# Day02笔记

## 数学运算

1. 运算符
2. 数学运算符： + - \* / %
3. 因为JavaScript中只有数字类型，不区分整形和浮点型
4. - \* / % 会触发隐式转换，字符串都会进行触发
5. 字符串能转就转，不能转就转成NaN

* 数学运算的加法（+）
* 除去字符串的情况，其他类型相加都是数学运算
* 可能触发隐式转换
* 字符串的拼接
* 任何类型只要跟字符串相加都是字符串的拼接
* 转换
* 隐式转换
* True =》1
* False =》0
* Undefined =》 NaN
* NULL =》0
* 强制（显式）转换 能转就转，不能转就转成NaN
* parseInt（）
* 字符串类型的小数会舍弃小数部分并转成数字类型
* 如果以数字开头，里面含有字符，会截取前面数字部分转成数字类型
* parseFloat（）
* Number（） 只能进行纯数字的字符串转换
* +

## 比较运算

* 比较运算符
* > >= < <= == != （=== !== JavaScript中独有的）
* 赋值运算符
* += -= \*= /=
* 前++ 存储新值，使用新值
* 后++ 存储新值，使用旧值

## 函数声明及函数表达式

* 区别
* 使用：没有任何区别的
* 对预解析有影响的
* 本质：函数声明式函数，函数表达式是声明的变量
* 区分：函数声明是以funtion开头，其他的全是函数表达式
* 一旦函数声明变成了函数表达式，就会放弃函数名（只能看，没有任何作用），函数表示式一般都是没有函数名的
* ()执行符号
* 不能去直接执行函数声明
* 可以去执行函数表达式

## 作用域、作用域链、闭包

* JavaScript中非常重要的两条链
* 作用域链： 从上往下查找
* 原型链：从下往上查找
* 作用域
* 全局作用域
* 函数作用域
* 块级作用域（ES6新增）
* 特点：逐行执行
* 基本语法检测 例如单词拼写错误，中文没有注释...低级错误
* 形成作用域（对象）的过程，预解析（预编译） 4步

1. 函数在执行前的一瞬间，会生成一个AO（active object活动对象）对象
2. 分析参数，形参作为AO的属性名，实参作为AO对象的属性值
3. 分析var声明，变量名作为AO对象的属性名，值为undefined，如果遇到同名，不会对AO对象作出任何改变
4. 分析函数声明，函数名作为AO对象的属性名，值为函数体，如果遇到同名，直接以函数声明为主，直接覆盖

* 逐行执行，就是根据预解析产生出来的对象进行逐行执行
* 全局作用域，对应的对象是GO（global object）。进入script标签以后，在所有代码之前会产生
* 产生的方式可以说跟AO完全相同，只是少了参数这一个步骤
* 在进入script标签，代码执行前的一瞬间，会产生一个GO对象
* 分析var声明，变量名作为GO对象的属性名，值为undefined，如果遇到同名，不做任何改变
* 分析函数声明，函数名作为GO对象的属性名，值为函数体，如果遇到同名，直接覆盖
* 因为JavaScript中只有全局作用域和函数作用域
* 在进行预解析的时候 只需要去管函数 if判断中的声明（变量声明及函数声明都得正常分析）if条件决定是后面的执行过程
* 注： var声明是没有影响的，但是函数声明存在兼容性问题(各个版本的浏览器产生的结果会不一样)
* 闭包: 通过一些方式可以访问到函数内部的变量的
* 正常来说，函数执行完成以后，会销毁
* 优点：可以变量的安全
* 缺点：但是闭包会带着父级的作用域保存到外面，导致函数执行完成以后内存不会被释放

## Arguments 和 形参有映射关系

* 当形参和实参没有对应的时候，没有对应的形参和arguments没有办法形成映射关系
* 开启了严格模式，形参和arguments就不会有映射关系

## DOM（getElementsByTagName中的问题）

* 面试中可能会问到：实时性的问题
* getElementsByTagName是返回副本，不是实时更新的