

IOS时代的JAVASCRIPT与OC通信


JAVASCRIPT 头文件分析

- ▶ JSContext : 提供JavaScript的运行需要的环境
- ▶ JSValue : JavaScript对象
- ▶ JSManagedValue : 以保证在Objective-C和JavaScript两侧都可以正确访问对象 (内存, 线程安全)
- ▶ JSVirtualMachine : 为JavaScript的运行提供了底层资源有自己独立的堆栈以及垃圾回收机制, 而且通过锁来实现线程安全, 如果需要并发执行js代码, 可以创建不同的JSVirtualMachine虚拟机对象来实现
- ▶ JSExport : 一个神奇的协议, 用于OC对象继承声明protocol后在JS中调用相对应的方法, 属性

JAVASCRIPTCORE 头文件 分析

- ▶ JSContext : 提供JavaScript的运行需要的环境
- ▶ JSValue : JavaScript对象
- ▶ JSManagedValue : 以保证在Objective-C和JavaScript两侧都可以正确访问对象 (内存, 线程安全)
- ▶ JSVirtualMachine : 为JavaScript的运行提供了底层资源有自己独立的堆栈以及垃圾回收机制, 而且通过锁来实现线程安全, 如果需要并发执行js代码, 可以创建不同的JSVirtualMachine虚拟机对象来实现
- ▶ JSExport : 一个神奇的协议, 用于OC对象继承声明protocol后在JS中调用相对应的方法, 属性

JSCONTEXT



```
var globalVar = "level0"  
function fun1(){  
  var value1 = "level1";  
  var fun2 = function(){  
    var value2 = "level2";  
  }  
}
```