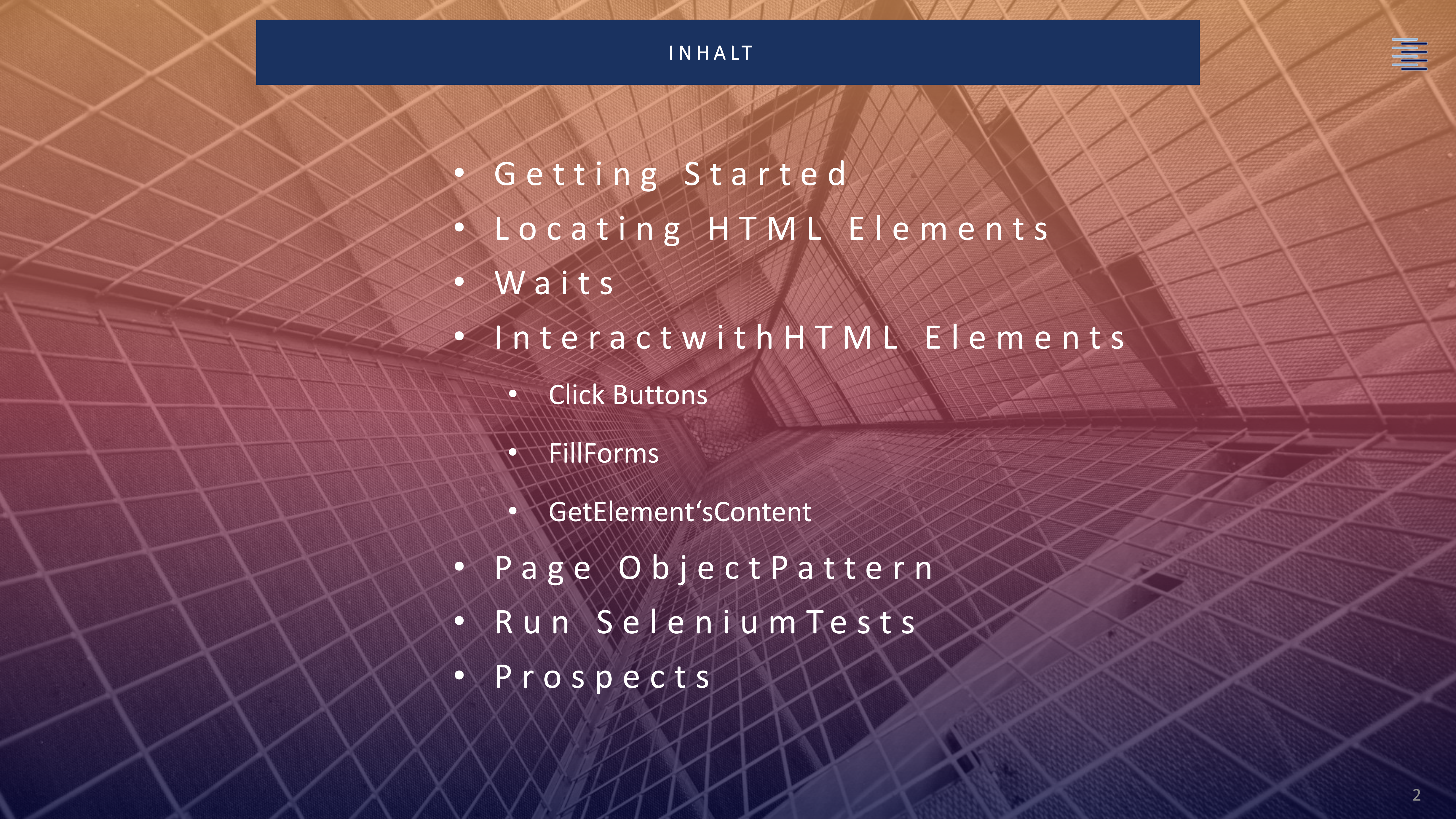
SOFTWARE-ENGINEERING

HAUSAUFGABE 2
EINFÜHRUNG

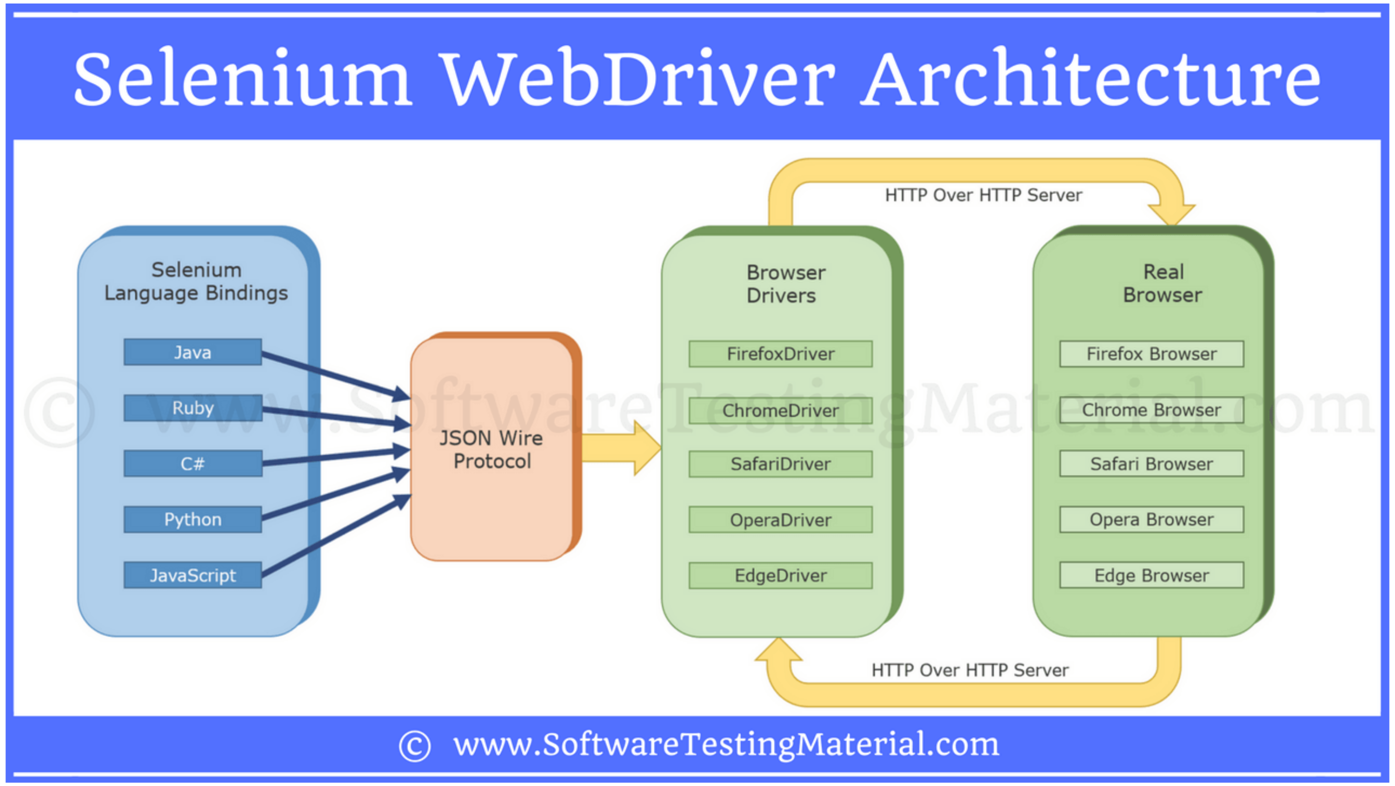
Hochschule für Angewandte Wissenschaften Hamburg
Department Medientechnik

