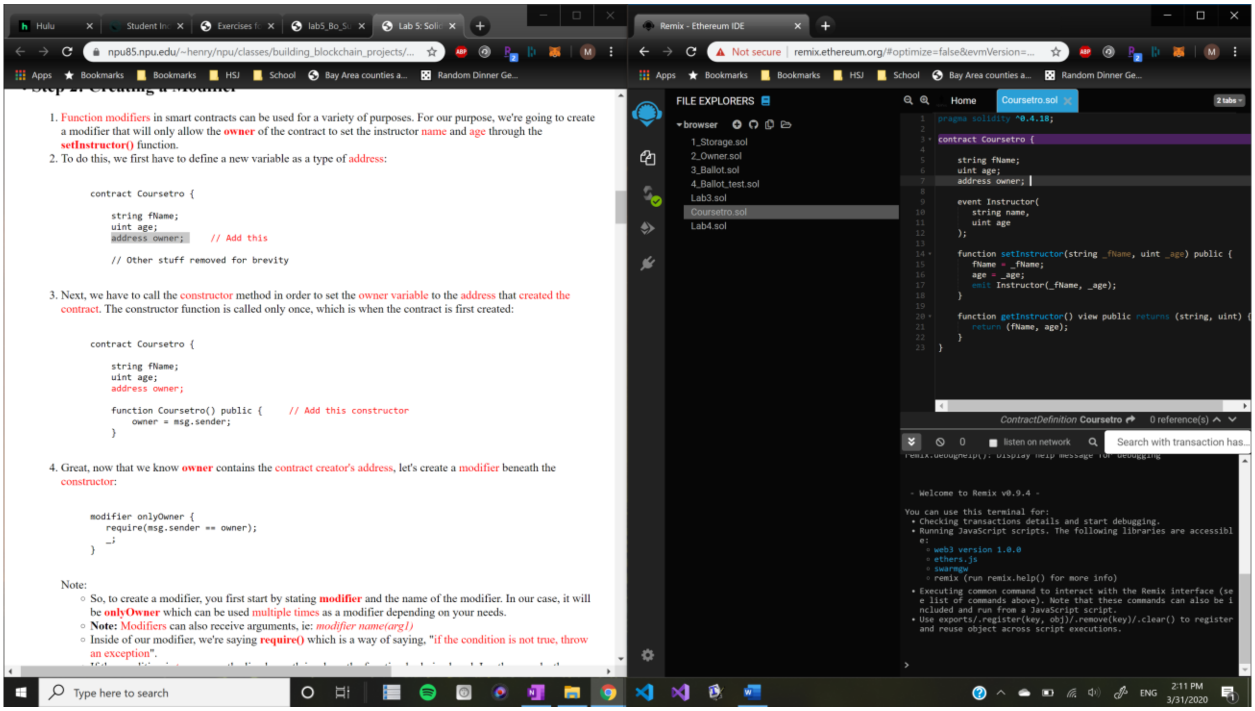
First attempt:



• Step 2: Creating a Modifier

1. Function modifiers in smart contracts can be used for a variety of purposes. For our purpose, we're going to create a modifier that will only allow the **owner** of the contract to set the instructor name and age through the **setInstructor()** function.
2. To do this, we first have to define a new variable as a type of address:

```
contract Coursetro {

    string fName;
    uint age;
    address owner;      // Add this

    // Other stuff removed for brevity
```

3. Next, we have to call the constructor method in order to set the owner variable to the address that created the contract. The constructor function is called only once, which is when the contract is first created:

```
contract Coursetro {

    string fName;
    uint age;
    address owner;

    function Coursetro() public {     // Add this constructor
        owner = msg.sender;
    }
```

4. Great, now that we know **owner** contains the contract creator's address, let's create a modifier beneath the constructor:

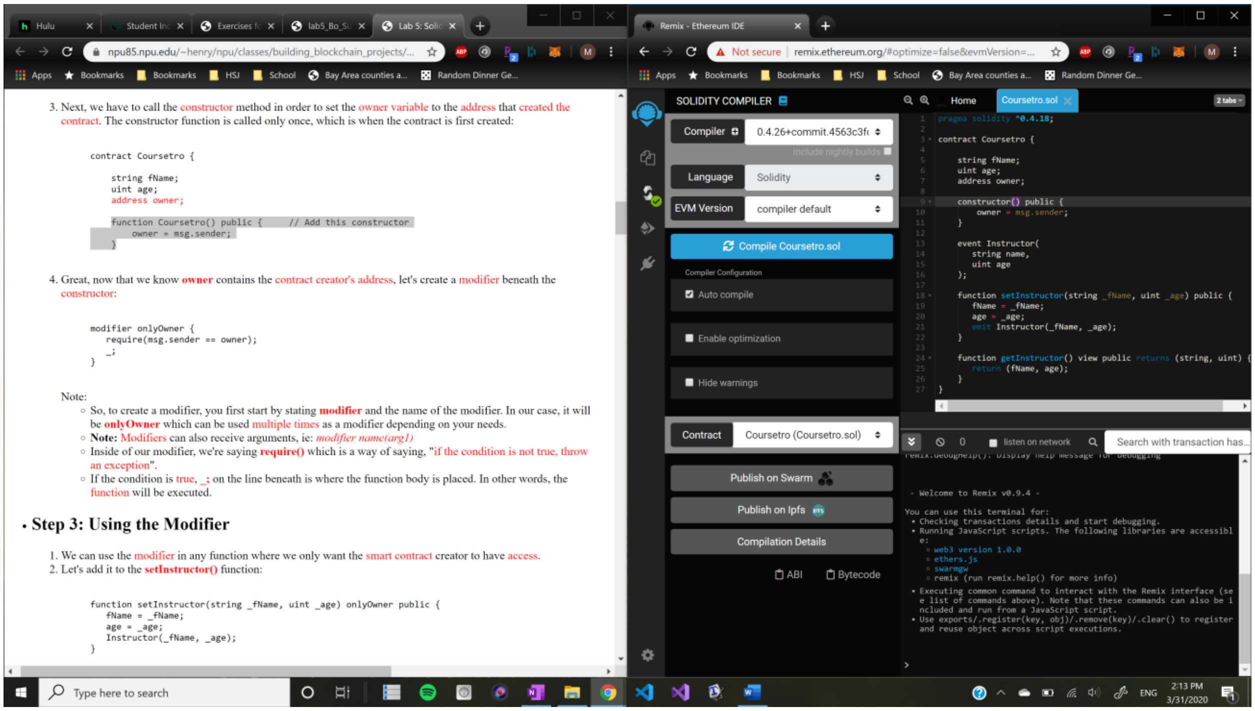
```
modifier onlyOwner {
   require(msg.sender == owner);
   _;
}
```

Note:

- So, to create a modifier, you first start by stating **modifier** and the name of the modifier. In our case, it will be **onlyOwner** which can be used multiple times as a modifier depending on your needs.
- **Note:** Modifiers can also receive arguments, ie: *modifier name(arg1)*
- Inside of our modifier, we're saying **require()** which is a way of saying, "if the condition is not true, throw an exception".

```
pragma solidity ^0.4.18;

contract Coursetro {

    string fName;
    uint age;
    address owner;

    event Instructor(
       string name,
       uint age
    );

    function setInstructor(string _fName, uint _age) public {
       fName = _fName;
       age = _age;
       emit Instructor(_fName, _age);
    }

    function getInstructor() view public returns (string, uint) {
       return (fName, age);
    }
}
```

```
 - Welcome to Remix v0.9.4 -

You can use this terminal for:
 • Checking transactions details and start debugging.
 • Running JavaScript scripts. The following libraries are accessibl
   e:
    ◦ web3 version 1.0.0
    ◦ ethers.js
    ◦ swarmgw
    ◦ remix (run remix.help() for more info)
 • Executing common command to interact with the Remix interface (se
   e list of commands above). Note that these commands can also be i
   ncluded and run from a JavaScript script.
 • Use exports/.register(key, obj)/.remove(key)/.clear() to register
   and reuse object across script executions.

>
```



3. Next, we have to call the constructor method in order to set the owner variable to the address that created the contract. The constructor function is called only once, which is when the contract is first created:

```
contract Coursetro {

    string fName;
    uint age;
    address owner;

    function Coursetro() public {     // Add this constructor
        owner = msg.sender;
    }
```

4. Great, now that we know **owner** contains the contract creator's address, let's create a modifier beneath the constructor:

```
modifier onlyOwner {
   require(msg.sender == owner);
   _;
}
```

Note:

- So, to create a modifier, you first start by stating **modifier** and the name of the modifier. In our case, it will be **onlyOwner** which can be used multiple times as a modifier depending on your needs.
- **Note:** Modifiers can also receive arguments, ie: *modifier name(arg1)*
- Inside of our modifier, we're saying **require()** which is a way of saying, "if the condition is not true, throw an exception".
- If the condition is true, **_;** on the line beneath is where the function body is placed. In other words, the function will be executed.

