
function g() {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables



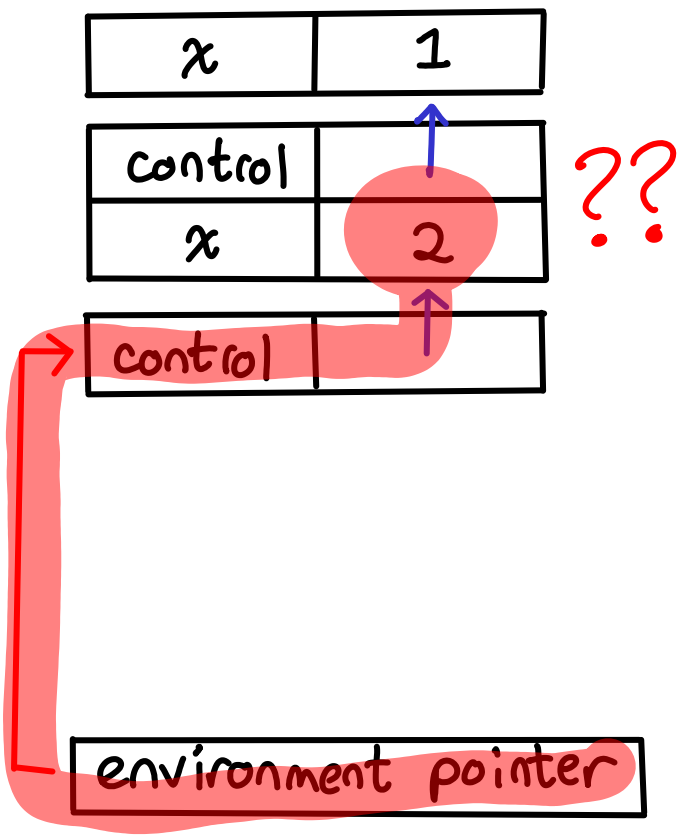

```
var x = 1;
function f() {
    console.log(x);
}
function g() {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables




```
var x = 1;
function f() {
    console.log(x);
}
function g() {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables




```
var x = 1;
function f() {
    console.log(x);
}
function g() {
    var x = 2;
    f();
}
g();
```

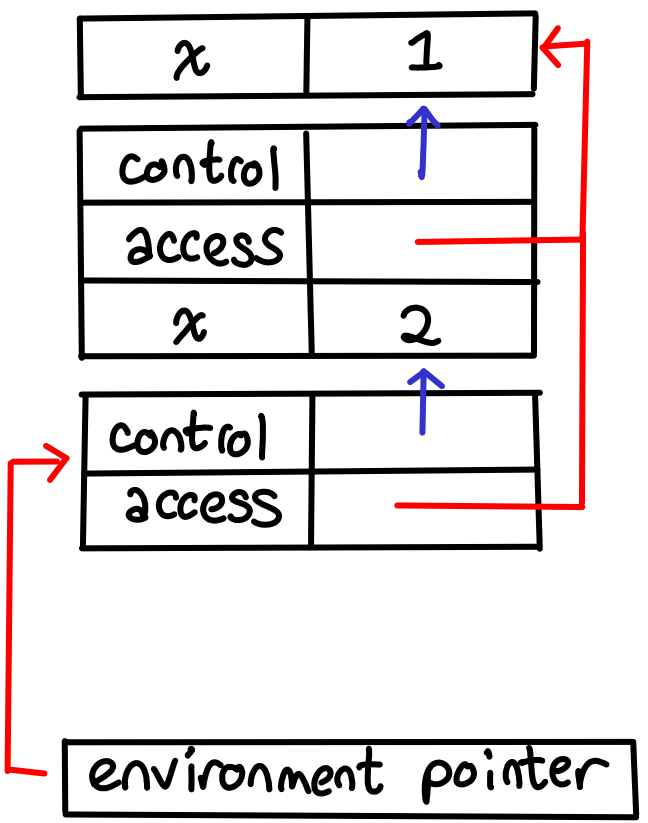
??

Congratulations:

You just invented

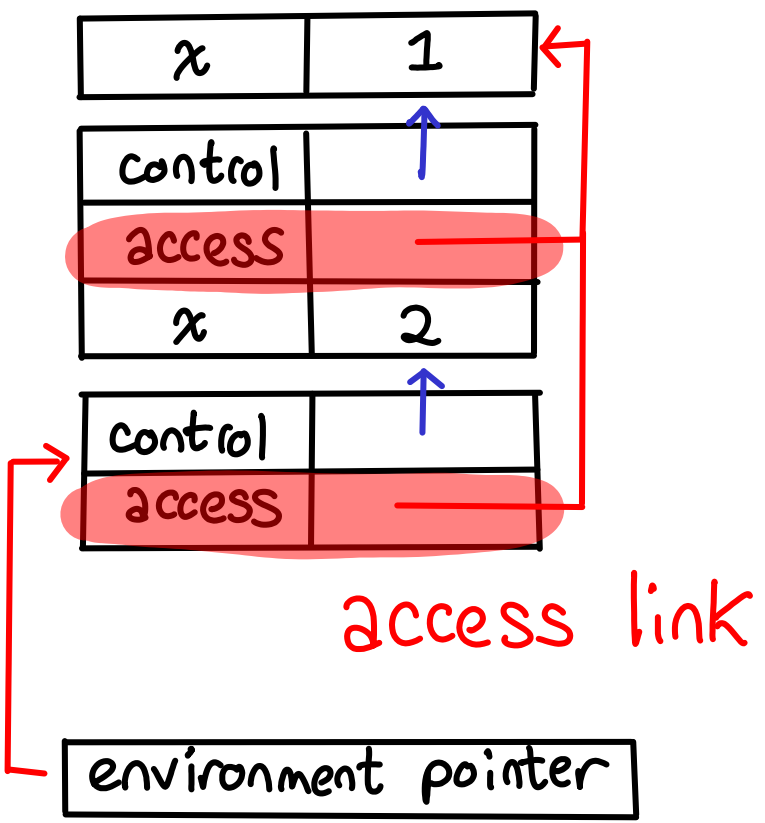
dynamic scoping

Ex 3 Free variables



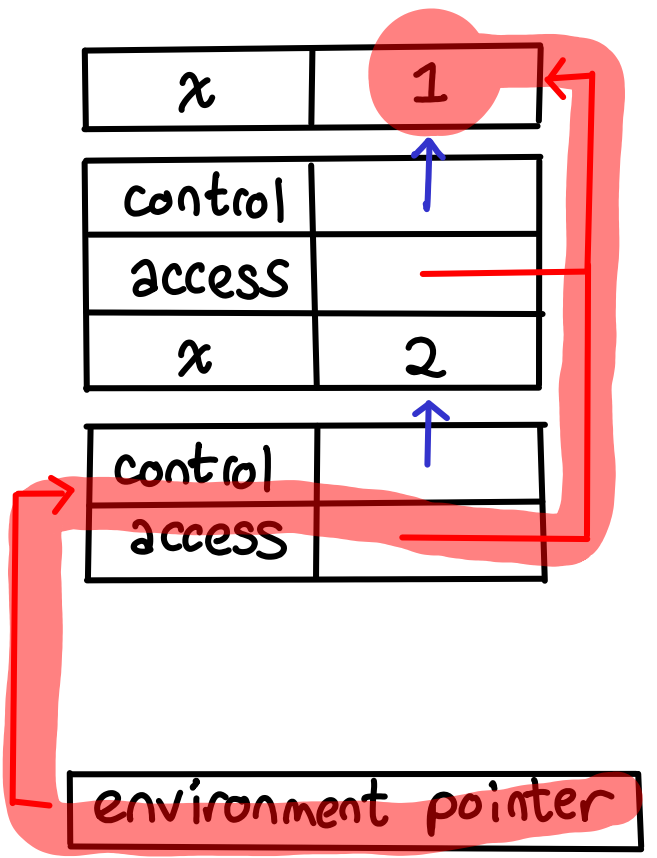

```
var x = 1;
function f() {
    console.log(x);
}
function g() {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables




