solidity自学：第四天

目录

内存与区块链: storage与memory

在函数体内定义一个可变长度数组时，它默认的类型是storage，地址拷贝

```
function test(uint[] arry) returns(uint){
        arrx = arry;//将内存的arry拷贝到区块链上的arrx变量
        //当我们在函数体内部定义了一个可变长度数组时，实际上，它默认的类型是storage类型，
        //它指向了区块链上的arrx。
        //所以当我们修改Z的元素的时候，我们实际上在操作区块链上的arrx
        uint[] storage Z = arrx;//通过指针实际上修改了区块链上arrx的值
        Z[0] = 100;
        //通过指针实际上修改了区块链上arrx的长度，说明Z和arrx其实是一样的。
        //操作Z的时候，会改变arrx的值
        Z.length = 1000;
    }
```

结构体 定义与初始化

```
    struct student{
        uint grade;
        string name;

        mapping(uint => string) map;//在初始化结构体的时候可以忽略掉
    }
    student aa;
    function Init() returns(uint,string,string){
        //student s = student(100,"Mike");
        //这里【student(100,"Mike")】默认的是memory类型，它无法赋值给storage类型的student s
        //应该改成：
        student memory s = student(100,"Mike");
        //s.map[52] = "helloWorld";  无法这么做
        //原因：memory的对象不可以直接操作struct结构体中的mapping
        //解决方法如下：
        aa = s;
        aa.map[10] = "helloWorld";
        return(s.grade,s.name,aa.map[10]);
        //结果："0": "uint256: 100",
        //      "1": "string: Mike",
        //      "2": "string: helloWorld"
    }
```

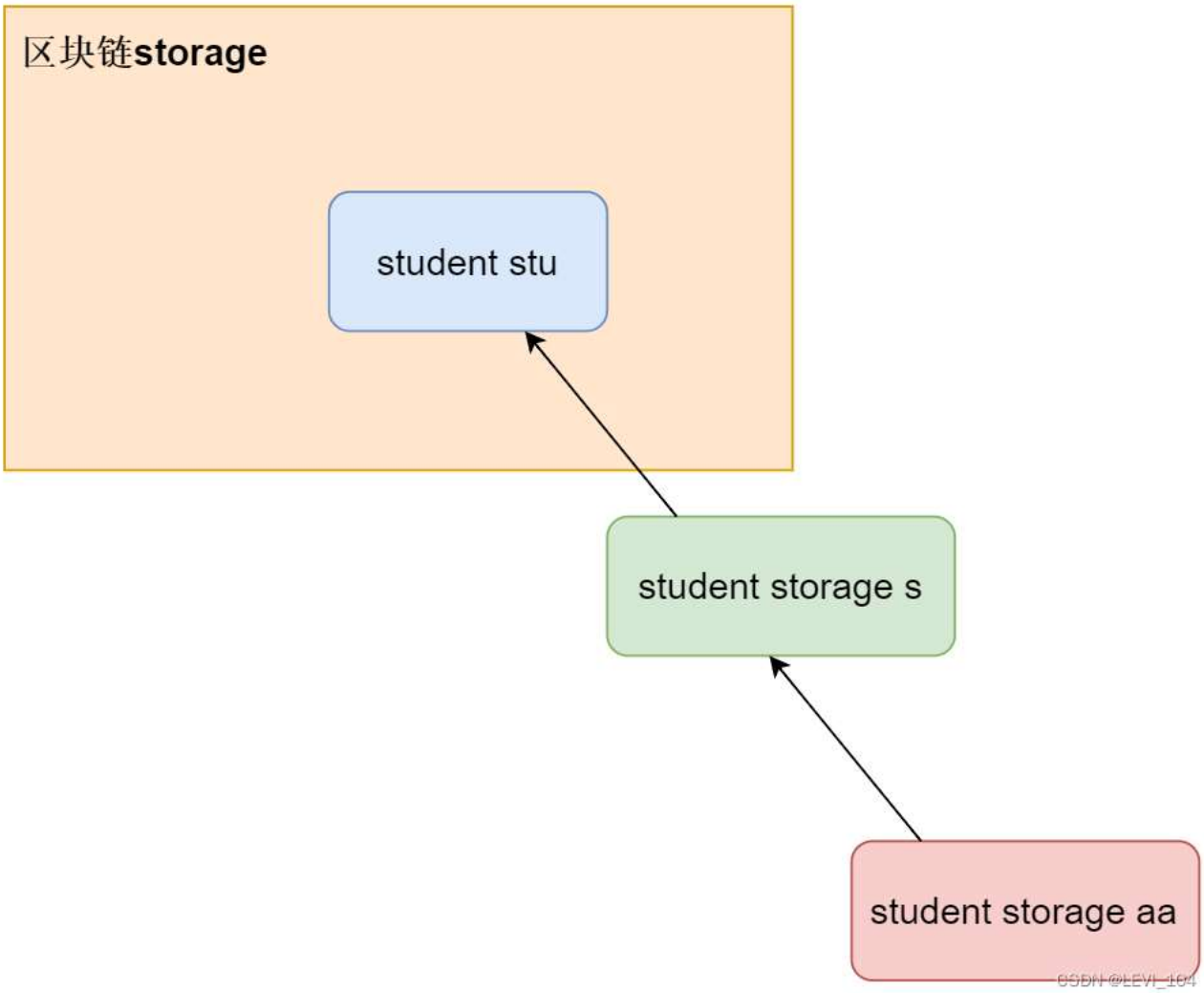
结构体作为函数参数

```
    struct student{
        uint grade;
        string name;
    }
    //如果形参是struct类型，那么一定要加internal修饰符
    function test(student s) internal{}
```