• **Step 3: Using the Modifier**

1. We can use the modifier in any function where we only want the smart contract creator to have access.
2. Let's add it to the **setInstructor()** function:

```
function setInstructor(string _fName, uint _age) onlyOwner public {
   fName = _fName;
   age = _age;
   Instructor(_fName, _age);
}
```

Compiler: 0.4.26+commit.4563c3fc

☐ Include nightly builds

Language: Solidity

EVM Version: compiler default

☑ Auto compile

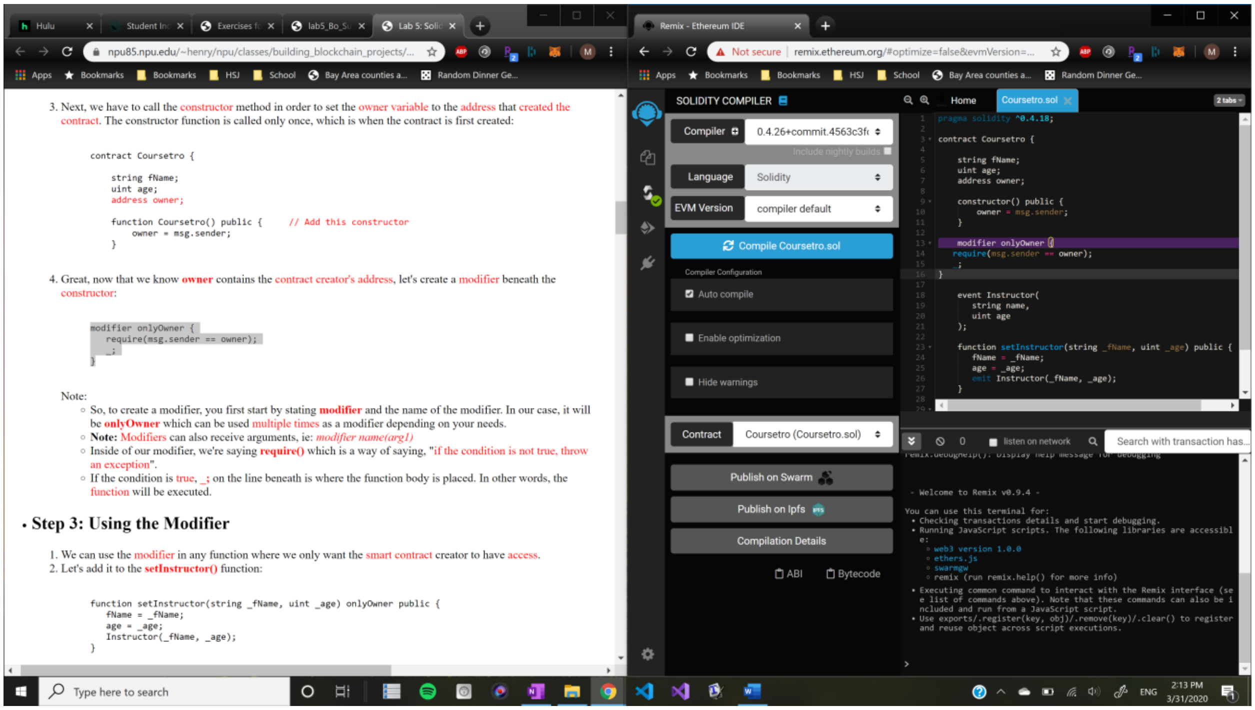
☐ Enable optimization

☐ Hide warnings

Contract: Coursetro (Coursetro.sol)

```
pragma solidity ^0.4.18;

contract Coursetro {

    string fName;
    uint age;
    address owner;

    constructor() public {
        owner = msg.sender;
    }

    event Instructor(
       string name,
       uint age
    );

    function setInstructor(string _fName, uint _age) public {
       fName = _fName;
       age = _age;
       emit Instructor(_fName, _age);
    }

    function getInstructor() view public returns (string, uint) {
       return (fName, age);
    }
}
```

```
 - Welcome to Remix v0.9.4 -

You can use this terminal for:
 • Checking transactions details and start debugging.
 • Running JavaScript scripts. The following libraries are accessibl
   e:
    ◦ web3 version 1.0.0
    ◦ ethers.js
    ◦ swarmgw
    ◦ remix (run remix.help() for more info)
 • Executing common command to interact with the Remix interface (se
   e list of commands above). Note that these commands can also be i
   ncluded and run from a JavaScript script.
 • Use exports/.register(key, obj)/.remove(key)/.clear() to register
   and reuse object across script executions.

>
```



3. Next, we have to call the constructor method in order to set the owner variable to the address that created the contract. The constructor function is called only once, which is when the contract is first created:

```
contract Coursetro {

    string fName;
    uint age;
    address owner;

    function Coursetro() public {     // Add this constructor
        owner = msg.sender;
    }
```

4. Great, now that we know **owner** contains the contract creator's address, let's create a modifier beneath the constructor:

```
modifier onlyOwner {
   require(msg.sender == owner);
   _;
}
```

Note:

- So, to create a modifier, you first start by stating **modifier** and the name of the modifier. In our case, it will be **onlyOwner** which can be used multiple times as a modifier depending on your needs.
- **Note:** Modifiers can also receive arguments, ie: *modifier name(arg1)*
- Inside of our modifier, we're saying **require()** which is a way of saying, "if the condition is not true, throw an exception".
- If the condition is true, _; on the line beneath is where the function body is placed. In other words, the function will be executed.

• Step 3: Using the Modifier

1. We can use the modifier in any function where we only want the smart contract creator to have access.
2. Let's add it to the **setInstructor()** function:

```
function setInstructor(string _fName, uint _age) onlyOwner public {
   fName = _fName;
   age = _age;
   Instructor(_fName, _age);
}
```

Compiler 0.4.26+commit.4563c3fc

Language Solidity

EVM Version compiler default

☑ Auto compile

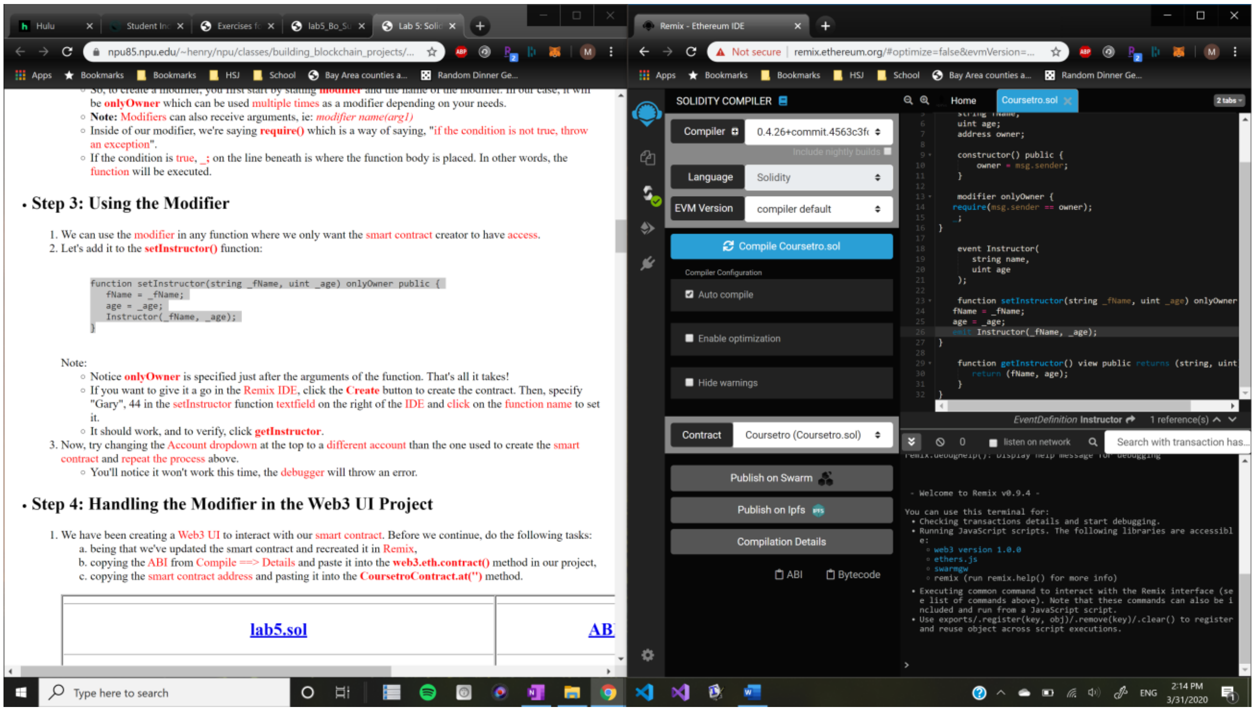
□ Enable optimization

□ Hide warnings

Contract Coursetro (Coursetro.sol)

```
pragma solidity ^0.4.18;

contract Coursetro {

    string fName;
    uint age;
    address owner;

    constructor() public {
        owner = msg.sender;
    }

    modifier onlyOwner {
   require(msg.sender == owner);
   _;
}

    event Instructor(
       string name,
       uint age
    );

    function setInstructor(string _fName, uint _age) public {
       fName = _fName;
       age = _age;
       emit Instructor(_fName, _age);
    }
```

```
remix.debugHelp(): Display help message for debugging

 - Welcome to Remix v0.9.4 -

You can use this terminal for:
 • Checking transactions details and start debugging.
 • Running JavaScript scripts. The following libraries are accessibl
   e:
    ◦ web3 version 1.0.0
    ◦ ethers.js
    ◦ swarmgw
    ◦ remix (run remix.help() for more info)
 • Executing common command to interact with the Remix interface (se
   e list of commands above). Note that these commands can also be i
   ncluded and run from a JavaScript script.
 • Use exports/.register(key, obj)/.remove(key)/.clear() to register
   and reuse object across script executions.

>
```



be **onlyOwner** which can be used multiple times as a modifier depending on your needs.

- **Note:** Modifiers can also receive arguments, ie: *modifier name(arg1)*
- Inside of our modifier, we're saying **require()** which is a way of saying, "if the condition is not true, throw an exception".
- If the condition is true, _; on the line beneath is where the function body is placed. In other words, the function will be executed.

• Step 3: Using the Modifier

1. We can use the modifier in any function where we only want the smart contract creator to have access.
2. Let's add it to the **setInstructor()** function:

```
function setInstructor(string _fName, uint _age) onlyOwner public {
   fName = _fName;
   age = _age;
   Instructor(_fName, _age);
}
```

Note:

- Notice **onlyOwner** is specified just after the arguments of the function. That's all it takes!
- If you want to give it a go in the Remix IDE, click the **Create** button to create the contract. Then, specify "Gary", 44 in the setInstructor function textfield on the right of the IDE and click on the function name to set it.
- It should work, and to verify, click **getInstructor**.

3. Now, try changing the Account dropdown at the top to a different account than the one used to create the smart contract and repeat the process above.
   - You'll notice it won't work this time, the debugger will throw an error.

