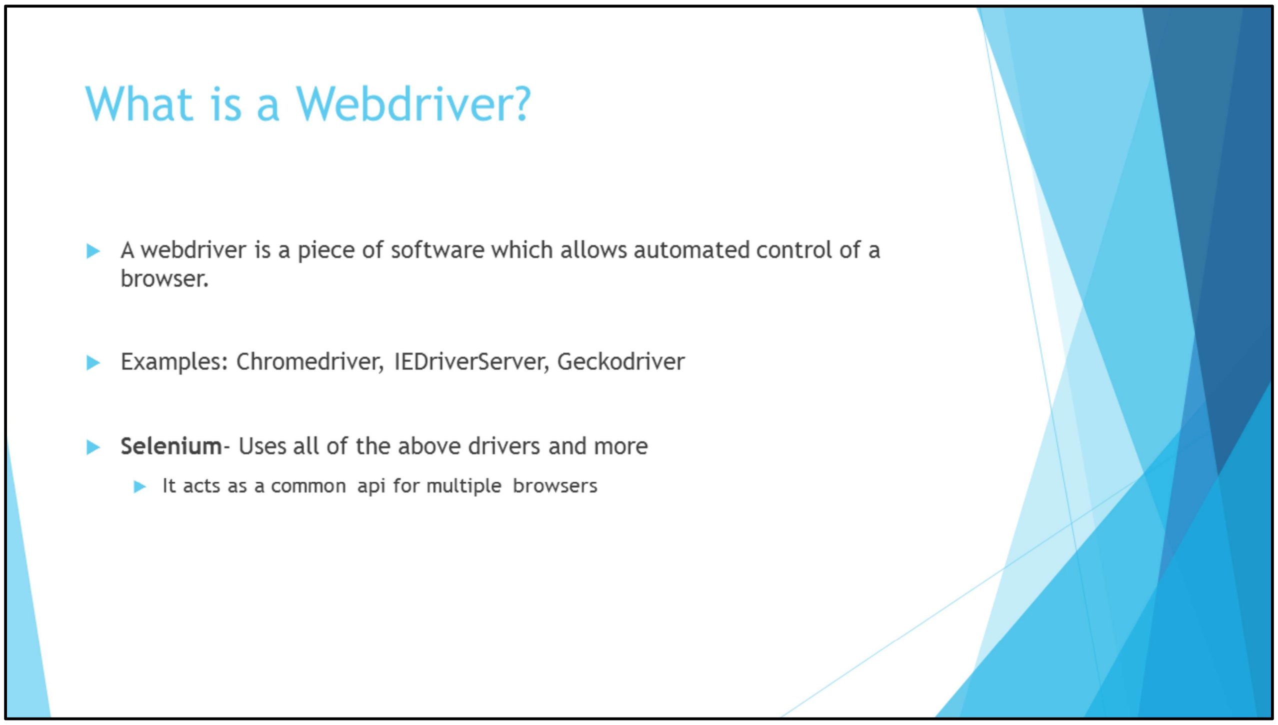
Webdriver: Page Object Design Pattern

Jared Sheffer

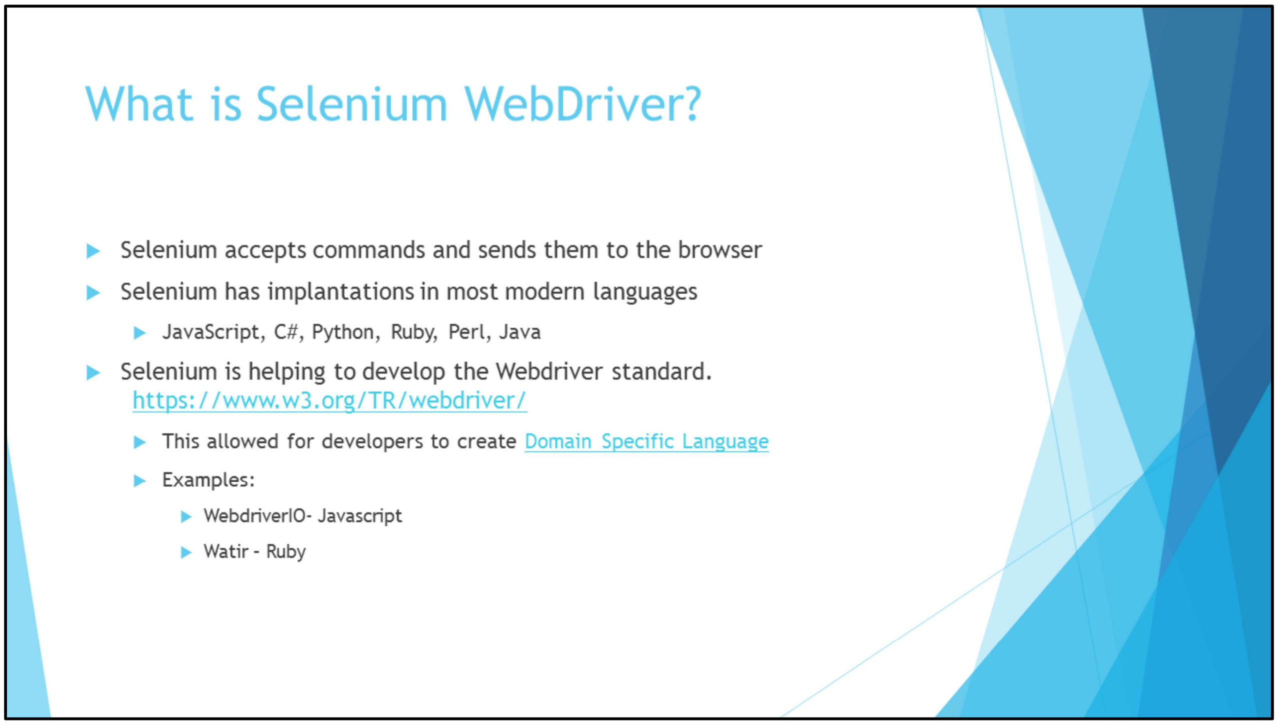


What is a Webdriver?

