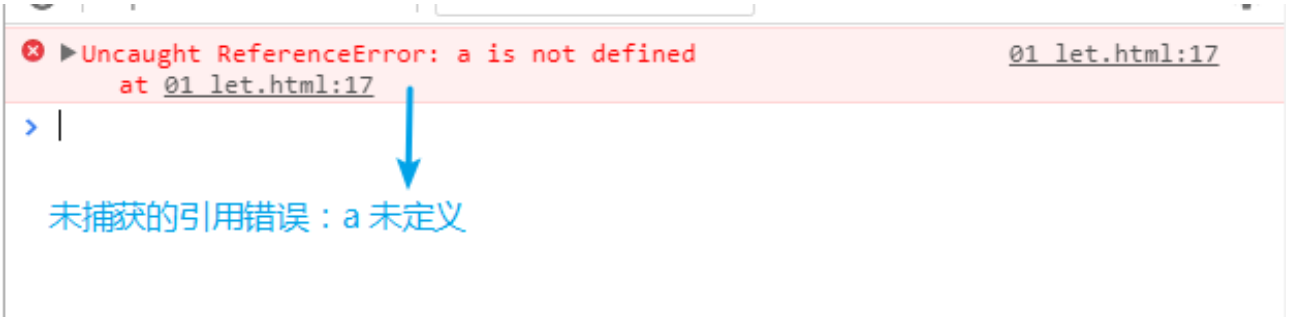
es6简单学习

一、let命令

1.1 let 命令作用域

es6新增let命令用来声明变量，所声明的变量只在let所在的代码块内有效

```
{
    let a = 10;
    var b = 1;
}
alert(a); //报错 Uncaught ReferenceError: a is not defined;
alert(b); // 1
```



1.2 解决i值的传递

```
let a = [];
for (let i=0;i<10;i++) {
    a[i] = function () {
        console.log(i);
    }
}
a[6]();//打印6,如果使用var打印10
```

for循环特别之处：循环语句部分是一个父作用域，而循环体内部是一个单独的子作用域

```
for(let i=0;i<3;i++) {
    // 函数内部的i和外部的i是分离的
    let i = 'abc';
    console.log(i);
    // abc
```

```
    // abc
    // abc
}
```

1.3 不存在变量声明提升

let命令改变了语法行为，它所声明的变量一定要在声明后使用，否则报错

```
// var的情况
console.log(foo); // undefined;
var foo = 2;
// let的情况
console.log(bar); // 报错：Uncaught ReferenceError
let bar = 2;
```

1.4 不允许重复声明

let 不允许在相同的作用域内，重复声明同一个变量

```
// 不能在函数内部重新声明参数
function fn(arg){
    let arg; // 报错
}
```

```
Uncaught SyntaxError: Identifier 'age' has already been declared
>
```

标识符age已经被声明

二、const命令

与let的相同点：

- 只在声明的块级作用域内有效
- 不存在声明提升（暂时性死区）
- 不可重复声明

> 常量一般都大写

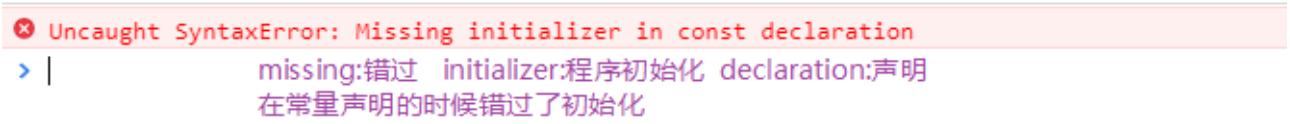
2.1 一但声明值不可更改

```
const PI = 3.14;
```

```
// Uncaught TypeError:Assignment to constant variable
PI = 3;// 报错
```

2.2 一但声明必须赋值

```
const foo;
// SyntaxError:Missing initializer in const declaration
```



2.3 声明复杂数据类型

将一个对象声明为常量

```
// foo对象本身是可变的，不可变的是foo指向的地址
const foo = {};
// 为foo添加属性，foo的地址并没有变化
foo.prop = 123;
console.log(foo.prop);
foo={};
// 将foo指向另一个对象，此时foo的地址发生了变化
```

将数组声明为常量

```
const a = [];
a.push('hello'); //可执行
a.length = 0;
// 将一个新的数组赋值给a，a的地址发生了改变
a = ['Dave']; // 报错
```

总结：当复杂数据类型对应的内存中的内容发生改变时，常量不会改变。
当对应的地址发生改变时，相当于改变常量，会报错

三、解构赋值

注意点

- ...的用法

```
let [head, ...tail] = [1, 2, 3, 4];
console.log(head); // 1
console.log(tail); // [2,3,4]
```

- 数组中未定义的一项值为undefined

```
let [x, y, ...z] = ['a'];
console.log(x); // a
// 数组中未定义的一项值为undefined
console.log(y); // undefined
console.log(z); // []
```

- 解构不成功变量的值为undefined

```
let [foo] = [];
let [bar,foo] = [1];
// foo解构不成功，值为undefined;
```

- 不完全解构

```
let [a,[b],d] = [1,[2,3],4];
console.log(a); // 1
console.log(b); // 2
console.log(d); // 4
```

- 对使用let声明后的变量进行赋值

```
// 对使用let定义的变量进行解构赋值
let foo;
({foo}={foo:1});//必须要加括号
console.log(foo);
let baz;
({bar:baz}={bar:1}); // 成功
console.log(baz); // 1
```

括号必须加，否则会被报错。（解析器会将起始的大括号理解成一个代码块，而不是赋值语句）