• Step 4: Handling the Modifier in the Web3 UI Project

1. We have been creating a Web3 UI to interact with our smart contract. Before we continue, do the following tasks:
   a. being that we've updated the smart contract and recreated it in Remix,
   b. copying the ABI from Compile ==> Details and paste it into the **web3.eth.contract()** method in our project,
   c. copying the smart contract address and pasting it into the **CoursetroContract.at('')** method.

lab5.sol	AB

Compiler 0.4.26+commit.4563c3fc

Language Solidity

EVM Version compiler default

☑ Auto compile

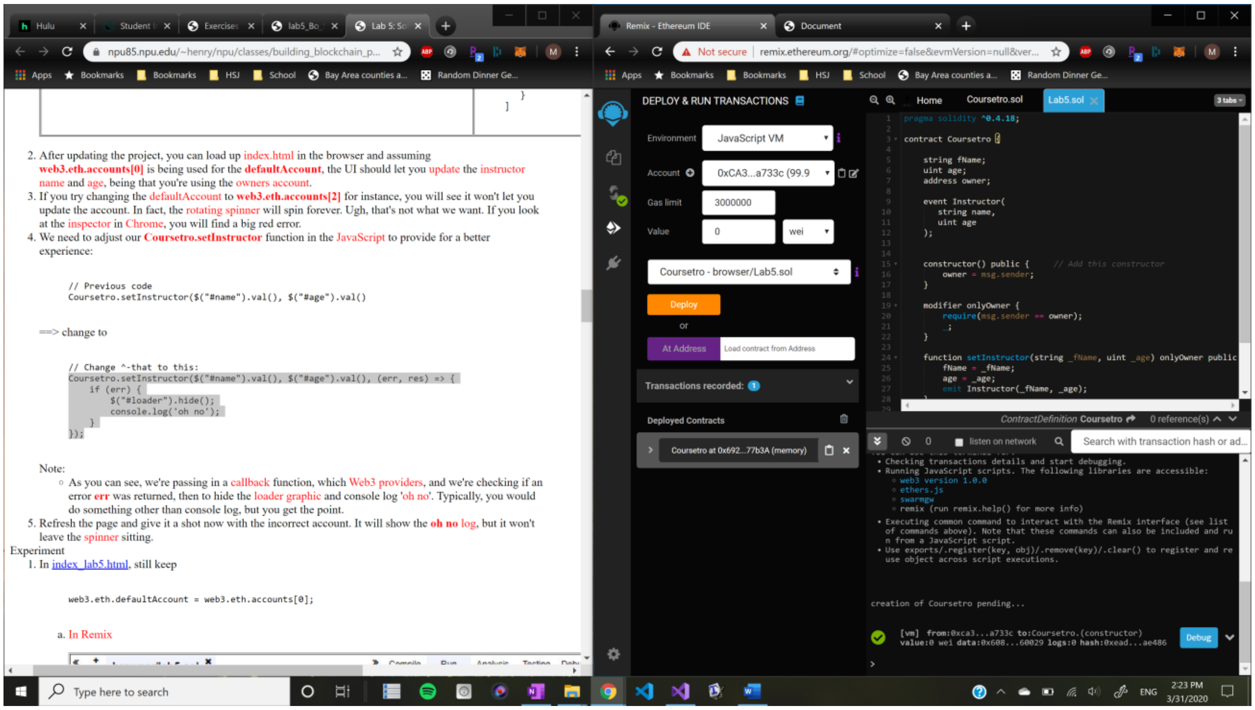
□ Enable optimization

□ Hide warnings

Contract Coursetro (Coursetro.sol)

```
    string fName;
    uint age;
    address owner;

    constructor() public {
        owner = msg.sender;
    }

    modifier onlyOwner {
   require(msg.sender == owner);
   _;
}

    event Instructor(
       string name,
       uint age
    );

    function setInstructor(string _fName, uint _age) onlyOwner
   fName = _fName;
   age = _age;
   emit Instructor(_fName, _age);
}

    function getInstructor() view public returns (string, uint
       return (fName, age);
    }
}
```

```
remix.debugHelp(): Display help message for debugging

 - Welcome to Remix v0.9.4 -

You can use this terminal for:
 • Checking transactions details and start debugging.
 • Running JavaScript scripts. The following libraries are accessibl
   e:
    ◦ web3 version 1.0.0
    ◦ ethers.js
    ◦ swarmgw
    ◦ remix (run remix.help() for more info)
 • Executing common command to interact with the Remix interface (se
   e list of commands above). Note that these commands can also be i
   ncluded and run from a JavaScript script.
 • Use exports/.register(key, obj)/.remove(key)/.clear() to register
   and reuse object across script executions.

>
```



npu85.npu.edu/~henry/npu/classes/building_blockchain_p...
Remix - Ethereum IDE
remix.ethereum.org/#optimize=false&evmVersion=null&ver...

2. After updating the project, you can load up index.html in the browser and assuming **web3.eth.accounts[0]** is being used for the **defaultAccount**, the UI should let you update the instructor name and age, being that you're using the owners account.
3. If you try changing the defaultAccount to **web3.eth.accounts[2]** for instance, you will see it won't let you update the account. In fact, the rotating spinner will spin forever. Ugh, that's not what we want. If you look at the inspector in Chrome, you will find a big red error.
4. We need to adjust our **Coursetro.setInstructor** function in the JavaScript to provide for a better experience:

```
// Previous code
Coursetro.setInstructor($("#name").val(), $("#age").val()
```

==> change to

```
// Change ^-that to this:
Coursetro.setInstructor($("#name").val(), $("#age").val(), (err, res) => {
    if (err) {
        $("#loader").hide();
        console.log('oh no');
    }
});
```

Note:
- As you can see, we're passing in a callback function, which Web3 providers, and we're checking if an error **err** was returned, then to hide the loader graphic and console log 'oh no'. Typically, you would do something other than console log, but you get the point.

5. Refresh the page and give it a shot now with the incorrect account. It will show the **oh no** log, but it won't leave the spinner sitting.

Experiment

1. In index_lab5.html, still keep

```
web3.eth.defaultAccount = web3.eth.accounts[0];
```

a. In Remix

DEPLOY & RUN TRANSACTIONS

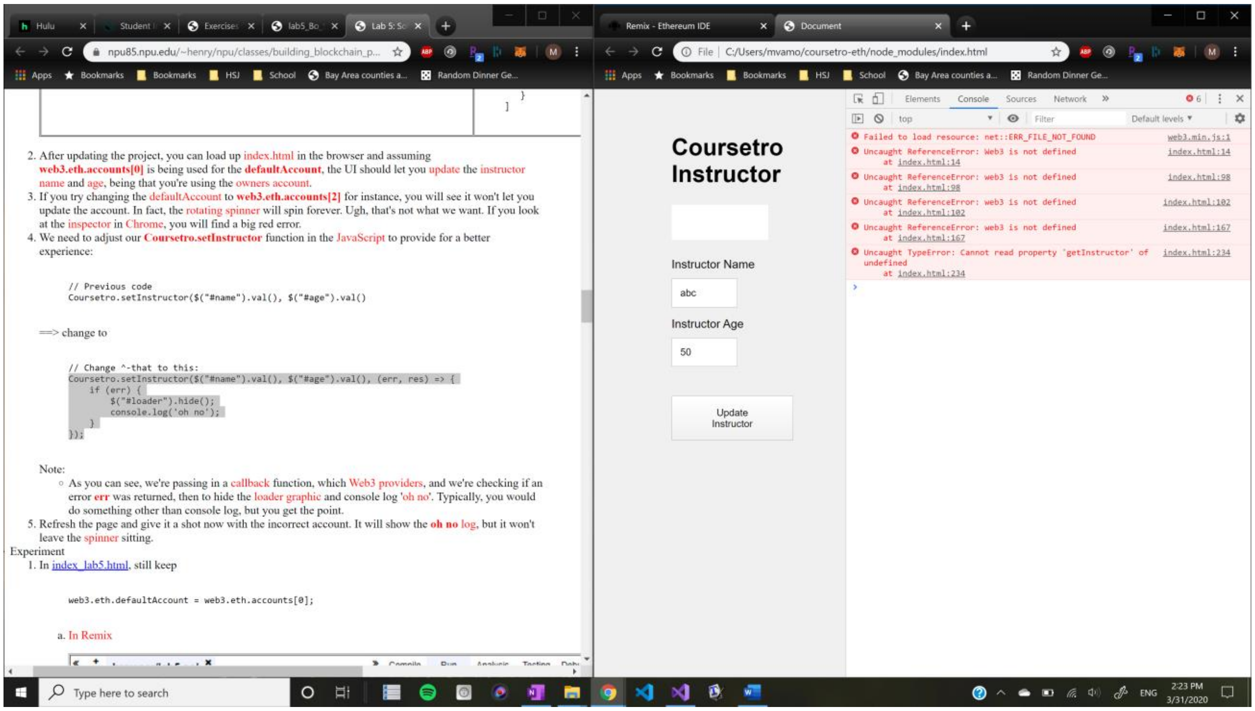
Environment: JavaScript VM
Account: 0xCA3...a733c (99.9
Gas limit: 3000000
Value: 0 wei
Coursetro - browser/Lab5.sol

```
pragma solidity ^0.4.18;

contract Coursetro {

    string fName;
    uint age;
    address owner;

    event Instructor(
       string name,
       uint age
    );


    constructor() public {     // Add this constructor
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    function setInstructor(string _fName, uint _age) onlyOwner public
        fName = _fName;
        age = _age;
        emit Instructor(_fName, _age);
```

ContractDefinition Coursetro
0 reference(s)
listen on network
Search with transaction hash or ad...

```
 • Checking transactions details and start debugging.
 • Running JavaScript scripts. The following libraries are accessible:
    ○ web3 version 1.0.0
    ○ ethers.js
    ○ swarmgw
    ○ remix (run remix.help() for more info)
 • Executing common command to interact with the Remix interface (see list
   of commands above). Note that these commands can also be included and ru
   n from a JavaScript script.
 • Use exports/.register(key, obj)/.remove(key)/.clear() to register and re
   use object across script executions.


creation of Coursetro pending...

[vm] from:0xca3...a733c to:Coursetro.(constructor)
value:0 wei data:0x608...60029 logs:0 hash:0xead...ae486
```

Debug
Type here to search
2:23 PM
3/31/2020



npu85.npu.edu/~henry/npu/classes/building_blockchain_p...
Document
File | C:/Users/mvamo/coursetro-eth/node_modules/index.html

2. After updating the project, you can load up index.html in the browser and assuming **web3.eth.accounts[0]** is being used for the **defaultAccount**, the UI should let you update the instructor name and age, being that you're using the owners account.
3. If you try changing the defaultAccount to **web3.eth.accounts[2]** for instance, you will see it won't let you update the account. In fact, the rotating spinner will spin forever. Ugh, that's not what we want. If you look at the inspector in Chrome, you will find a big red error.
4. We need to adjust our **Coursetro.setInstructor** function in the JavaScript to provide for a better experience:

```
// Previous code
Coursetro.setInstructor($("#name").val(), $("#age").val()
```

==> change to

```
// Change ^-that to this:
Coursetro.setInstructor($("#name").val(), $("#age").val(), (err, res) => {
    if (err) {
        $("#loader").hide();
        console.log('oh no');
    }
});
```

Note:
- As you can see, we're passing in a callback function, which Web3 providers, and we're checking if an error **err** was returned, then to hide the loader graphic and console log 'oh no'. Typically, you would do something other than console log, but you get the point.