Dr. Larissa Putzar



INHALT

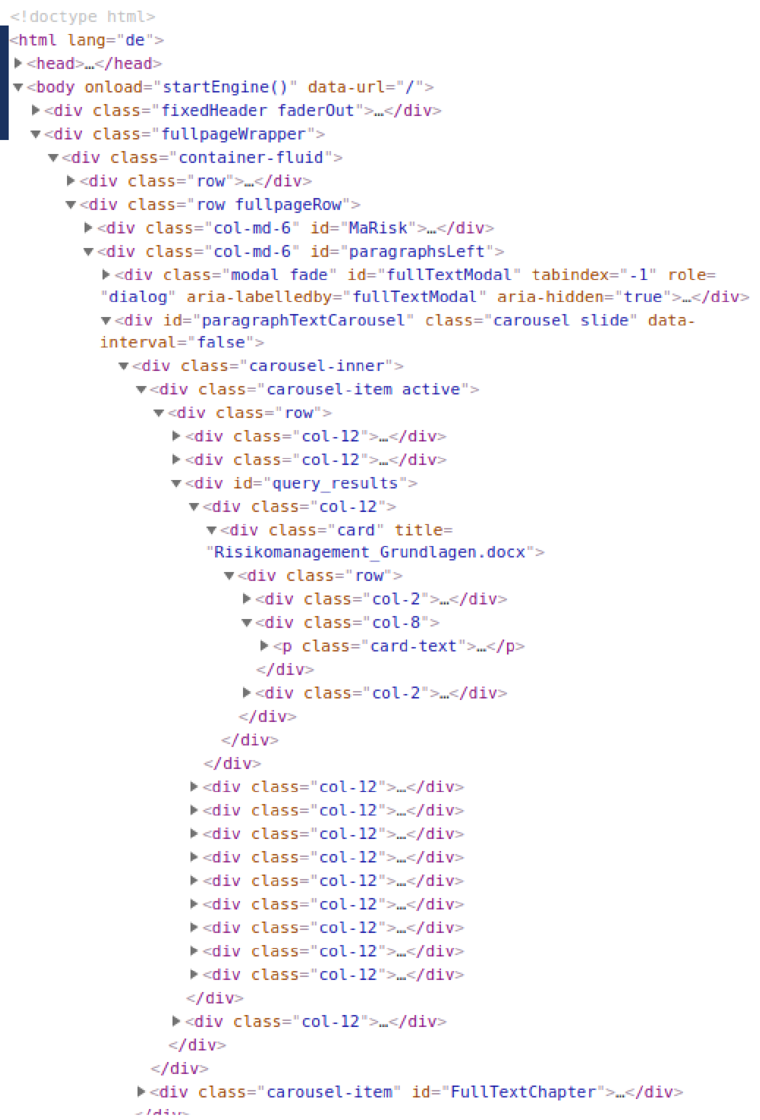
GETTING STARTED



- The language bindings provided by the Selenium project ("the driver").
- The executable which acts as a bridge between "chrome" and the "driver".
- There is the browser itself ("chrome").

INHALT

LOCATING HTML ELEMENTS - SINGLE ELEMENTS



```
<!doctype html>
<html lang="de">
 ▶<head>…</head>
 ▼<body onload="startEngine()" data-url="/">
   ▶<div class="fixedHeader faderOut">…</div>
   ▼<div class="fullpageWrapper">
     ▼<div class="container-fluid">
       ▶<div class="row">…</div>
       ▼<div class="row fullpageRow">
         ▶<div class="col-md-6" id="MaRisk">…</div>
         ▼<div class="col-md-6" id="paragraphsLeft">
           ▶<div class="modal fade" id="fullTextModal" tabindex="-1" role=
           "dialog" aria-labelledby="fullTextModal" aria-hidden="true">…</div>
           ▼<div id="paragraphTextCarousel" class="carousel slide" data-
           interval="false">
             ▼<div class="carousel-inner">
               ▼<div class="carousel-item active">
                 ▼<div class="row">
                   ▶<div class="col-12">…</div>
                   ▶<div class="col-12">…</div>
                   ▼<div id="query_results">
                     ▼<div class="col-12">
                       ▼<div class="card" title=
                       "Risikomanagement_Grundlagen.docx">
                         ▼<div class="row">
                           ▶<div class="col-2">…</div>
                           ▼<div class="col-8">
                             ▶<p class="card-text">…</p>
                           </div>
                           ▶<div class="col-2">…</div>
                         </div>
                       </div>
                     </div>
                     ▶<div class="col-12">…</div>
                     ▶<div class="col-12">…</div>
                     ▶<div class="col-12">…</div>
                     ▶<div class="col-12">…</div>
                     ▶<div class="col-12">…</div>
                     ▶<div class="col-12">…</div>
                     ▶<div class="col-12">…</div>
                     ▶<div class="col-12">…</div>
                     ▶<div class="col-12">…</div>
                   </div>
                   ▶<div class="col-12">…</div>
                 </div>
               </div>
               ▶<div class="carousel-item" id="FullTextChapter">…</div>
```