```
var x = 1;
function f() {
    console.log(x);
}
function g() {
    var x = 2;
    f();
}
g();
```

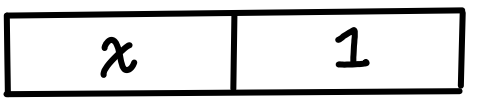
Ex 3 Free variables




```
var x = 1;
function f() {
    console.log(x);
}
function g() {
    var x = 2;
    f();
}
g();
```

But how do we know how to wire up the access links?

Ex 3 Free variables

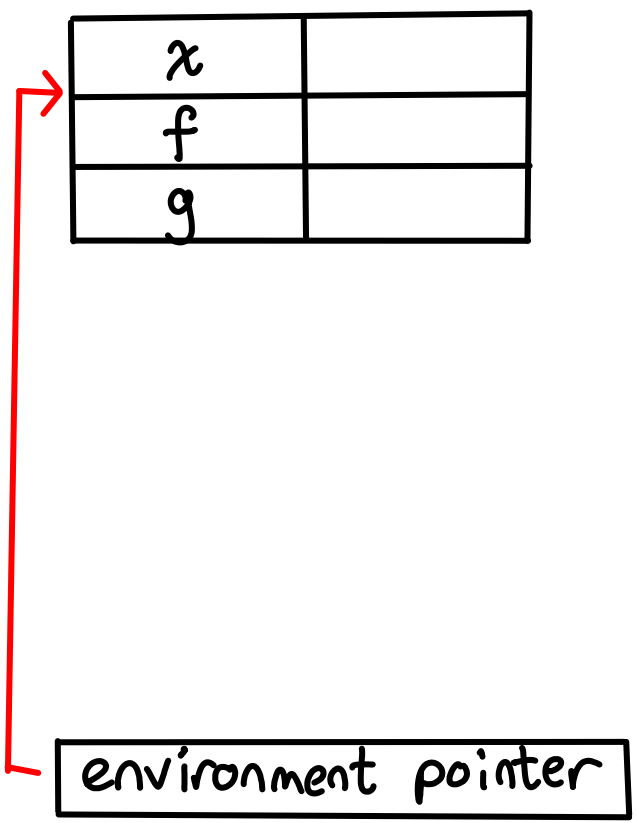



```
var x = 1;
function f() {
    console.log(x);
}
function g() {
    var x = 2;
    f();
}
g();
```

static?

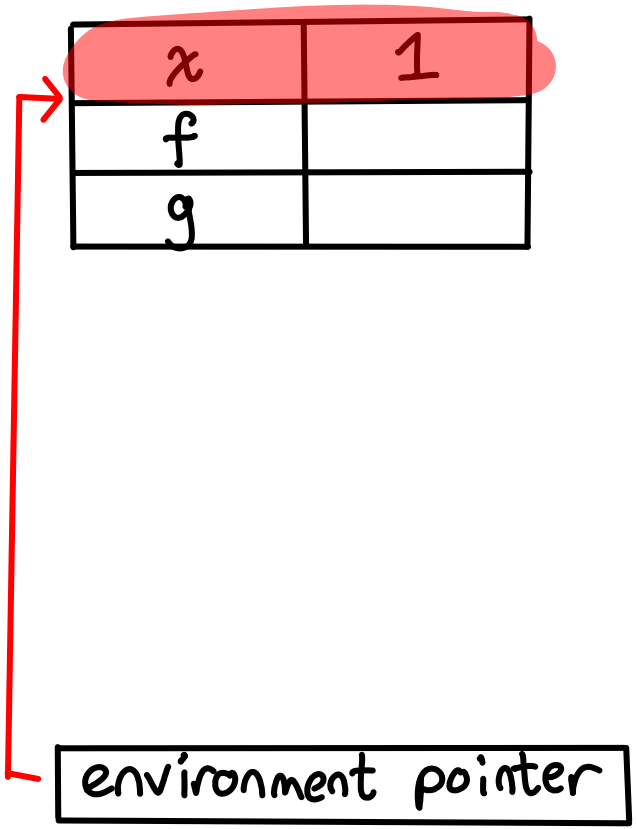
Functions are Data

Ex 3 Free variables



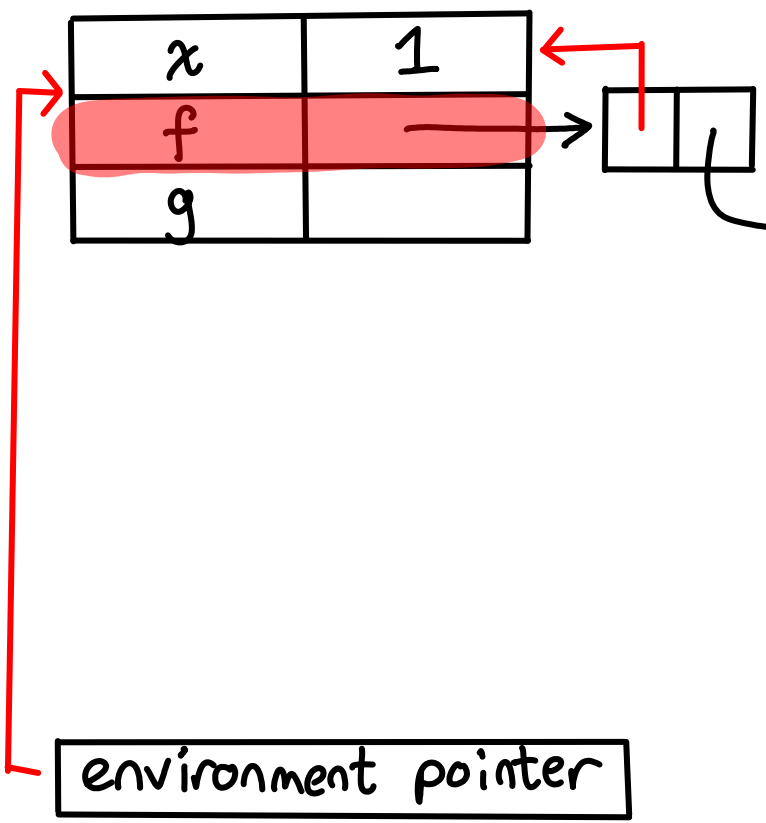

```
var x = 1;
var f = function () {
    console.log(x);
}
var g = function () {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables



