Tutorial 6 – Full Stack Applications

This week we will use the Pet Shop tutorial you implemented earlier to explore all the elements of a full stack DApp and how the parts all fit together. The purpose of this tutorial is for you to gain an appreciation of the various layers and the multiple technologies and programming languages that are often involved.

You will make some minor coding modifications to implement a few simple changes to the application, touching on each relevant layer. The Pet Shop tutorial is built using the following languages/formats:

- Solidity – the programming language for the Smart Contracts
- Javascript – much of the user interface code
- HTML and CSS – the visual elements of the user interface
- JSON files – storage of data about the animals

You will need to have completed the Pet Shop from the previous tutorial.

MAKE A COPY of the Pet Shop tutorial folder using filer explorer (Windows) or Finder (Mac) and name it something different such as pet-shop-enhanced.

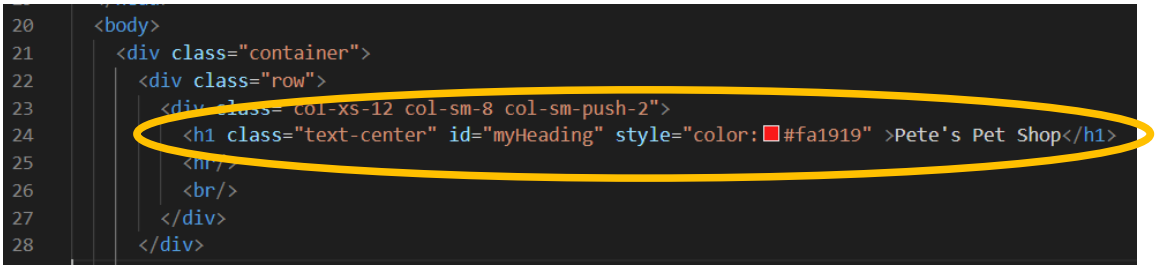
Before you start altering the code:

- make sure Ganache is running
- open your command prompt and navigate to the new folder
- open visual studio code
- start the mini web server (npm run dev)
- open the pet shop URL in your browser (localhost:3000)
- make sure Metamask is set to the Ganache network

1. Change the Heading Colour

The first change is a simple one. Change the colour of the heading. To do this, use vs.code to navigate to the index.html page, found in the src folder.

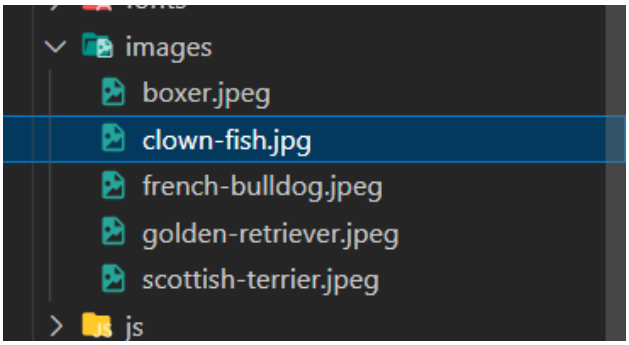
You will add two extra HTML tags in the line that creates the heading on line 24. One is an id tag which will be used later in the tutorial, and the other is a color tag. Once you add the color tag, Vs.code should display a colour picker and you can choose whatever colour you like for the heading.



Each time you change the values and save, you should see the page in your browser automatically change.

2. Insert another pet picture

The pictures of the pets are stored in a folder along with everything else. Find another image of your favourite pet and insert it into the images folder.

