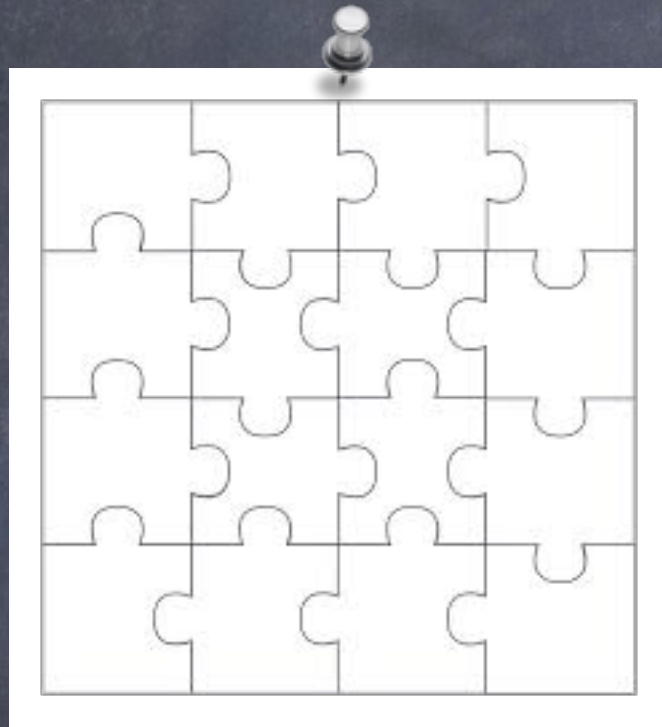


Javascript Basics

fanzhongkai@baidu.com



Puzzle



Important

JavaScript

ECMAScript

DOM

BOM

Rendering Engine

Webkit (KHTML)

Blink

Trident

Gecko

~~Presto~~

JavaScript Engine

V8

SpiderMonkey

*Tool – web inspector

- console.info (log, dir)
- debugger (break point)

Identifiers

- The first character must be a letter, an underscore (`_`, `underscore`), or a dollar sign (`$`, `jquery`).
- All other characters may be letters, underscores, dollar signs, or numbers.

*strict mode

- ECMAScript 5
- this is a pragma that tells supporting JavaScript engines to change into strict mode.

Statements

- Using code blocks with control statements

```
if (test)
    alert(test);    //valid, but error-prone and should be avoided

if (test){          //preferred
    alert(test);
}
```

; and {}, DO NOT OMIT THEM!

*Variables

• Global vs Local – var

```
function test(){  
    var message = "hi"; //local variable  
}  
test();  
alert(message); //error!
```

```
function test(){  
    message = "hi"; //global variable  
}  
test();  
alert(message); //"hi"
```

• Declare – assign

Data types

*Data types

Undefined Null Boolean Number String
object

typeof (operator not function)

- "undefined" if the value is undefined
- "boolean" if the value is a Boolean
- "string" if the value is a string
- "number" if the value is a number
- "object" if the value is an object (other than a function) or null
- "function" if the value is a function

*Undefined

- The Undefined type has only one value, which is the special value undefined. When a variable is declared using var but not initialized, it is assigned the value of undefined as follows:

```
var message;  
alert(message == undefined);    //true
```

Puzzle:

typeof(a) == "undefined"

a ---- declared ? not assigned value?

both are possible

It is advisable to **always initialize variables.**

Null

- Empty object pointer
- Any time an object is expected but is not available, null should be used in its place

Boolean

- Boolean(), convert value to true or false

DATA TYPE	VALUES CONVERTED TO TRUE	VALUES CONVERTED TO FALSE
Boolean	true	false
String	Any nonempty string	" " (empty string)
Number	Any nonzero number (including infinity)	0, NaN (See the "NaN" section later in this chapter.)
Object	Any object	null
Undefined	n/a	undefined

Number

- octal (0), hex (0x)
- Float
 - ECMAScript always looks for ways to convert values into integers
 - You should never test for specific floating-point values
- Number.MIN_VALUE & Number.MAX_VALUE
- NaN
 - First, any operation involving NaN always returns NaN
 - Second, NaN is not equal to any value, including NaN.

