day01 JavaScript入门

JavaScript概述

JavaScript是一个**解释型的语言**，JavaScript是一个**脚本语言**（侵入性 xss攻击），JavaScript是一个**弱类型语言**（没有强制的类型）。JavaScript由**BOM**（browser object model 浏览器对象模型），**DOM**（document object model 文档对象）**，ECMAScript**（基础语法）。 ECMAScript主要版本有ES3（所有的浏览器都支持），ES5（大部分浏览器支持），ES6（部分浏览器支持）。（babel.js他是一个专门用来转换ECMAScript相关版本的一个脚本工具）。

JavaScript的相关书写

- 内嵌写法 （不建议的）

```
<!-- 内嵌写法 -->
<a href="javaScript:void">点击</a>
```

- 内联写法 （写主要代码）

```
<!-- 内联写法 script标签可以写在任意的位置 建议写在最后 代码执行流程 从上到下的 文档流-->
<script>
    console.log('你好 世界')
</script>
```

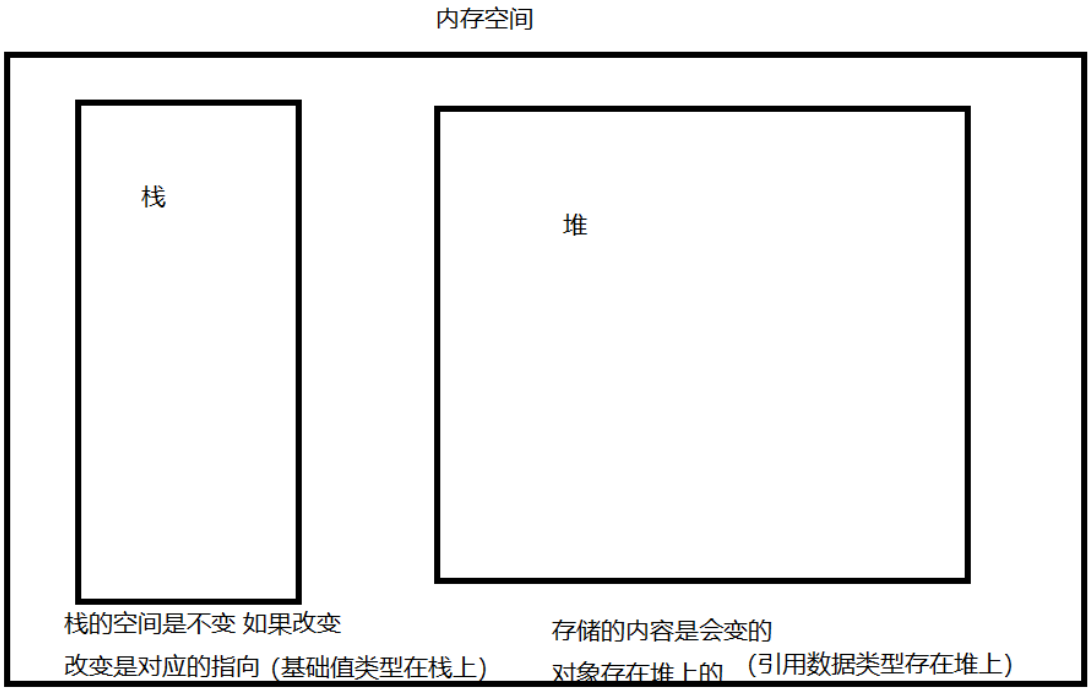
- 外联写法 （抽取的公共代码）

```
<!-- 外联的写法 -->
<script src="./demo.js"></script>
```