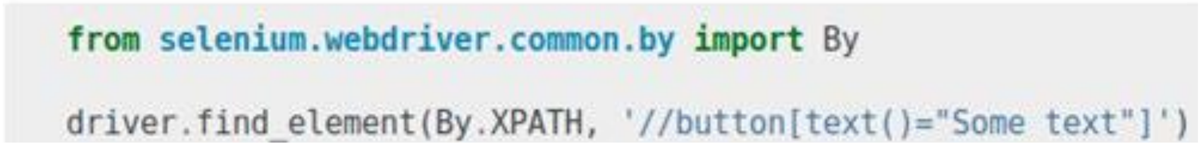
First of all you have to get the HTML-DOM: *driver.get(http://127.0.0.1:5000/results/)*

Selenium provides the following methods to locate elements in a page:

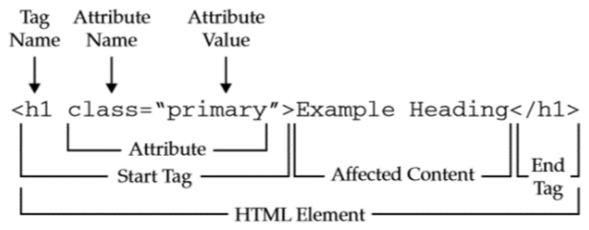
- find_element_by_id(<String>)
- find_element_by_name(<String>)
- find_element_by_tag_name(<String>)
- find_element_by_class_name(<String>)
- find_element_by_css_selector(<String>)
- find_element_by_xpath(<String>

Selenium provides this method for locators:

- find_element(<Locator>)



```python
from selenium.webdriver.common.by import By

driver.find_element(By.XPATH, '//button[text()="Some text"]')
```



LOCATING HTML ELEMENTS - SINGLE ELEMENTS

By ID

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
  </form>
 </body>
<html>
```

The form element can be located like this:

```
login_form = driver.find_element_by_id('loginForm')
```

By Name

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
</body>
<html>
```

The username & password elements can be located like this:

```
username = driver.find_element_by_name('username')
password = driver.find_element_by_name('password')
```

By Tage Name

```
<html>
 <body>
  <h1>Welcome</h1>
  <p>Site content goes here.</p>
</body>
<html>
```

The heading (h1) element can be located like this:

```
heading1 = driver.find_element_by_tag_name('h1')
```

Using Locator

```
from selenium.webdriver.common.by import By

driver.find_element(By.ID    , 'text')
```

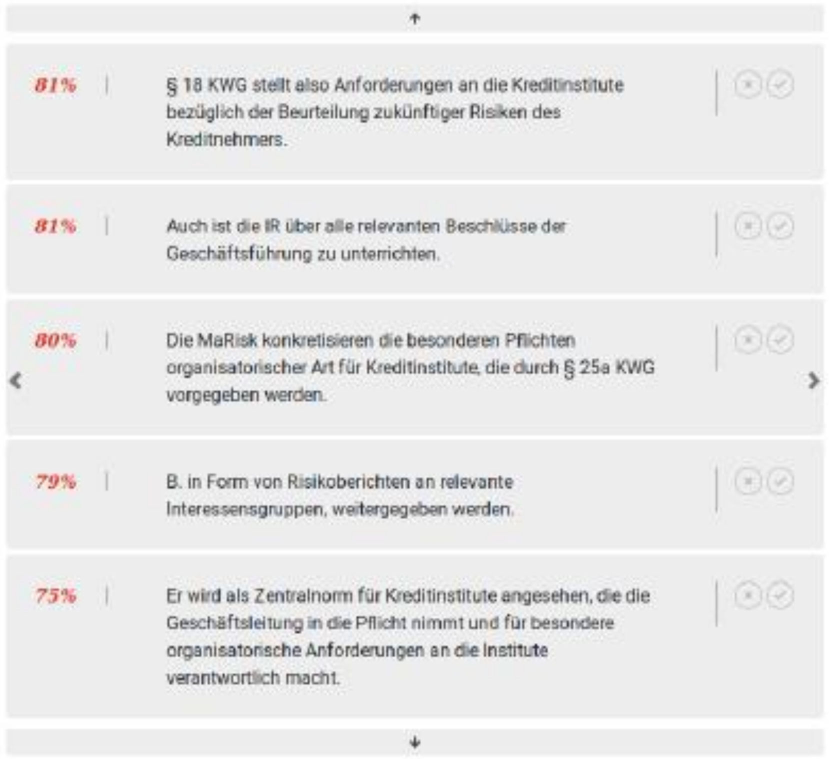
These are the attributes available for *By* class:

```
ID = "id"
XPATH = "xpath"
LINK_TEXT = "link text"
PARTIAL_LINK_TEXT = "partial link text"
NAME = "name"
TAG_NAME = "tag name"
CLASS_NAME = "class name"
CSS_SELECTOR = "css selector"
```

LOCATING HTML ELEMENTS - MULTIPLE ELEMENTS

- Find multiple elements:
  - *find_elements_by_name*(<String>)
  - *find_elements_by_xpath*(<String>)
  - *find_elements_by_tag_name*(<String>)
  - *find_elements_by_class_name*(<String>)
  - *find_elements_by_css_selector*(<String>)
- These methods will return a list
  - Access individual list's elements
  - Iterate through list
- Selenium provides this method for locators:
  - find_elements(<Locator>)
- Example:

```
# Iterate through the given results
for box in container.find_elements_by_class_name("col-12"):
    form = box.find_element_by_tag_name("form")
    label_tag = form.find_elements_by_tag_name("label")
    checkbox = label_tag[1]

    # Click on the checkbox
    checkbox.click()
```