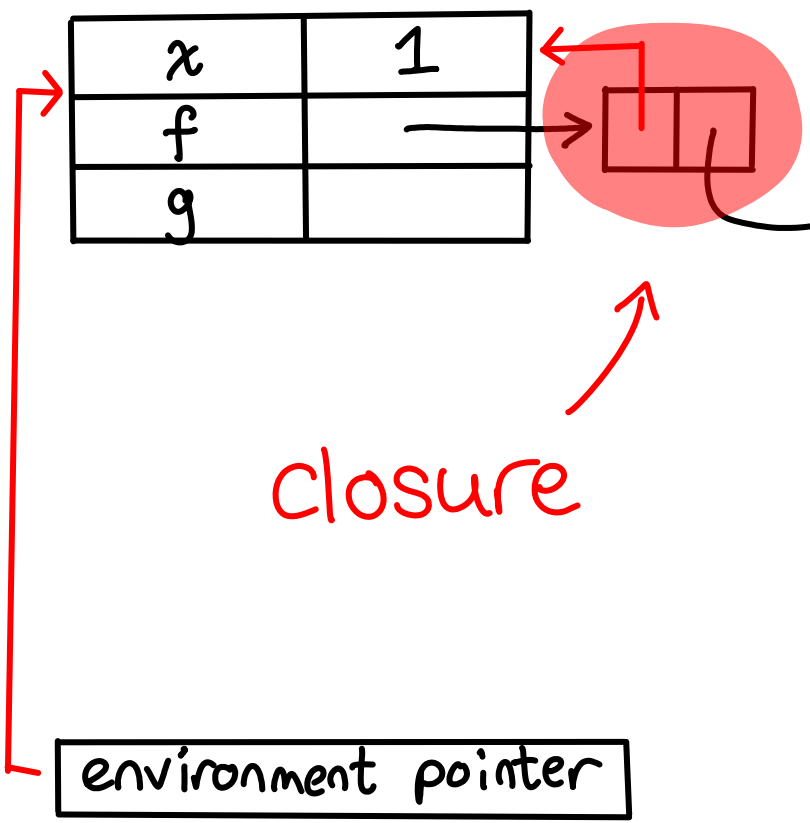

```
var x = 1;
var f = function () {
    console.log(x);
}
var g = function () {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables



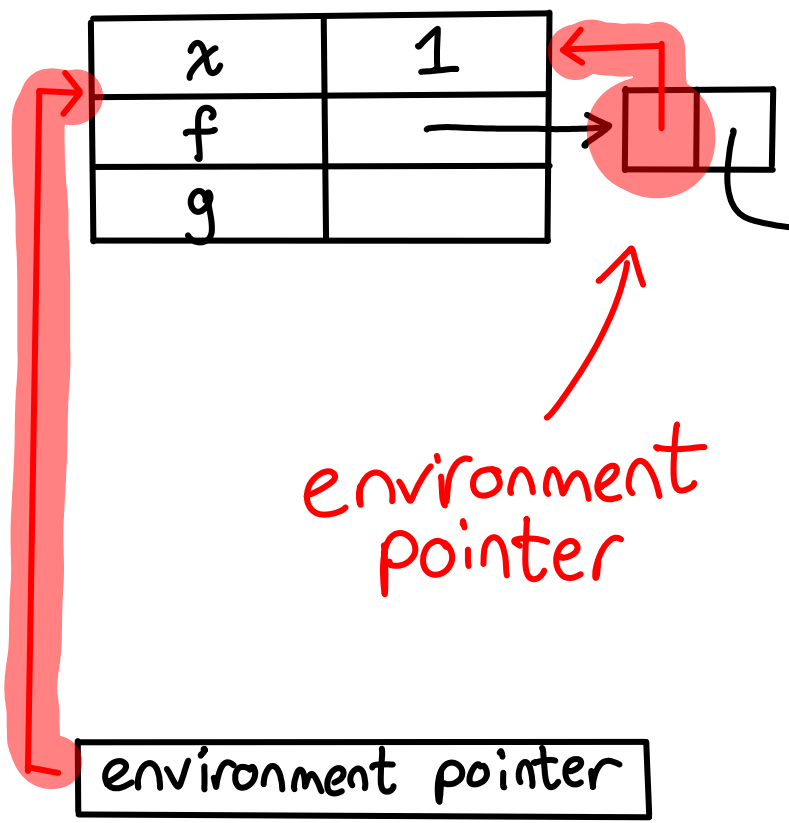

```
var x = 1;
var f = function () {
    console.log(x);
}
var g = function () {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables



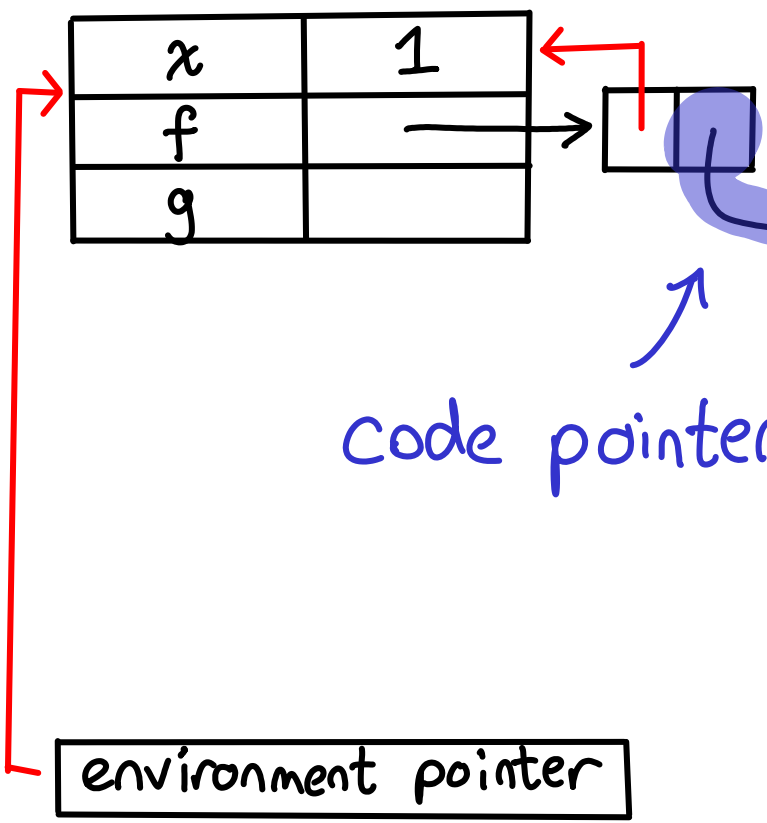

```
var x = 1;
var f = function () {
    console.log(x);
}
var g = function () {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables



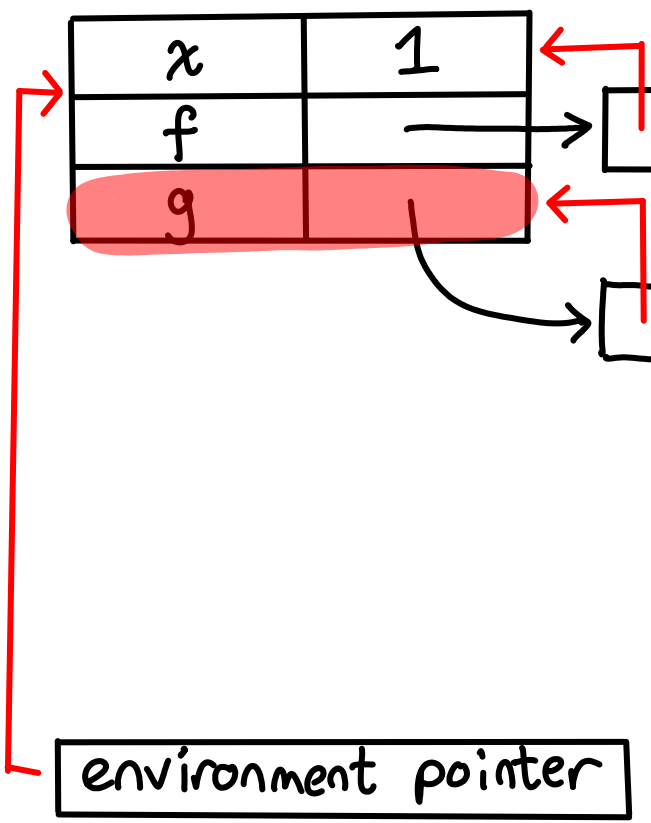