JavaScript的变量

变量就是一个存储单位，他会根据你赋的值在内存中开启空间（根据赋值得到对应的类型）

内存空间



变量声明

采用var关键词来进行声明 （var关键词声明的是伪全局变量）

```
var 变量名 = 变量值
```

变量命名的相关规范

- 不能是关键词和保留字

- 不能以数字开头
- 只能由数字,字母,下滑线,$构成
- 使用驼峰命名法 （首字母小写其他的首字母大写）
- 语义化命名 （见名知意）

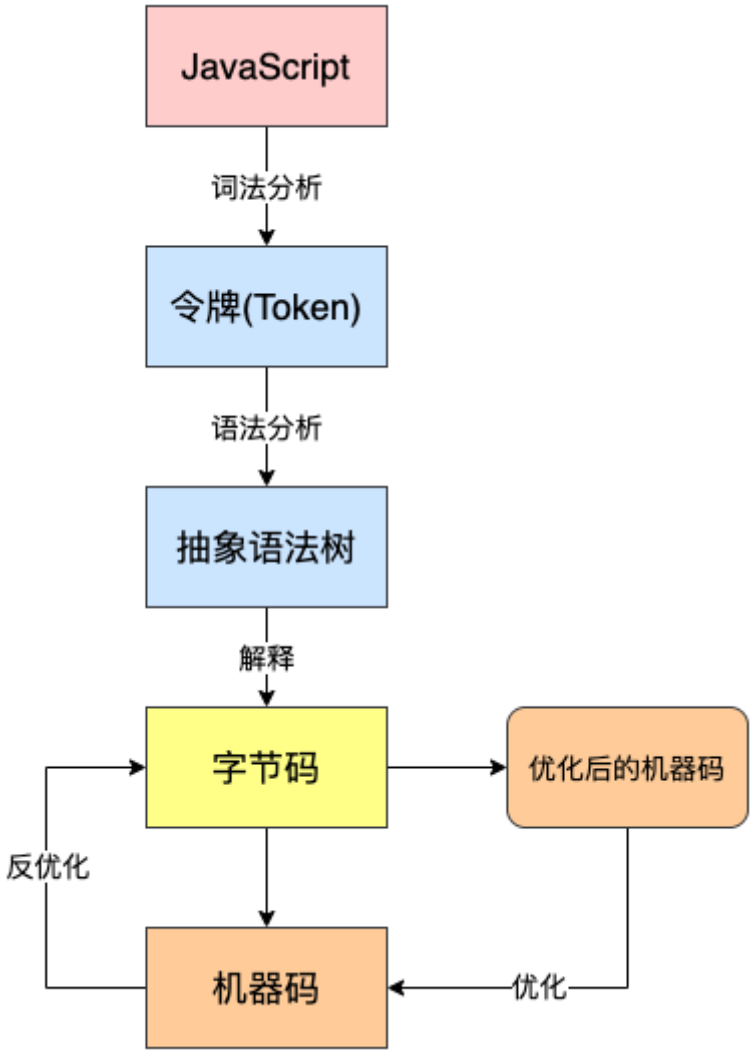
v8引擎的解析过程

```
var a = 1
```

- 分段（分为一个个token）

  ```
  关键词 var
  名字  a
  符号 =
  数字 1
  ```

- 组成ast （抽象语法树）去掉无效的符号
- 根据抽象语法树声明字节码解析



JavaScript的数据类型 （根据赋值来决定类型）

基础值类型（存储在栈上）

- 数值类型（所有的数字都是数值类型）number

```
var number = 10
```

- 字符串类型 （使用单引号或双引号括起来的都是字符串类型）string

```
var str = 'hello world'
var str1 = "hello world"
```

- 布尔类型 (true 或者 false) boolean

```
var bool = true
```

- null类型 （取值为null 空类型（gc垃圾回收机制））

```
var obj = null
```

- undefined类型 （未定义 他是null派生子类）

```
var un
```

引用数据类型 （存储在堆上）

object 、function 、date ....

typeof 类型检测（原理二进制解析）

```
//数值类型
var number = 10
//类型检测 typeof (原理二进制解析) 返回的是一个字符串
console.log(typeof number);
//字符串
var str = 'hello'
var str = '' //空字符串不是空
console.log(typeof str);
//布尔类型
var bool = true
console.log(typeof bool)
//null类型 (空对象引用 typeof值为object)
var obj = null
console.log(typeof obj);
//undefined 未定义
var un = undefined
console.log(typeof un);
// console.log(un == null);
console.log(typeof typeof un);//返回的是一个string
```

javascript中的注释

单行注释 ctrl+/

```
//单行注释
```

多行注释 shift+alt+a

```
/*
多行注释
*/
```

类型转换

基础值类型转换引用数据类型 （装箱）

- String
- Number
- Boolean

```
var number = 10
console.log(typeof number)
//只要是被new都是引用数据类型
var obj = new Number(number)
console.log(typeof obj);
//String
var str = 'hello'
console.log(typeof new String(str))
//Boolean
var bool = true
console.log(typeof new Boolean(bool))
```