Number conversions

- `Number()`
- `parseInt()`
- `parseFloat()`

String

- How many steps? (efficient)

```
var lang = "Java";  
lang = lang + "Script";
```

- Converting to a string

- toString()

- String()

- + ""

*Object

- “new Object” & “new Object()” both ok
- Properties and methods of object instances:
 - **constructor** — The function that was used to create the object.
 - **hasOwnProperty(propertyName)** — Indicates if the given property exists on the object instance (not on the prototype).
 - **isPrototypeOf(object)** — Determines if the object is a prototype of another object.
 - **propertyIsEnumerable(propertyName)** — Indicates if the given property can be enumerated using the for-in statement (discussed later in this chapter).
 - **toLocaleString()** — Returns a string representation of the object that is appropriate for the locale of execution environment.
 - **toString()** — Returns a string representation of the object.
 - **valueOf()** — Returns a string, number, or Boolean equivalent of the object. It often returns the same value as **toString()**.

Operator

Unary operators

• ++, --



"1" -- "1";

• +, - (Number())

```
var s1 = "2";
var s2 = "z";
var b = false;
var f = 1.1;
var o = {
  valueOf: function() {
    return -1;
  }
};

s1++; //value becomes numeric 3
s2++; //value becomes NaN
b++; //value becomes numeric 1
f--; //value becomes 0.10000000000000009 (due to floating-point inaccuracies)
o--; //value becomes numeric -2
```


Boolean operators

- `!!` equals `Boolean()`
- `&&` operator is a short-circuited operation
- `||`

Equality operators

- identically equal & equal (===, ==, !==, !=)

EXPRESSION	VALUE
<code>null == undefined</code>	<code>true</code>
<code>"NaN" == NaN</code>	<code>false</code>
<code>5 == NaN</code>	<code>false</code>
<code>NaN == NaN</code>	<code>false</code>
<code>NaN != NaN</code>	<code>true</code>
<code>false == 0</code>	<code>true</code>
<code>true == 1</code>	<code>true</code>
<code>true == 2</code>	<code>false</code>
<code>undefined == 0</code>	<code>false</code>
<code>null == 0</code>	<code>false</code>
<code>"5" == 5</code>	<code>true</code>

Others

- Bitwise (\sim , $\&$, $|$, \wedge , \ll , \gg , \ggg)
- Multiplicative ($*$, $/$, $\%$)
- Additive ($+$, $-$)
- Relational ($>$, \geq , \leq , $<$)
- Conditional ($?:$)
- Assignment ($=$)
- Comma ($,$)
- force ——— $()$

Statements

*Expression Statement

expression?

1. General expression statement

```
1 + 2 + 3;  
3 * (1 + 2 + 3);  
void(1 + 2 + 3);  
eval('1 + 2 + 3;');  
2;  
;
```


*Expression Statement


2. Assignment statement

`a = 1 + 2;`

*Expression Statement

3. Declaration statement

`var a = 1 + 2;`



```
var a = (1, 2, 3);  
a = 1, 2, 3;
```


*Expression Statement

4. Function call statement

```
function a() {}  
a();  
  
a = function() {}  
a();  
  
(function() {  
})();  
  
(function() {  
})();  
  
void function() {  
}();
```

(...)

for-in

- Object properties in ECMAScript are unordered
- null, undefined

label

```
var num = 0;

outermost:
for (var i=0; i < 10; i++) {
    for (var j=0; j < 10; j++) {
        if (i == 5 && j == 5) {
            break outermost;
        }
        num++;
    }
}

alert(num);    //55
```


with

```
with(location){  
    var qs = search.substring(1);  
    var hostName = hostname;  
    var url = href;  
}
```

In strict mode, the with statement is not allowed and is considered a syntax error

Others

- if
- do-while
- while
- for
- break/continue
- switch

Catch Exception

try
catch
finally
throw

*Return Value

no return value:
;
var a = b;
function a() {}
break/continue/
label

```
eval('1+2;var x=5;;;;function f() {}');
```

```
eval('1+2;var x=5;;function f() {}void 0')
```


Functions

*Functions – arguments

- “like” an array
- arguments.length (number of arguments passed in)
- All arguments in ECMAScript are passed by value.
- modify arguments... will...