```
var x = 1;
var f = function () {
    console.log(x);
}
var g = function () {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables



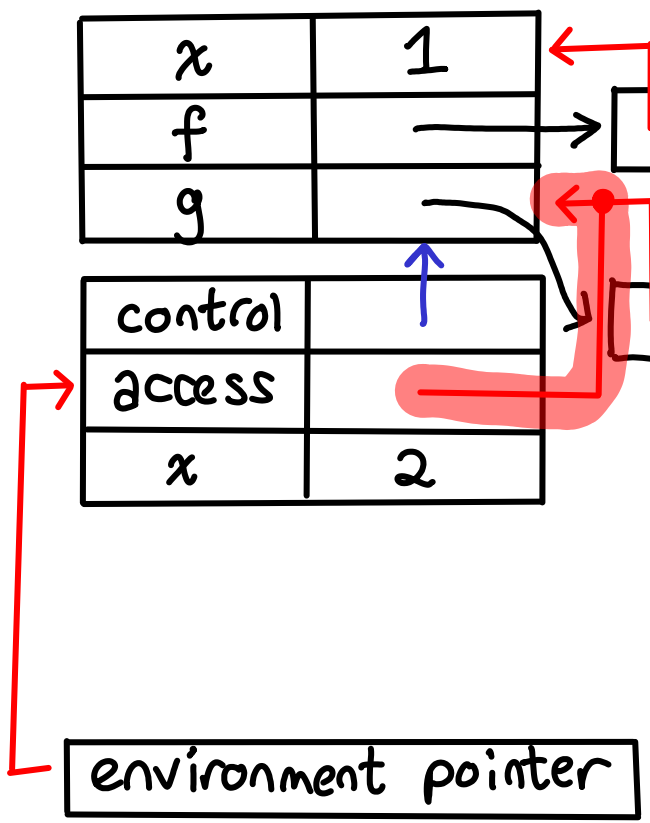

```
var x = 1;
var f = function () {
    console.log(x);
}
var g = function () {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables



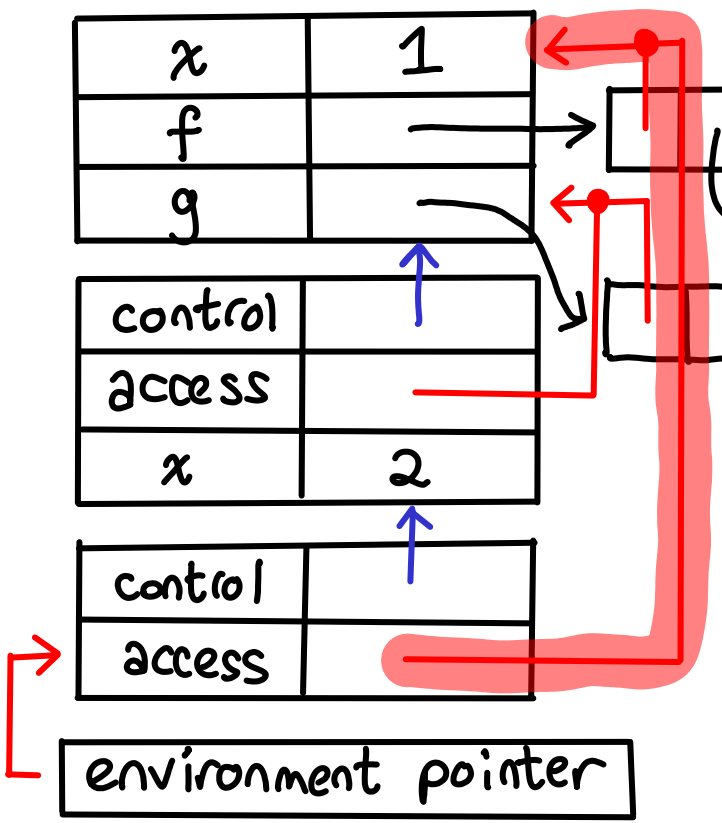

```
var x = 1;
var f = function () {
    console.log(x);
}
var g = function () {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables



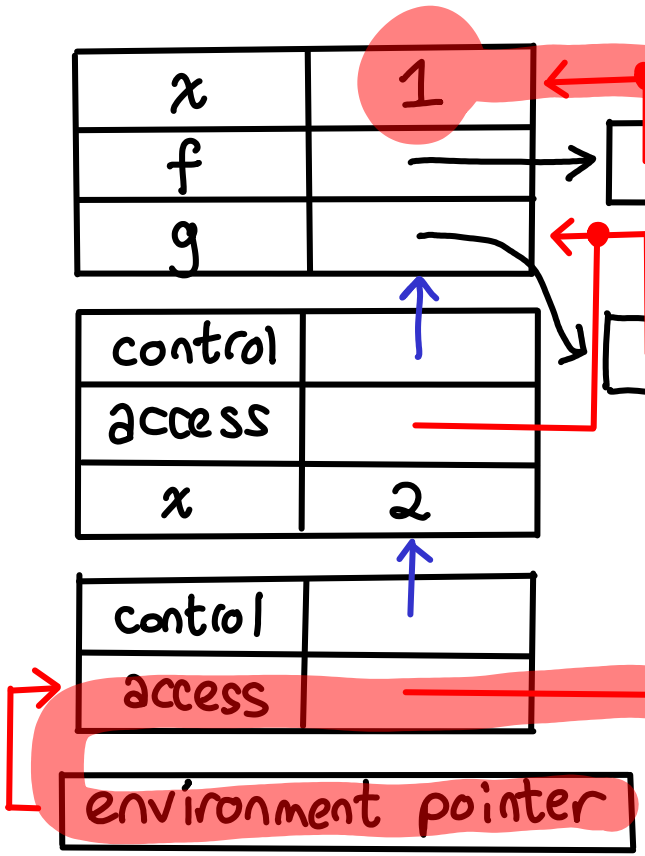

```
var x = 1;
var f = function () {
    console.log(x);
}
var g = function () {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables



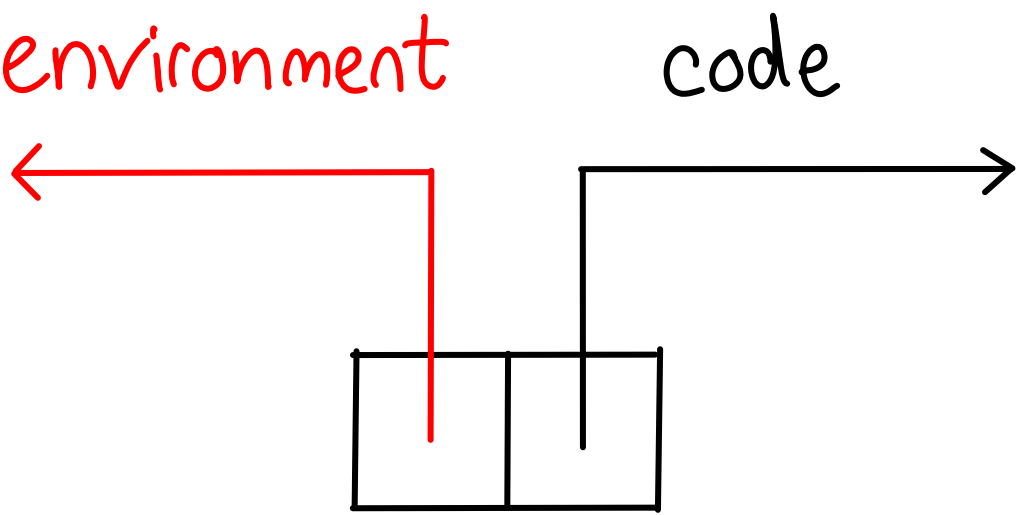

