链滴

面试开发常用的 JavaScript 知识点总结

作者：Vanessa
原文链接：https://ld246.com/article/1470722230732
来源网站：链滴

<h2 id="No1-语法和类型">No1.语法和类型</h2>
<h2 id="1-声明定义">1.声明定义</h2>
　　变量类型：var，定义变量；let，定义块域(scope)本地变量；const，定义只读常量。
　　变量格式：以字母、下划线“_”或者 $ 符号开头，大小写敏感。
　　变量赋值：声明但未赋值的变量在使用时值为 undefined，未声明的变量直接使用会抛异常
　　未赋值变量作计算：结果为 NaN。例如：
<pre><code class="highlight-chroma"><span class="highlight
cl">var x, y = 1;
console.log(x + y);
//结果为NaN，因为x没有赋值。
</code></pre>
<h2 id="2-作用域">2.作用域</h2>
　　变量作用域：在 ES6 之前没有块声明域，变量作用于函数块或者全局。如下面的代码输入的
为 5。
<pre><code class="highlight-chroma"><span class="highlight
cl">if (true) {
    var x = 5;
}
console.log(x); // 5
</code></pre>
　　ES6 变量作用域：ES6 支持块作用域，但需要使用 let 声明变量。下面的代码输出结果抛出
常。
<pre><code class="highlight-chroma"><span class="highlight
cl">if (true) {
    let y = 5;
}
console.log(y); //
eferenceError: y is not defined1234
</code></pre>
　　变量上浮：在一个方法或者全局代码中，我们在生命变量之前使用变量时并没有抛异常，而
返回 undefined。这是因为 javascript 自动把变量的声明上浮到函数或者全局的最前面。如下面的代
:
<pre><code class="highlight-chroma"><span class="highlight
cl">/**
* 全局变量上浮
*/
console.log(x ===
undefined); // logs "true"
var x = 3;

