Learning Objectives

1. **Understand the fundamentals of Playwright** and its automation capabilities.
2. **Learn Playwright's JavaScript API** for web testing and automation.
3. **Create, run, and debug automated test cases** using best practices.
4. **Set up effective CI/CD pipelines** for test automation.

Learning Timeline

(Estimated completion: 3-4 weeks with 1-2 hours per day)

Week 1: Basics of Playwright and JavaScript Refresher

Day 1-2: Set Up the Environment

- **Install Node.js** (if not already installed)
- **Install Visual Studio Code (VS Code)** or another IDE if needed
- **Install Playwright**:

  ```
  npm init playwright@latest
  ```

- **Set up Git** (optional but recommended for version control)

Day 3-4: JavaScript Basics Refresher (Skip if proficient)

- Review key JavaScript concepts:
  - Variables, functions, and async/await patterns
  - DOM manipulation and selectors
  - Promises and error handling
- **Resources**:
  - MDN Web Docs for JavaScript syntax
  - JavaScript.info for deeper dives

Day 5-7: Start "Getting Started with Web Automation with Playwright and JavaScript" on Udemy

- Begin Laurentiu Raducu's course on Udemy.
  - **Focus:** Modules 1-3: Understanding Playwright Basics, Setting Up, and Writing Your First Test.
  - **Objectives**:
    - Understand what Playwright is and its features.
    - Write and execute simple scripts.

    - **Hands-On**: Run provided examples in your setup and replicate examples independently.

Week 2: Core Playwright Concepts and Test Writing

Day 8-10: Continue with the Udemy Course (Modules 4-5)

- **Topics**: Working with selectors, handling pop-ups, and taking screenshots.
    - Practice handling dynamic elements and interacting with the UI.
    - Test on multiple browsers (Chromium, Firefox, WebKit).
    - Take screenshots and handle UI components such as forms, modals, and navigation.

Day 11-12: Deep Dive into Playwright API Documentation

- Explore Playwright's official documentation.
    - Learn about the key classes and methods: `page`, `context`, `browser`, etc.
    - Understand how to handle advanced interactions, such as waiting strategies and event handling.

Day 13-14: Practical Hands-On Practice

- Write test cases to:
    - Open a website, navigate between pages, and validate elements.
    - Fill in forms and verify form validation.
    - Take screenshots of specific UI components.
- **Suggested Practice Website**: Use the-internet.herokuapp.com to test common UI interactions.

Week 3: Advanced Playwright Features and Test Optimization

Day 15-17: Complete Udemy Course (Modules 6-8)

- **Topics**:
    - Advanced Playwright configurations and debugging techniques.
    - Emulate mobile devices and geolocation.
    - Run tests in headless mode for CI/CD integration.
- **Hands-On**:
    - Set up configurations for multiple browsers and devices.
    - Write scripts with headless mode enabled and understand the debugging tools available.

Day 18-19: Write Advanced Test Cases

- Write scripts with more complex interactions:
  - Test file upload/download functionality.
  - Test iframes and embedded content.
- Refine and optimize your existing scripts.

Day 20-21: Test Frameworks and Reporting

- Set up **Jest or Mocha** for running Playwright tests.
- Integrate test reporting (e.g., Allure or Playwright's built-in reporting).
- Learn how to interpret test reports and analyze test coverage.

Week 4: CI/CD Integration, Best Practices, and Advanced Learning

Day 22-24: Integrate Playwright with CI/CD

- Choose a CI/CD tool (e.g., GitHub Actions, CircleCI, or Jenkins).
- Set up Playwright to run tests on commit or pull requests.
  - **Resources**:
    - Playwright GitHub Actions documentation
    - Jenkins CI/CD for Node.js and Playwright

Day 25-27: Debugging and Troubleshooting

- Learn advanced debugging tips:
  - Use Playwright's tracing and video recording capabilities.
  - Analyze failure logs and screenshots for troubleshooting.

Day 28: Review, Recap, and Explore Further Learning

- Revisit any sections of Laurentiu Raducu's course as needed.
- Review test scripts and refactor for readability and efficiency.
- Explore advanced topics like cross-browser testing strategies and multi-tab interactions.

Additional Resources

- **Playwright GitHub Repository**: Gain insights from community discussions and contributions.
- **YouTube Channels**:
  - Check for additional tutorials from YouTube channels like "Automation Bro" and "Testing Academy" for Playwright tips and tricks.