- A webdriver is a piece of software which allows automated control of a browser.
- Examples: Chromedriver, IEDriverServer, Geckodriver
- **Selenium**- Uses all of the above drivers and more
  - It acts as a common api for multiple browsers

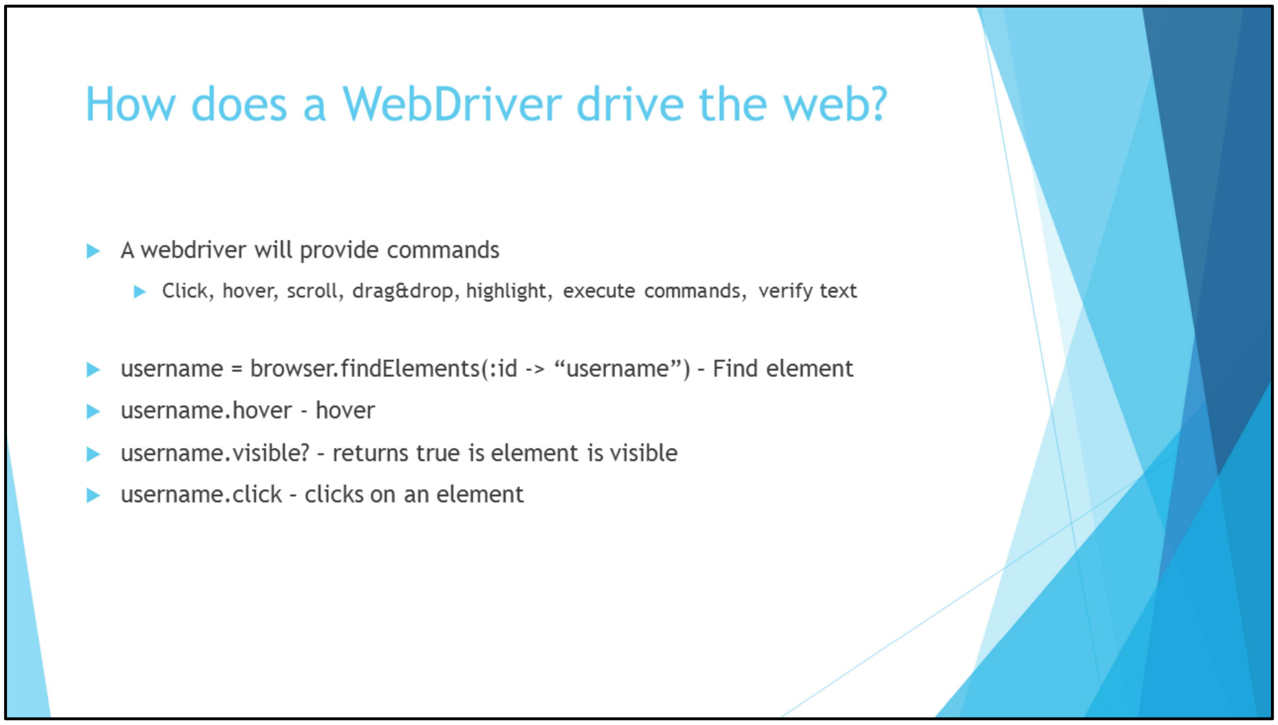
Most web browsers will provide their own webdriver for example there is the chromedriver for chrome, iedriverserver for ie, geckodriver for firefox

Selenium Web Driver is a tool for writing automated tests of websites. It aims to mimic the behavior of a real user, and as such interacts with the HTML of the application.