5. Refresh the page and give it a shot now with the incorrect account. It will show the **oh no** log, but it won't leave the spinner sitting.

Experiment

1. In index_lab5.html, still keep

```
web3.eth.defaultAccount = web3.eth.accounts[0];
```

a. In Remix

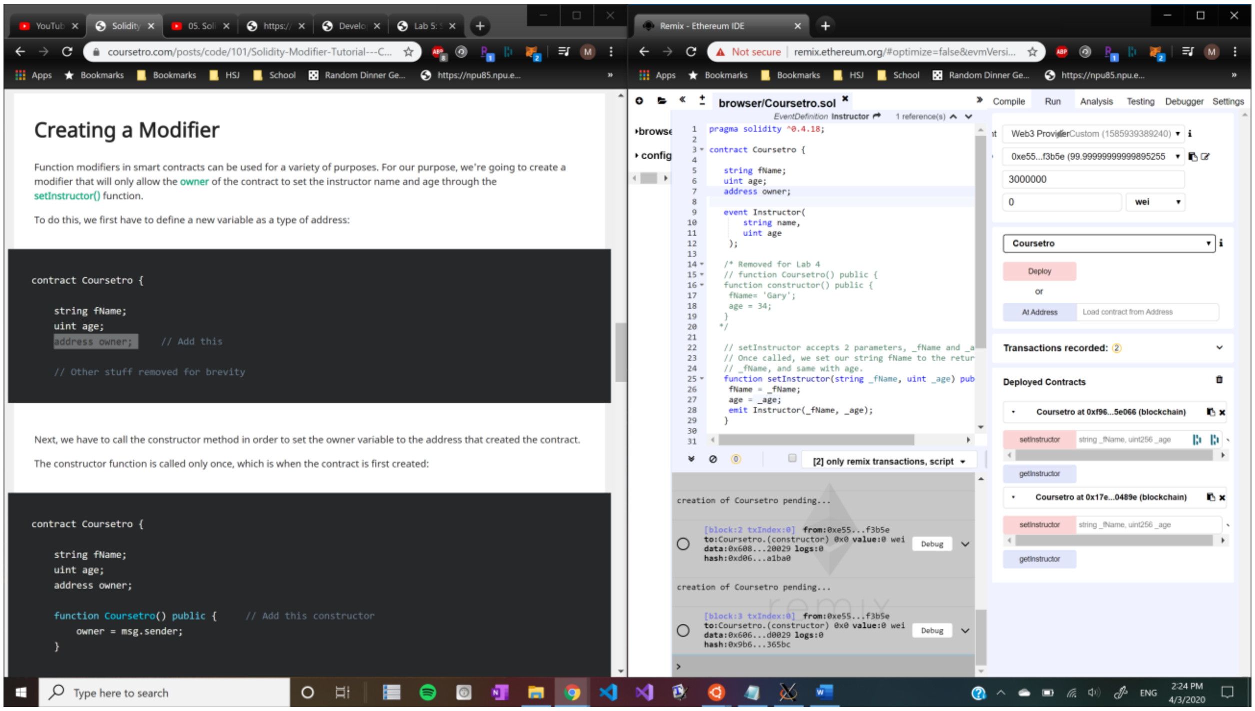
Coursetro Instructor

Instructor Name: abc

Instructor Age: 50

```
Failed to load resource: net::ERR_FILE_NOT_FOUND                    web3.min.js:1
Uncaught ReferenceError: Web3 is not defined                        index.html:14
    at index.html:14
Uncaught ReferenceError: web3 is not defined                        index.html:98
    at index.html:98
Uncaught ReferenceError: web3 is not defined                       index.html:102
    at index.html:102
Uncaught ReferenceError: web3 is not defined                       index.html:167
    at index.html:167
Uncaught TypeError: Cannot read property 'getInstructor' of        index.html:234
undefined
    at index.html:234
```

Second attempt (Using Ubuntu and Xming):



Creating a Modifier

Function modifiers in smart contracts can be used for a variety of purposes. For our purpose, we're going to create a modifier that will only allow the owner of the contract to set the instructor name and age through the setInstructor() function.

To do this, we first have to define a new variable as a type of address:

```
contract Coursetro {

    string fName;
    uint age;
    address owner;      // Add this

    // Other stuff removed for brevity
```

Next, we have to call the constructor method in order to set the owner variable to the address that created the contract.

The constructor function is called only once, which is when the contract is first created:

```
contract Coursetro {

    string fName;
    uint age;
    address owner;

    function Coursetro() public {      // Add this constructor
        owner = msg.sender;
    }
```

browser/Coursetro.sol

EventDefinition Instructor 1 reference(s)

```
pragma solidity ^0.4.18;

contract Coursetro {

   string fName;
   uint age;
   address owner;

   event Instructor(
       string name,
       uint age
    );

   /* Removed for Lab 4
   // function Coursetro() public {
   function constructor() public {
    fName= 'Gary';
    age = 34;
   }
  */

   // setInstructor accepts 2 parameters, _fName and _a
   // Once called, we set our string fName to the retur
   // _fName, and same with age.
   function setInstructor(string _fName, uint _age) pub
    fName = _fName;
    age = _age;
    emit Instructor(_fName, _age);
   }
```

[2] only remix transactions, script

```
creation of Coursetro pending...

[block:2 txIndex:0] from:0xe55...f3b5e
to:Coursetro.(constructor) 0x0 value:0 wei
data:0x608...20029 logs:0
hash:0xd06...a1ba0

creation of Coursetro pending...

[block:3 txIndex:0] from:0xe55...f3b5e
to:Coursetro.(constructor) 0x0 value:0 wei
data:0x606...d0029 logs:0
hash:0x9b6...365bc
```

Web3 Provider Custom (1585939389240)
0xe55...f3b5e (99.99999999999895255
3000000
0 wei

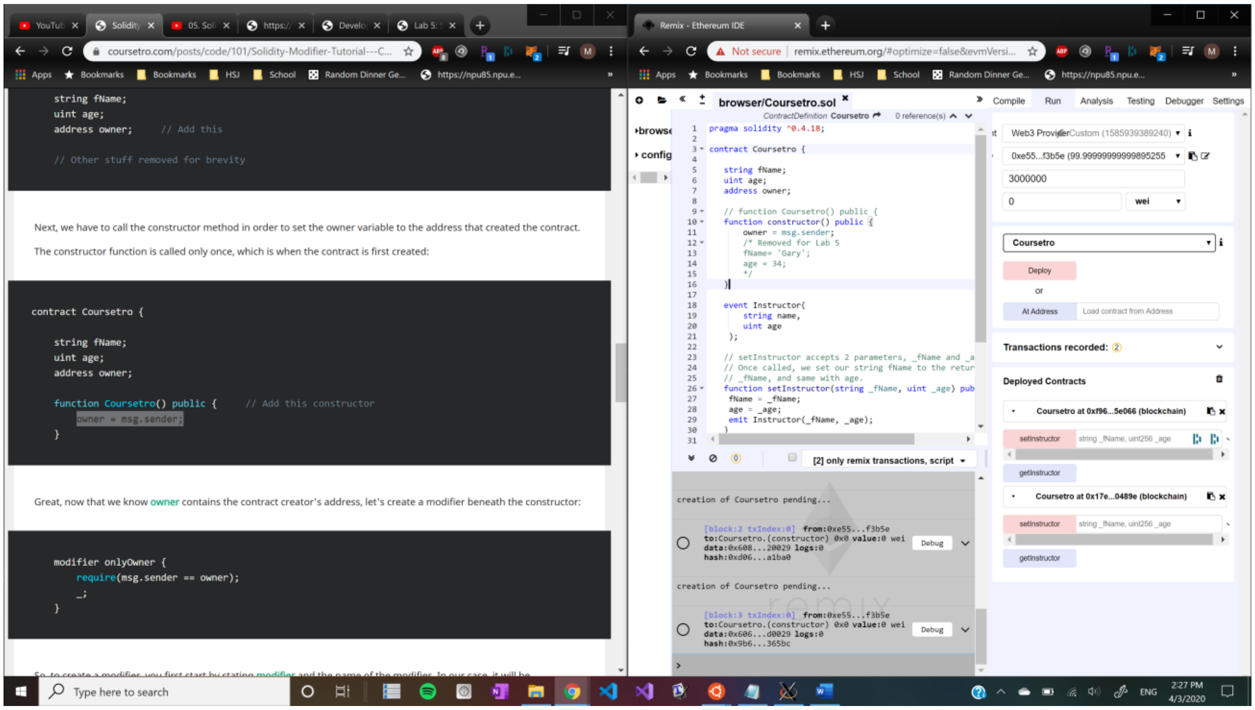
Coursetro

Transactions recorded: 2

Deployed Contracts

Coursetro at 0xf96...5e066 (blockchain)
setInstructor string _fName, uint256 _age
getInstructor

Coursetro at 0x17e...0489e (blockchain)
setInstructor string _fName, uint256 _age
getInstructor



```
    string fName;
    uint age;
    address owner;      // Add this

    // Other stuff removed for brevity
```

Next, we have to call the constructor method in order to set the owner variable to the address that created the contract.