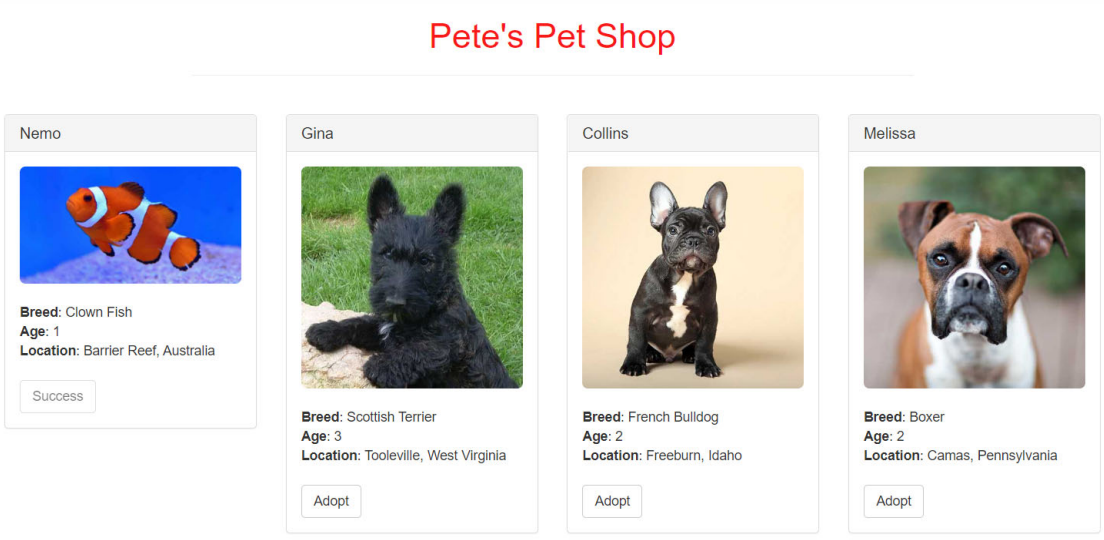


3. Change the first pet to your new pet

The list of pets that are displayed on the web page are defined in a JSON file called pets.json which is in the src folder. Edit the file and change the details of the first pet to be your new pet with the picture pointing to the new image file you added to the images folder. You do not need to insert any new lines, just edit the existing first pet.

```
src > {…} pets.json > {} 1
[
  {
    "id": 0,
    "name": "Nemo",
    "picture": "images/clown-fish.jpg",
    "age": 1,
    "breed": "Clown Fish",
    "location": "Barrier Reef, Australia"
  },
```

Save changes and then go back and refresh your browser to view the new list.



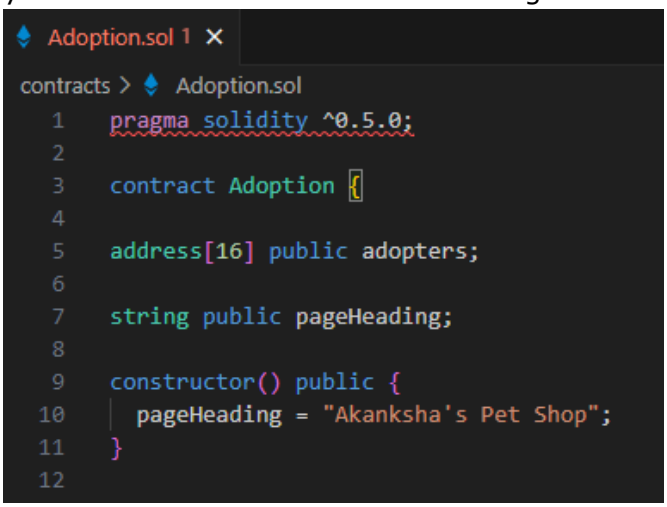
4. Adding a heading variable to the Smart Contract and displaying it on the home page

Next we want to be able to show a home page heading using a heading title set inside the smart contract, rather than just always being 'Pet's Pet Shop'.

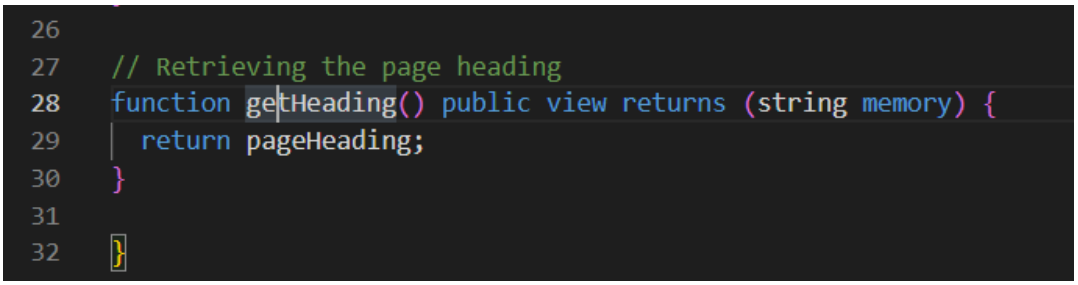
Doing this task will show you all the layers of code required to get a value inside the smart contract to actually appear in the user interface. It is not simple.