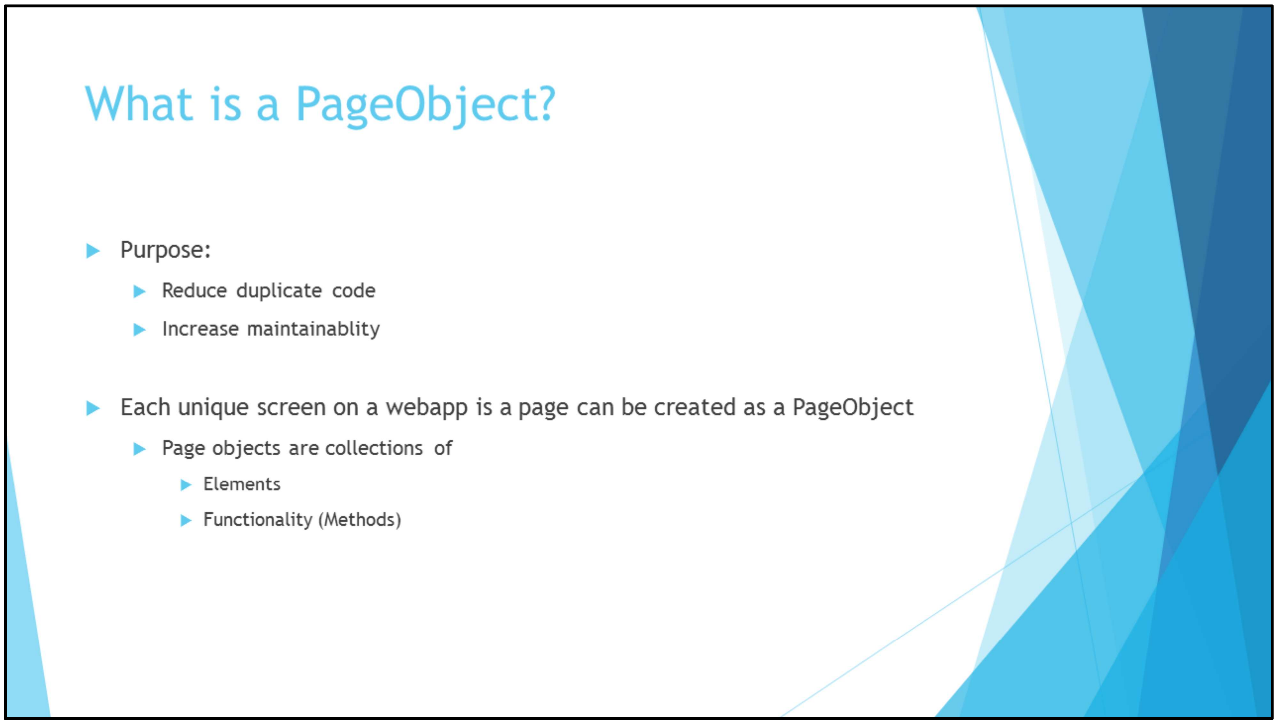
What is Selenium WebDriver?

- Selenium accepts commands and sends them to the browser
- Selenium has implantations in most modern languages
  - JavaScript, C#, Python, Ruby, Perl, Java
- Selenium is helping to develop the Webdriver standard. https://www.w3.org/TR/webdriver/
  - This allowed for developers to create Domain Specific Language
  - Examples:
    - WebdriverIO- Javascript
    - Watir - Ruby



How does a WebDriver drive the web?

- A webdriver will provide commands
  - Click, hover, scroll, drag&drop, highlight, execute commands, verify text
- username = browser.findElements(:id -> “username”) - Find element
- username.hover - hover
- username.visible? - returns true is element is visible
- username.click - clicks on an element

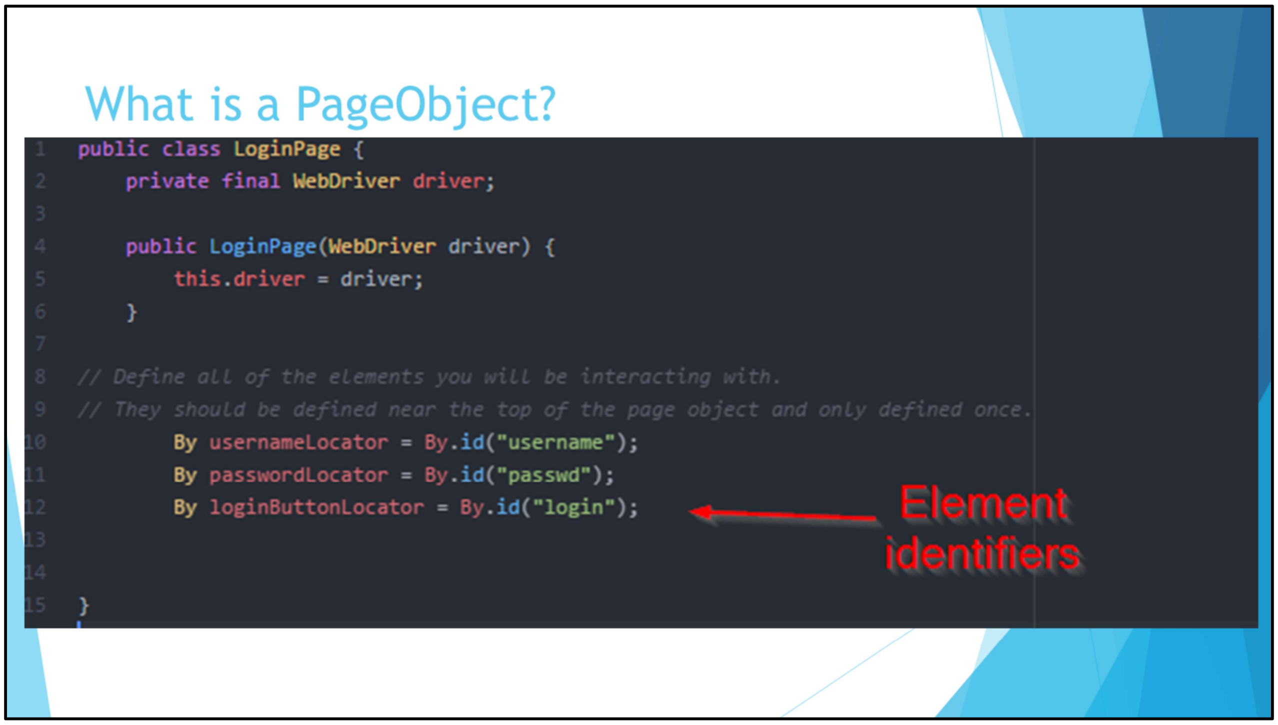


What is a PageObject?

- Purpose:
  - Reduce duplicate code
  - Increase maintainablity
- Each unique screen on a webapp is a page can be created as a PageObject
  - Page objects are collections of
    - Elements
    - Functionality (Methods)

A Page Object simply models these as objects within the test code.

