1

Great job! Now you know what a component is: a class with the `@Component` decorator, a selector, HTML template and CSS styles. Here's a recap of the root **AppComponent** code:

TS

```
@Component({
  selector: 'app-root',
  standalone: true,
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

}
```

In the next lesson, you'll create your own component!

5 COMMENTS

CONTINUE

2

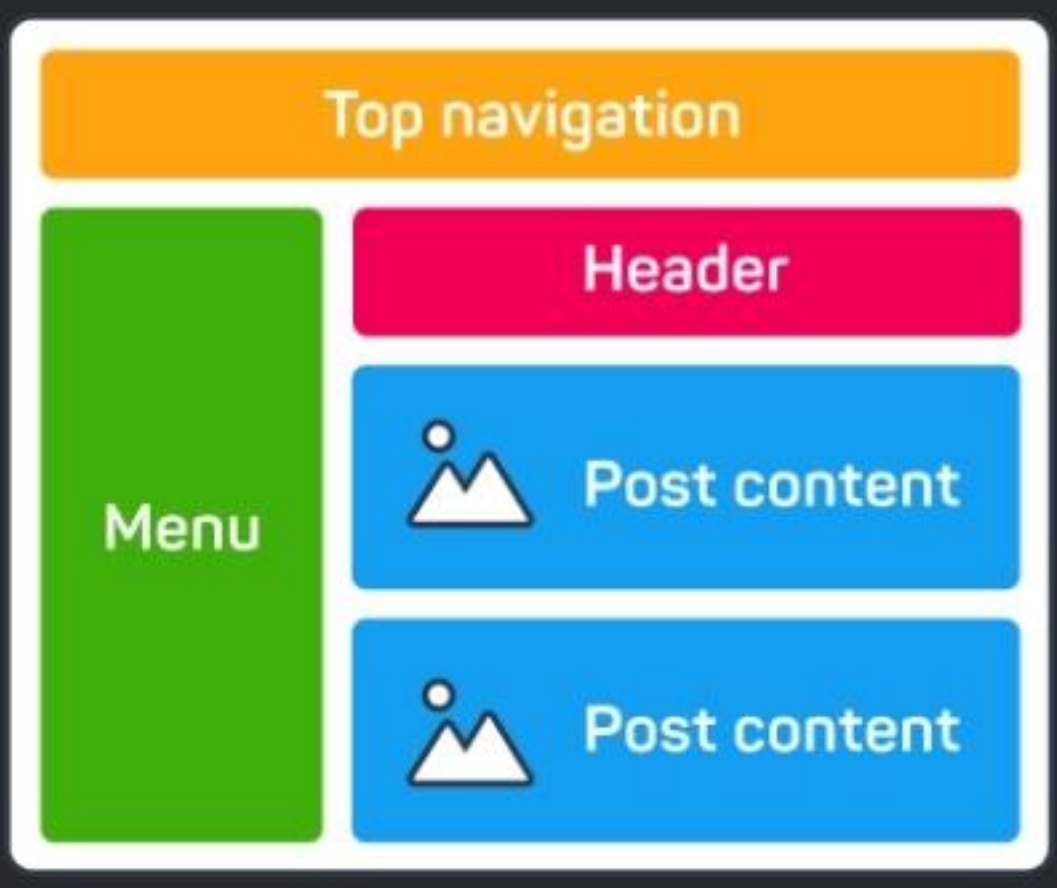
An Angular app has a **root** component, which is the parent of all the other components used to build the page. The root component is automatically created with a new Angular project.

Here is the `app.component.ts` file, which defines a root component called `AppComponent`

```ts
@Component({
  selector: 'app-root',
  standalone: true,
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent { }
```

The `@Component` **decorator** specifies metadata for the component class, including the HTML and CSS templates.

2



Components are the most basic building blocks of an Angular app's user interface (UI).

Each post on a social media app looks and behaves the same, because they are components.

Angular components are reusable pieces of code.

1 COMMENT

CONTINUE

2

Which is the main entry point of Angular apps?

`main.css`

`angular.cli`

`main.ts`

4 COMMENTS

5 in a row!

CONTINUE

2

Rearrange the commands to create a new Angular app called `shop`, then navigate to the project folder and run the app in the browser

```
ng new shop
```

```
cd shop
```

```
ng serve --open
```

ANSWER

CHECK

2

A new Angular project has been created in the Code Playground for you.