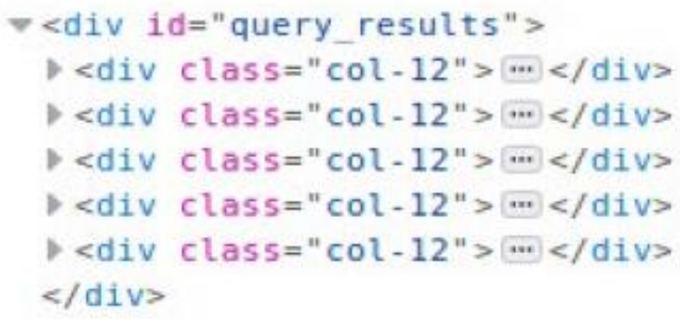

81% | § 18 KWG stellt also Anforderungen an die Kreditinstitute bezüglich der Beurteilung zukünftiger Risiken des Kreditnehmers.

81% | Auch ist die IR über alle relevanten Beschlüsse der Geschäftsführung zu unterrichten.

80% | Die MaRisk konkretisieren die besonderen Pflichten organisatorischer Art für Kreditinstitute, die durch § 25a KWG vorgegeben werden.

79% | B. in Form von Risikoberichten an relevante Interessensgruppen, weitergegeben werden.

75% | Er wird als Zentralnorm für Kreditinstitute angesehen, die die Geschäftsleitung in die Pflicht nimmt und für besondere organisatorische Anforderungen an die Institute verantwortlich macht.


```
▾<div id="query_results">
  ▸<div class="col-12">…</div>
  ▸<div class="col-12">…</div>
  ▸<div class="col-12">…</div>
  ▸<div class="col-12">…</div>
  ▸<div class="col-12">…</div>
 </div>
```

INHALT

WAITS

Most of the web apps are using AJAX techniques

When a page is loaded by the browser, the elements within that page may load at different time intervals.

This makes locating elements difficult: if an element is not yet present in the DOM, a locate function will raise an*ElementNotVisibleException*exception.

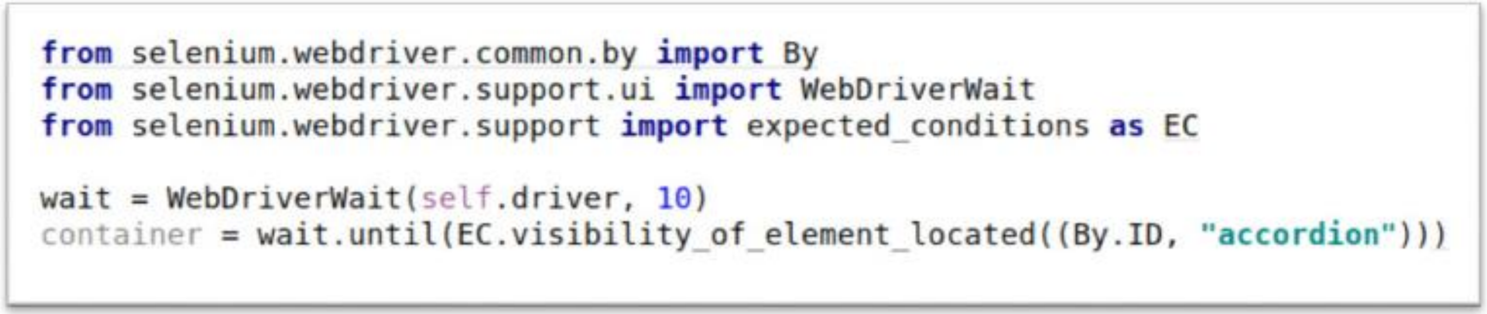
Selenium Webdriverprovides two types of waits -implicit & explicit.

- An implicit wait makes WebDriver poll the DOM for a certain amount of time when trying to locate an element. If the element is not available within the specified Time a NoSuchElementException will be raised.
- An explicit wait is a code that you define to wait for a certain condition to occur before proceeding further in the code. It is more extendible in the means that you can set it up to wait for any condition you might like (presence, visibility, clickability, …)

EXPLICIT WAIT-SYNTAX

Locator

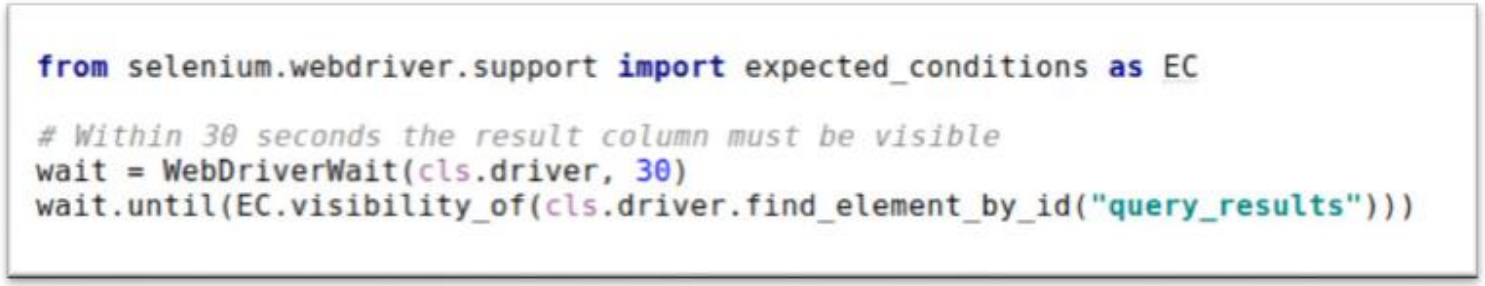
- Locator is a tuple of (By.<Locator>, <Name>)
- If you pass locator, WebDriver will use it to create Element object.
- If waiting condition is true within the given duration, element is saved in variable



```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

wait = WebDriverWait(self.driver, 10)
container = wait.until(EC.visibility_of_element_located((By.ID, "accordion")))
```