```
var x = 1;
var f = function () {
    console.log(x);
}
var g = function () {
    var x = 2;
    f();
}
g();
```

Ex 3 Free variables

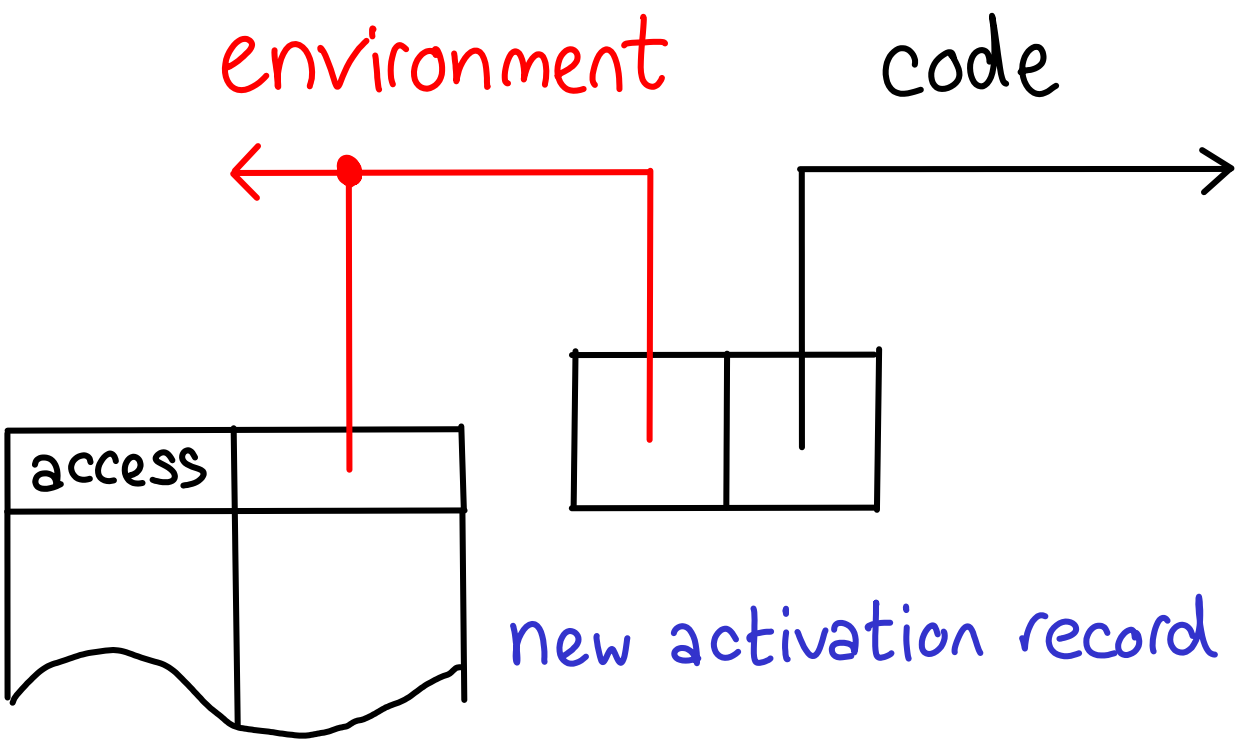



```
var x = 1;
var f = function () {
    console.log(x);
}
var g = function () {
    var x = 2;
    f();
}
g();
```

Closure (the "double-bubble" model)



Closure (the "double-bubble" model)



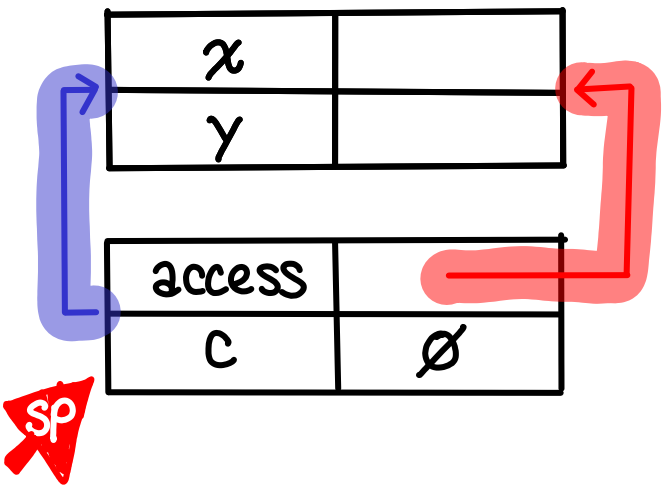
Ex 4 Higher-order functions

x	
y	

SP

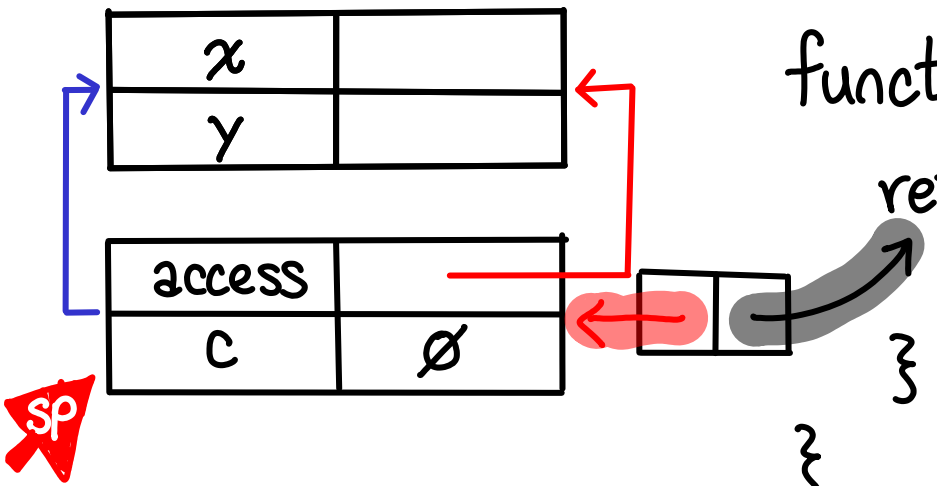
```
function f(c) {
    return function() {
        return c++;
    }
}
var x = f(0);
var y = f(1);
console.log(x());
```

Ex 4 Higher-order functions



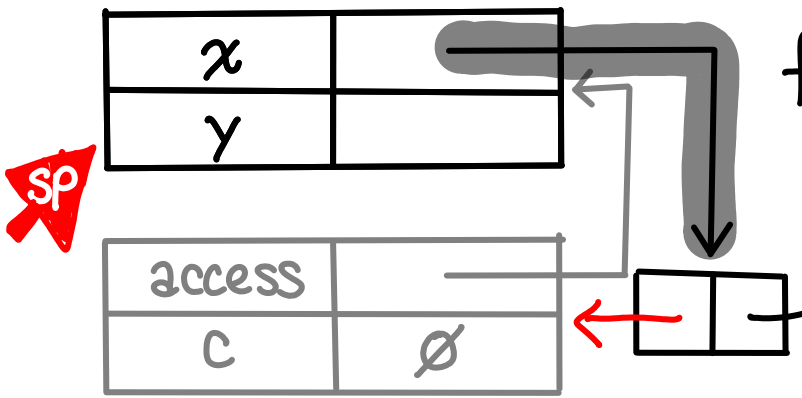

