1. Car.sol 使用版本0.4.26

|  |
| --- |
| pragma solidity >0.4.22;  contract Car{  string brand;  uint public price;  constructor(string initBrand,uint initprice) public {  brand=initBrand;  price=initprice;  }  function setBrand(string newBrand) public{  brand=newBrand;  }  function setPrice(uint newPrice) public{  price = newPrice;  }  function getBrand() public view returns(string){  return brand;  }  } |

1. 初代代币实现 使用版本0.4.26

|  |
| --- |
| pragma solidity>0.4.22;  contract Coin{  address public minter;//铸币者  mapping(address => uint) public balances;//一个哈希表映射  event Sent(address from,address to,uint amount);//一个事件  constructor()public {minter=msg.sender;}  function mint(address receiver ,uint amount) public {//铸币  require(msg.sender==minter);  balances[receiver] += amount;  }  function send(address receiver,uint amount)public{//转币  require(amount<=balances[msg.sender]);  balances[msg.sender]-=amount;  balances[receiver]+=amount; //存在安全风险 整数溢出  emit Sent(msg.sender,receiver,amount);//可以更好的监听区块链  }  } |

1. 改进版代币合约 版本0.4.26

|  |
| --- |
| pragma solidity >0.4.22;  contract Coin{  mapping(address => uint256) balances;  constructor(uint initialSupply)public{  balances[msg.sender]=initialSupply;  }  function send(address receiver,uint amount)public returns(bool success){  require(balances[msg.sender]>=amount);  require(balances[receiver]+amount>=balances[receiver]);  balances[msg.sender]-=amount;  balances[receiver]+=amount;  return true;  }  } |

ERC20代币合约：[以太坊ERC20代币合约案例 - 邹华栋 - 博客园 (cnblogs.com)](https://www.cnblogs.com/jameszou/p/10131443.html)

|  |
| --- |
| **pragma** solidity ^**0.4**.16;    **interface** tokenRecipient { **function** receiveApproval(**address** \_from, **uint256** \_value, **address** \_token, **bytes** \_extraData) **public**; }//接口怎么用？  /\*\*  \* 一份简单的ERC20代币模版  \*\*/  **contract** TokenERC20 {  **string** **public** name;  **string** **public** symbol;  **uint8** **public** decimals = 18; // 18 精度默认值  **uint256** **public** totalSupply; //总币值    **mapping** (**address** => **uint256**) **public** balanceOf; //  **mapping** (**address** => **mapping** (**address** => **uint256**)) **public** allowance;//授权传输    **event** Transfer(**address** **indexed** from, **address** **indexed** to, **uint256** value);//代币传输    **event** Burn(**address** **indexed** from, **uint256** value);//代币销毁      **function** TokenERC20(**uint256** initialSupply, **string** tokenName, **string** tokenSymbol) **public** {//构造函数  totalSupply = initialSupply \* 10 \*\* **uint256**(decimals);//按照最低单位记录 避免处理浮点数  balanceOf[msg.sender] = totalSupply;  name = tokenName;  symbol = tokenSymbol;  }    //只能在内部调用  **function** \_transfer(**address** \_from, **address** \_to, **uint** \_value) **internal** {  require(\_to != **0x0**);//不能向零地址转账  require(balanceOf[\_from] >= \_value);//必须要有钱才能转  require(balanceOf[\_to] + \_value > balanceOf[\_to]);//防止加法溢出  **uint** previousBalances = balanceOf[\_from] + balanceOf[\_to];//转钱之前的代币数额  balanceOf[\_from] -= \_value;  balanceOf[\_to] += \_value;  Transfer(\_from, \_to, \_value);  assert(balanceOf[\_from] + balanceOf[\_to] == previousBalances);//断言 后面的必须要为真，前后代币合必须相等，保证原子性  }  //外部可见的transfer  **function** transfer(**address** \_to, **uint256** \_value) **public** {  \_transfer(msg.sender, \_to, \_value);  }    //用别人的钱转给别人 授权转账  **function** transferFrom(**address** \_from, **address** \_to, **uint256** \_value) **public** **returns** (**bool** success) {  require(\_value <= allowance[\_from][msg.sender]); // Check allowance 保证授权地址给你的额度不能超过value  allowance[\_from][msg.sender] -= \_value;//从授权中先扣钱  \_transfer(\_from, \_to, \_value);//调用原来的方法  **return** true;  }    //给别人设置额度  **function** approve(**address** \_spender, **uint256** \_value) **public**  **returns** (**bool** success) {  allowance[msg.sender][\_spender] = \_value;  **return** true;  }    **function** approveAndCall(**address** \_spender, **uint256** \_value, **bytes** \_extraData) **public** **returns** (**bool** success) {//\_spender传入智能合约的地址  tokenRecipient spender = tokenRecipient(\_spender);  **if** (approve(\_spender, \_value)) {  spender.receiveApproval(msg.sender, \_value, this, \_extraData);//调用实现合约的具体代码  **return** true;  }  }    **function** burn(**uint256** \_value) **public** **returns** (**bool** success) {  require(balanceOf[msg.sender] >= \_value);  balanceOf[msg.sender] -= \_value;  totalSupply -= \_value;  Burn(msg.sender, \_value);  **return** true;  }    **function** burnFrom(**address** \_from, **uint256** \_value) **public** **returns** (**bool** success) {  require(balanceOf[\_from] >= \_value);  require(\_value <= allowance[\_from][msg.sender]);  balanceOf[\_from] -= \_value;  allowance[\_from][msg.sender] -= \_value;  totalSupply -= \_value;  Burn(\_from, \_value);  **return** true;  }  } |

插入：solidity的接口使用：

