编写合约的部署脚本

上节课我们已经通过编译从 solidity 源码得到了字节码，接下来我们会试图完成一个自动化脚本，将合约部署到区块链网络中。

部署的必要条件

与以太坊节点的通信连接

我们需要启动一个以太坊节点，连接到想要的网络，然后开放 HTTP-RPC 的 API（默认 8545 端口）给外部调用；或者也可以用第三方提供的可用节点入口，以太坊社区有人专门为开发者提供了节点服务。目前我们直接用 ganache，不需要考虑这些问题，但如果配置其它网络，这个配置就是必要的。

余额大于 0 的账户

因为以太坊上的任何交易都需要账户发起，账户中必须有足够的余额来支付手续费（Transaction Fee），如果余额为 0 部署会失败。当然，我们目前用的是 ganache，里面默认有 10 个账户，每个账户 100ETH，不存在这个问题，但如果要部署到其它网络（私链、测试网络、主网）就必须考虑这个问题。

安装依赖

搞清楚部署的必要条件之后，我们需要安装必要的依赖包。首先是 web3.js，web3.js 的 1.0.0 版本尚未发布，但是相比 0.2x.x 版本变化非常大，1.x 中大量使用了 Promise，可以结合 async/await 使用，而 0.x 版本只支持回调，因为使用 async/await 能让代码可读性更好，我们这次选择使用 1.0.0 版本。

```
npm install web3
```

编写部署脚本

做好准备工作之后，我们开始编写合约部署脚本，在 scripts 目录下新建脚本文件 deploy.js：

```
const path = require('path');
const Web3 = require('web3');
const web3 = new Web3(new Web3.providers
                      .HttpProvider('http://localhost:8545'));
// 1. 拿到 abi 和 bytecode
const contractPath = path.resolve(__dirname,
                                  '../compiled/Car.json');
const { interface, bytecode } = require(contractPath);

(async () => {
       // 2. 获取钱包里面的账户
       const accounts = await web3.eth.getAccounts();
       console.log('部署合约的账户：', accounts[0]);
       // 3. 创建合约实例并且部署
       var result = await new
                        web3.eth.Contract(JSON.parse(interface))
              .deploy({ data: bytecode, arguments: ['AUDI'] })
              .send({ from: accounts[0], gas: '1000000' });
       console.log('合约部署成功：', result);
})();
```

我们来熟悉一下 v1.0.0 版本中的部署操作。由于 1.0.0 版本中调用返回全部是 promise，所以我们这里用到了 ES7 中的 async/await 来处理所有异步调用。

第二步获取钱包账户，存为本地变量，然后选取 accounts[0] 作为部署合约的账户；我们应该确保这个账户中以太余额充足。

第三步中，我们用 promise 的链式调用完成了创建抽象合约对象、创建部署交易对象（deploy）和发送部署交易三个步骤，其中只有 send 一步是真正的异步请求调用。分开写就是这样：

```
const contract = new web3.eth.Contract(JSON.parse(interface));
const transaction = contract.deploy({ data: bytecode, arguments:
                                    ['AUDI'] });
const result = await transaction.send({ from: accounts[0], gas:
                                        1000000 });
```

运行脚本

在根目录下运行写好的部署脚本：

node scripts/deploy.js

查看结果，可以看到合约已经成功部署。我们发现返回结果有些复杂，所以可以对代码稍作改进，截取 address 返回，并计算一下部署花了多少时间：

```
const path = require('path');
const Web3 = require('web3');
const web3 = new Web3(new Web3.providers
                      .HttpProvider('http://localhost:8545'));
// 1. 拿到 bytecode
const contractPath = path.resolve(__dirname,
                                  '../compiled/Car.json');
const { interface, bytecode } = require(contractPath);

(async () => {
        // 2. 获取钱包里面的账户
        const accounts = await web3.eth.getAccounts();
        console.log('部署合约账户:', accounts[0]);
        // 3. 创建合约实例并且部署
```

```
        console.time('合约部署耗时');
        var result = await new
                  web3.eth.Contract(JSON.parse(interface))
                  .deploy({ data: bytecode, arguments: ['AUDI'] })
                  .send({ from: accounts[0], gas: '1000000' });
        console.timeEnd('合约部署耗时');
        console.log('合约部署成功:', result.options.address);
})();
```