Element

- Element is a WebElement

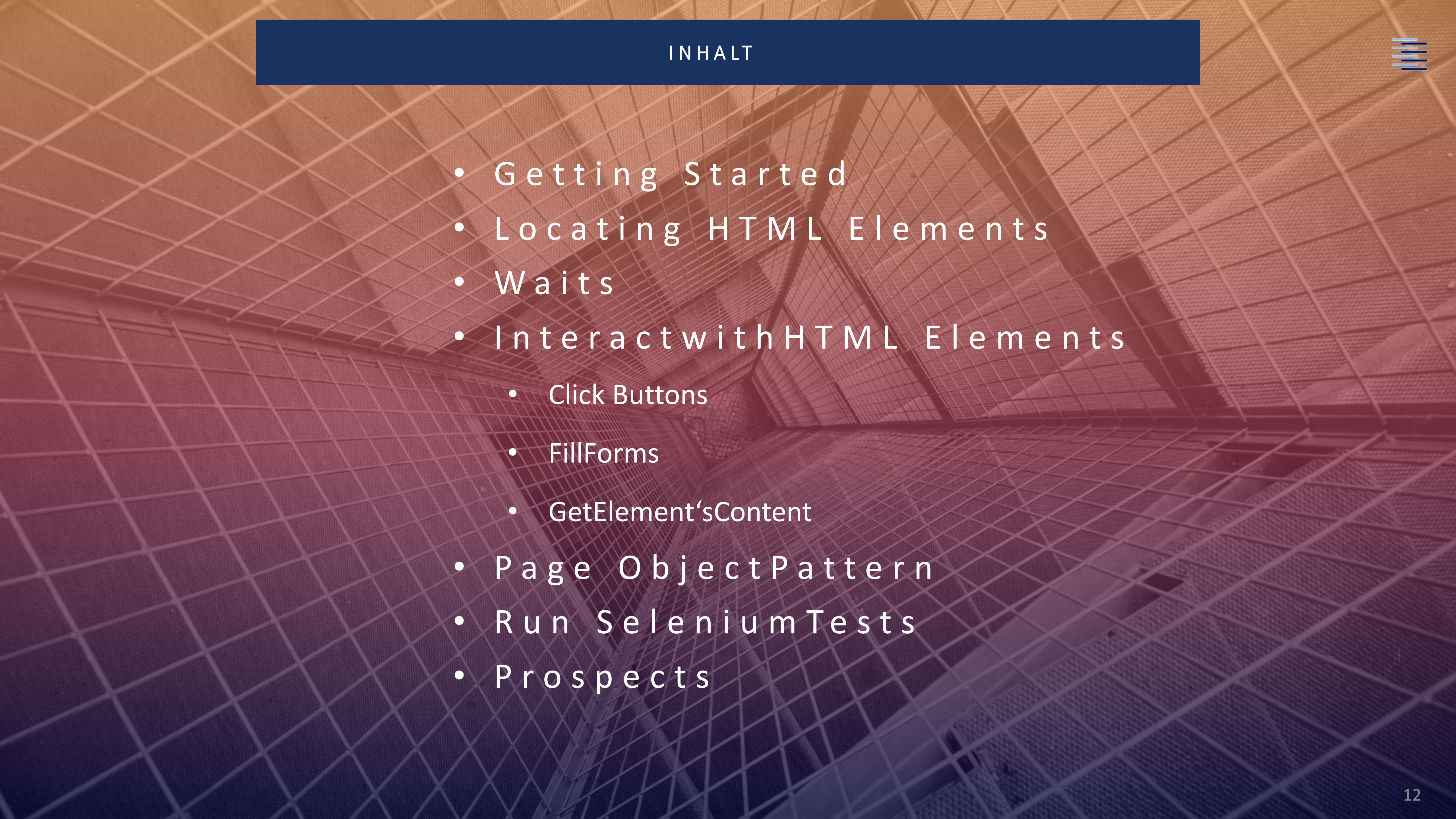


```
from selenium.webdriver.support import expected_conditions as EC

# Within 30 seconds the result column must be visible
wait = WebDriverWait(cls.driver, 30)
wait.until(EC.visibility_of(cls.driver.find_element_by_id("query_results")))
```

EXPLICIT WAIT–FURTHER CONDITIONS

Expected Condition	Explanation
presence_of_element_located(locator)	An expectation for checking that an element is present on the DOM of a page. This does not necessarily mean that the element is visible
visibility_of_element_located(locator)	An expectation for checking that an element is present on the DOM of a page and visible. Visibility means that the element is not only displayed but also has a height and width that is greater than 0.
visibility_of(element)	An expectation for checking that an element is present on the DOM of a page and visible. Visibility means that the element is not only displayed but also has a height and width that is greater than 0.
text_to_be_present_in_element(locator, text_)	An expectation for checking if the given text is present in the specified element.
invisibility_of_element_located(locator)	An Expectation for checking that an element is either invisible or not present on the DOM.
element_to_be_clickable(locator)	An Expectation for checking an element is visible and enabled such that you can click it.
element_to_be_selected(element)	An expectation for checking the selection is selected
…	…
alert_is_present	Expect an alert to be present

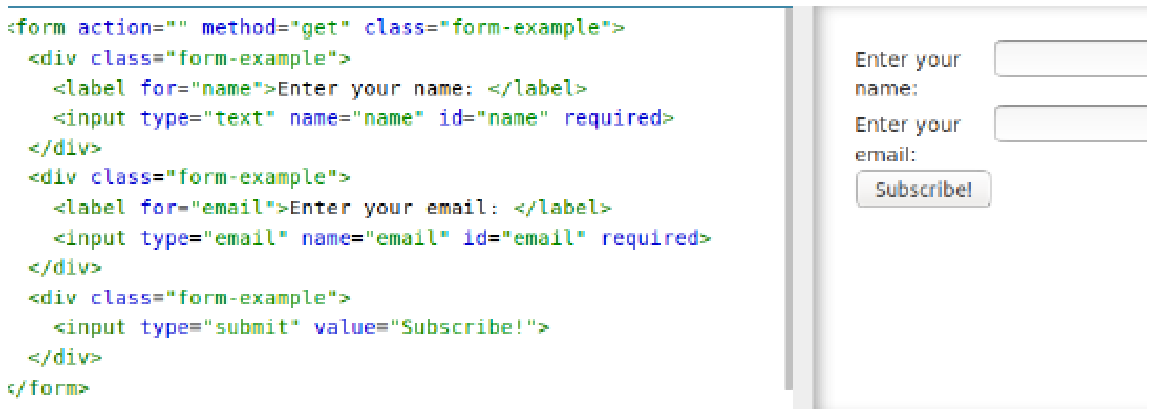
Detaillierte Ausführung unter Kapitel 7.39: https://selenium-python.readthedocs.io/api.html#module-selenium.webdriver.support.expected_conditions