storage转storage

```
    struct student{
```

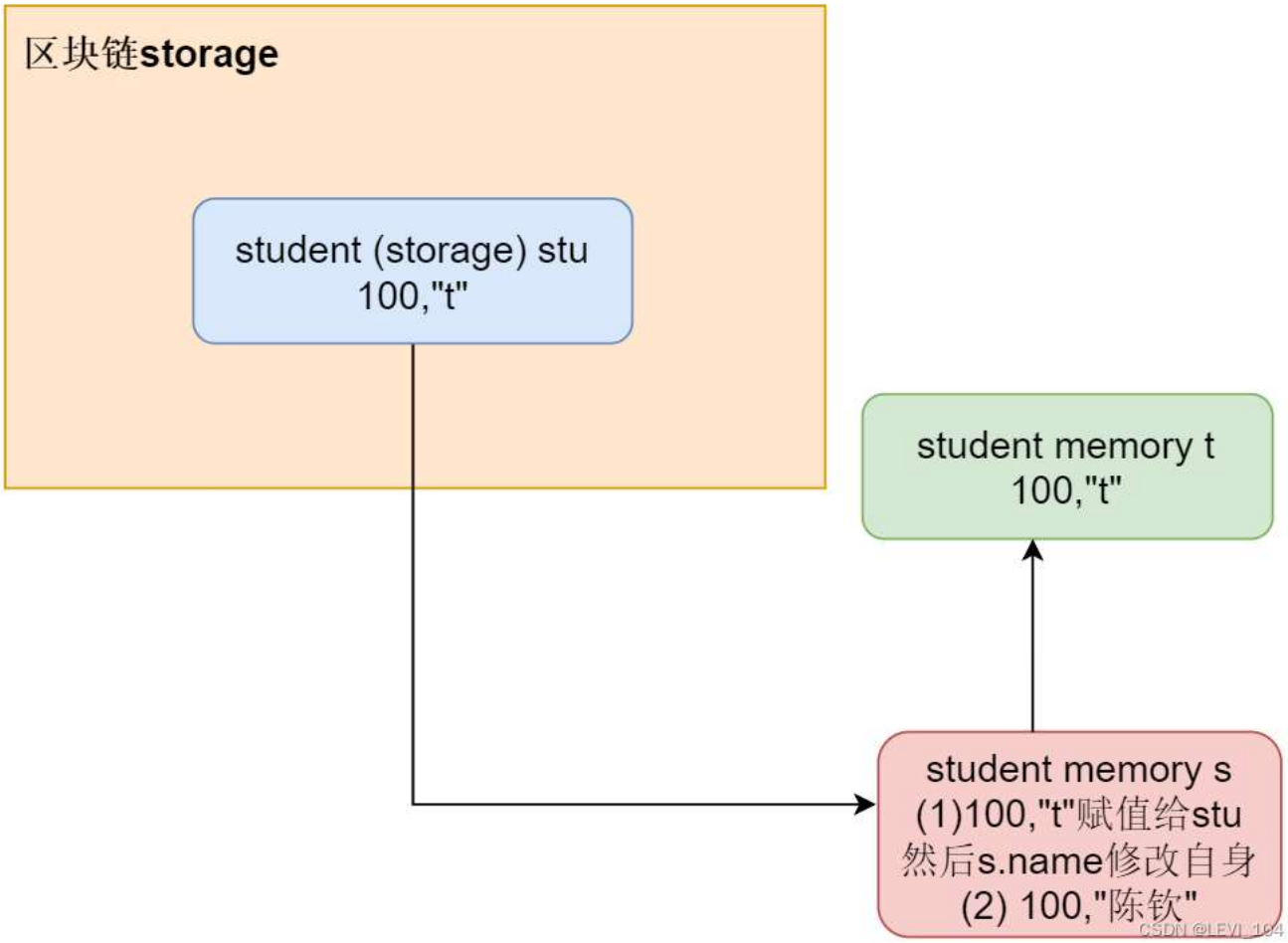
```
        uint grade;
        string name;
    }
    student stu;//这里是在区块链中开辟了一个storage
    //如果形参是struct类型，那么一定要加internal修饰符
    function test(student storage s) view internal{
        //student storage s: s是指针，用来指向区块链storage中的stu
        //                   然后下面修改就是修改指针指向的stu
        student storage aa = s;
        aa.name = "陈钦";
    }
    function call() returns(string){
        test(stu);
        return stu.name;
    }
```



