分别为经典语法和类c原型方式。

* **经典语法**

定义函数名，返回参数类型，入参类型`constprintf=lib.func('printf','int',['str','...']);`

* **类C语法**

在类中定义方法类似，`lib.func('int printf(const char *fmt, ...)');`

推荐使用`类C语法`更加方便，不受参数的限制，更易于修改。

DLL调用类型
---

1. 同步调用

本机函数，您就可以像调用任何其他 JS 函数一样简单地调用它。

```
const atoi = lib.func('int atoi(const char *str)');
let value = atoi('1257');
```

2. 异步调用

有一些耗时的操作，可以使用异步调用回调的方式处理。

```
```

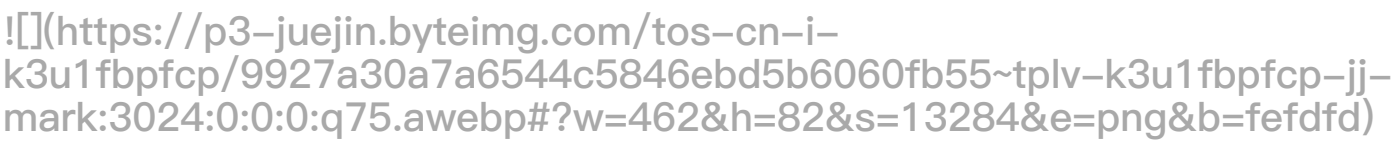
```
const atoi = lib.func('int atoi(const char *str)');

atoi.async('1257', (err, res) => {
    console.log('Result:', res);
})

```

JS类型值传递

JS基础类型时不支持值传递的，遇到需要传递指针变量时，直接调用是无法获取到变更后的值。相应的koffi也提供了非常方便的值包装。

1. 数组包装

项目中采用比较方便的数组包装来进行值传递。包装基础对象到数组中，变更后取出第一位就能获取到变更后的值。



需要定义返回的值的获取长度，防止出现只获取到部分的返回结果。

2. 引用类型包装

把基础类型包装成引用对象。传递到函数中。

```
let cardsenr = koffi.alloc('int', 64);
let cardRet = dcCard(icdev, 0, c sult:', result);

    return 'hello electron-egg';
  }
}

ExampleController.toString = () => '[class ExampleController]';
module.exports = ExampleController;
```

```

JS前端
====

通过ipc的方式，乡调用api一样调用node后端接口。

定义路由
----

```
import { ipc } from '@/utils/ipcRenderer';

const ipcApiRoute = {
  test: 'controller.d8.test',
  init: 'controller.d8.init',
  reset: 'controller.d8.reset',
  exit: 'controller.d8.exit',
  command:'controller.d8.command'
}
```

调用
--

```
ipc.invoke(this.ipcApiRoute.init).then(r => {
  // r为返回的数据
  if(r >= 0) {
    this.setInitRet(r);
    this.scrollToBottom("连接成功，返回码：" + r);
    this.connectStr = "连接成功";
    this.setDeviceStatus('success');
  } else {
    this.scrollToBottom("连接失败，返回码：" + r);
  }
})
```
```

通过ipc的invode方法调用方法即可。后续就可以愉快的编写我们的业务代码了。

参照
==

[Koffi帮助文档](http://cxyroad.com/”https://nongchatea.gitbook.io/koffi-chinese/javascript-hui-tiao”)

[electron-egg帮助文档](http://cxyroad.com/”https://www.kaka996.com/”)