The constructor function is called only once, which is when the contract is first created:

```
contract Coursetro {

    string fName;
    uint age;
    address owner;

    function Coursetro() public {      // Add this constructor
        owner = msg.sender;
    }
```

Great, now that we know owner contains the contract creator's address, let's create a modifier beneath the constructor:

```
    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }
```

So, to create a modifier, you first start by stating modifier and the name of the modifier. In our case, it will be

browser/Coursetro.sol

ContractDefinition Coursetro 0 reference(s)

```
pragma solidity ^0.4.18;

contract Coursetro {

   string fName;
   uint age;
   address owner;

   // function Coursetro() public {
   function constructor() public {
       owner = msg.sender;
       /* Removed for Lab 5
       fName= 'Gary';
       age = 34;
       */
   }

   event Instructor(
       string name,
       uint age
    );

   // setInstructor accepts 2 parameters, _fName and _a
   // Once called, we set our string fName to the retur
   // _fName, and same with age.
   function setInstructor(string _fName, uint _age) pub
    fName = _fName;
    age = _age;
    emit Instructor(_fName, _age);
   }
```

[2] only remix transactions, script

```
creation of Coursetro pending...

[block:2 txIndex:0] from:0xe55...f3b5e
to:Coursetro.(constructor) 0x0 value:0 wei
data:0x608...20029 logs:0
hash:0xd06...a1ba0

creation of Coursetro pending...

[block:3 txIndex:0] from:0xe55...f3b5e
to:Coursetro.(constructor) 0x0 value:0 wei
data:0x606...d0029 logs:0
hash:0x9b6...365bc
```

Web3 Provider Custom (1585939389240)
0xe55...f3b5e (99.99999999999895255
3000000
0 wei

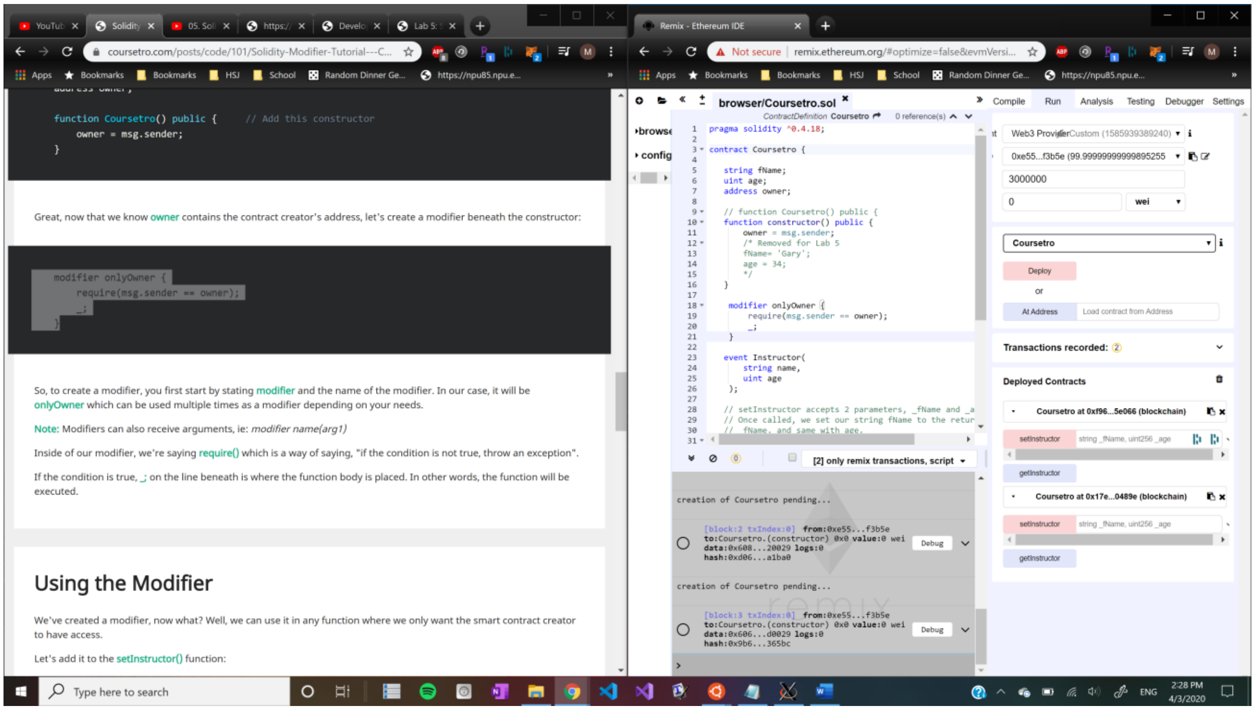
Coursetro

Transactions recorded: 2

Deployed Contracts

Coursetro at 0xf96...5e066 (blockchain)
setInstructor string _fName, uint256 _age
getInstructor

Coursetro at 0x17e...0489e (blockchain)
setInstructor string _fName, uint256 _age
getInstructor



```
    function Coursetro() public {     // Add this constructor
        owner = msg.sender;
    }
```

Great, now that we know owner contains the contract creator's address, let's create a modifier beneath the constructor:

```
    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }
```

So, to create a modifier, you first start by stating modifier and the name of the modifier. In our case, it will be onlyOwner which can be used multiple times as a modifier depending on your needs.

Note: Modifiers can also receive arguments, ie: *modifier name(arg1)*

Inside of our modifier, we're saying require() which is a way of saying, "if the condition is not true, throw an exception".

If the condition is true, _; on the line beneath is where the function body is placed. In other words, the function will be executed.

Using the Modifier

We've created a modifier, now what? Well, we can use it in any function where we only want the smart contract creator to have access.

Let's add it to the setInstructor() function:

```
pragma solidity ^0.4.18;

contract Coursetro {

   string fName;
   uint age;
   address owner;

   // function Coursetro() public {
   function constructor() public {
       owner = msg.sender;
       /* Removed for Lab 5
       fName= 'Gary';
       age = 34;
       */
   }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

   event Instructor(
       string name,
       uint age
    );

   // setInstructor accepts 2 parameters, _fName and _a
   // Once called, we set our string fName to the retur
   //  fName, and same with age.
```

[2] only remix transactions, script

```
creation of Coursetro pending...

[block:2 txIndex:0] from:0xe55...f3b5e
to:Coursetro.(constructor) 0x0 value:0 wei
data:0x608...20029 logs:0
hash:0xd06...a1ba0

creation of Coursetro pending...

[block:3 txIndex:0] from:0xe55...f3b5e
to:Coursetro.(constructor) 0x0 value:0 wei
data:0x606...d0029 logs:0
hash:0x9b6...365bc
```

Web3 Provider Custom (1585939389240)
0xe55...f3b5e (99.99999999999895255
3000000
0 wei

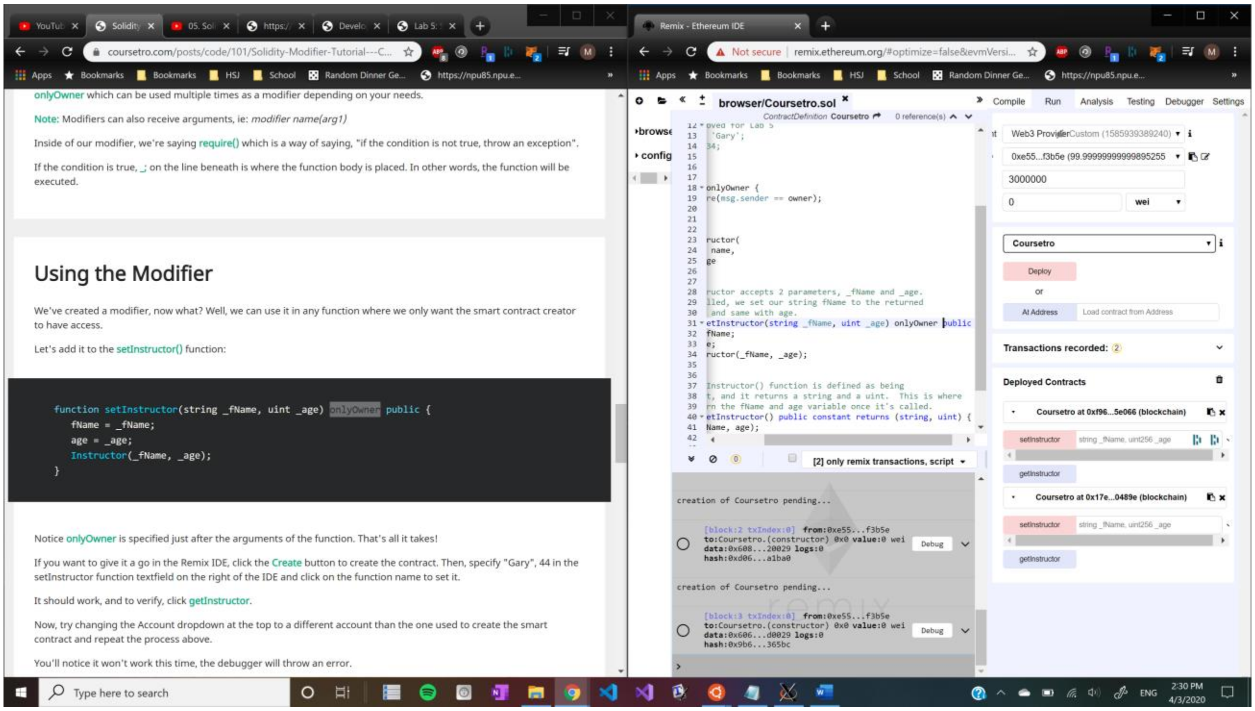
Coursetro
Deploy
or
At Address Load contract from Address

Transactions recorded: 2

Deployed Contracts

Coursetro at 0xf96...5e066 (blockchain)
setInstructor string _fName, uint256 _age
getInstructor

Coursetro at 0x17e...0489e (blockchain)
setInstructor string _fName, uint256 _age
getInstructor



onlyOwner which can be used multiple times as a modifier depending on your needs.

Note: Modifiers can also receive arguments, ie: *modifier name(arg1)*

Inside of our modifier, we're saying require() which is a way of saying, "if the condition is not true, throw an exception".

If the condition is true, _; on the line beneath is where the function body is placed. In other words, the function will be executed.

Using the Modifier

We've created a modifier, now what? Well, we can use it in any function where we only want the smart contract creator to have access.

Let's add it to the setInstructor() function:

```
    function setInstructor(string _fName, uint _age) onlyOwner public {
       fName = _fName;
       age = _age;
       Instructor(_fName, _age);
    }
```

Notice onlyOwner is specified just after the arguments of the function. That's all it takes!

