Understanding and Adapting the Stock Checker Script

This document explains the core components of the `check_bestbuy_stock_playwright.py` script and provides guidance on how to modify it to monitor different retail items or adapt it to other websites.

How the Script Works

The script leverages Playwright, a powerful library for browser automation. Here's a breakdown of its key functionalities:

1. **Browser Launch:** It launches a Chromium browser instance (either visible or headless).
2. **Navigation:** It navigates to the specified product URL.
3. **User Agent:** It uses a random user agent to mimic a real user's browser, which can help bypass some basic bot detection.
4. **Page Loading:** It waits for the page's DOM content to be loaded, ensuring that the basic HTML structure is available.
5. **Element Detection (Selectors):** This is the most crucial part. The script attempts to find specific HTML elements on the page that indicate the stock status (e.g., an "Add to Cart" button or a "Sold Out" message). It uses CSS selectors to locate these elements.
6. **Stock Determination:** Based on whether it finds the "Add to Cart" or "Sold Out" element, it prints the corresponding stock status.
7. **Looping and Delay:** The script runs in an infinite loop, performing the stock check at regular intervals (e.g., every 5 minutes), with a `time.sleep()` to prevent overwhelming the website.

Adapting the Script for Different Retail Items or Websites

The primary challenge when adapting this script is identifying the correct **selectors** for the stock status indicators on different websites.

1. Identify the Product URL

First, you'll need the direct URL of the product page you want to monitor.

2. Inspect Elements to Find Selectors

This is the most critical step and requires using your web browser's developer tools:

- **Open the Product Page:** Navigate to the product page in your web browser.

- **Right-Click and Inspect:** Right-click on the "Add to Cart" button (if in stock) or the "Sold Out" message/button (if out of stock) and select "Inspect" or "Inspect Element" from the context menu.
- **Examine the HTML:** The developer tools will open, showing you the HTML code for the element you clicked on. Look for unique attributes that identify this element. Good candidates are:
  - `id` attributes (e.g., `id="add-to-cart-button"`)
  - `class` attributes (e.g., `class="btn btn-primary add-to-cart"`)
  - `data-*` attributes (e.g., `data-test-id="add-to-cart"`, `data-sku-id="12345"`)
  - The text content of the element (e.g., a `<span>` containing "In Stock").

  Example (from Best Buy): For the "Sold Out" button, we found: `<button data-test-id="sold-out" ...><span>Sold Out</span></button>` A good selector for this is `button[data-test-id="sold-out"]`.
- **Test Your Selector (Optional but Recommended):** In the browser's developer tools, go to the "Console" tab. You can test your CSS selector by typing `document.querySelector('YOUR_SELECTOR_HERE')`. If it returns the element, your selector is likely correct.

3. Modify the Script

Open `check_bestbuy_stock_playwright.py` in a text editor and make the following changes:

- **Update `URL`:** Change the `URL` variable at the top of the script to your new product URL.

  ```
  URL = "https://www.example.com/new-product-page"
  ```
- **Update Selectors in `check_stock()` function:** Locate the `page.wait_for_selector()` calls. You will need to replace the existing selectors with the ones you identified in Step 2.

  ```
  # Example for a new website
  try:
      # Wait for the 'Add to Cart' button to be visible
      page.wait_for_selector('YOUR_ADD_TO_CART_SELECTOR_HERE', timeout=10000)
      add_to_cart_present = True
  except:
      pass

  try:
      # Wait for the 'Sold Out' indicator to be visible
      page.wait_for_selector('YOUR_SOLD_OUT_SELECTOR_HERE', timeout=10000)
      sold_out_present = True
  except:
      pass
  ```

Remember to use single quotes around your selector string if it contains double quotes, or vice-versa.

- **Adjust `time.sleep()` (Optional):** You can change the `time.sleep(300)` value at the bottom of the script to adjust how frequently the script checks for stock (e.g., `60` for every minute, `3600` for every hour).

4. Test Your Modified Script

Run the script with `python check_bestbuy_stock_playwright.py` and observe its behavior. Start with `headless=False` to see the browser and ensure it's navigating correctly and finding the elements.

General Tips for Adaptation

- **Dynamic Content:** Some websites load content dynamically after the initial page load. If your selectors aren't working, try waiting for a more general element to appear first, or use `page.wait_for_load_state('networkidle')` (use with caution, as it can be slow).
- **Anti-Bot Measures:** Many e-commerce sites employ sophisticated anti-bot measures. If your script is consistently blocked, you might need to explore more advanced techniques like using proxies, CAPTCHA solving services, or more complex browser automation patterns.
- **Error Handling:** Always include `try-except` blocks to gracefully handle potential errors like network issues or elements not being found.
- **Be Respectful:** Avoid making requests too frequently, as this can put a strain on the website's servers and might lead to your IP being blocked. Adhere to the website's `robots.txt` file if you intend to do extensive scraping.