Reduces the amount of duplicated code and means that if the UI changes, the fix need only be applied in one place.

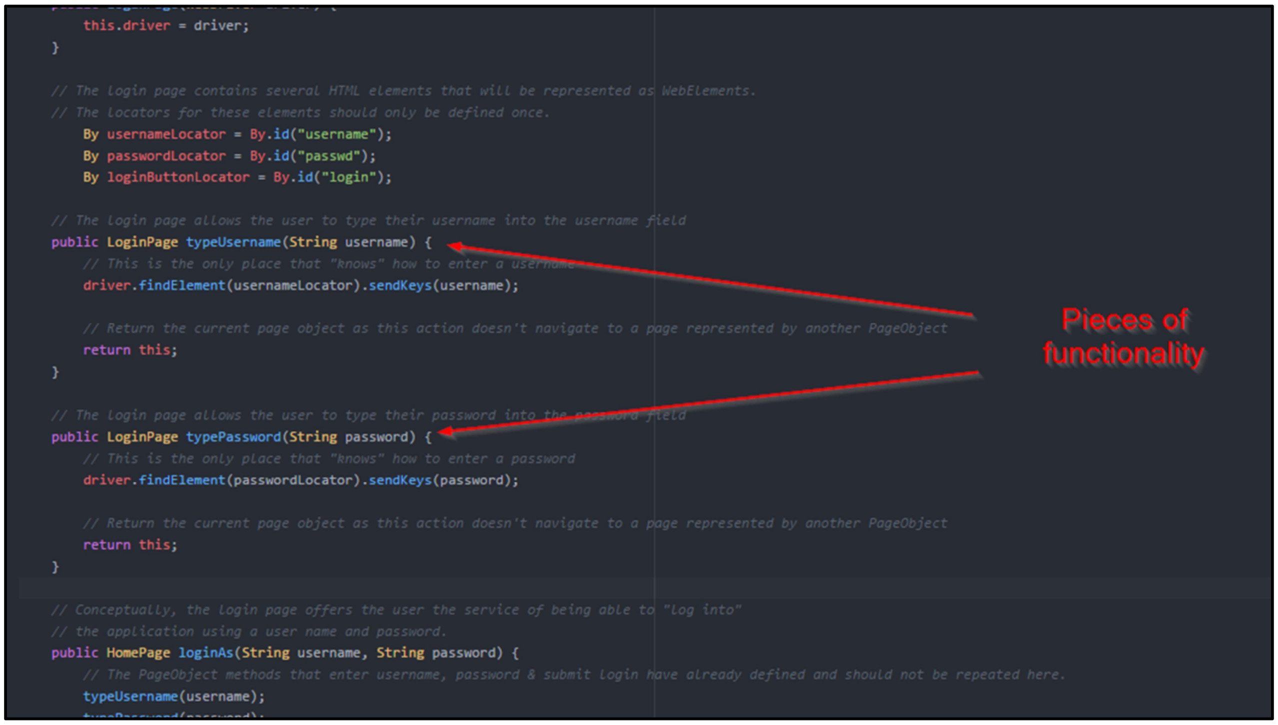


What is a PageObject?

```java
public class LoginPage {
    private final WebDriver driver;

    public LoginPage(WebDriver driver) {
        this.driver = driver;
    }

// Define all of the elements you will be interacting with.
// They should be defined near the top of the page object and only defined once.
        By usernameLocator = By.id("username");
        By passwordLocator = By.id("passwd");
        By loginButtonLocator = By.id("login");


}
```

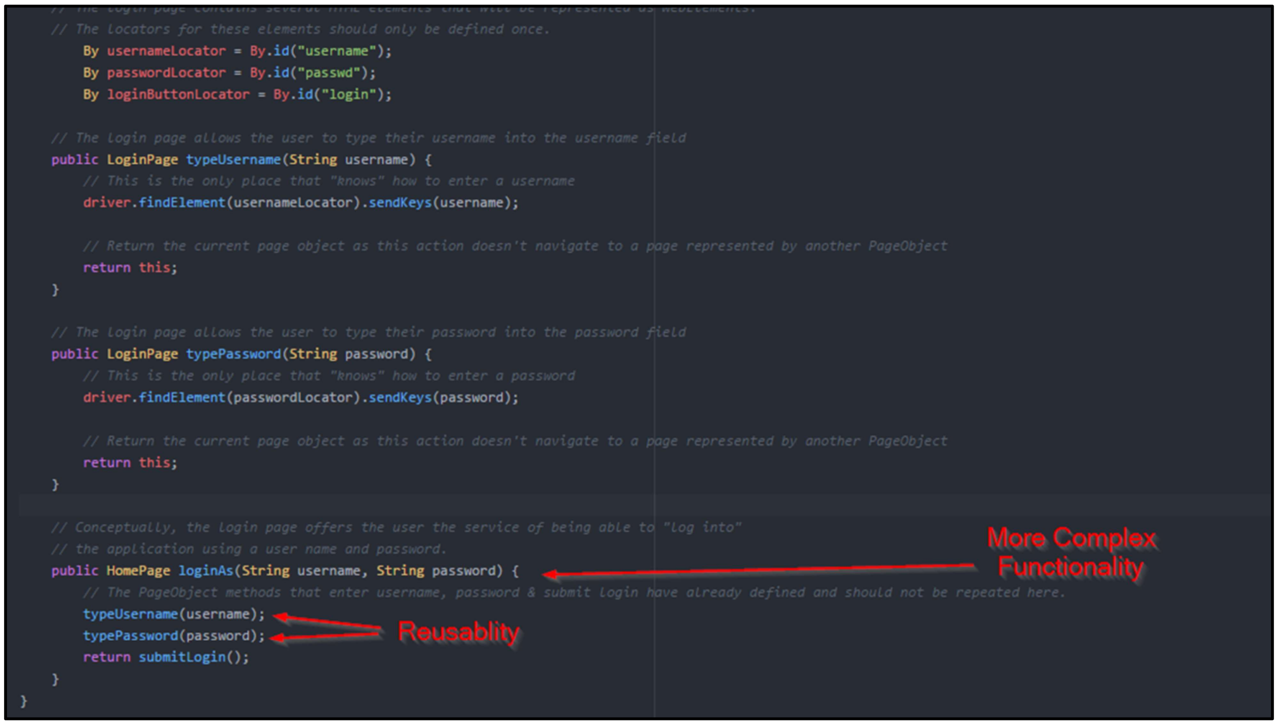
The main part of the pageobject is the element locators or identifers.
This is how the developer identifies the element on the page
This should the only place locators are used or defined.