INHALT

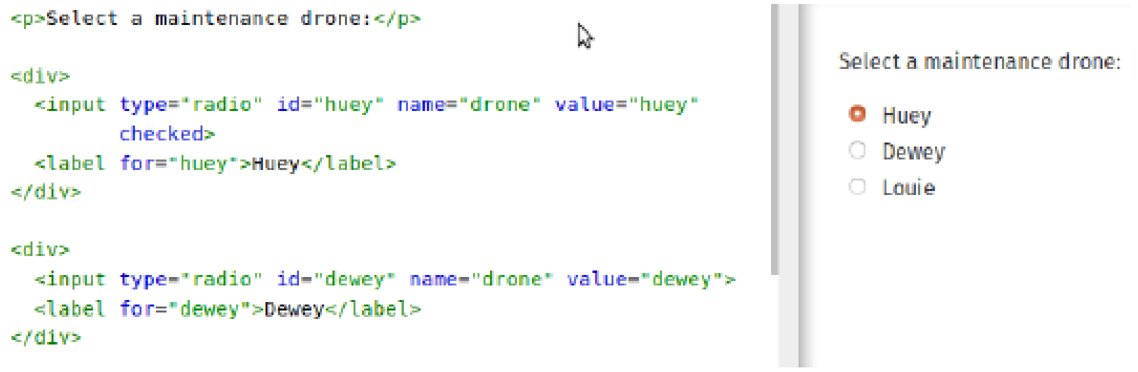
INTERACT WITH HTML-ELEMENTS - FILLFORMS

- **Fill form**
  - Element must be input tag
  - element.send_keys(<String>)
- **Submit form**
  - click() has to be done on the submit button
  - submit() can be done on any form element



```
<form action="" method="get" class="form-example">
  <div class="form-example">
    <label for="name">Enter your name: </label>
    <input type="text" name="name" id="name" required>
  </div>
  <div class="form-example">
    <label for="email">Enter your email: </label>
    <input type="email" name="email" id="email" required>
  </div>
  <div class="form-example">
    <input type="submit" value="Subscribe!">
  </div>
</form>
```

- **Click on label**
  - Element must be label tag
  - element.click()