```
function f(c) {
    return function() {
        return c++;
    }
}
var x = f(ø);
var y = f(1);
console.log(x());
```

Ex 4 Higher-order functions



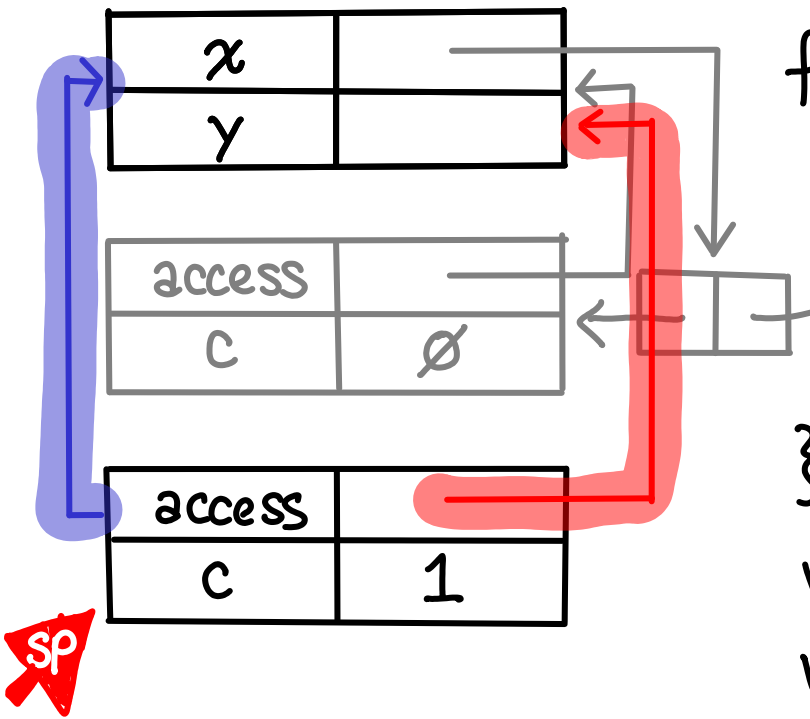

```
function f(c) {
    return function() {
        return c++;
    }
}
var x = f(ø);
var y = f(1);
console.log(x());
```

Ex 4 Higher-order functions



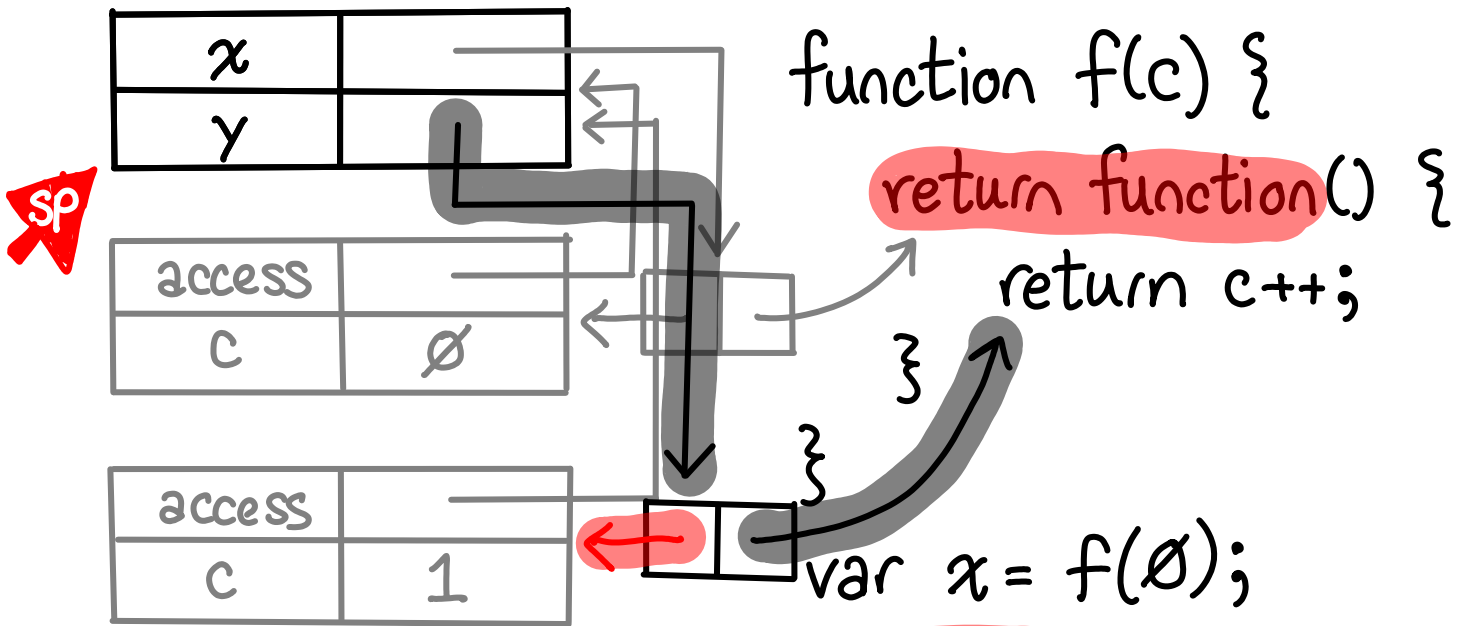

```
function f(c) {
  return function() {
    return c++;
  }
}

var x = f(ø);
var y = f(1);
console.log(x());
```

Ex 4 Higher-order functions

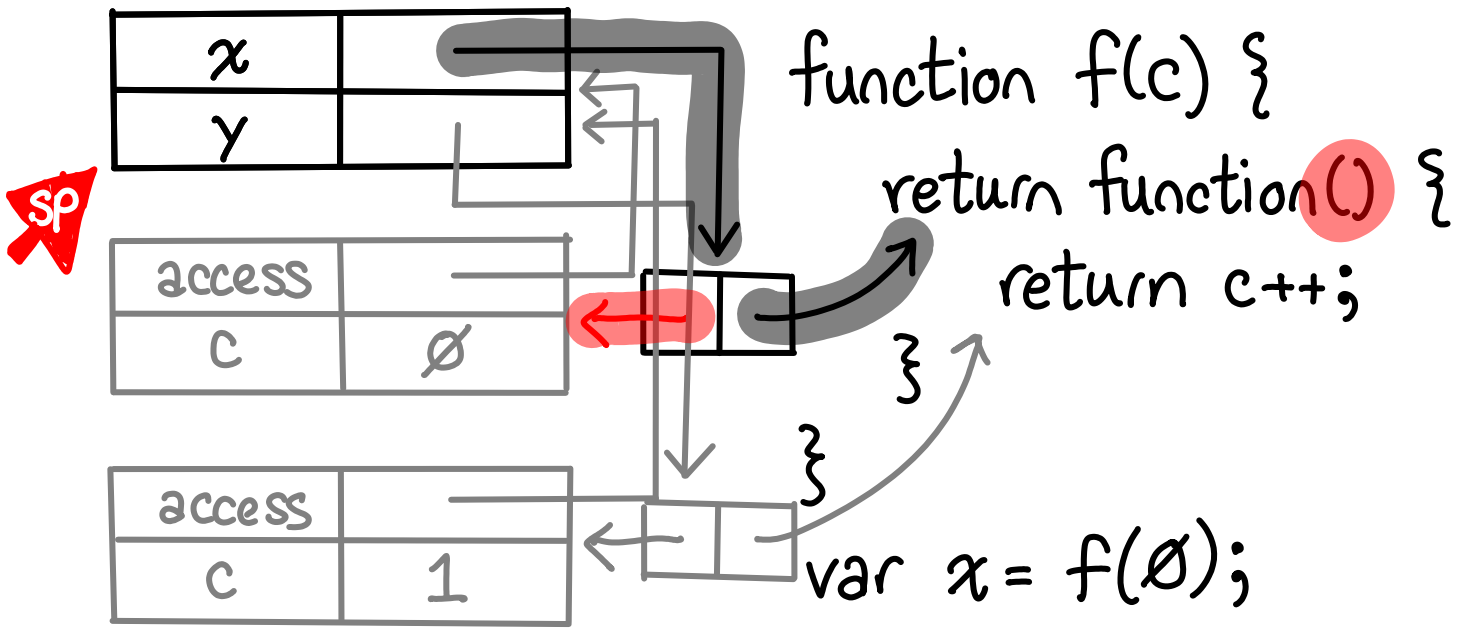



