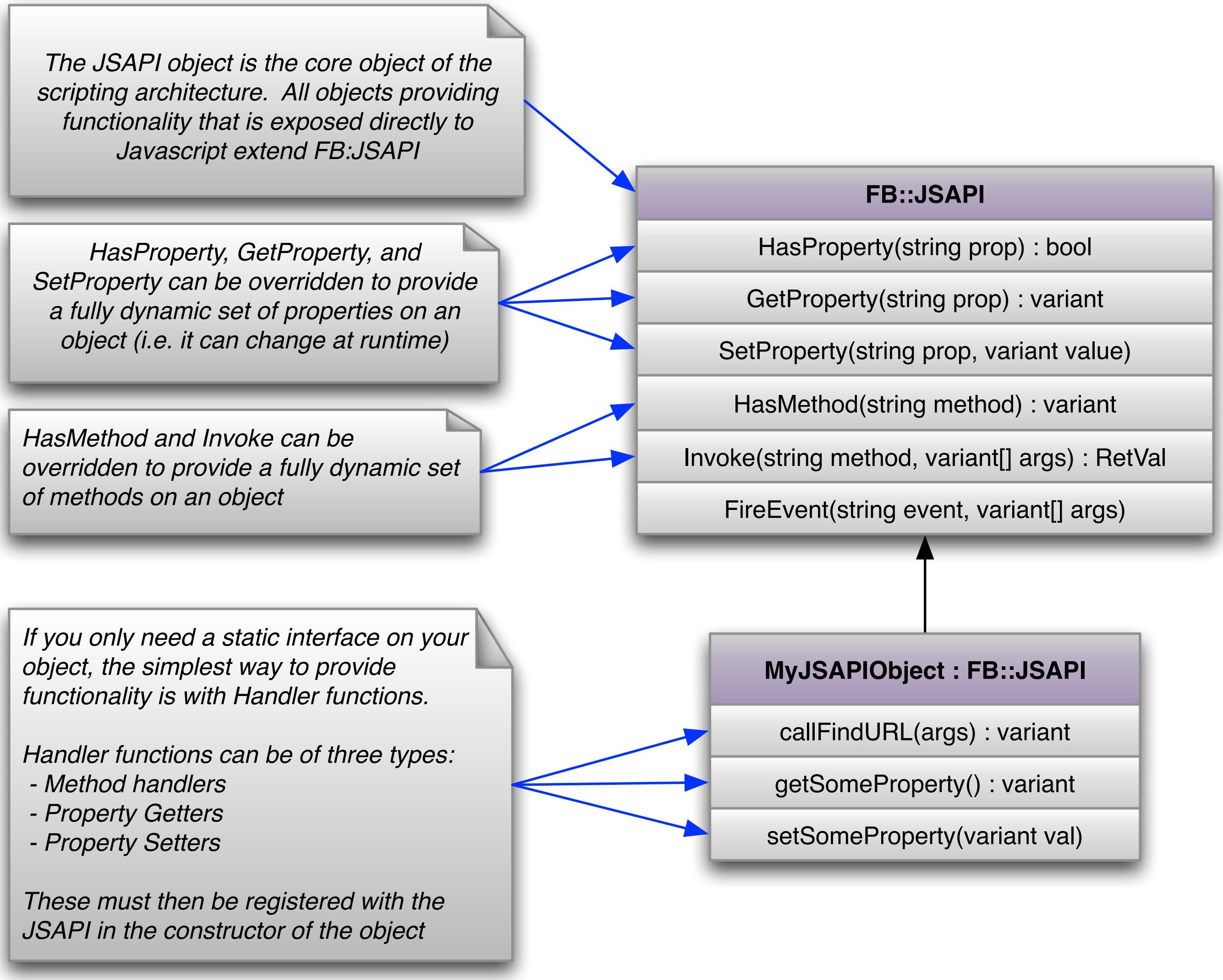
JSAPI (Javascript API) class structure



The JSAPI object is the core object of the scripting architecture. All objects providing functionality that is exposed directly to Javascript extend FB:JSAPI

HasProperty, GetProperty, and SetProperty can be overridden to provide a fully dynamic set of properties on an object (i.e. it can change at runtime)

HasMethod and Invoke can be overridden to provide a fully dynamic set of methods on an object

FB::JSAPI

HasProperty(string prop) : bool

GetProperty(string prop) : variant

SetProperty(string prop, variant value)

HasMethod(string method) : variant

Invoke(string method, variant[] args) : RetVal

FireEvent(string event, variant[] args)

If you only need a static interface on your object, the simplest way to provide functionality is with Handler functions.

Handler functions can be of three types:
- Method handlers
- Property Getters
- Property Setters

These must then be registered with the JSAPI in the constructor of the object

MyJSAPIObject : FB::JSAPI

callFindURL(args) : variant

getSomeProperty() : variant

setSomeProperty(variant val)

```
MyJSAPIObject::MyJSAPIObject(void)
{
   registerMethod( "toString", (CallMethodPtr)&MyJSAPIObject::callFindURL );

   registerProperty( "valid", (GetPropPtr)&MyJSAPIObject::getSomeProperty,
                              (SetPropPtr)&MyJSAPIObject::setSomeProperty);
}
```

From any method inside MyJSAPIObject, you can fire an event by calling the FireEvent method:

```
this->FireEvent("onload", vector_list<FB::variant>("String param", 4, 23.3));
```

All event handlers that have been attached to this object from Javascript will be fired with the 3 specified parameters.