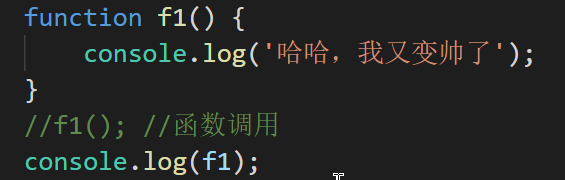
在函数中，函数名中存的是函数代码段，代码如下，以及结果。





// console.log(a); //会报错 这里的a没有引号所以应该为自变量，而前面没有定义赋值，所以会报错。

// var a;

// console.log(a); //结果是undefined 因为前面定义了没赋值

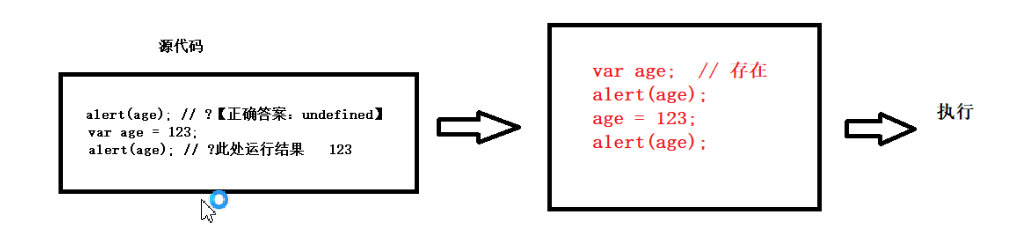
//域解析

//案例1

alert(age); // **答案：undefiend**

var age = 123;

alert(age); //**123**

****

**//案例2**

**console.log(a + 2);**

**var a = 2;**

**console.log(a + 2);**

**变形之后**

**var a;**

**console.log(a+2); //NaN 没赋值 若不加2 就为undefined ,只要undefined与其他相加都是NaN**

**a=2;**

**console.log(a+2); //4**

**//案例3**

**fn();**

**var fn = function() { //这里的fn是变量名 后面可以是任何东西**

**alert(1);**

**}**

**//变形后**

**var fn;**

**fn(); //fn() 指函数调用，这里的fn是函数，往上找的fn为变量名，而应该是函数，所以会报错**

**fn = function() {**

**alert(1);**

**}**

**//案例4**

**function num() {**

**console.log(10);**

**}**

**var num = 20;**

**console.log(num());**

**//变形后**

**function num() {**

**console.log(10);**

**} //以上是一个函数，不调用的话不会执行**

**var num;**

**num = 20;**

**console.log(num()); //这里的num()，num 是个函数，而往上找num是变量名，所以会报错**

**//案例5**

//案例

function num() {

console.log(10); //10 [1]

}

console.log(num()); //这里的num()是函数调用，所以上面的函数体激活了[1]输出了，输出10。

//而函数里没返回所以为undefined

**//案例5**

///////小测试

function num() {

console.log(10); //10

}

num();

var num = 20;

console.log(num());

//变形后

function num() {

console.log(10);

} //以上是一个函数，不调用的话不会执行

var num;

num = 20;

console.log(num()); //这里的num()，num 是个函数，而往上找num是变量名，所以会报错。

**//案例6**

案例

function getSum(x, y) {

console.log(x); // 没有赋值 undefined

return x + y;

}

console.log(getSum()); //返回两个undefined 是NaN

fn(); //?

function fn() {

alert(2);

}

**//案例7**

///案例

function fn(a,b) {

var a;

var b;

a = 100;

var b = 200;

console.log(b);

}

console.log(fn());

///////////////////////////////////////////////////////

function fn(a) { //这里的a是形参，这是函数体里面的一个变量，看成 var a;所以是局部变量而不是隐式全局变量

// var a

a = 100;

var b = 200;

}

console.log(a); //所以找不到全局变量a,所以出错了。

console.log(b);

///////////////////////////////////////////////////////

function fn() {

a = 100;

var b = 200;

}

fn(); //调用了，所以此时函数里的a,为隐式全局变量，所以此时

console.log(a); //a为100

console.log(b); //这里的函数里b是局部变量所以，找不到会出错

**//案例8**

///////案例

var a = 123;

function test() {

console.log(a + 2); //? NaN //若只是a的话为undefined 加2为NaN

var a = 2;

console.log(a + 2); //? 4 //局部变量2+2=4

}

test();

//变形后

function test() {

console.log(a + 2); //

var a;

a = 2;

console.log(a + 2);

}

var a;

a = 123;

test(); //函数调用了上面函数被激发了，

**//案例9**

////案例

function test(f) {

var f;

f(123);

}

test(function(v) {

alert(v + 3); //?