/**
* 方法变量上浮
*/
var myvar = "my v
lue";
// 打印变量myvar
果为：undefined
(function() {
    console.log(my
ar); // undefined
    var myvar = "lo
al value";
})();

```
</span></span></code></pre>
<p>　　上面代码和下面代码是等价的：</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">/**
</span></span><span class="highlight-line"><span class="highlight-cl">* 全局变量上浮
</span></span><span class="highlight-line"><span class="highlight-cl">*/
</span></span><span class="highlight-line"><span class="highlight-cl">var x;
</span></span><span class="highlight-line"><span class="highlight-cl">console.log(x ===
undefined); // logs "true"
</span></span><span class="highlight-line"><span class="highlight-cl">x = 3;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">/**
</span></span><span class="highlight-line"><span class="highlight-cl">* 方法变量上浮
</span></span><span class="highlight-line"><span class="highlight-cl">*/
</span></span><span class="highlight-line"><span class="highlight-cl">var myvar = "my v
lue";
</span></span><span class="highlight-line"><span class="highlight-cl">(function() {
</span></span><span class="highlight-line"><span class="highlight-cl">    var myvar;
</span></span><span class="highlight-line"><span class="highlight-cl">    console.log(my
ar); // undefined
</span></span><span class="highlight-line"><span class="highlight-cl">    myvar = "local
alue";
</span></span><span class="highlight-line"><span class="highlight-cl">})();
</span></span></code></pre>
<p>　　全局变量：在页面中，全局对象是 window，所以我们访问全局变量可通过 window.variabl
。例如：</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">version = "1.0.0";
</span></span><span class="highlight-line"><span class="highlight-cl">console.log(wind
w.version); //输出1.0.0
</span></span></code></pre>
<h2 id="No2-数据结构和类型">No2.数据结构和类型</h2>
<h2 id="1-数据类型">1.数据类型</h2>
<p>　　6 个基础类型：Boolean（true 或者 false）、null（js 大小写敏感，和 Null、NULL 是有
别的）、undefined、Number、String、Symbol（标记唯一和不可变）</p>
<p>　　一个对象类型：object。</p>
<p>　　object 和 function：对象作为值的容器，函数作为应用程序的过程。</p>
<h2 id="2-数据转换">2.数据转换</h2>
<p>　　函数：字符串转换为数字可使用 parseInt 和 parseFloat 方法。</p>
<p>　　parseInt：函数签名为 parseInt(string, radix)，radix 是 2-36 的数字表示数字基数，例如
进制或者十六进制。返回结果为 integer 或者 NaN，例如下面输出结果都为 15。</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">parseInt("0xF", 16);
</span></span><span class="highlight-line"><span class="highlight-cl">parseInt("F", 16);
</span></span><span class="highlight-line"><span class="highlight-cl">parseInt("17", 8);
</span></span><span class="highlight-line"><span class="highlight-cl">parseInt(021, 8);
</span></span><span class="highlight-line"><span class="highlight-cl">parseInt("015", 10)

</span></span><span class="highlight-line"><span class="highlight-cl">parseInt(15.99, 10)

</span></span><span class="highlight-line"><span class="highlight-cl">arseInt("15,123",
0);
</span></span><span class="highlight-line"><span class="highlight-cl">parseInt("FXX123",
16);
```

```
</span></span><span class="highlight-line"><span class="highlight-cl">parseInt("1111", 2)

</span></span><span class="highlight-line"><span class="highlight-cl">parseInt("15*3", 1
);
</span></span><span class="highlight-line"><span class="highlight-cl">parseInt("15e2", 1
);
</span></span><span class="highlight-line"><span class="highlight-cl">parseInt("15px", 1
);
</span></span></code></pre>
<p>　　parseFloat：函数签名为 parseFloat(string)，返回结果为数字或者 NaN。例如：</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">parseFloat("3.14"); //返回数字
</span></span><span class="highlight-line"><span class="highlight-cl">parseFloat("314e-
"); //返回数字
</span></span><span class="highlight-line"><span class="highlight-cl">parseFloat("more
on-digit characters"); //返回NaN
</span></span></code></pre>
<h2 id="3-数据类型文本化">3.数据类型文本化</h2>
<p>　　文本化类型：Array、Boolean、Floating-point 、integers、Object、RegExp、String。<
p>
<p>　　Array 中额外的逗号情况：["Lion", , "Angel"]，长度为 3，[1]的值为 undefiend。['home', ,
'school', ]，最后一个逗号省略所以长度为 3。[ , 'home', , 'school']，长度为 4。['home', , 'school', ,
]，长度为 4。</p>
<p>　　integer 整数：整数可以表达为十进制、八进制、十六进制、二进制。例如：</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">0, 117 and -345 //十进制
</span></span><span class="highlight-line"><span class="highlight-cl">015, 0001 and -0o
7 //八进制
</span></span><span class="highlight-line"><span class="highlight-cl">0x1123, 0x00111
nd -0xF1A7 //十六进制
</span></span><span class="highlight-line"><span class="highlight-cl">0b11, 0b0011 and
-0b11 1234 //二进制
</span></span></code></pre>
<p>　　浮点数：[(+|-)][digits][.digits][(E|e)[(+|-)]digits]。例如：</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">3.1415926，-.123456789，-3.1E+12（3100000000000），.1e-23（1e-24）
</span></span></code></pre>
<p>　　对象：对象的属性获取值可通过“.属性”或者“[属性名]”获取。例如：</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">var car = { manyCars: {a: "Saab", "b": "Jeep"}, 7: "Mazda" };
</span></span><span class="highlight-line"><span class="highlight-cl">console.log(car.m
nyCars.b); // Jeep
</span></span><span class="highlight-line"><span class="highlight-cl">console.log(car[7])
 // Mazda
</span></span></code></pre>
<p>　　对象属性：属性名可以是任意字符串或者空字符串，无效的名字可通过引号包含起来。复杂名字不能通过.获取，但可以通过[]获取。例如：</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">var unusualPropertyNames = {
</span></span><span class="highlight-line"><span class="highlight-cl">    "": "An empty st
ing",
</span></span><span class="highlight-line"><span class="highlight-cl">    "!": "Bang!"
</span></span><span class="highlight-line"><span class="highlight-cl">}
</span></span><span class="highlight-line"><span class="highlight-cl">console.log(unusu
```

lPropertyNames."); // SyntaxError: Unexpected string
console.log(unusu
lPropertyNames[""]); // An empty string
console.log(unusu
lPropertyNames.!); // SyntaxError: Unexpected token !
console.log(unusu
lPropertyNames["!"]); // Bang!
</code></pre>
　　转意字符：下面的字符串输出结果包含了双引号，因为使用了转意符号“\”。
<pre><code class="highlight-chroma"><span class="highlight
cl">var quote = "He read \"The Cremation of Sam McGee\" by R.W. Service.";
console.log(quote)

//输出：He read "
he Cremation of Sam McGee" by R.W. Service.1。
</code></pre>
　　字符串换行法：直接在字符串行结束时添加“\”，如下代码所示：
<pre><code class="highlight-chroma"><span class="highlight
cl">var str = "this string \
is broken \
across multiple\
lines."
console.log(str); //
this string is broken across multiplelines.
</code></pre>
<h2 id="No3-控制流和错误处理">No3.控制流和错误处理</h2>
<h2 id="1-块表达式">1.块表达式</h2>
　　作用：块表达式一般用于控制流，像 if、for、while。下面的代码中{x++;}就是一个块声明