Values in the arguments object are automatically reflected by the corresponding named arguments,

- This doesn't mean that both access the same memory space, though; their memory spaces are separate but happen to be kept in sync.
- This effect goes only one way: changing the named argument does not result in a change to the corresponding value in arguments.

The length of the arguments object is set based on the number of arguments passed in, not the number of named arguments listed for the function.

```
function b (x, y, a) {  
    arguments[2] = 10;  
    console.log(a);  
}  
b(1,2,3);
```

```
function foo(a, b) {  
    arguments[1] = 2;  
    alert(b);  
}  
foo(1);
```


GOOD JOB
— from 陈华榕

1.1 ARGUMENTS

【问题】写出 `console log` 的值。说出为什么结果会是这样。该测试题在 JS 解析引擎的任何模式下都会有如是结果么？

【回答】在浏览器控制台执行分别以下代码：

```
function b(x,y,a) { arguments[2] = 10; console.log(a); } b(1,2,3);  
function b(x,y,a) { "use strict"; arguments[2] = 10; console.log(a); }  
b(1,2,3);
```

对应是普通模式与严格模式。在各种浏览器下的执行结果如下（对于不支持审查元素、开发人员工具或类似功能的浏览器，通过 `alert` 或 `document.write` 输出 `console.log` 的结果）：

浏览器	普通模式	严格模式
Chrome	10	3
Firefox	10	3
Opera	10	3
IE10	10	3
IE9	10	10
IE8	10	10
IE7	测试失败	测试失败
IE6	10	10

可以看到，除了 IE7 测试失败外，其余浏览器在普通模式下均给出了相同的结果，对 `arguments[2]` 的修改等同于对 `a` 的修改。

IE9 及以前版本不支持严格模式，故给出的结果不变。在其他支持严格模式的浏览器中，严格模式下都给出了相同的结果，对 `arguments[2]` 的修改并没有影响 `a`。

出现以上现象，是因为在普通模式下，`arguments[2]` 与 `a` 其实是同一个变量，而在严格模式下，它们成了初始值相同，但完全独立的两个变量。通过以下代码进行单步执行，并 watch `arguments` 和 `a`，可以验证这一点：

```
function b(x,y,a) {  
    // "use strict"; // 去掉本行首的双斜杠可启用严格模式  
    arguments[2] = 10;  
    a = 8;  
    document.write(a);  
}  
b(1,2,3);
```


No overloading

```
function addSomeNumber(num){ return num + 100;
}  
function addSomeNumber(num) { return num + 200;
}  
var result = addSomeNumber(100);
```


使用模拟函数重载来编写一个具有如下功能的函数：

- 1，如果输入参数大于三个，返回最后一个参数。
- 2，如果输入参数小于三个且全部为数字，则返回排序后的数组，如果最后一个数为奇数则降序排列，反之升序排列。
- 3，如果输入参数小于三个且包含字符串，则将所有参数强制转化为字符串联接返回。

GOOD JOB
— from 陈华榕

```
function myFunc() {  
    var arguLen = arguments.length;  
    if (arguLen > 3) {    // 第 1 种情况  
        return arguments[arguLen - 1];  
    }  
    else if (arguLen < 3) {  
        var numFlag = true;    // 用于判断是否均为数字  
        var strFlag = false; // 用于判断是否包含字符串  
        for (var i = 0; i < arguLen; ++i) {  
            if (typeof arguments[i] != "number") {  
                numFlag = false;  
            }  
            if (typeof arguments[i] === "string") {  
                strFlag = true;  
                break;  
            }  
        }  
        if (numFlag) {    // 第 2 种情况  
            var args = [].slice.call(arguments, 0);    // 转成数组  
            if (args[arguLen - 1] % 2 == 0) {  
                return args.sort(function(a,b){return a-b;}); // 升序  
            }  
            else {  
                return args.sort(function(a,b){return b-a;}); // 降序  
            }  
        }  
        else if (strFlag) { // 第 3 种情况  
            var result = "";  
            for (var i = 0; i < arguLen; ++i) {  
                result += String(arguments[i]);  
            }  
            return result;  
        }  
    }  
    return;  
}
```