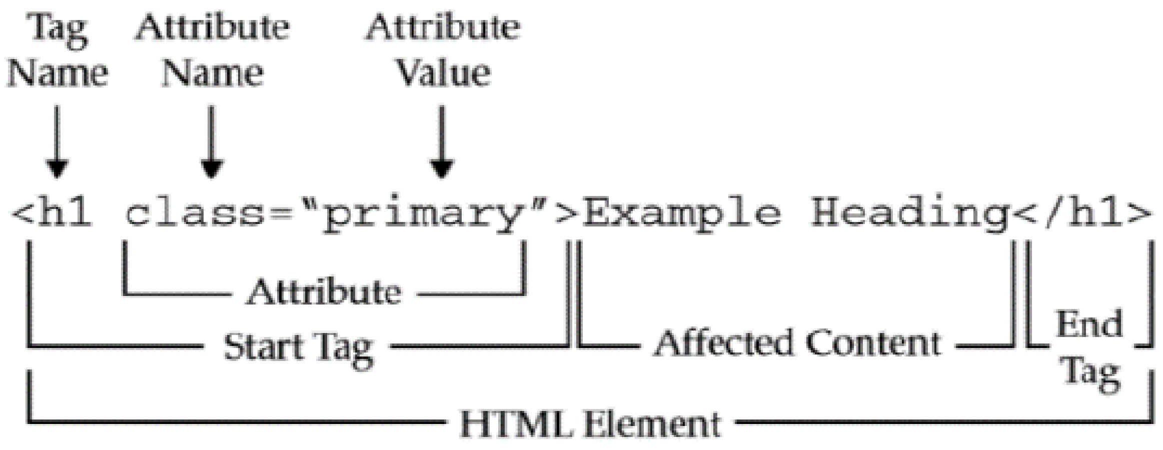


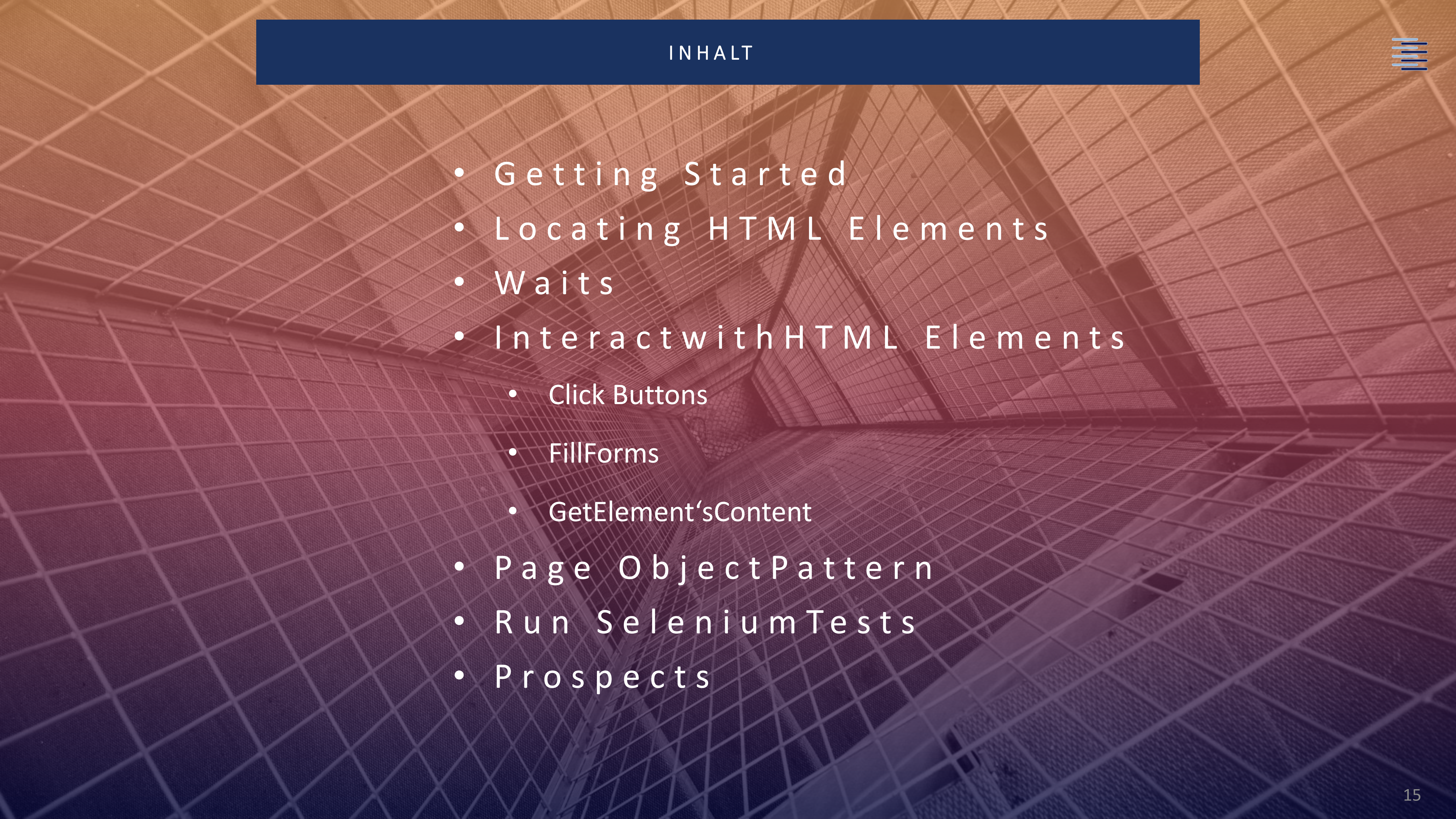
```
<p>Select a maintenance drone:</p>

<div>
  <input type="radio" id="huey" name="drone" value="huey"
         checked>
  <label for="huey">Huey</label>
</div>

<div>
  <input type="radio" id="dewey" name="drone" value="dewey">
  <label for="dewey">Dewey</label>
</div>
```

INTERACT WITH HTML-ELEMENTS -GET ELEMENT'SCONTENT

- Get attribute value
  - placeholder_inputfield = element.get_attribute(„placeholder")

- Get content
  - text = element.get_attribute("textContent")



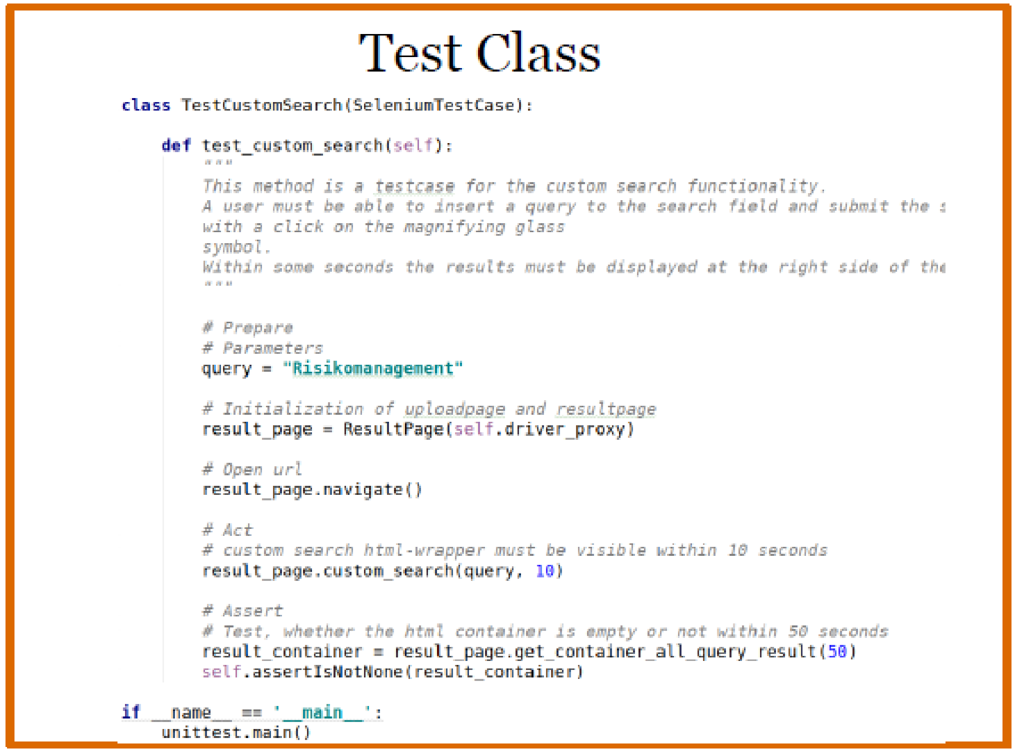


INHALT

PAGE OBJECT PATTERN

Benefits of Page Object Pattern:

- Creating reusable code that can be shared across multiple test cases
- Reducing the amount of duplicated code
- If the user interface changes, the fix needs changes in only one place and the test case does not need to change