```
function f(c) {
    return function() {
        return c++;
    }
}
var x = f(ø);
var y = f(1);
console.log(x());
```

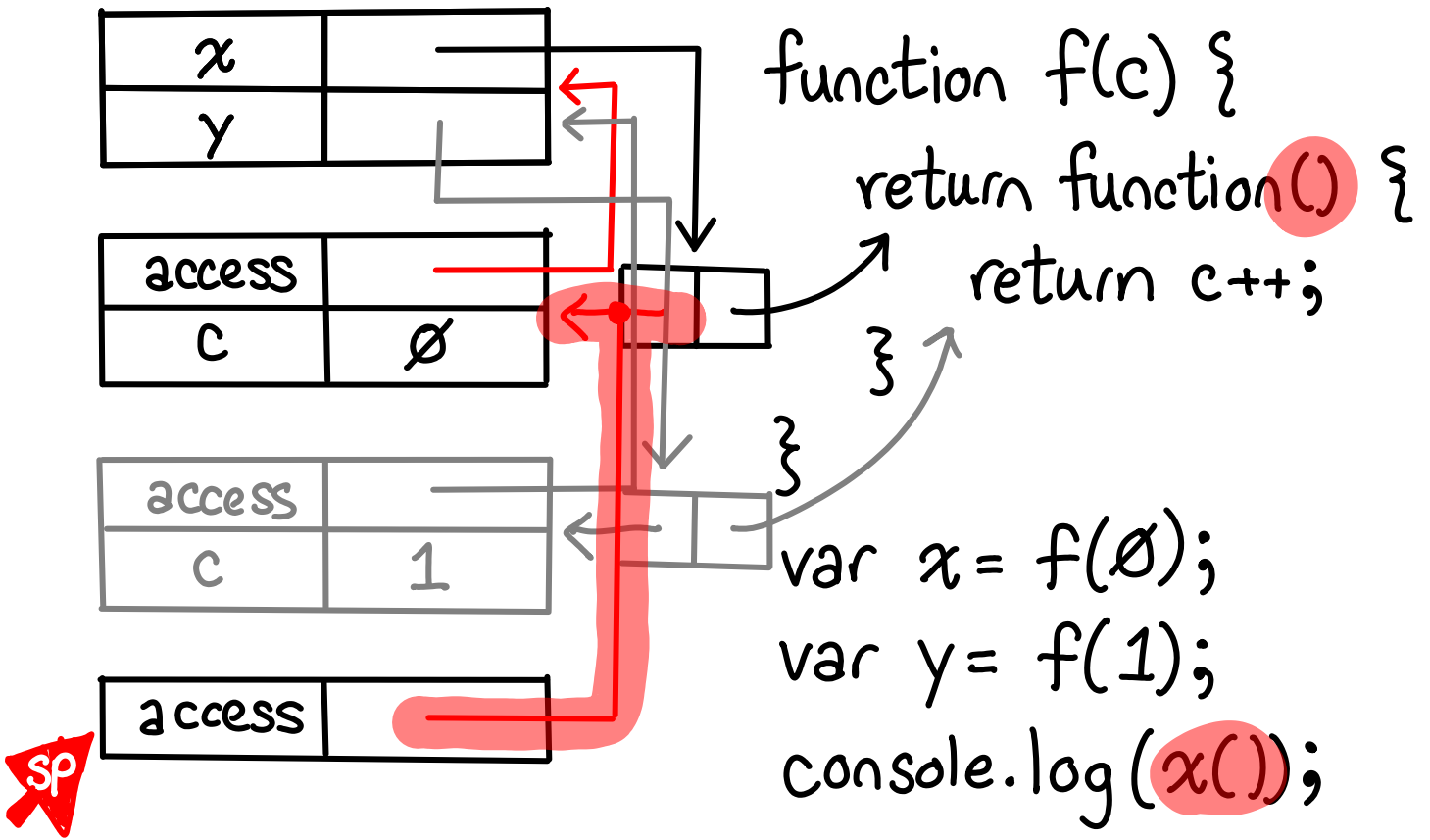
Ex 4 Higher-order functions



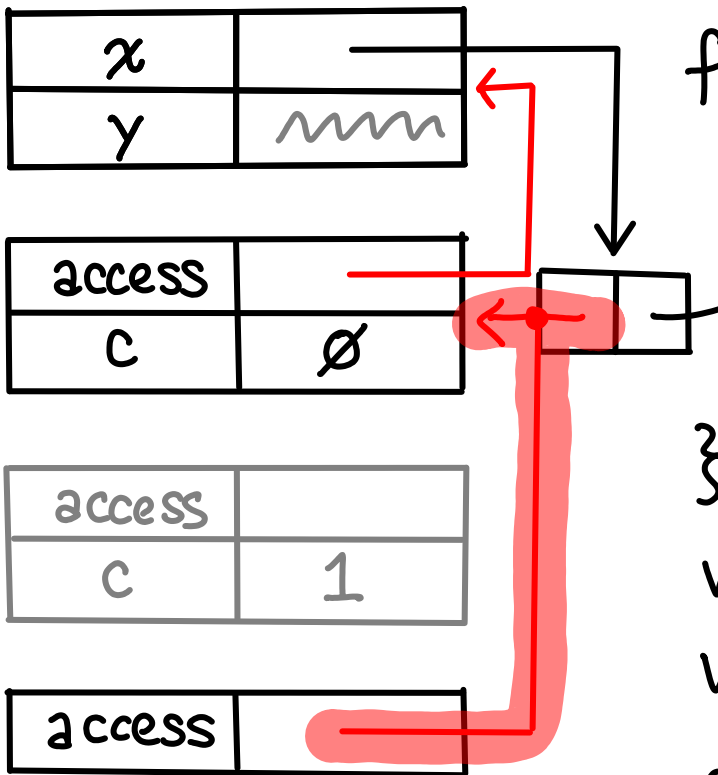
Ex 4 Higher-order functions



Ex 4 Higher-order functions

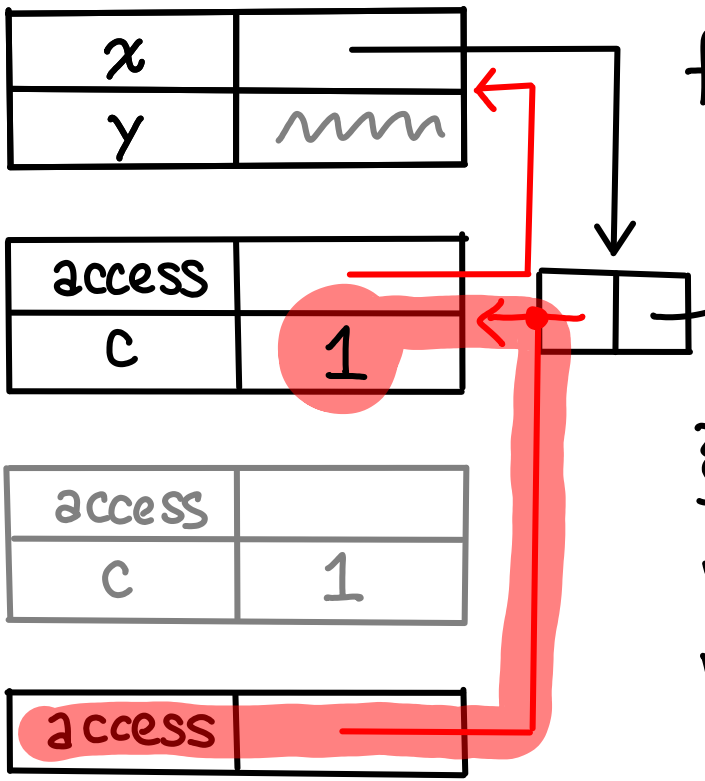


Ex 4 Higher-order functions



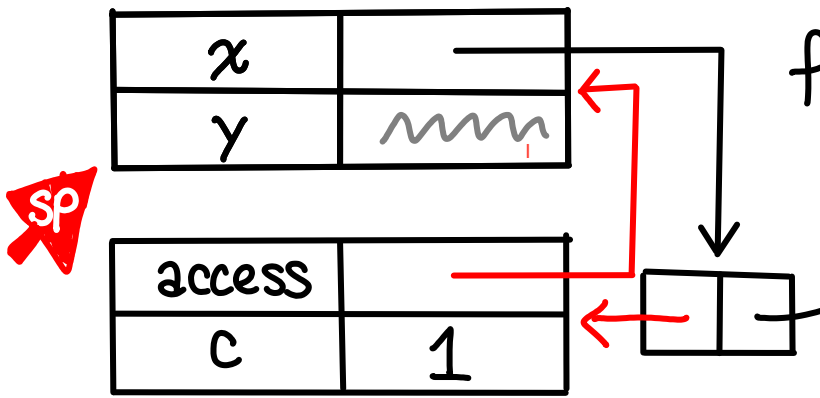