By doing this it means if the locator ever changes you only change it in one place instead of multiple places.



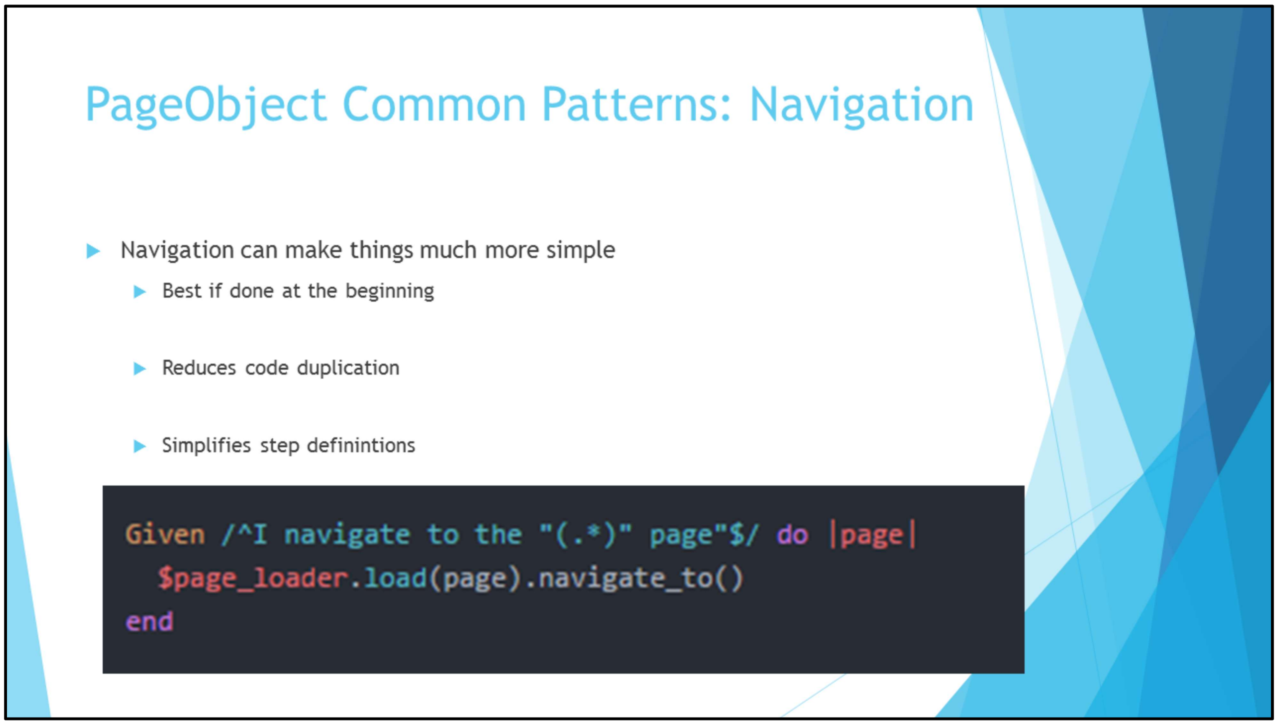
```java
        this.driver = driver;
    }

    // The login page contains several HTML elements that will be represented as WebElements.
    // The locators for these elements should only be defined once.
        By usernameLocator = By.id("username");
        By passwordLocator = By.id("passwd");
        By loginButtonLocator = By.id("login");

    // The login page allows the user to type their username into the username field
    public LoginPage typeUsername(String username) {
        // This is the only place that "knows" how to enter a username
        driver.findElement(usernameLocator).sendKeys(username);

        // Return the current page object as this action doesn't navigate to a page represented by another PageObject
        return this;
    }

    // The login page allows the user to type their password into the password field
    public LoginPage typePassword(String password) {
        // This is the only place that "knows" how to enter a password
        driver.findElement(passwordLocator).sendKeys(password);

        // Return the current page object as this action doesn't navigate to a page represented by another PageObject
        return this;
    }

    // Conceptually, the login page offers the user the service of being able to "log into"
    // the application using a user name and password.
    public HomePage loginAs(String username, String password) {
        // The PageObject methods that enter username, password & submit login have already defined and should not be repeated here.
        typeUsername(username);
```