CODE PLAYGROUND **HTML**

```
<html>
  <head>
    <title>My app</title>
...

Tap to try |
```

To create the project on your local machine, run the `ng new` command and provide a name for your app.

7 COMMENTS

CONTINUE

2

Can you change the text displayed on the page?

Open the code, change the HTML heading text to anything you like, then check out the result in the Preview

CODE PLAYGROUND HTML

```
<h1>Hello from Angular!</h1>

Tap to try |
```

Congrats! You just made your first Angular app!

5 COMMENTS

CONTINUE

2

The source files reside in the `src/` subfolder.

- `index.html` is the main HTML of the app
- `main.ts` is the entry point of your app and runs when your app is launched
- `styles.css` is the main CSS file, which includes the styles for your app

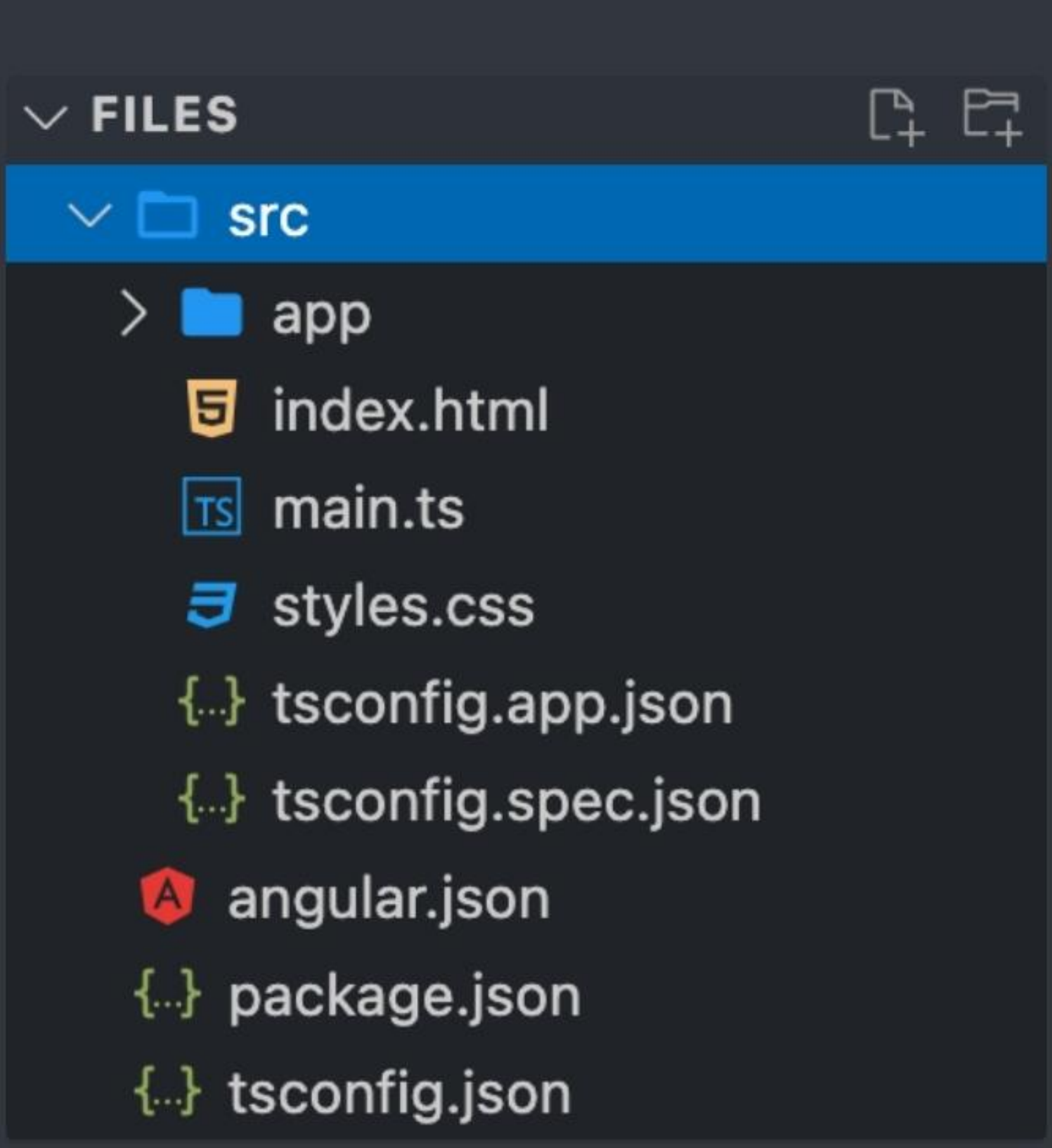
1 COMMENT

CONTINUE

2

Angular projects are made of different files.