If you want to give it a go in the Remix IDE, click the Create button to create the contract. Then, specify "Gary", 44 in the setInstructor function textfield on the right of the IDE and click on the function name to set it.

It should work, and to verify, click getInstructor.

Now, try changing the Account dropdown at the top to a different account than the one used to create the smart contract and repeat the process above.

You'll notice it won't work this time, the debugger will throw an error.

```
oved for Lab 5
 'Gary';
34;



onlyOwner {
re(msg.sender == owner);



ructor(
 name,
ge



ructor accepts 2 parameters, _fName and _age.
lled, we set our string fName to the returned
 and same with age.
etInstructor(string _fName, uint _age) onlyOwner public
fName;
e;
ructor(_fName, _age);



Instructor() function is defined as being
t, and it returns a string and a uint.  This is where
rn the fName and age variable once it's called.
etInstructor() public constant returns (string, uint) {
Name, age);
```

[2] only remix transactions, script

```
creation of Coursetro pending...

[block:2 txIndex:0] from:0xe55...f3b5e
to:Coursetro.(constructor) 0x0 value:0 wei
data:0x608...20029 logs:0
hash:0xd06...a1ba0

creation of Coursetro pending...

[block:3 txIndex:0] from:0xe55...f3b5e
to:Coursetro.(constructor) 0x0 value:0 wei
data:0x606...d0029 logs:0
hash:0x9b6...365bc
```

Web3 Provider Custom (1585939389240)
0xe55...f3b5e (99.99999999999895255
3000000
0 wei

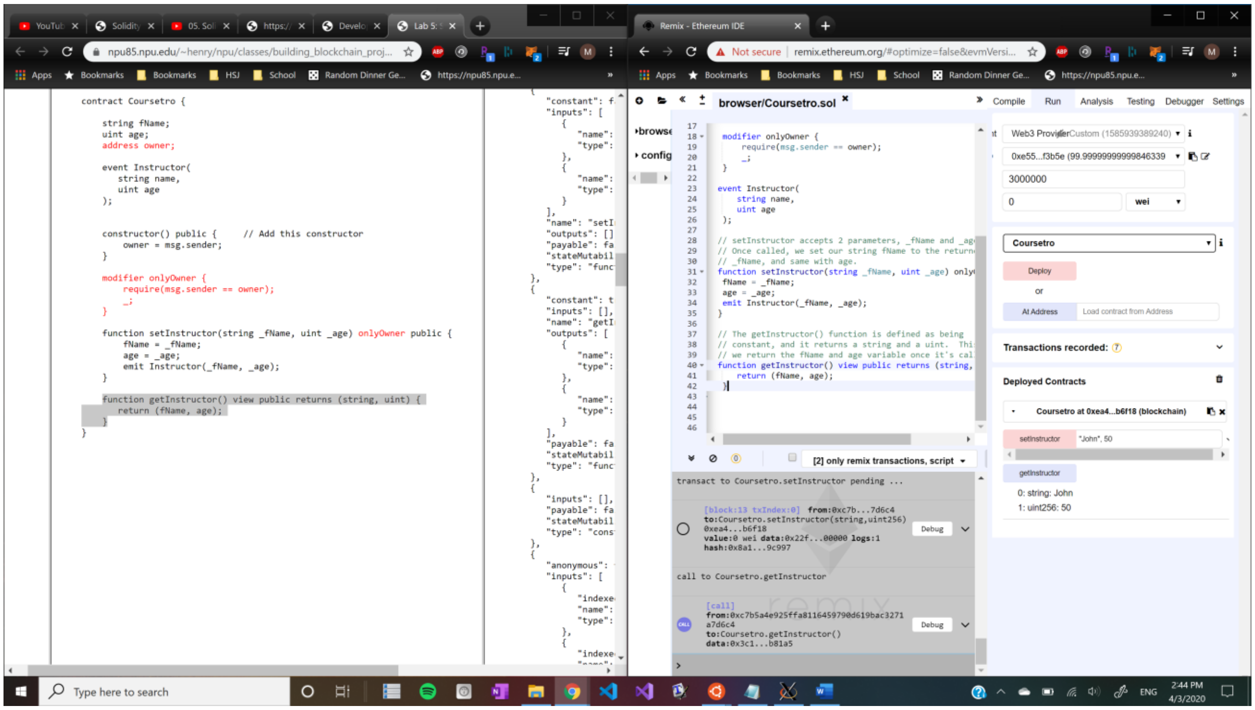
Coursetro
Deploy
or
At Address Load contract from Address

Transactions recorded: 2

Deployed Contracts

Coursetro at 0xf96...5e066 (blockchain)
setInstructor string _fName, uint256 _age
getInstructor

Coursetro at 0x17e...0489e (blockchain)
setInstructor string _fName, uint256 _age
getInstructor



```
contract Coursetro {

    string fName;
    uint age;
    address owner;

    event Instructor(
       string name,
       uint age
    );


    constructor() public {     // Add this constructor
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    function setInstructor(string _fName, uint _age) onlyOwner public {
        fName = _fName;
        age = _age;
        emit Instructor(_fName, _age);
    }

    function getInstructor() view public returns (string, uint) {
       return (fName, age);
    }
}
```

```
  {
     "constant": f
     "inputs": [
        {
           "name":
           "type":
        },
        {
           "name":
           "type":
        }
     ],
     "name": "setI
     "outputs": []
     "payable": fa
     "stateMutabil
     "type": "func
  },
  {
     "constant": t
     "inputs": [],
     "name": "getI
     "outputs": [
        {
           "name":
           "type":
        },
        {
           "name":
           "type":
        }
     ],
     "payable": fa
     "stateMutabil
     "type": "func
  },
  {
     "inputs": [],
     "payable": fa
     "stateMutabil
     "type": "cons
  },
  {
     "anonymous":
     "inputs": [
        {
           "indexe
           "name":
           "type":
        },
        {
           "indexe
```

```
 modifier onlyOwner {
     require(msg.sender == owner);
     _;
 }

event Instructor(
    string name,
    uint age
 );

// setInstructor accepts 2 parameters, _fName and _age
// Once called, we set our string fName to the return
// _fName, and same with age.
function setInstructor(string _fName, uint _age) only
 fName = _fName;
 age = _age;
 emit Instructor(_fName, _age);
}

// The getInstructor() function is defined as being
// constant, and it returns a string and a uint.  Thi
// we return the fName and age variable once it's cal
function getInstructor() view public returns (string,
    return (fName, age);
 }
```

[2] only remix transactions, script

```
transact to Coursetro.setInstructor pending ...

[block:13 txIndex:0] from:0xc7b...7d6c4
to:Coursetro.setInstructor(string,uint256)
0xea4...b6f18
value:0 wei data:0x22f...00000 logs:1
hash:0x8a1...9c997

call to Coursetro.getInstructor

[call]
from:0xc7b5a4e925ffa8116459790d619bac3271
a7d6c4
to:Coursetro.getInstructor()
data:0x3c1...b81a5
```

Web3 Provider Custom (1585939389240)
0xe55...f3b5e (99.99999999999846339
3000000
0 wei

Coursetro
Deploy
or
At Address — Load contract from Address

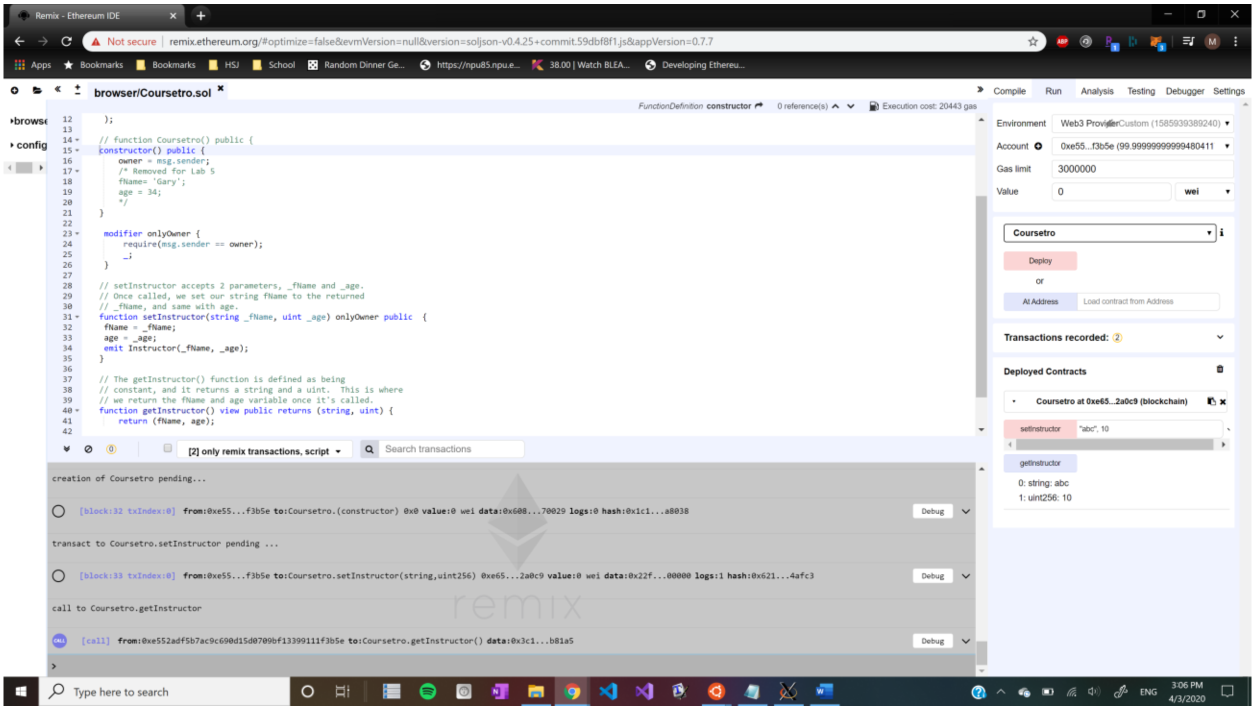
Transactions recorded: 7

Deployed Contracts

Coursetro at 0xea4...b6f18 (blockchain)

setInstructor "John", 50
getInstructor

0: string: John
1: uint256: 50



FunctionDefinition constructor 0 reference(s) Execution cost: 20443 gas

```
   );

  // function Coursetro() public {
  constructor() public {
      owner = msg.sender;
      /* Removed for Lab 5
      fName= 'Gary';
      age = 34;
      */
  }

   modifier onlyOwner {
       require(msg.sender == owner);
       _;
   }

  // setInstructor accepts 2 parameters, _fName and _age.
  // Once called, we set our string fName to the returned
  // _fName, and same with age.
  function setInstructor(string _fName, uint _age) onlyOwner public  {
   fName = _fName;
   age = _age;
   emit Instructor(_fName, _age);
  }

  // The getInstructor() function is defined as being
  // constant, and it returns a string and a uint.  This is where
  // we return the fName and age variable once it's called.
  function getInstructor() view public returns (string, uint) {
      return (fName, age);
```