4.1 Add a heading variable to the Adoption smart contract

Use vs.code to edit Adoption.sol that you had previously created. You need to add some code at the start of the contract, just after the line that declares the adopters. One line of code will declare a variable to store the pageHeading. Then you will need to add a constructor to give the heading a value.



Next, you will need to add a method at the end of the contract, just before the last '}' bracket, to retrieve the heading value:



Make sure you save changes when you are done.

4.2 Re-compile and Re-deploy the smart contract

Check that the contract still compiles on the command line with truffle.
If you are still running the mini web server for the frontend, close it with ctl-C and then:

```
truffle compile –all
```

As the contract code has changed, you will need to re-deploy the smart contract to the blockchain on the command line.

```
truffle migrate --reset
```

4.3 Add a new loadHeading function into App.js

You will also need to add Javascript code into the user interface to retrieve the heading from the smart contract and insert it into the page. This will be added to the app.js file that you originally pasted code into.

Add a comma to the last bracket of the handleAdopt() function and paste the new function just after that. The new code to paste in is:

```
loadHeading: function() {
    var adoptionInstance;

    const headingTitle = document.getElementById("myHeading");

    App.contracts.Adoption.deployed().then(function(instance) {

      adoptionInstance = instance;

      return adoptionInstance.getHeading.call();
    }).then(function(newHeading) {
      headingTitle.textContent = newHeading;

    }).catch(function(err) {
      headingTitle.textContent = "Error loading Heading";
    });
  }
```

After inserting the comma and pasting this code, it should look like the following.

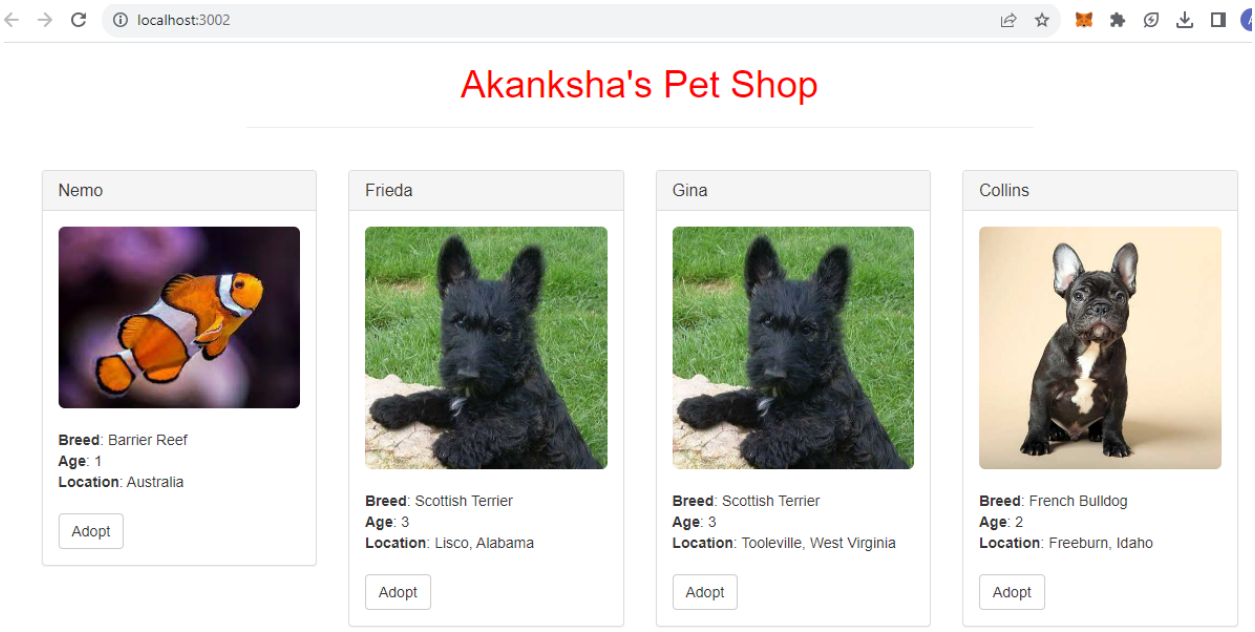
```
111         console.log(err.message);
112       });
113     });
114   },
115
116   loadHeading: function() {
117     var adoptionInstance;
118
119     const headingTitle = document.getElementById("myHeading");
120
121     App.contracts.Adoption.deployed().then(function(instance) {
122
123       adoptionInstance = instance;
124
125       return adoptionInstance.getHeading.call();
126     }).then(function(newHeading) {
127       headingTitle.textContent = newHeading;
128
129     }).catch(function(err) {
130       headingTitle.textContent = "Error loading Heading";
131     });
132   }
133
134
135 };
136
137 $(function() {
138   $(window).load(function() {
139     App.init();
140   });
141 });
142
```

