JAVASCRIPTCORE


JSVALUE


家


知

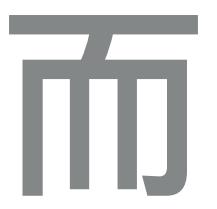
J

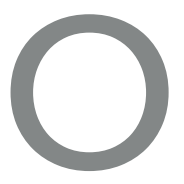

的

,

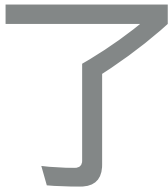
就

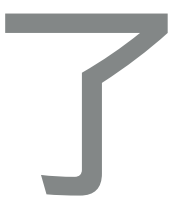

什





t





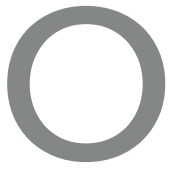

种

,

就


被

的



t


和

J

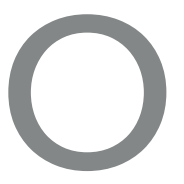

起

,

就


的

,




和


的


换


换


冬

```
<pre>
@textblock
   Objective-C type  |   JavaScript type
 --------------------+---------------------
         nil         |     undefined
        NSNull       |       null
       NSString      |       string
       NSNumber      |   number, boolean
     NSDictionary    |   Object object
       NSArray       |    Array object
        NSDate       |     Date object
       NSBlock (1)   |   Function object (1)
         id (2)      |   Wrapper object (2)
        Class (3)    | Constructor object (3)
@/textblock
</pre>
```

- 大家知道JS里面是弱类型的，也就是只有在代码执行时才能知道一个变量具体是什么类型，而Objective-C是强类型了，为了处理这种类型差异，JSValue就被引入了。JSValue的作用就是在Objective-C对象和JavaScript对象之间起转换作用

- JS对象在JSVirtualMachine中的一个强引用，其实就是Hybrid对象。我们对JS的操作都是通过它。并且每个JSValue都是强引用一个context同时，OC和JS对象之间的转换也是通过它，相应的类型转换如图：

JSVALUE

- 大家知道JS里面是弱类型的，也就是只有在代码执行时才能知道一个变量具体是什么类型，而Objective-C是强类型了，为了处理这种类型差异，JSValue就被引入了。JSValue的作用就是在Objective-C对象和JavaScript对象之间起转换作用

- JS对象在JSVirtualMachine中的一个强引用，其实就是Hybrid对象。我们对JS的操作都是通过它。并且每个JSValue都是强引用一个context同时，OC和JS对象之间的转换也是通过它，相应的类型转换如图：

```
<pre>
@textblock
   Objective-C type   |   JavaScript type
 ---------------------+---------------------
         nil          |     undefined
        NSNull        |        null
       NSString       |       string
       NSNumber       |   number, boolean
     NSDictionary     |   Object object
       NSArray        |    Array object
        NSDate        |     Date object
       NSBlock (1)    |   Function object (1)
         id (2)       |   Wrapper object (2)
        Class (3)     | Constructor object (3)
@/textblock
</pre>
```

OC 和 JAVASCRIPT 通讯