<pre><code class="highlight-chroma"><span class="highlight
cl">while (x < 10) {
　　x++;
}
</code></pre>
　　ES6 之前没有块域范围：在 ES6 之前，在 block 中定义的变量实际是包含在方法或者全局
，变量的影响超出了块作用域的范围。例如下面的代码最终执行结果为 2，因为 block 中声明的变量
用于方法。
<pre><code class="highlight-chroma"><span class="highlight
cl">var x = 1;
{
　　var x = 2;
}
console.log(x); //
utputs 2
</code></pre>
　　ES6 之后有块域范围：在 ES6 中，我们可以把块域声明 var 改成 let，让变量只作用域 block
范围。
<h2 id="2-逻辑判断">2.逻辑判断</h2>
　　判断为 false 的特殊值：false、undefined、null、0、NaN、""。
　　简单 boolean 和对象 Boolean 类型：简单 boolean 类型的 false 和 true 与对象 Boolean
类型的 false 和 true 是有区别，两者是不相等的。如下面的例子：
<pre><code class="highlight-chroma"><span class="highlight
cl">var b = new Boolean(false);
if (b) // 返回true

if (b == true) //
回false
</code></pre>
<h2 id="No4-异常处理">No4.异常处理</h2>
<h2 id="1-异常类型">1.异常类型</h2>
<p>   抛出异常语法：抛异常可以是任意类型。如下所示。</p>
<pre><code class="highlight-chroma"><span class="highlight
cl">throw "Error2"; // 字符串类型
throw 42; // 数字
型
throw true; // 布
类型
throw {toString: f
nction() { return "I'm an object!"; } }; //对象类型
</code></pre>
<p>   自定义异常：</p>
<pre><code class="highlight-chroma"><span class="highlight
cl">// 创建一个对象类型UserException
function UserExce
tion(message) {
 this.message =
message;
 this.name = "U
erException";
}

//重写toString方
，在抛出异常时能直接获取有用信息
UserException.pro
otype.toString = function() {
 return this.nam
 + ': "' + this.message + '"';
}

// 创建一个对象实
并抛出它
throw new UserEx
eption("Value too high");
</code></pre>
<h2 id="2-语法">2.语法</h2>
<p>   关键字：使用 try{}catch(e){}finally{}语法，和 C#语法相似。</p>
<p>   finally 返回值：如果 finaly 添加了 return 语句，则不管整个 try.catch 返回什么，返回值
是 finally 的 return。如下所示：</p>
<pre><code class="highlight-chroma"><span class="highlight
cl">function f() {
 try {
   console.log(0
;
   throw "bogu
";
 } catch(e) {
   console.log(1
;
<span class="highlight-line"><span class="highlight-cl">   return true; //

返回语句被暂停，直到finally执行完成
        console.log(2
; // 不会执行的代码
    } finally {
        console.log(3
;
        return false; /
覆盖try.catch的返回
        console.log(4
; //不会执行的代码
    }
    // "return false"
s executed now
    console.log(5);
/ not reachable
}
f(); // 输出 0, 1, 3;
返回 false
</code></pre>
        finally 吞并异常：如果 finally 有 return 并且 catch 中有 throw 异常。throw 的异常不会捕获，因为已经被 finally 的 return 覆盖了。如下代码所示：
<pre><code class="highlight-chroma"><span class="highlight
cl">function f() {
    try {
        throw "bogu
";
    } catch(e) {
        console.log('
aught inner "bogus"');
        throw e; // th
ow语句被暂停，直到finally执行完成
    } finally {
        return false; /
 覆盖try.catch中的throw语句
    }
    // 已经执行了"re
urn false"
}

try {
    f();
} catch(e) {
    //这里不会被执
，因为catch中的throw已经被finally中的return语句覆盖了
    console.log('ca
ght outer "bogus"');
}
// 输出
// caught inner "b
gus"
</code></pre>
        系统 Error 对象：我们可以直接使用 Error{name, message}对象，例如：throw (new Error(
The message'));
转自 <a href="https://link.ld246.com/forward?goto=http%3A%2F%2Fmp.weixin.qq.com

2Fs%3F__biz%3DMjM5MzMyNzg0MA%3D%3D%26mid%3D2650402498%26idx%3D2%26sn3D7f8702fa8d5eb0723db8a7f912e06a6e%26scene%3D0%23wechat_redirect" target="_blank" rel="nofollow ugc">WEB 开发者</p>

原文链接：面试开发常用的 JavaScript 知识点总结