})

//解析：

//test(function(v) {alert(v + 3);}) 是函数调用，function(v) {alert(v + 3);}是实参，和这里的形参

//f相等，又相等于var f; f(123); 变形后为

function test(f) {

var f;

(function(v) {

alert(v + 3);

})(123);

}

test();

//(function(v) {alert(v + 3);})是函数自调用，(123)是实参，所以输出123+3=126

**//////////////////////////案例10**

**function Person(name, age) {**

**this.name = name;**

**this.age = age;**

**}**

**var p1 = new Person('zs', 18);**

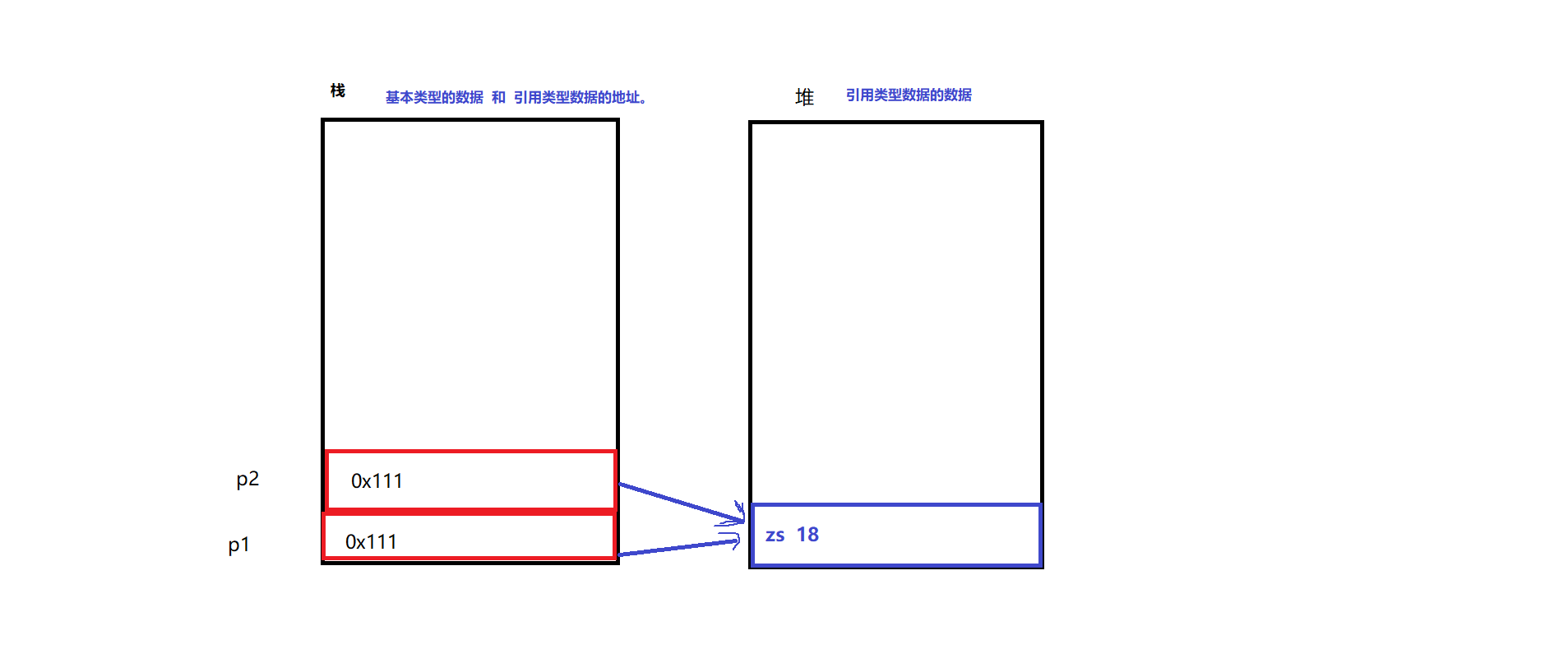
**var p2 = p1; //开内存空间后p1,p2占同一位置**

**p2.name = 'ls'; //所以把p2更改了,p1也会改**

**console.log(p1.name); ///所以p1输出ls**

**console.log(p2.name);**

**的执行结果是： ls ls**

**图片解析如下:**