We can then reuse the locators within methods.
We have two methods here which are unique pieces of functionality which will likely be used in multiple places
They each take an string and enter it into the correct field



```java
    // The locators for these elements should only be defined once.
        By usernameLocator = By.id("username");
        By passwordLocator = By.id("passwd");
        By loginButtonLocator = By.id("login");

    // The login page allows the user to type their username into the username field
    public LoginPage typeUsername(String username) {
        // This is the only place that "knows" how to enter a username
        driver.findElement(usernameLocator).sendKeys(username);

        // Return the current page object as this action doesn't navigate to a page represented by another PageObject
        return this;
    }

    // The login page allows the user to type their password into the password field
    public LoginPage typePassword(String password) {
        // This is the only place that "knows" how to enter a password
        driver.findElement(passwordLocator).sendKeys(password);

        // Return the current page object as this action doesn't navigate to a page represented by another PageObject
        return this;
    }

    // Conceptually, the login page offers the user the service of being able to "log into"
    // the application using a user name and password.
    public HomePage loginAs(String username, String password) {
        // The PageObject methods that enter username, password & submit login have already defined and should not be repeated here.
        typeUsername(username);
        typePassword(password);
        return submitLogin();
    }
}
```

Then building off of that we use those methods as a larger piece of functionality to actually do a whole login



PageObject Common Patterns: Navigation

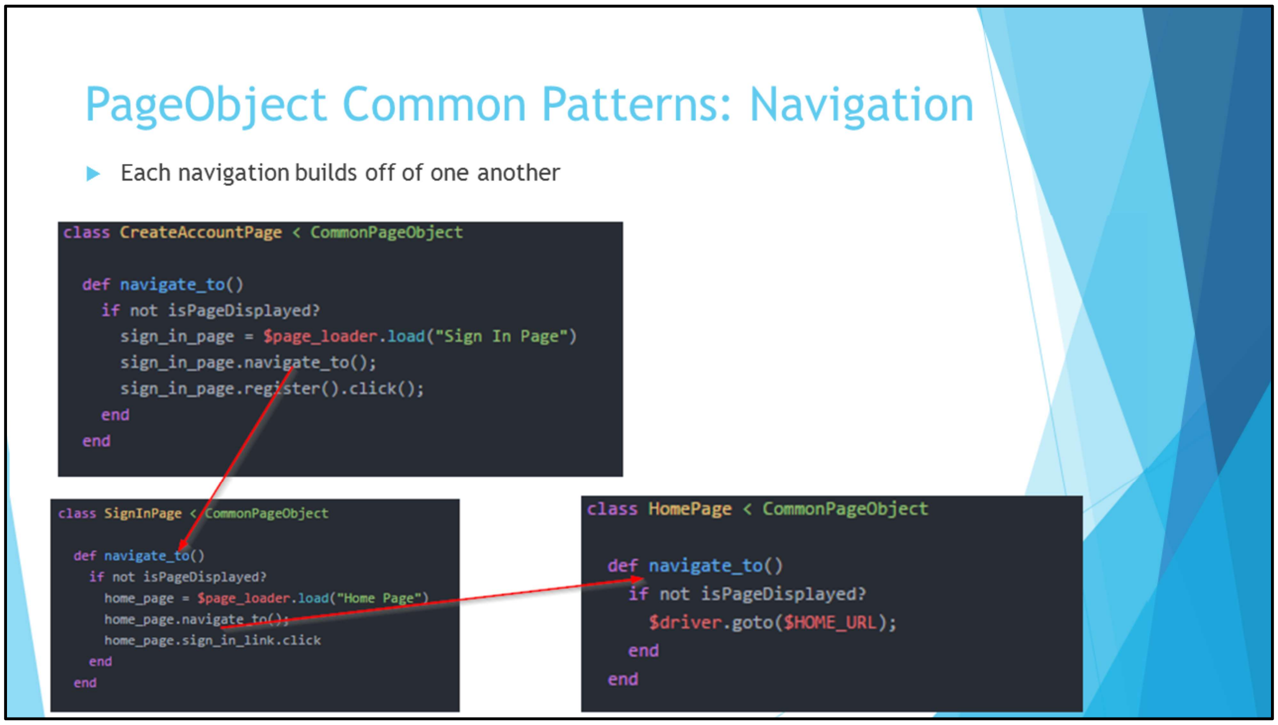
- Navigation can make things much more simple
  - Best if done at the beginning
  - Reduces code duplication
  - Simplifies step definintions

```
Given /^I navigate to the "(.*)" page"$/ do |page|
  $page_loader.load(page).navigate_to()
end
```

A Page Object simply models these as objects within the test code.

Reduces the amount of duplicated code and means that if the UI changes, the fix need only be applied in one place.

This is one of my steps which I include in every project I lead. This is a only line step which allows me to navigate to any page within my application



PageObject Common Patterns: Navigation

- Each navigation builds off of one another

```
class CreateAccountPage < CommonPageObject

  def navigate_to()
    if not isPageDisplayed?
      sign_in_page = $page_loader.load("Sign In Page")
      sign_in_page.navigate_to();
      sign_in_page.register().click();
    end
  end
```