4.4 Add a call to the new function

You will also need to insert a line of code to call on the loadHeading function when the page loads. This will be inserted at the start of the markAdopted function.

```
70
71   markAdopted: function() {
72     var adoptionInstance;
73
74     App.loadHeading();
75
76     App.contracts.Adoption.deployed().then(function(instance) {
77       adoptionInstance = instance;
78
```

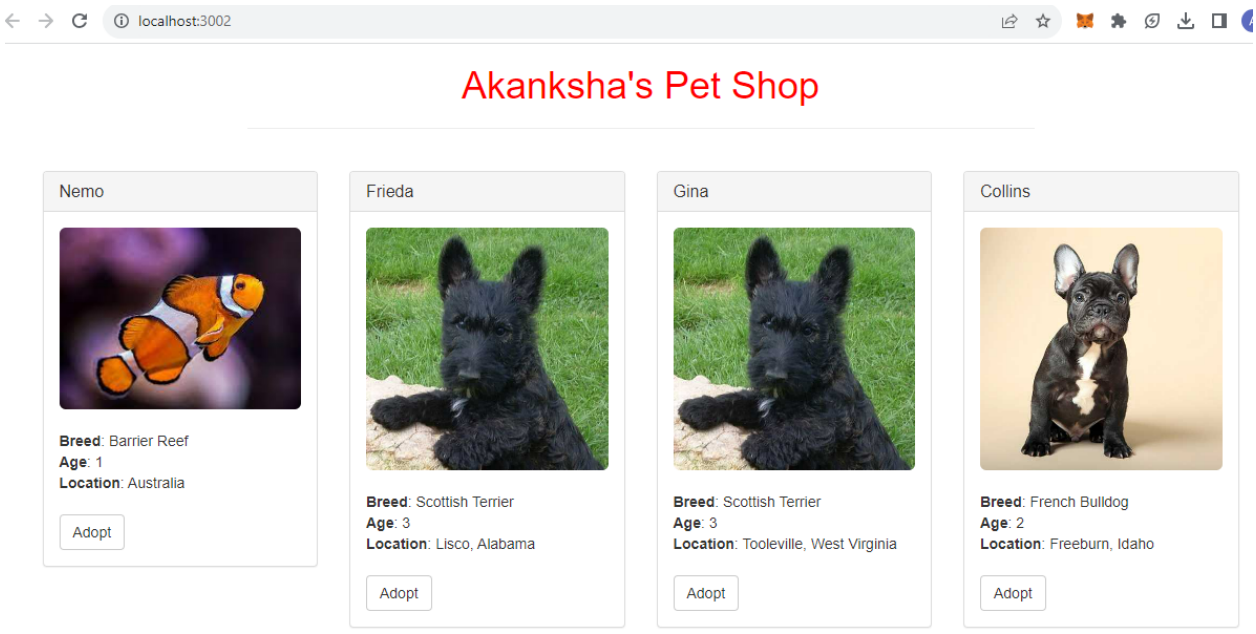
Now make sure you save all changes and then start up the mini web server again using 'npm run dev', and hopefully you can see your shiny new heading on the home page.



5. *Evidence for Labtask 2*

Provide the following screenshots:

- New pet shop home page showing a different colour and heading. Your colour and title must be different to the default and different to the colour in this tutorial screenshot.



- Vs.code with a screen shot of the Adoption.sol code