1. 作用域和闭包
2. 作用域是什么

* 作用域是根据名称查找变量的一套规则。
* 如果查找的目的是对变量进行赋值，那么就会使用LHS赋值。如果目的是获取变量的值，就会使用RHS查询。
* 编译分3个阶段：词法分析(将代码拆分为词法单元)—语法分析(转换成一个抽象语法树AST)—代码生成(将AST转换成可执行代码即机器语言)。
* 任何js片段在执行前都会进行编译，大部分情况下编译发生在代码执行前几微秒。
* 引擎从头到尾负责整个js程序的编译及执行过程。编译器负责语法分析及代码生成，作用域负责收集并维护由所有声明的标识符组成的一系列查询，并实施一套非常严格的规则，确定当前执行的代码对这些标识符的访问权限。
* var a=2;首先编译器会在当前作用域中声明一个变量(如果之前没有声明过)，然后在运行时，引擎会在作用域中查找该变量，如果能够找到就会对它赋值。
* 当一个块或者函数嵌套在另一个块或者函数中，就发生了作用域的嵌套。遍历嵌套作用域链的规则：引擎从当前的执行作用域中开始查找变量，如果找不到就向上一级继续查找，当抵达最外层的全局作用域时，无论找没找到，查询都会停止。
* 如果RHS查询在所有嵌套的作用域中遍寻不到所需的变量，引擎就会抛出ReferenceError异常。当引擎执行LHS查询时，如果在全局作用域中也找不到目标变量，全局作用域就中就会创建一个具有该名称的变量(非严格模式下)，严格模式下会产生ReferenceError异常。
* ReferenceError同作用域判别失败有关，TypeError代表作用域判别成功了但对结果的操作是不合法的。

1. 词法作用域

* 词法作用域意味着作用域是由书写代码时函数声明的位置来决定的。
* 2个机制可以欺骗词法作用域：eval()和with
* 这2个机制的副作用：引擎无法在编译时对作用域查找进行优化从而导致运行变慢，在严格模式下失效。
* 作用域2种：词法作用域和动态作用域
* 词法作用域就是定义在词法阶段的作用域。
* 没有任何函数可以同时出现在2个父级函数中
* 在多层嵌套的作用域中可以定义同名标识符，这是遮蔽效应，即内部标识符遮蔽了外部的标识符。
* 词法作用域只会查找一级标识符，即foo.bar.baz词法作用域先查找foo标识符，然后对象属性访问规则查找bar…
* with可能会造成全局作用域变量污染，隐式声明全局变量。

1. 函数作用域和块作用域

* 函数的作用域是指，属于这个函数的全部变量都可以在整个函数的范围内使用及复用。
* 从所写的代码中挑选任意片段，然后用函数声明对它进行包装，实际上就是把这些代码隐藏起来了。遵循最小特权原则(最小授权或最小暴露)。避免同名标识符的冲突。
* 避免变量冲突的方法：命名空间、模块管理
* 如果function是声明中的第一个词，那么就是一个函数声明，否则就是一个函数表达式。
* 匿名函数
* arguments.callee调用函数自身
* IIFE代表立即执行函数表达式。进阶用法：把它们当作函数调用并传参数进去。还可以解决undfined标识符的默认值被错误覆盖导致的异常。还可以倒置代码的运行顺序，把需要运行的函数放在第二位，在IIFE执行之后当参数传递进去。
* 块级作用域：with try-catch let const
* 如果用let代替var则需要在代码重构中付出额外的精力。

1. 提升

* 包括变量和函数在内的所有声明都会在任何代码被执行前首先被处理。
* var a=2;🡪var a;a=2;第一个定义声明是在编译阶段进行的，第二个赋值声明会被留在原地等待执行。
* 每个作用域都会进行提升操作。变量只能提升到当前作用域的最前面。
* 函数声明会提升，但函数表达式不会提升。
* 函数首先被提升，然后才是变量。

1. 作用域闭包

* 当函数可以记住并访问所在的词法作用域，即使函数是在当前词法作用域之外执行，这时就产生了闭包。
* 模块2个主要特征：为创建内部作用域而调用了一个包装函数，包装函数的返回值必须至少包括一个对内部函数的引用。
* 一般情况下，函数执行后内部作用域会被销毁。但闭包的话，这个内部作用域不会被销毁。
* IIFE严格来讲不是闭包，因为它不是在它本身的词法作用域外执行的，是在定义时所在的定义域内执行的。里面的变量是通过普通的词法作用域查找而非闭包被发现的。
* 闭包循环

for(var i=0;i<5;i++){

(function(j){

setTimeout(function(){

console.log(j);

},1000);

})(i)

}

for(var i=0;i<5;i++){

let j=i;

setTimeout(function timer(){

console.log(j);

},1000)

}

for(let i=0;i<5;i++){

setTimeout(function timer(){

console.log(i);

},1000)

}

1. 项目构建
2. 安装node

官网下载

重启生效

1. 安装gulp

gulp是一个基于流(stream)的任务自动化构建工具

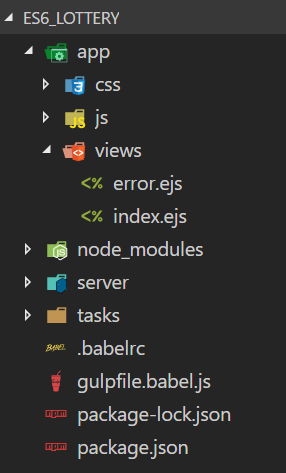
$npm install –global gulp

$ npm install –save-dev gulp

1. 安装babel

$npm install --save-dev babel-cli babel-preset-env

1. 安装webpack
2. 创建项目目录



1. 初始化package.json

$npm init

1. 安装express

$npm install –g express

$npm install –g express-generator@3

express4.0与之前差距较大，无法以node app.js为启动方式

1. 使用ejs模板引擎

$express –e .

注：在server文件夹里

1. 小数减法bug



es6新增属性极小的常量 Number.EPSILON 判断是否为可忽略的误差

只要差值小于最小数就说明正确

注：Number.MIN\_VALUE ECMAScript所能表示的最小数值

1. 回流与重绘

<https://www.cnblogs.com/stitchgogo/p/7920828.html>

1. css制作图形

<http://www.cnblogs.com/xiaohuochai/p/5028101.html>

1. 丢失this的解决方法

var self=this

箭头函数

bind