In this lesson, you'll explore the file structure of an Angular project.



4 COMMENTS

CONTINUE

2

Complete to create a new Angular app named 'photoEditor', then navigate to the project folder and run the app in the browser

```
ng new photoEditor
cd photoEditor
ng serve --open
```

7 COMMENTS

3 in a row!

CONTINUE

2

To create a new Angular app, we need to use the command `ng new` and provide it the name for our new app. This creates all the necessary files and folders for an Angular app.

Fill in the blanks to create a new Angular app called `photoEditor`

ng new photoEditor

2

The first step of the setup is to install the **Angular CLI** (command-line interface), which is used to perform various Angular operations, such as creating and running apps, testing and deploying.

Complete to install the Angular CLI using the `npm install` command

npm install -g

@angular/cli

ANSWER

CHECK

2

semantically equal.

```
class A {
    private int x;
    public  boolean
 equals(Object o) {
        return  ((A)o).x == this.x;
    }
    public static void main(String[ ]
args)  {
     A a = new A();
     a.x = 9;
     A b = new  A  ();
     b.x = 5;
     System.out.println(a.  equals
(b));
```

ANSWER

CHECK

3

The automatically generated hashCode() method is used to determine where to store the object internally. Whenever you implement **equals**, you MUST also implement **hashCode**.

We can run the test again, using the **equals** method:

CODE PLAYGROUND **JAVA**

```java
public static void main(String[ ] args)
{
  Animal a1 = new Animal("Robby");
  Animal a2 = new Animal("Robby");
  System.out.println(a1.equals(a2));
}
```

Tap to try |

You can use the same menu to generate other useful methods, such as **getters** and **setters** for your class attributes.

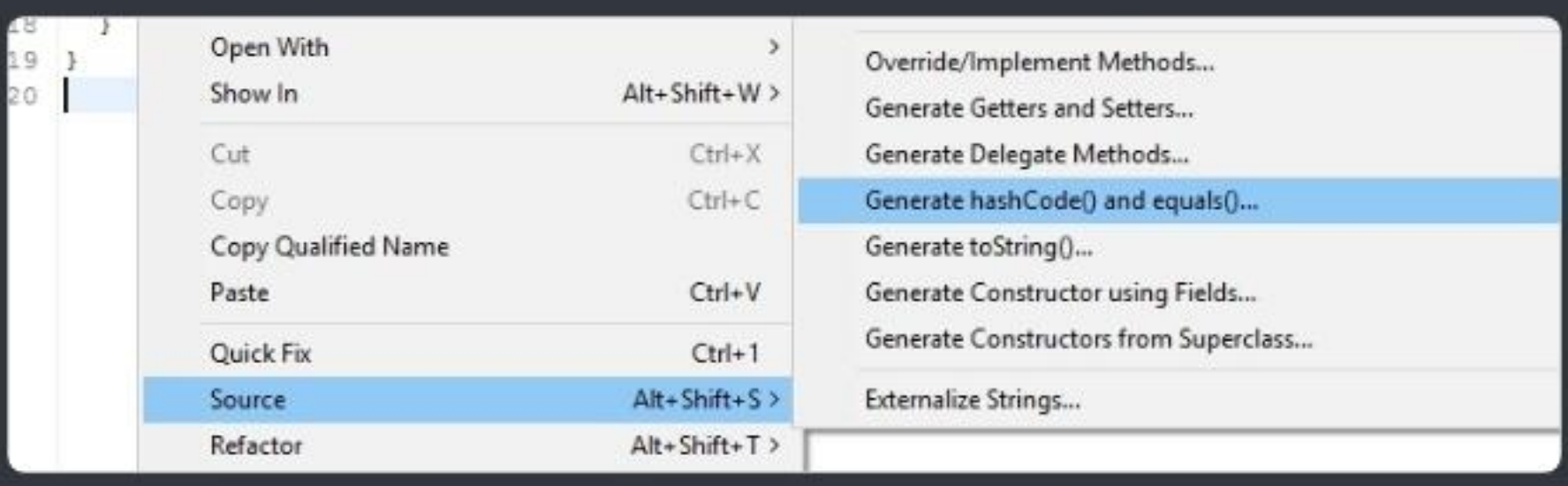
27 COMMENTS

CONTINUE

3

equals()

