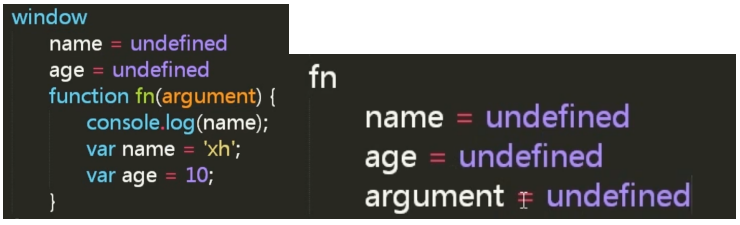
js的解析机制

```
var name = "xm",
    age = 10;
function fn(argument){
    console.log(name);
    var name = "xh";
    var age = 10;
}
fn();
//undefined
```

JS的解析过程：预解析然后逐行读代码

上面的代码中有两个作用域：window和fn，在window中的变量为name，age和fn，在fn中的变量为name，age和argument，分别在两个作用域中进行预解析



变量和函数名冲突的时候进行预解析只保留函数，相当于将变量删除

当函数和函数冲突时，谁在后面就留谁

作用域问题（1）

```
 console.log(a);//a()
var a = 1;
console.log(a);//1因为这是对一个变量的赋值
function a(){
    console.log(2);
};
console.log(a);//1
var a = 3;
console.log(a);//3
function a(){
    console.log(4);
};
console.log(a);//3
a();//报错说a不是一个函数，因为a=3是一个变量
```

在进行预解析的时候由于函数名和变量名冲突，所以解析后只剩下函数体，即预解析后的a是一个函数function a(){ console.log(4); };

作用域问题（2）

js的预解析是分script标签的，先解析上面的script标签里的代码再解析下面的script代码，也就是说下面的script标签里可以读取上面script标签的变量，引用相关的函数。

作用域问题（3）

```
var a = 1;
function fn(){
    console,log(a);//1 由作用域链进行查找在fn中没有a就到fn外面去查找
    a = 2;
}
fn();
console.log(a);//2
```

作用域问题（4）

```
var a = 1;
function fn(a){
    console.log(a);//undefined 因为形参相当于进行var的变量声明
    a = 2;//对局部变量的修改
}
fn();
console.log(a);//1 全局变量没有改变
```