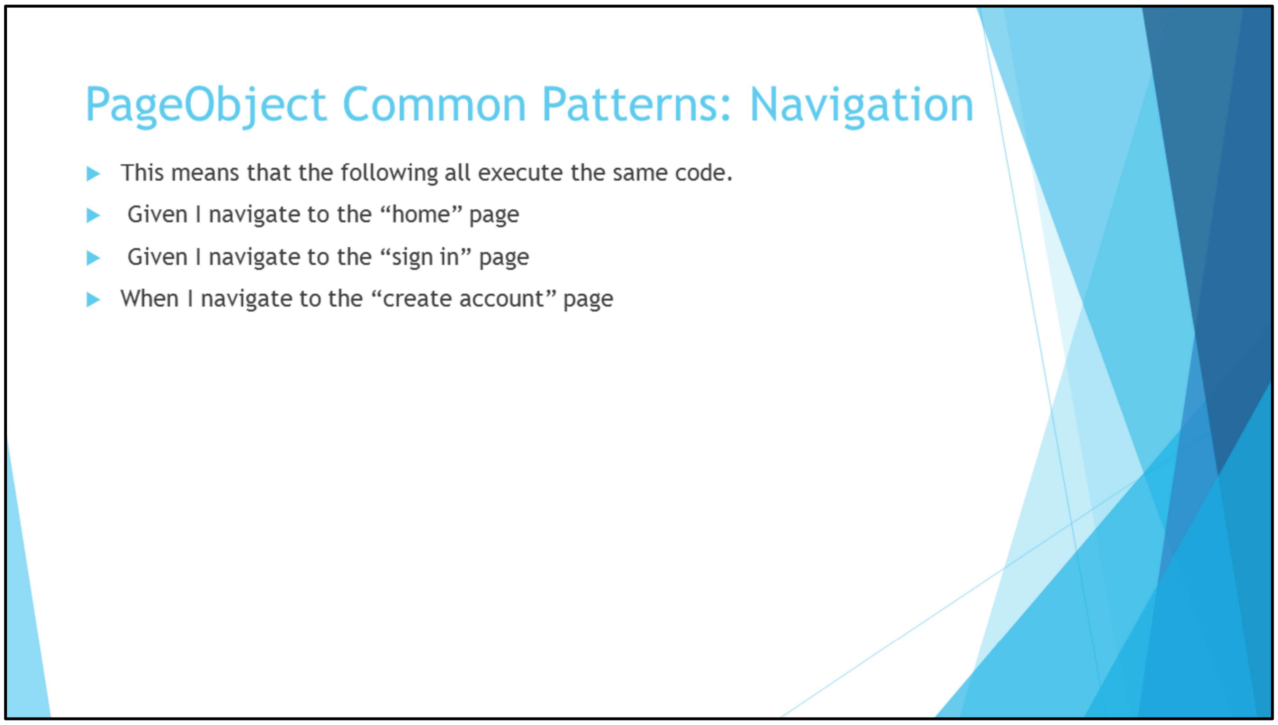
```
class SignInPage < CommonPageObject

  def navigate_to()
    if not isPageDisplayed?
      home_page = $page_loader.load("Home Page")
      home_page.navigate_to();
      home_page.sign_in_link.click
    end
  end
```

```
class HomePage < CommonPageObject

  def navigate_to()
    if not isPageDisplayed?
      $driver.goto($HOME_URL);
    end
  end
```

In this example when you call createAccountPage.navigate_to
 It will check to see if it is already on the createAccount page and if not it will execute its logic to get there and it knows there is really only one path to create account and that is through sign in
So createAccount.navigateToWill call SigninPage.navigateTo and click register.

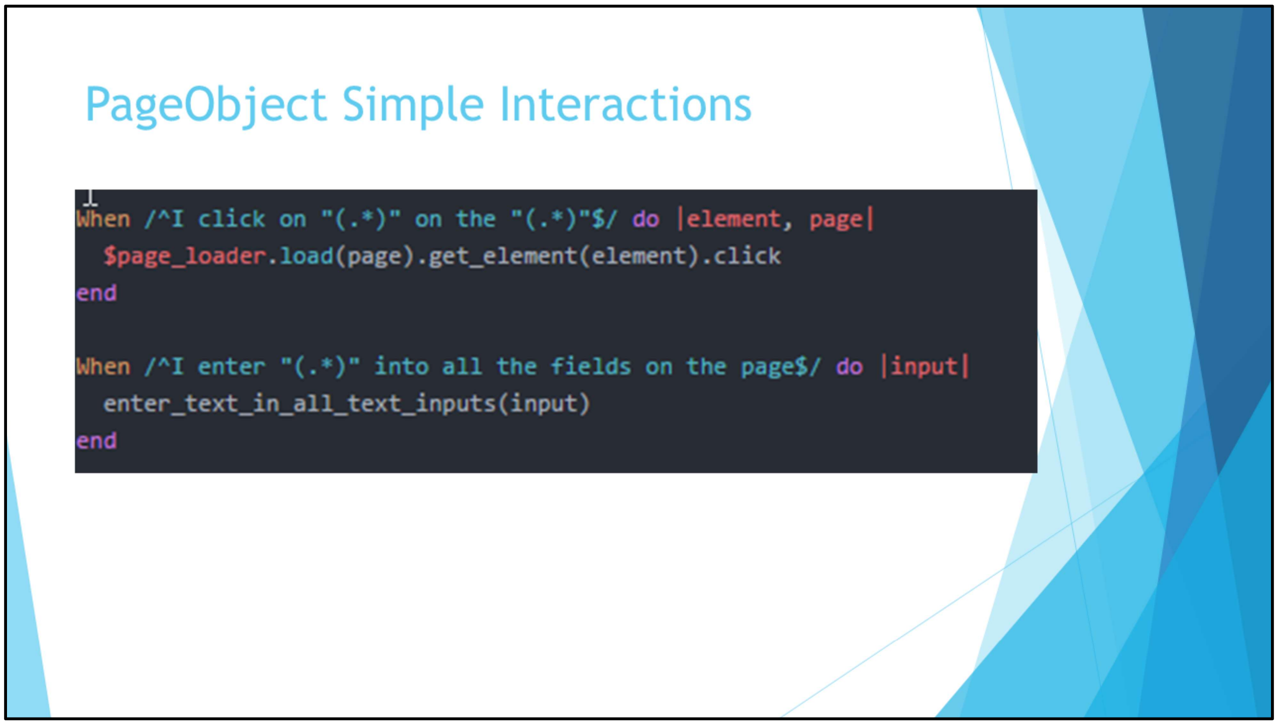
So ultimately if it cant find what page it is on them createAccountpage will call createAccount.navigateTo which will call SigninPage.navigateTo → Homepage.navigateTo