Each object has a predefined **equals()** method that is used for semantical equality testing. But, to make it work for our classes, we need to override it and check the conditions we need. There is a simple and fast way of generating the equals() method, other than writing it manually. Just right click in your class, go to **Source->Generate hashCode() and equals()...**



This will automatically create the necessary methods.

JAVA

27 COMMENTS

CONTINUE

3

Rearrange the code to have an inner class Hand, which has a method called "shake" that prints "Hi".

```
public class Person {
```

```
    class Hand {
```

```
    public void shake() {
```

```
    System.out.println("Hi");  }
```

```
    }   }
```

1 COMMENT

Job well done

CONTINUE

```java
class Robot {
    int id;
    Robot(int i) {
        id = i;
        Brain b = new Brain();
        b.think();
    }

    private class Brain {
        public void think() {
            System.out.println(id + " is thinking");
        }
    }
}

public class Program {
    public static void main(String[] args) {
        Robot r = new Robot(1);
    }
}
```

OUTPUT

```
1 is thinking
```

Anonymous Classes

PROBLEM CODE RESULT

```java
public class Main
{
  public static void main(String[] args) {

        Purchase customer = new Purchase();
        Purchase specialCustomer = new Purchase(){
            //your code goes here
            @Override public int totalAmount(int price)  {
             return price - (price*20)/100;
        }
        };


System.out.println(customer.totalAmount(1000));

System.out.println(specialCustomer.totalAmount(100000));
  }
}

class Purchase {
    int price;

    public int totalAmount(int price) {
        return price - (price*10)/100;
    }
}
```

TAB < > ! / . RUN

Anonymous Classes

You are a store manager.
You are offering a 10% discount on all items in the store. Today, you have had a total of two customers. To the first, you honored the 10% discount on all purchased items. The second customer, however, purchased a lot of items and you want to give him a bigger discount -- 20% -- to show your appreciation.
Complete the program by creating two Purchase objects - 1 for the regular customer, and 1 for a special one, and override the totalAmount() method for the special customer on the fly to set the proper 20% discount.

Method calls are already given.

START SOLVING

3

another object of that class, the start method's implementation will be the one defined in the class.

CODE PLAYGROUND JAVA

```
class Machine {
  public void start() {
    System.out.println("Starting...");
  }
}
public static void main(String[ ] args)
{
  Machine m1 = new Machine() {
    @Override public void start() {
      System.out.println("Woooooo");
    }
  };
  Machine m2 = new Machine();
  m2.start();
  }
```

Tap to try

Run the code and see how it works!

COMMENTS

CONTINUE

Crie um botão com o texto Send para que o usuário envie sua consulta.