memory转storage

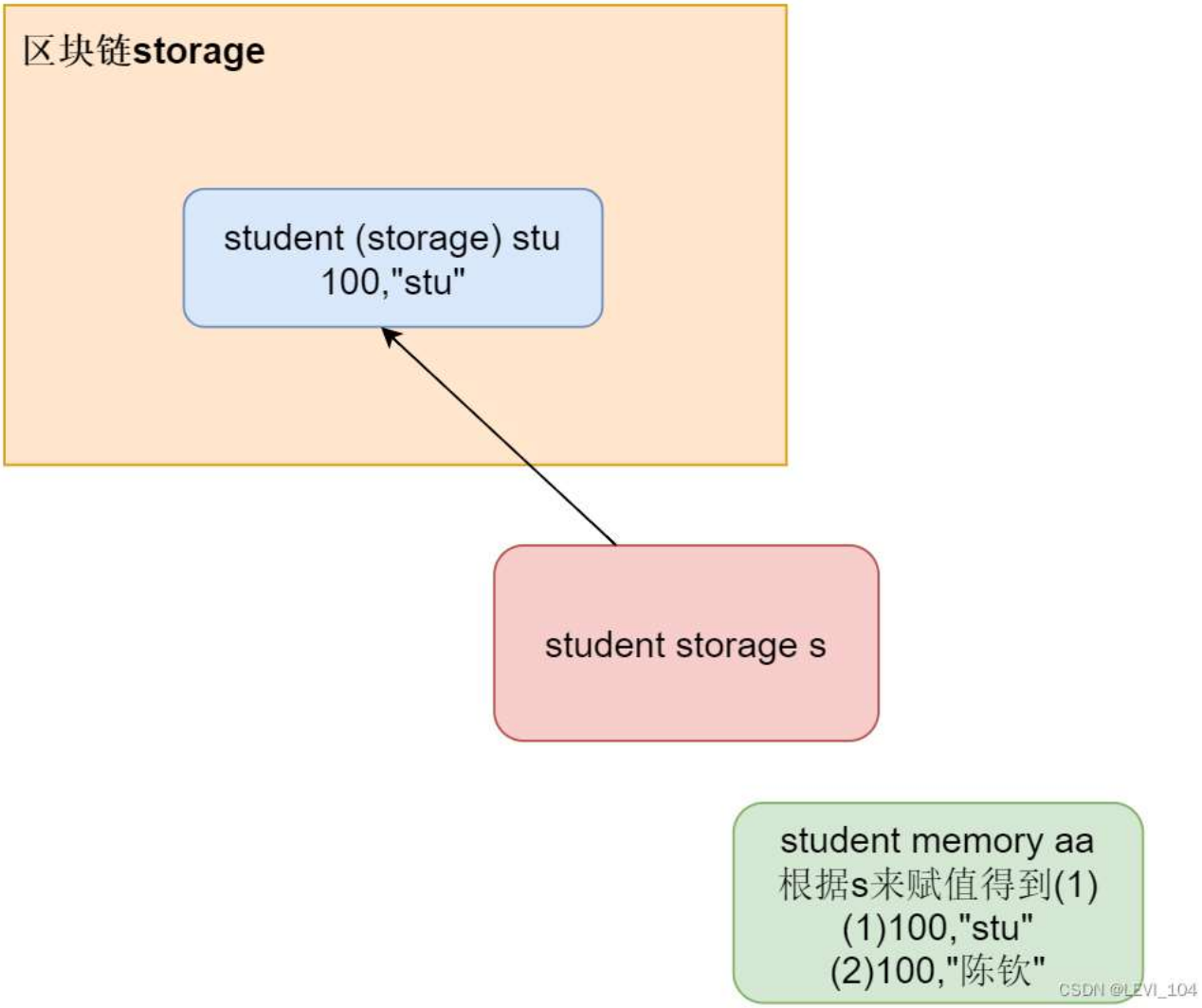
```
    struct student{
        uint grade;
        string name;
    }
    student stu;//这里是在区块链中开辟了一个storage
    //如果形参是struct类型，那么一定要加internal修饰符
    function test(student memory s) view internal returns(string){
        stu = s;
        s.name = "陈钦";
    }
    function call() returns(string){
        student memory t = student(100,"t");
        test(t);
        return stu.name;
```

```
    }
```



storage转memory

```
    struct student{
        uint grade;
        string name;
    }
    student stu = student(100,"stu");//这里是在区块链中开辟了一个storage
    //如果形参是struct类型，那么一定要加internal修饰符
    function test(student storage s) view internal returns(string){
        //student storage s: s是指针，用来指向区块链storage中的stu
        //                   然后下面修改就是修改指针指向的stu
        student memory aa = s;
        aa.name = "陈钦";
    }
    function call() returns(string){
        test(stu);
        return stu.name;
    }
```



memory转memory

```
    struct student{
        uint grade;
        string name;
    }
    //如果形参是struct类型，那么一定要加internal修饰符
    function test(student memory s) view internal returns(string){
        student memory Mike = s;
        Mike.name = "陈钦";
    }
    function call() returns(string){
        student memory John = student(100,"John");
        test(John);
        return John.name;
    }
```