[2] only remix transactions, script — Search transactions

```
creation of Coursetro pending...

[block:32 txIndex:0] from:0xe55...f3b5e to:Coursetro.(constructor) 0x0 value:0 wei data:0x608...70029 logs:0 hash:0x1c1...a8038

transact to Coursetro.setInstructor pending ...

[block:33 txIndex:0] from:0xe55...f3b5e to:Coursetro.setInstructor(string,uint256) 0xe65...2a0c9 value:0 wei data:0x22f...00000 logs:1 hash:0x621...4afc3

call to Coursetro.getInstructor

[call] from:0xe552adf5b7ac9c690d15d0709bf13399111f3b5e to:Coursetro.getInstructor() data:0x3c1...b81a5
```

Environment: Web3 Provider Custom (1585939389240)
Account: 0xe55...f3b5e (99.99999999999480411
Gas limit: 3000000
Value: 0 wei

Coursetro
Deploy
or
At Address — Load contract from Address

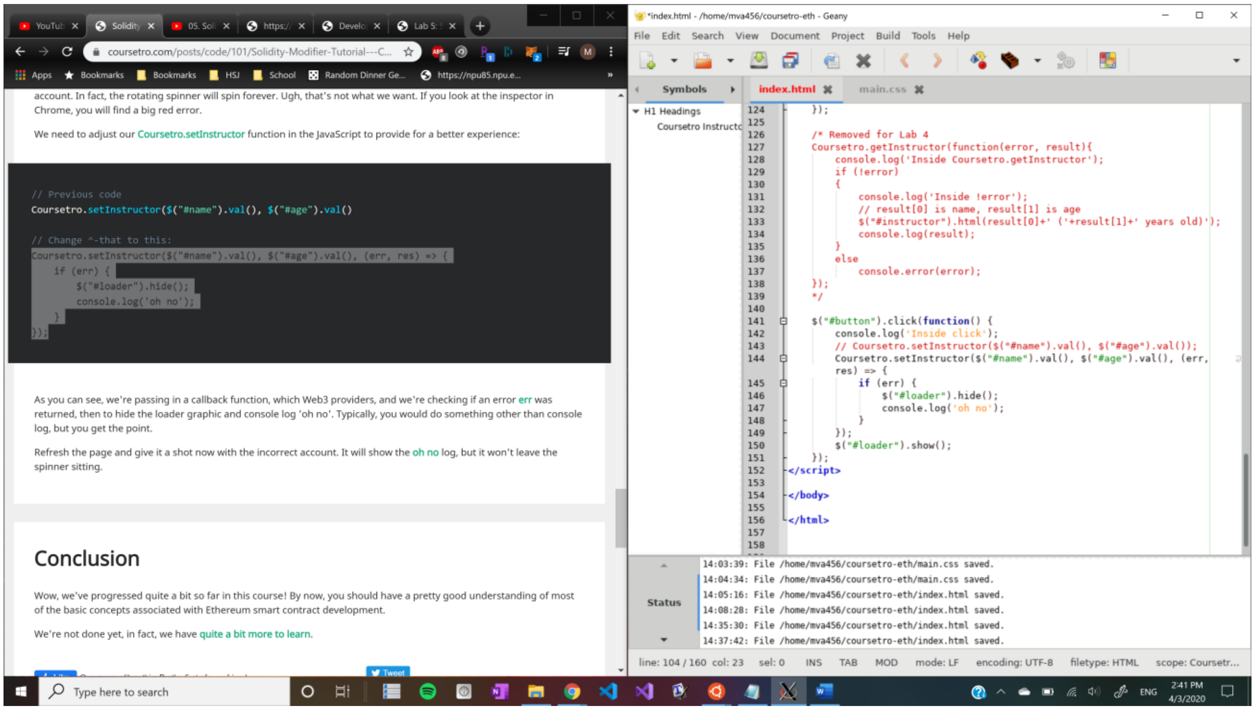
Transactions recorded: 2

Deployed Contracts

Coursetro at 0xe65...2a0c9 (blockchain)

setInstructor "abc", 10
getInstructor

0: string: abc
1: uint256: 10


coursetro.com/posts/code/101/Solidity-Modifier-Tutorial---C...
*index.html - /home/mva456/coursetro-eth - Geany
File Edit Search View Document Project Build Tools Help
Symbols
index.html
main.css
H1 Headings
Coursetro Instructo

account. In fact, the rotating spinner will spin forever. Ugh, that's not what we want. If you look at the inspector in Chrome, you will find a big red error.

We need to adjust our Coursetro.setInstructor function in the JavaScript to provide for a better experience:

```
// Previous code
Coursetro.setInstructor($("#name").val(), $("#age").val()

// Change ^-that to this:
Coursetro.setInstructor($("#name").val(), $("#age").val(), (err, res) => {
    if (err) {
        $("#loader").hide();
        console.log('oh no');
    }
});
```

As you can see, we're passing in a callback function, which Web3 providers, and we're checking if an error err was returned, then to hide the loader graphic and console log 'oh no'. Typically, you would do something other than console log, but you get the point.

Refresh the page and give it a shot now with the incorrect account. It will show the oh no log, but it won't leave the spinner sitting.

Conclusion

Wow, we've progressed quite a bit so far in this course! By now, you should have a pretty good understanding of most of the basic concepts associated with Ethereum smart contract development.

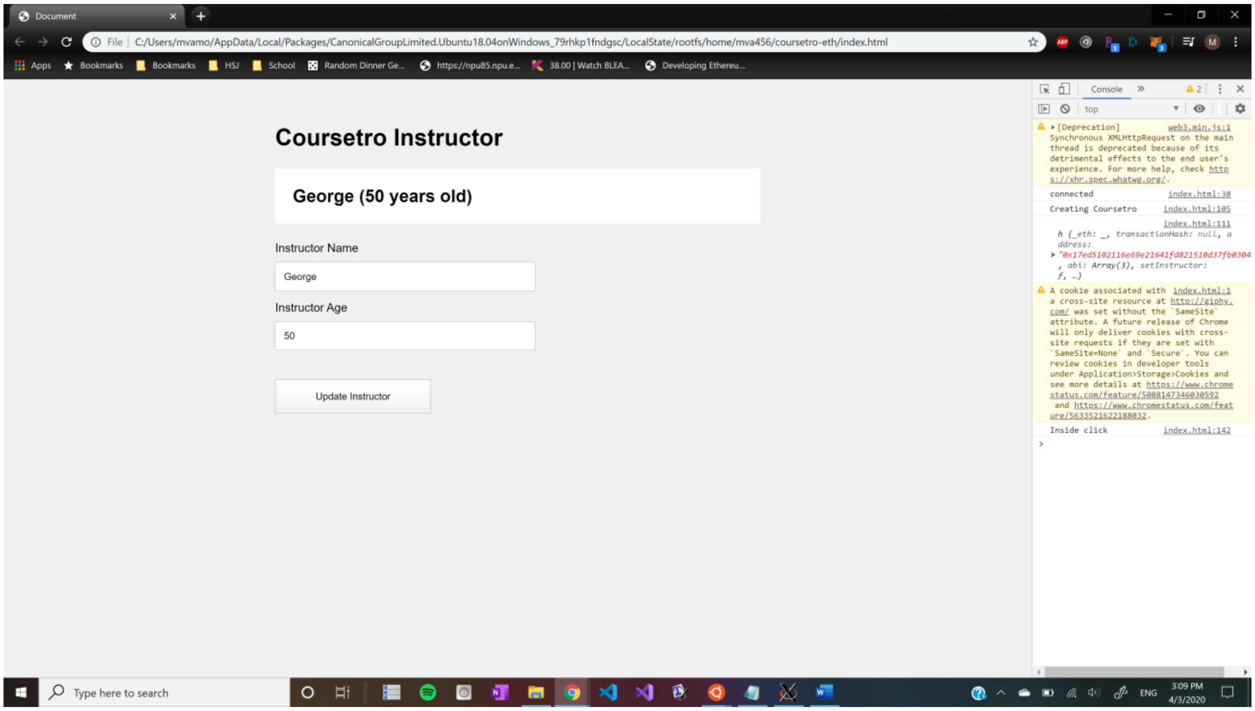
We're not done yet, in fact, we have quite a bit more to learn.

```
    });

    /* Removed for Lab 4
    Coursetro.getInstructor(function(error, result){
        console.log('Inside Coursetro.getInstructor');
        if (!error)
        {
            console.log('Inside !error');
            // result[0] is name, result[1] is age
            $("#instructor").html(result[0]+' ('+result[1]+' years old)');
            console.log(result);
        }
        else
            console.error(error);
    });
    */

    $("#button").click(function() {
        console.log('Inside click');
        // Coursetro.setInstructor($("#name").val(), $("#age").val());
        Coursetro.setInstructor($("#name").val(), $("#age").val(), (err,
        res) => {
            if (err) {
                $("#loader").hide();
                console.log('oh no');
            }
        });
        $("#loader").show();
    });
</script>

</body>

</html>
```

```
14:03:39: File /home/mva456/coursetro-eth/main.css saved.
14:04:34: File /home/mva456/coursetro-eth/main.css saved.
14:05:16: File /home/mva456/coursetro-eth/index.html saved.
14:08:28: File /home/mva456/coursetro-eth/index.html saved.
14:35:30: File /home/mva456/coursetro-eth/index.html saved.
14:37:42: File /home/mva456/coursetro-eth/index.html saved.
```

line: 104 / 160 col: 23 sel: 0 INS TAB MOD mode: LF encoding: UTF-8 filetype: HTML scope: Coursetr...