```
function f(c) {
    return function() {
        return c++;
    }
}
var x = f(ø);
var y = f(1);
console.log(x());
```

Ex 4 Higher-order functions




```
function f(c) {
  return function() {
    return c++;
  }
}
var x = f(0);
var y = f(1);
console.log(x());
```

Ex 4 Higher-order functions



access	
c	1

```
function f(c) {
  return function() {
    return c++;
  }
}
var x = f(0);
var y = f(1);
console.log(x());
```

1

You can store data with activation frames

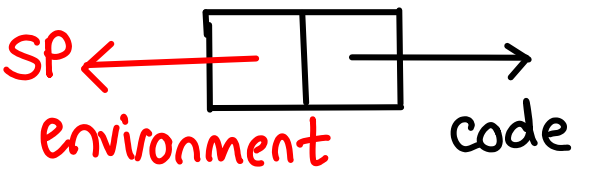
However, the caller must then manage both the chart function (assuming you have multiple types of charts to pick from) and the configuration object. To bind the chart configuration to the chart function, we need a closure:

```
function chart(config) {
  return function() {
    // generate chart here, using `config.width` and `config.height`
  };
}
```

"Towards Reusable Charts" — Mike Bostock

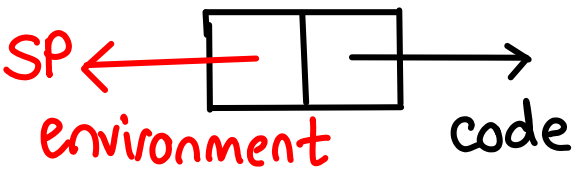
Environment Model

Function definition

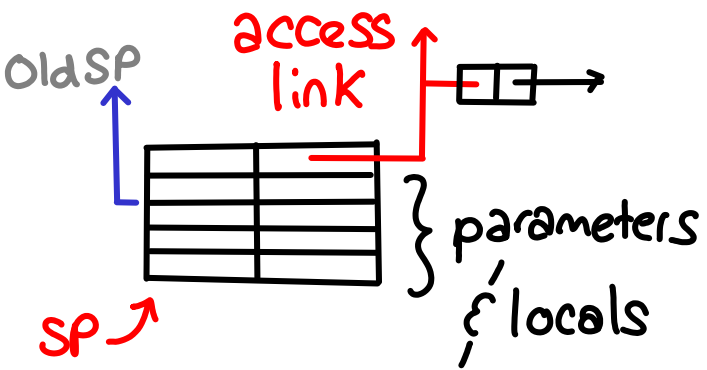


Environment Model

Function definition

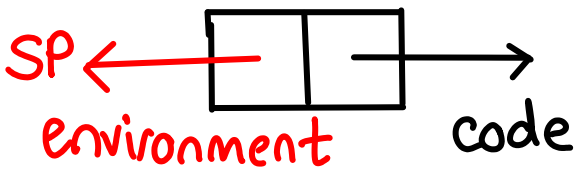


Function application

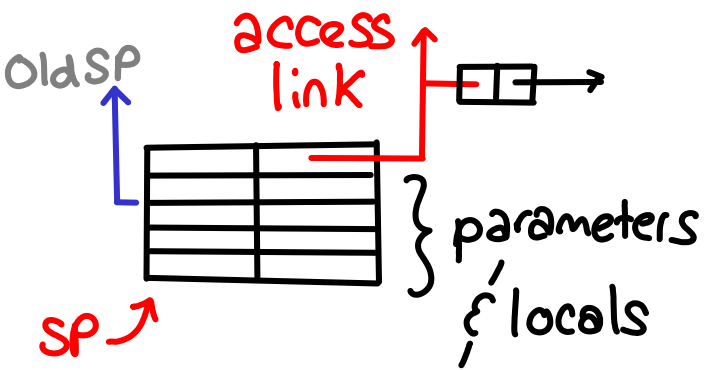


Environment Model

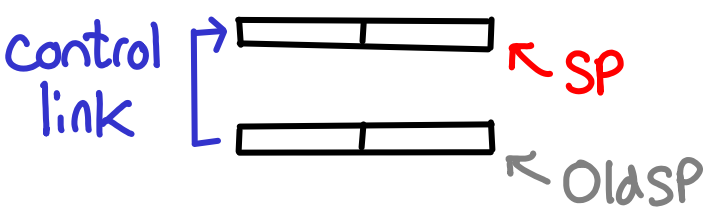
Function definition



Function application



Function return



Bonus

- Parameter passing
- GC concerns
- JavaScript "blocks" aren't scopes
- Variable Hoisting