Then click the singin link on the homepage
And click the register link on the sign in page



PageObject Common Patterns: Navigation

- This means that the following all execute the same code.
- Given I navigate to the “home” page
- Given I navigate to the “sign in” page
- When I navigate to the “create account” page



PageObject Simple Interactions

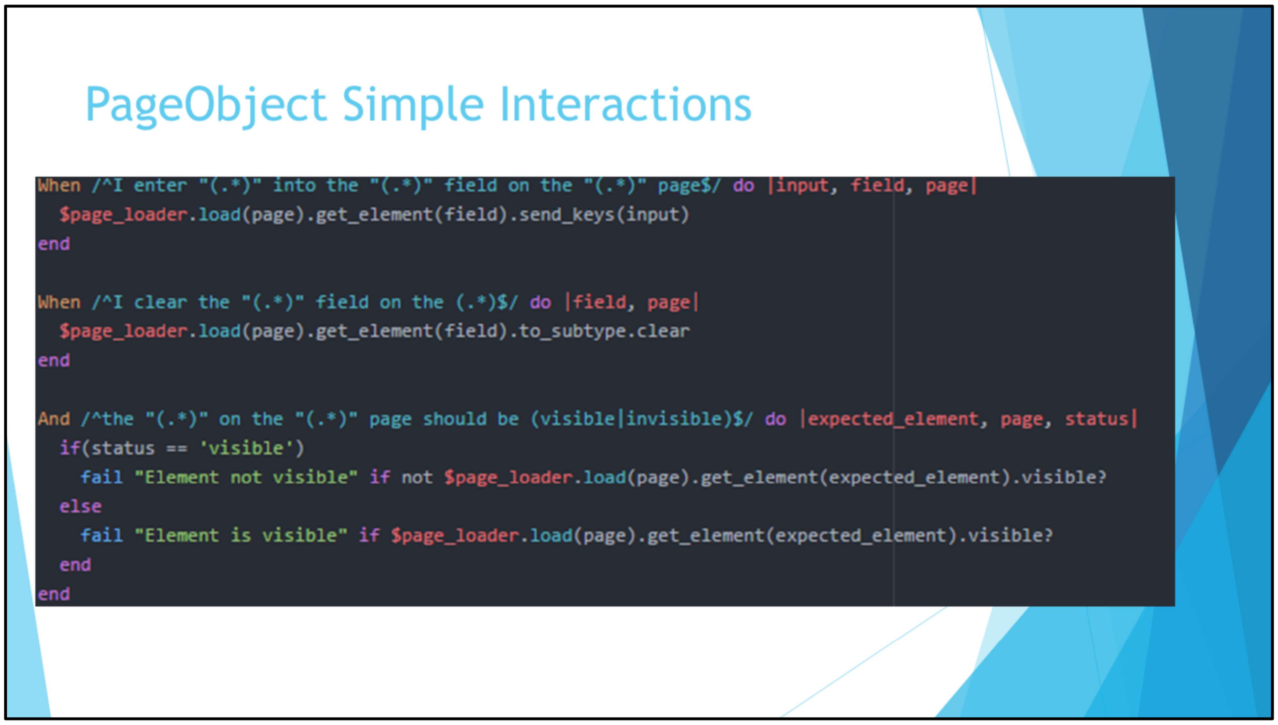
```
When /^I click on "(.*)" on the "(.*)"$/ do |element, page|
  $page_loader.load(page).get_element(element).click
end

When /^I enter "(.*)" into all the fields on the page$/ do |input|
  enter_text_in_all_text_inputs(input)
end
```

The page object model allows me to do things like this

I click on “something” on the “some” page.
I never have to write any other step which does that

I also have a method which will insert a given text into all the fields on the page.

This is useful if I need to fill out a form but it doesn’t matter what is in them.



PageObject Simple Interactions

```ruby
When /^I enter "(.*)" into the "(.*)" field on the "(.*)" page$/ do |input, field, page|
  $page_loader.load(page).get_element(field).send_keys(input)
end

When /^I clear the "(.*)" field on the (.*)$/ do |field, page|
  $page_loader.load(page).get_element(field).to_subtype.clear
end

And /^the "(.*)" on the "(.*)" page should be (visible|invisible)$/ do |expected_element, page, status|
  if(status == 'visible')
    fail "Element not visible" if not $page_loader.load(page).get_element(expected_element).visible?
  else
    fail "Element is visible" if $page_loader.load(page).get_element(expected_element).visible?
  end
end
```

The top one lets me enter any text into any field on any page with one line of code

The next one lets me clear the text out.

And the last step will allow me to validate the visibility of any element on any page

These are examples of how a Page Object can help simplify your step definintions.

Questions?

The top one lets me enter any text into any field on any page with one line of code

The next one lets me clear the text out.

And the last step will allow me to validate the visibility of any element on any page