IOS7 时代的 JAVASCRIPT 与 OC 通信

JSEXPORT PROTOCOL

```objc
//在VC中进行测试
@interface ViewController () <JSExportTest>

@property (nonatomic, strong) JSProtocolObj *obj;
@property (nonatomic, strong) JSContext *context;

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    //创建context
    self.context = [[JSContext alloc] init];
    //设置异常处理
    self.context.exceptionHandler = ^(JSContext *context, JSValue *exception) {
        [JSContext currentContext].exception = exception;
        NSLog(@"exception:%@",exception);
    };
    //将obj添加到context中
    self.context[@"OCObj"] = self.obj;
    //JS里面调用Obj方法，并将结果赋值给Obj的sum属性
    [self.context evaluateScript:@"OCObj.sum = OCObj.addB(2,3)"];

}

```


的

,


赋

S

的


协


的

,

J

S


B

t

S


换


成

S


的

JSEXPORT PROTOCOL

```
109  //在VC中进行测试
110  @interface ViewController () <JSExportTest>
111
112  @property (nonatomic, strong) JSProtocolObj *obj;
113  @property (nonatomic, strong) JSContext *context;
114
115  @end
116
117  @implementation ViewController
118
119  - (void)viewDidLoad {
120      [super viewDidLoad];
121      //创建context
122      self.context = [[JSContext alloc] init];
123      //设置异常处理
124      self.context.exceptionHandler = ^(JSContext *context, JSValue *exception) {
125          [JSContext currentContext].exception = exception;
126          NSLog(@"exception:%@",exception);
127      };
128      //将obj添加到context中
129      self.context[@"OCObj"] = self.obj;
130      //JS里面调用Obj方法，并将结果赋值给Obj的sum属性
131      [self.context evaluateScript:@"OCObj.sum = OCObj.addB(2,3)"];
132
133  }
134
```

在JS中进行调用这个对象的方法，并将结果赋值sum。唯一要注意的是OC的函数命名和JS函数命名规则问题。协议中定义的是add: b:，但是JS里面方法名字是addB(a,b)。可以通过JSExportAs这个宏转换成JS的函数名字

JSEXPORTAS