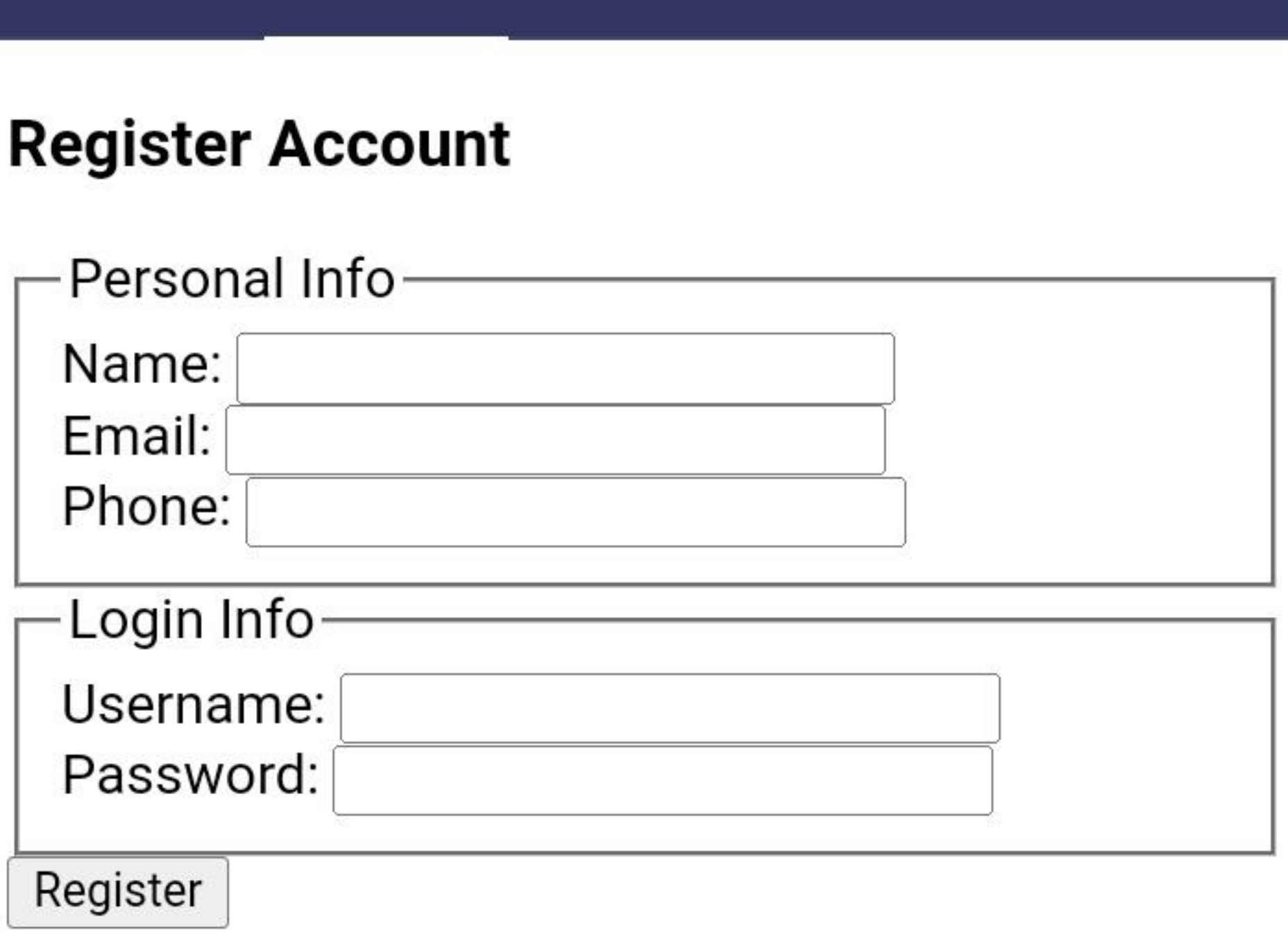
index.html Browser

```
<!doctype html>
<html>
 <body>
  <form>
    Name: <input type="text">
    <br>
    Phone: <input type="tel">
    <br>
    Email: <input type="email">
    <br>
    Message: <input type="text">
    <br>
    <button> Send </button>
  </form>
 </body>
</html>
```

</> CONTINUAR

Para enviar as informações de entrada, adicionamos um elemento `button` dentro do `form` . Codifique um `button` com o texto "Registrar" na parte inferior do formulário.

index.html Browser



index.html Browser

```
<!doctype html>
<html>
 <body>
  <h3>Register Account</h3>
  <form>
   <fieldset>
    <legend>Personal Info</legend>
    Name: <input type="text"><br>
    Email: <input type="email"><br>
    Phone: <input type="tel"><br>
    </fieldset>
    <fieldset>
     <legend>Login Info</legend>
     Username: <input
type="text"><br>
     Password: <input
type="password"><br>
    </fieldset>
    <button> Register </button>
  </form>
 </body>
</html>
```

</> CONTINUAR

Usamos **conjuntos de campos** para agrupar entradas relacionadas. Um `fieldset` exibirá uma borda ao redor dos elementos relacionados. Codifique os elementos `fieldset` .

index.html Browser

```
<!doctype html>
<html>
 <body>
  <h3>Register Account</h3>
  <form>
   <fieldset> Name: <input
type="text"><br> Email: <input
     type="email"><br> Phone:
<input type="tel"><br>
   </fieldset>
   <fieldset> Username: <input
type="text"><br> Password: <input
     type="password"><br>
   </fieldset>
   <button>Register</button>
  </form>
</body>
```

</>

CONTINUAR