引用数据类型转为基础值类型 （拆箱）

- toString 转为字符串

```
//obj对象转为字符串 对象中有东西 属性(方法)
console.log(typeof obj.toString());//转为字符串
console.log(typeof String(obj));//转为字符串
//obj对象转为对应的数值
console.log(typeof Number(obj))
//obj对象转为对应的布尔类型
console.log(typeof Boolean(obj))
console.log(Boolean(obj));
```

基础值类型之间的转换

转为字符串（String 及 toString方法）

- 数值转字符串

  ```
  // 数值转换字符串
  var number1 = 10
  var str1 = String(number1)
  console.log(typeof str1);
  ```

- 布尔类型转为字符串

  ```
  //boolean类型转为字符串
  var bool = true
  var str2 = String(bool)
  console.log(typeof str2);
  ```

- null及undefined转为字符串

```
//null转为字符串
var nullObj = null
var str3 = String(nullObj)
console.log(typeof str3);
//undefined转为字符串
var un
var str4 = String(un)
console.log(typeof str4);
```

- toString转换

```
//利用toString来转换
console.log(typeof number1.toString());
console.log(typeof bool.toString());
//null和undefined没有toString方法
// console.log(typeof nullObj.toString());
// console.log(typeof un.toString());
```

转为数值类型 （Number 及 parseInt 和 parseFloat）

- 字符串转为数值 无法转换为NaN
- boolean类型转为数值 true为1 false为0
- null转为数值 值为0
- undefined转为数值 值为NaN

```
console.log(Number('124345'))
console.log(Number(null))//null转为数值为0
console.log(Number(true))//true为1 false为0
//NaN是数值类型
console.log(typeof NaN)//number
console.log(Number(undefined))//NaN  Not a Number
//任意类型转为数值的时候 无法进行转换就会出现NaN
console.log(Number('abc'))//NaN
console.log(Number('1234abc'))//NaN
//parseInt 转整型 (切割前面的内容) parseFloat(保留小数) 转浮点型
console.log(parseInt('1234.123abc'))
console.log(parseInt('a123.123abc'))//NaN
console.log(parseFloat('a123.123'))//NaN
console.log(parseFloat('1234.123abc'))
```

转为boolean类型 （Boolean在条件表达式下自动转为布尔类型）

- 数值转为boolean类型 非0及NaN都是true
- 字符串转为boolean类型 非空字符就是true
- null和undefined转为boolean类型 都是false

```
//数值转为boolean类型 非0及NaN都是true
console.log(Boolean(123))
console.log(Boolean(0))
console.log(Boolean(NaN))
//字符串转为boolean类型 非空字符就是true
console.log(Boolean(''))
console.log(Boolean(' '))
console.log(Boolean('123'))
//null和undefined转为boolean类型
console.log(Boolean(null))
console.log(Boolean(undefined))
```

Number

- NaN 无法被转换为数值的时候出现的
- infinity 无穷大

```
//常量值
console.log(Number.MAX_SAFE_INTEGER)
console.log(Number.MAX_VALUE)
console.log(Number.MIN_SAFE_INTEGER)
console.log(Number.MIN_VALUE)
console.log(Number.NaN) //NaN的值
console.log(Number.NaN,NaN)
//无法被转换为数值的时候出现NaN abc 转为数值  dfg 转为数值
console.log(NaN == NaN)//false
//无穷大
console.log(Number.NEGATIVE_INFINITY,-Infinity) //负无穷大
console.log(Number.POSITIVE_INFINITY,Infinity) //正无穷大
console.log(-Infinity == Number.NEGATIVE_INFINITY)
```

