Nood.js C++ 模块 addon插件（二） - 程序园

上篇介绍了Node.js C++模块中编码部分的介绍。在此之前先罗列一些这方面的资料供参考入门。

v8数据结构的手册

node C++模块入门

从C++的角度了解node

（google 百度去，这些资料可以引导我们写出一个完整的Node.js的C++扩展了。但是也许下面的内容能够让你写的过程中更轻松。）

首先我们来理解

函数参数

C++模块中可以被node调用的方法都是如下形式的

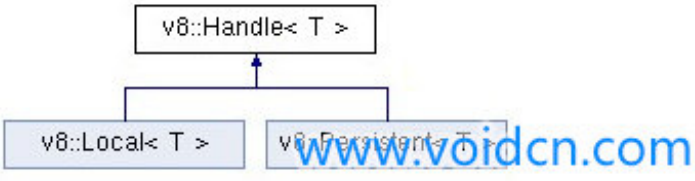
```
Handle<Value> Method(const Arguments& args)
{

}
```

传入参数args对象有两个常用的操作：

```
Local<Value> arg = args[0];
int length = args.Length();
```

在上述的代码中出现了local ，v8的继承关系如下:

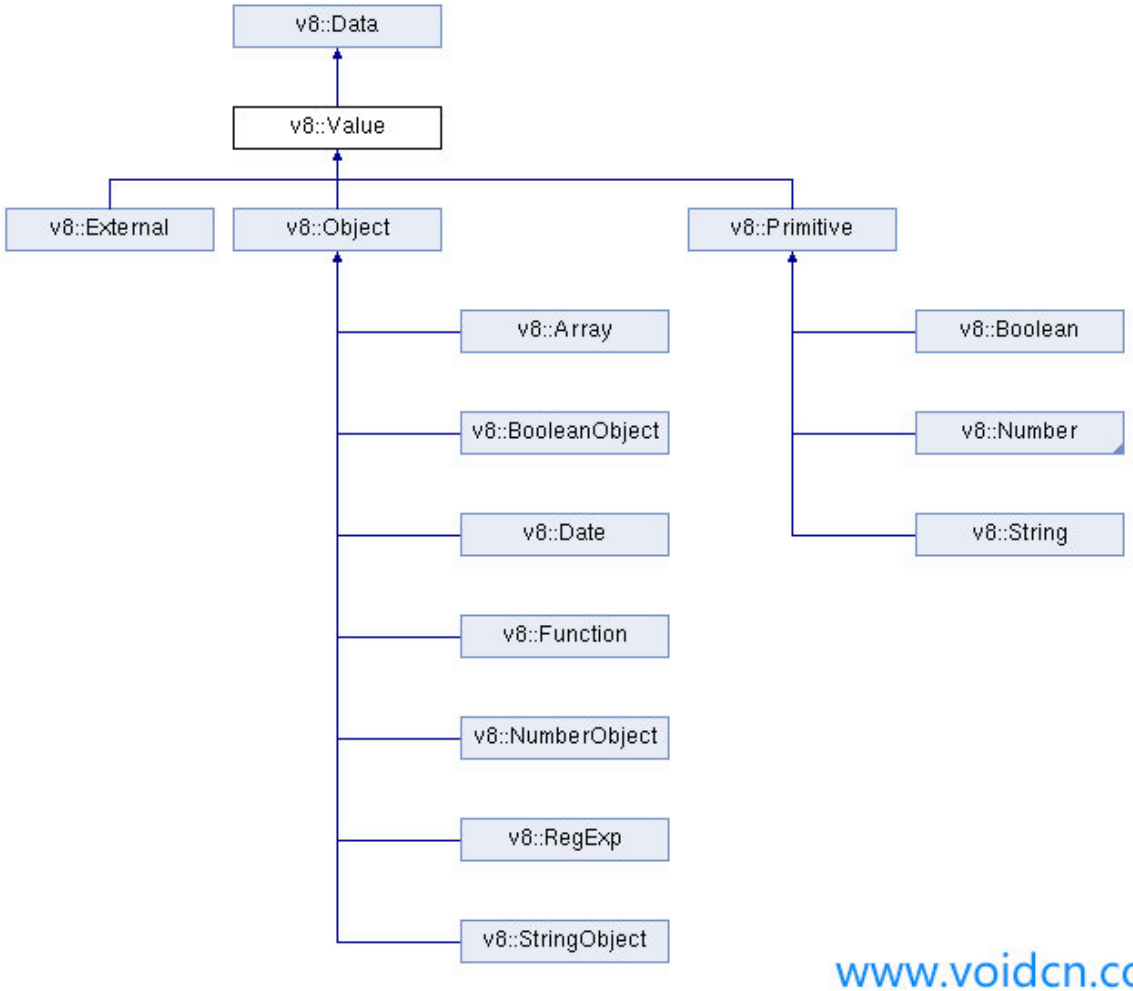


Handle是v8维护一个对象引用，v8会负责对象的回收，可以看作是v8中的智能指针。local世纪城自它的一个轻量

级的对象引用

数据类型转换

v8中整体的数据结构关系图如下：



可以看到，所有的*String, Number, Boolean, Object, Array*等对象都是从*Value*继承而来。因此从*Arguments*中获取的*Local*对象可以轻松的判断其js中的具体类型，并进行转换。

类型判断：

```
Local<Value> arg = args[0];
bool isArray = arg->IsArray();
bool isBoolean = arg->IsBoolean();
bool isNumber = arg->IsNumber();
bool isInt32 = arg->IsInt32();
```

v8提供了一系列的接口用来做类型判断，可以在其文档内找到所有的判断接口。

类型转换：

在经过类型判断之后，就可以根据结果进行类型的转换了：

```
Local<Value> arg = args[0];
Local<Object> = arg->ToObject();
Local<Boolean> = arg-
>ToBoolean();
Local<Number> = arg->ToNumber();
Local<Int32> = arg->ToInt32 ();
```

同样的，v8提供了一系列的接口用来做类型转换，可以在其文档内找到所有的转换接口。注意，v8并没有提供直接的从Value到Array的转换，但是我们发现，Array是继承自Object的，而其实Array对象并没有提供比Object更多的接口。联系到js中，会发现，js中的Array和Object操作也是一样的相似。因此尽管v8没有提供从Value到Array的转换，但是转换成Object就已经足够了,因为完全可以把Array当作一个Object来操作。

