<http://blog.csdn.net/myaccella/article/details/6956534>

由于QWebFrame 提供了一个addToJavaScriptWindowObject()接口， 所以在QT Webkit下添加一个自定义的  
Javascript对象就变得非常的容易，要注意在跨页面的时候，该对象会被清除，所以你只要重新加回来  
就可以了， 这一点跟GTK类似， 只要捕获WindowObjectCleared信号，在该信号处理函数里面执行  
addToJavaScriptWindowObject()就可以了。

这种方法不用去修改Webkit的核心代码WebCore或者JavascriptCore, 所有工作都在应用层面实现，  
可以说是简单又实用。

具体可以参考如下源代码：

**[html]** [view plain](http://blog.csdn.net/myaccella/article/details/6956534) [copy](http://blog.csdn.net/myaccella/article/details/6956534)

1. // 实现一个新类
2. class QFoo : public QObject
3. {
4. Q\_OBJECT
5. public:
6. QFoo(QObject \*parent) {}
7. void initialize(QWebView \*wv);
9. public slots:
10. void print(const QString &msg);
12. private slots:
13. void addObject();
15. private:
16. QWebFrame \*wf;
17. };

20. void QFoo::initialize(QWebView \*wv)
21. {
22. wf = wv-**>**page()-**>**mainFrame();
23. // Add to the javascript window object
24. addObject();
25. // connect the signal for javascript window object cleared, which happened
26. // at page switching etc.
27. connect(wf, SIGNAL(javaScriptWindowObjectCleared()), this, SLOT(addObject()));
28. }
30. void QFoo::addObject()
31. {
32. wf-**>**addToJavaScriptWindowObject(QString("QFoo"), this);
33. }
35. void QFoo::print(const QString &msg)
36. {
37. qDebug() **<<** "Output the message " **<<** **msg**;
38. }

41. // 测试主函数
42. int main(int argc, char \*argv[])
43. {
44. QApplication test(argc, argv);
46. QWebView \*wv = new QWebView();
48. QFoo \*foo = new QFoo(wv);
49. foo-**>**initialize(wv);
51. wv-**>**setUrl(QUrl("foo.html"));
52. wv-**>**show();
54. return test.exec();
55. }

HTML测试代码很简单：

<script>  
QFoo.print("Hello world!");  
</script>