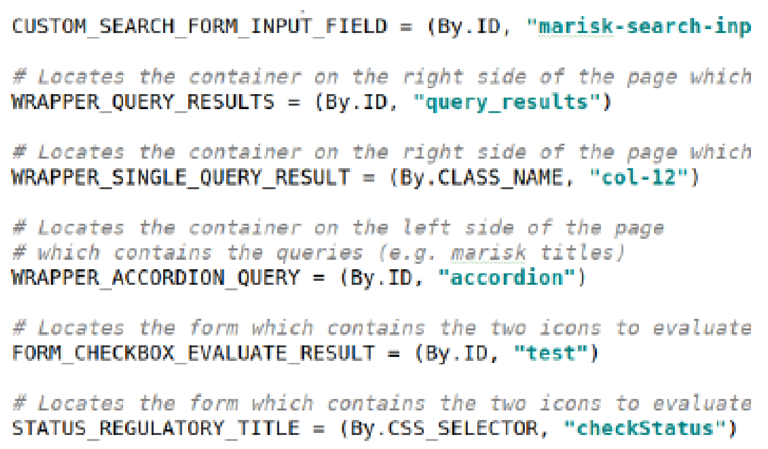
Test Class

```python
class TestCustomSearch(SeleniumTestCase):

    def test_custom_search(self):
        """
        This method is a testcase for the custom search functionality.
        A user must be able to insert a query to the search field and submit the s
        with a click on the magnifying glass
        symbol.
        Within some seconds the results must be displayed at the right side of the
        """

        # Prepare
        # Parameters
        query = "Risikomanagement"

        # Initialization of uploadpage and resultpage
        result_page = ResultPage(self.driver_proxy)

        # Open url
        result_page.navigate()

        # Act
        # custom search html-wrapper must be visible within 10 seconds
        result_page.custom_search(query, 10)

        # Assert
        # Test, whether the html container is empty or not within 50 seconds
        result_container = result_page.get_container_all_query_result(50)
        self.assertIsNotNone(result_container)

if __name__ == '__main__':
    unittest.main()
```

PAGE OBJECTPATTERN

Locators

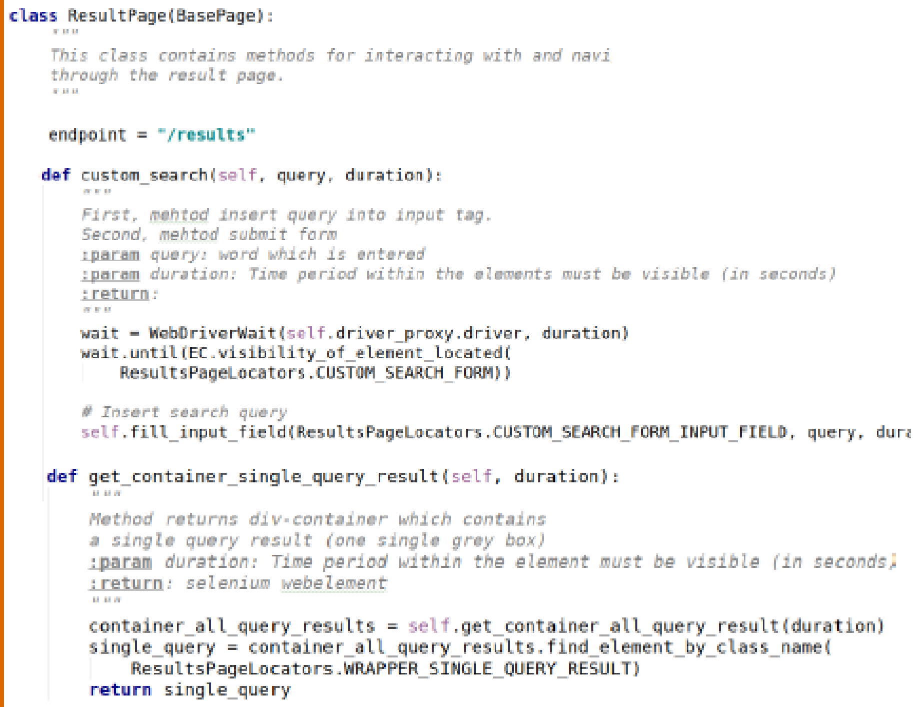
- One of the practices is to separate the locator strings from the place where they are being used.
- So it is easy to reuse a Locator



```
CUSTOM_SEARCH_FORM_INPUT_FIELD = (By.ID, "marisk-search-inp

# Locates the container on the right side of the page which
WRAPPER_QUERY_RESULTS = (By.ID, "query_results")

# Locates the container on the right side of the page which
WRAPPER_SINGLE_QUERY_RESULT = (By.CLASS_NAME, "col-12")

# Locates the container on the left side of the page
# which contains the queries (e.g. marisk titles)
WRAPPER_ACCORDION_QUERY = (By.ID, "accordion")

# Locates the form which contains the two icons to evaluate
FORM_CHECKBOX_EVALUATE_RESULT = (By.ID, "test")

# Locates the form which contains the two icons to evaluate
STATUS_REGULATORY_TITLE = (By.CSS_SELECTOR, "checkStatus")
```

Page Objects

- The page object pattern intends creating an object for each web page.
- A layer of separation between the test code and technical implementation is created.
- Each action on a website is a method



```
class ResultPage(BasePage):
    """
    This class contains methods for interacting with and navi
    through the result page.
    """

    endpoint = "/results"

   def custom_search(self, query, duration):
        """
        First, mehtod insert query into input tag.
        Second, mehtod submit form
        :param query: word which is entered
        :param duration: Time period within the elements must be visible (in seconds)
        :return:
        """
        wait = WebDriverWait(self.driver_proxy.driver, duration)
        wait.until(EC.visibility_of_element_located(
            ResultsPageLocators.CUSTOM_SEARCH_FORM))

        # Insert search query
        self.fill_input_field(ResultsPageLocators.CUSTOM_SEARCH_FORM_INPUT_FIELD, query, dura

    def get_container_single_query_result(self, duration):
        """
        Method returns div-container which contains
        a single query result (one single grey box)
        :param duration: Time period within the element must be visible (in seconds
        :return: selenium webelement
        """
        container_all_query_results = self.get_container_all_query_result(duration)
        single_query = container_all_query_results.find_element_by_class_name(
            ResultsPageLocators.WRAPPER_SINGLE_QUERY_RESULT)
        return single_query
```

INHALT

- Getting Started
- Locating HTML Elements
- Waits
- InteractwithHTML Elements
  - Click Buttons
  - FillForms
  - GetElement‘sContent
- Page ObjectPattern
- Run SeleniumTests
- Prospects