运算符

算术运算符

```
+ - * / % ++ --
```

- +特殊的算术运算符

  对于有字符串的值进行+ 那么对应的+号做的连接 返回的是字符串

  如果没有字符串那么其他的会被转为number类型进行运算（其他类型还是会被转为number）
- 其他的算术运算 （会将对应的值转为number类型进行运算 如果出现NaN 那么结果就是NaN）
- ++前置先执行+1操作再执行本行代码 ++后置就是先执行本行代码再执行++

```
console.log(1+true)//2
console.log(1+true+null)//2
console.log(1+true+undefined)//NaN
//出现字符串做的为连接
console.log(1+2+3+undefined+'undefined'+true+10)//NaNundefinedtrue10
console.log(1+''+2) //12
console.log(1*1+2+3+(3-4))//5
console.log(1*null)
console.log(null-undefined+10*100+(1000/10))//NaN
//取余 取模  小数取大数得到本身 大数取小数得余数
```

```
console.log(3%4)//3
console.log(4%3)//1
console.log(10%2+10)//10
//++的意思在原本的基础上自增1
var a = 10
console.log(++a)//11
//++ 前置和后置的区别
// ++前置先执行+1操作再执行本行代码 ++后置就是先执行本行代码再执行++
console.log(a++)//11
console.log(a)//12
console.log(12+13+(a++)-(a--)+(++a))//37 12+13+12-13+13
//执行顺序
// 先算括号里面 再执行方法 再算*/ % 再算+-
console.log(a*12+undefined-(10+5)%(5).toString())//NaN
console.log(a.toString()+undefined-(9*9)%10)//NaN
```

赋值运算符

```
= += -= *= /= %=
```

```
// 赋值运算在最后执行
// += -= *= /= %=
var i = 10
// i+=10 ==> i = 10+10
console.log(i+=10) //20
console.log(i-=5)//15
console.log(i/=5)//3
console.log(i%=2)//1
```

逻辑运算符

- &&都为真就是真(取得最后一个真) 只要有一个是假得就是假(第一个假)
- ||有一个是真就是真(取第一个真) 如果全部为假就是假(最后一个假)
- ！ 取反

```
&& || !
```

```
// &&都为真就是真(取得最后一个真) 只要有一个是假得就是假(第一个假)
console.log(1&&'hello'&&true&&3)//3
console.log(1&&'hello'&&0&&false)//0
//|| 有一个是真就是真(取第一个真)  如果全部为假就是假(最后一个假)
console.log(1||'hello'||true||3)//1
console.log(1||'hello'||0||false)//1
console.log(0||false)//false
console.log(!1)//取反 返回的是boolean类型的值
```

比较运算符 （返回boolean类型）

- NaN！ = NaN
- 对象之间不能直接比对（比对的是地址值）

```
> < == >= <= != ===
```

```
console.log(1>2)//false
console.log(1>=1)//true
console.log(1 && 1>1 || 2>2)//false
//== 和 ===  ==是表示值相等 ===表示是在==的基础上类型也要相等
console.log(1 == '1')//true
console.log(1 === '1')//false
//NaN==NaN false NaN!=NaN true
console.log(NaN==NaN)//false
console.log(NaN!=NaN)//true
//0.1+0.2 != 0.3 二进制解析 二进制运算(减法快 )
console.log(0.1 == 0.1)
console.log(0.2 == 0.2)
console.log(0.1+0.2 == 0.3)//false
```

位运算符（解析成二进制进行运算）

```
& ^ | < >
```

位移运算符（二进制上进行位移）

```
>> <<
```

```
// 位移运算
// 2 10
console.log(2>>2)//0
//左移 2 10  1000 (快速把2变成8)
console.log(2<<2)
```

..

扩展内容

进制转换

常用进制 二进制 四进制 八进制 十进制 十六进制

进制转换 从其他进制转为10进制 乘进制法

11111 转为 10进制

```
1*2的0次方 + 1*2的1次方 ....
1+2+4+8+16 = 31
```

将10进制转换为其他进制 除进制取余法

100 转为 2进制

```
除2取余法
```

将十进制转为其他进制得到字符串 toString

将其他进制转为10进制得到数值 parseInt

```
var n = 100
//转为2进制 toString方法 传入对应的转换的进制
var str = n.toString(16)
console.log(str)
//将其他进制转为10进制 parseInt 传入的其他进制内容 自己的进制 转换的进制
console.log(parseInt(str,16,10))
```