而所有的v8中的Boolean/Number/Int32等对象都有方法转换成C++原生的bool/double/int等类型。当然，同样有反过来转换的接口。因此从javascript与C++跨语言的数据类型转换就完全不是问题了。

Object与Array

基础类型相对来说比较简单，而Object和Array相对来说需要更多的接口方法来进行设置和内容的获取。

```
Handle<v8::Object> v8Obj = v8::Object::New();
v8Obj->Set(v8::String::NewSymbol("key"), v8::Integer::New(1));

if (v8Obj->Has(v8::String::New("key"))) {
  Handle<v8::Value> value = v8Obj-
>Get(v8::String::New("key"));

Handle<Array> keys = v8Obj->GetOwnPropertyNames();

int len = keys->Length();

v8Obj->Delete(v8::String::New("key"));
```

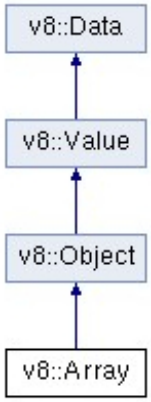
是不是觉得上面的代码十分眼熟？

```
var obj = {};
obj.key = 1;
if (obj.hasOwnProperty('key')) {
  var value = obj.key;
}
var keys = Object.keys(obj);
var len = keys.length;
delete obj.key;
```

可以看到，js的代码完整的映射到了C++代码之上。

而Array也是一样的，先用一张图描述Array和Object的关系：

在V8中，Array的接口基本就只是多了一个*Length*方法，获取数组的长度，而其他的方法都是继承自Object，所以Array的操作和Object非常类似。




```
Handle<v8::Array> v8Arr =
v8::Array::New(length);
int length = 10;
for (int i = 0; i != length; ++i) {
  v8Arr->Set(i, v8::Integer::New(i));
}

Handle<Value> item = v8Arr->Get(10);
```

至此，js中的数据结构对应到v8中的部分基本已经罗列完毕，js与c++的数据交换也完全不是问题了，此时用c++写Node.js扩展基本已经没有问题了