Status
Type here to search
2:41 PM
4/3/2020


Document
File | C:/Users/mvamo/AppData/Local/Packages/CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndgsc/LocalState/rootfs/home/mva456/coursetro-eth/index.html
Console
top
Type here to search
3:09 PM
4/3/2020

Coursetro Instructor

George (50 years old)

Instructor Name

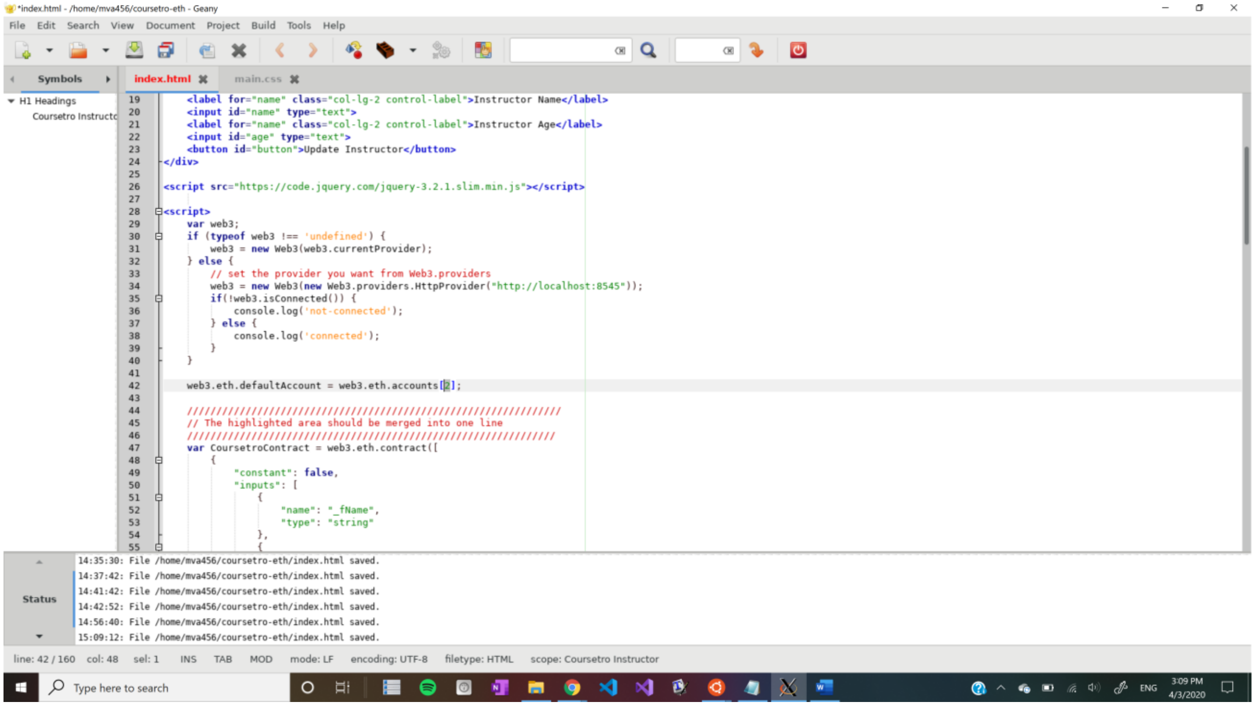
George

Instructor Age

50

Update Instructor

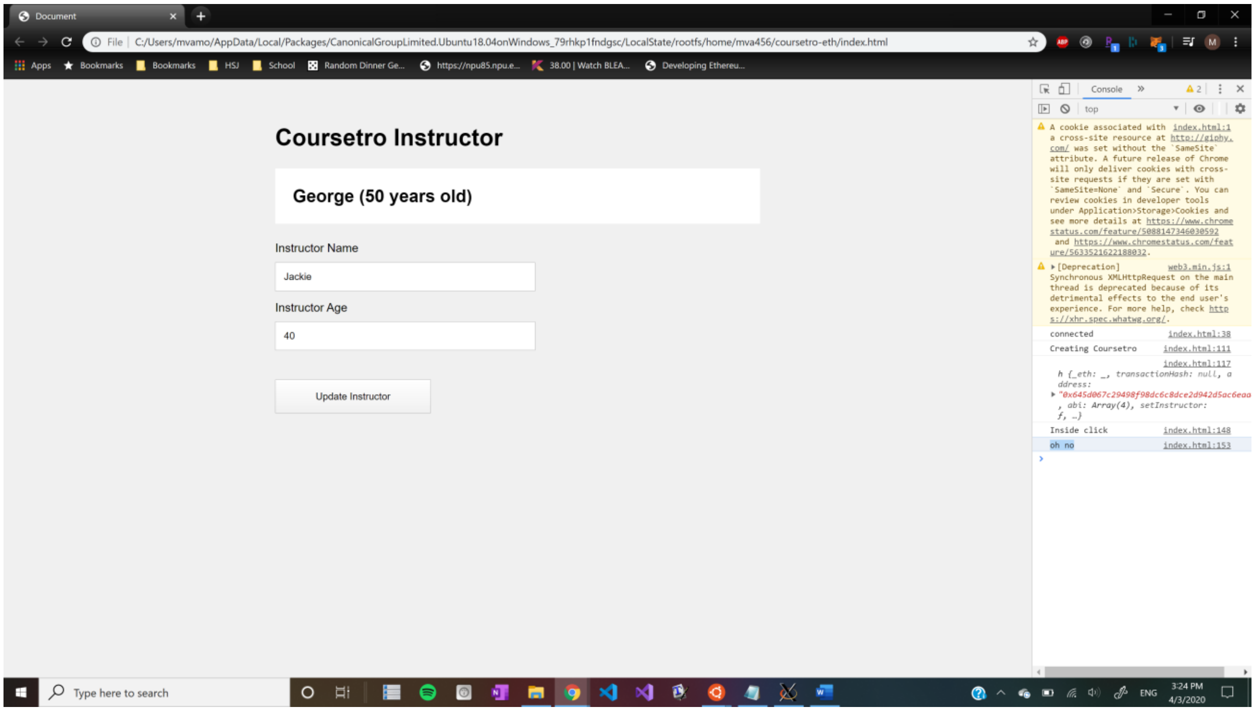
```
▸ [Deprecation]                          web3.min.js:1
Synchronous XMLHttpRequest on the main
thread is deprecated because of its
detrimental effects to the end user's
experience. For more help, check https://xhr.spec.whatwg.org/.
connected                                index.html:38
Creating Coursetro                       index.html:105
                                         index.html:111
  h {_eth: _, transactionHash: null, address:
▸ "0x17ed5102116e69e21641fd821510d37fb0304
  , abi: Array(3), setInstructor:
  f, …}
A cookie associated with                 index.html:1
a cross-site resource at http://giphy.com/ was set without the `SameSite`
attribute. A future release of Chrome
will only deliver cookies with cross-
site requests if they are set with
`SameSite=None` and `Secure`. You can
review cookies in developer tools
under Application>Storage>Cookies and
see more details at https://www.chromestatus.com/feature/5088147346030592
 and https://www.chromestatus.com/feature/5633521622188032.
Inside click                             index.html:142
>
```


*index.html - /home/mva456/coursetro-eth - Geany
File Edit Search View Document Project Build Tools Help
Symbols
H1 Headings
Coursetro Instructo
index.html
main.css
Status

```
    <label for="name" class="col-lg-2 control-label">Instructor Name</label>
    <input id="name" type="text">
    <label for="name" class="col-lg-2 control-label">Instructor Age</label>
    <input id="age" type="text">
    <button id="button">Update Instructor</button>
</div>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>

<script>
    var web3;
    if (typeof web3 !== 'undefined') {
        web3 = new Web3(web3.currentProvider);
    } else {
        // set the provider you want from Web3.providers
        web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
        if(!web3.isConnected()) {
            console.log('not-connected');
        } else {
            console.log('connected');
        }
    }

    web3.eth.defaultAccount = web3.eth.accounts[2];

    ///////////////////////////////////////////////////////////////////
    // The highlighted area should be merged into one line
    //////////////////////////////////////////////////////////////////
    var CoursetroContract = web3.eth.contract([
        {
            "constant": false,
            "inputs": [
                {
                    "name": "_fName",
                    "type": "string"
                },
                {
```

```
14:35:30: File /home/mva456/coursetro-eth/index.html saved.
14:37:42: File /home/mva456/coursetro-eth/index.html saved.
14:41:42: File /home/mva456/coursetro-eth/index.html saved.
14:42:52: File /home/mva456/coursetro-eth/index.html saved.
14:56:40: File /home/mva456/coursetro-eth/index.html saved.
15:09:12: File /home/mva456/coursetro-eth/index.html saved.
```

line: 42 / 160 col: 48 sel: 1 INS TAB MOD mode: LF encoding: UTF-8 filetype: HTML scope: Coursetro Instructor


Document
File
C:/Users/mvamo/AppData/Local/Packages/CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndgsc/LocalState/rootfs/home/mva456/coursetro-eth/index.html
Apps
Bookmarks
Bookmarks
HSJ
School
Random Dinner Ge...
https://npu85.npu.e...
38.00 | Watch BLEA...
Developing Ethereu...
Console
top

Coursetro Instructor

George (50 years old)

Instructor Name

Jackie

Instructor Age

40

Update Instructor

```
A cookie associated with  index.html:1
a cross-site resource at http://giphy.
com/ was set without the `SameSite`
attribute. A future release of Chrome
will only deliver cookies with cross-
site requests if they are set with
`SameSite=None` and `Secure`. You can
review cookies in developer tools
under Application>Storage>Cookies and
see more details at https://www.chrome
status.com/feature/5088147346030592
 and https://www.chromestatus.com/feat
ure/5633521622188032.
▸[Deprecation]              web3.min.js:1
Synchronous XMLHttpRequest on the main
thread is deprecated because of its
detrimental effects to the end user's
experience. For more help, check http
s://xhr.spec.whatwg.org/.
connected                  index.html:38
Creating Coursetro        index.html:111
                          index.html:117
  h {_eth: _, transactionHash: null, a
  ddress:
▸ "0x645d067c29498f98dc6c8dce2d942d5ac6eaa
  , abi: Array(4), setInstructor:
  f, …}
Inside click              index.html:148
oh no                     index.html:153
›
```