* Sự khác nhau của Sync và Async:

|  |  |
| --- | --- |
| **Sync** | **Async** |
| Chạy tuần tự từng bước một(đồng bộ), bước sau phải chờ bước trước | Chạy đồng loạt, bước trên không cần chờ bước dưới (bất đồng bộ), không cần phải chờ |
| Chạy đồng bộ nên sẽ sinh ra trạng thái chờ và không cần thiết ở một số trường hợp => có thể gây chậm hệ thống , tràn bộ nhớ do phải lưu trữ các trạng thái chờ không mong muốn | Chạy bất đồng bộ nên có thể xử lý nhiều việc cùng một lúc không cần phải chờ => tăng số lượng việc xử lý trong cùng một lúc , nhưng dễ dàng xảy ra lỗi tiến trình nếu không kiểm soát tốt |

* So sánh một số đặc trưng của typescript vs java

|  |  |
| --- | --- |
| **Typescript** | **Java** |
| Là javascript thêm type, typescript là một trình kiểm tra static type | Là một ngôn ngữ lập trình thuần hướng đối tượng |
| Không có modifier cho class | Có modifier cho class |
| Bị vấn đề cicular import vì không có modifier cho class | Không bị vấn đề cicular import vì class có modifier |
| Modifier của instance variable và method :   * Public là default * Protected phạm vi là trong class và kế thừa của nó | Modifier của instance variable và method :   * Default là default * Protected phạm vi là trong package và kế thừa của nó |
| Khai báo type:   * Khai báo type của variables, methods đằng sau dấu `:` theo sau variables, methods * Không bắt buộc phải khai báo type * Dùng let, const hay var để khai báo biến * Vd : let xyz : string; | Khai báo type:   * Khai báo type đằng trước variables, methods * Bắt buộc phải khai báo type * Vd: int ggg; |
| Chỉ được một constructor cho một class | được nhiều constructor cho một class do cơ chế overloading |
| Complile ra javascript version es5 | Compile ra bytecode |

* Async, Await, và Promise
* Promise (ES6) là một cơ chế xử lý bất đồng bộ của Javascript được đưa vào để thay thế callback (được xây dựng trên callback) , là một lời hứa trả về kết quả (resolve là hàm được gọi khi promise hoàn thành, reject là hàm được gọi khi xảy ra lỗi)
* Async , Await là một cơ chế xử lý bất đồng bộ mới hơn Promise (ES7) được xây lên từ nền là Promise .
* Async , Await hay đi chung với nhau, nếu như có Await thì thì chương trình chờ đến khi Promise được xử lý và trả về giá trị, nếu không có Await có nghĩa là việc thực khi sẽ không bị tạm dừng để chờ Promise được xử lý và trả về giá trị.
  + Async with promise Await >> Sync.
  + Async with promise without Await >> Async
* Callback là một hàm sẽ được thực hiện sau khi một hàm khác được thực hiện, có nghĩa là gọi lại hàm đã thực thi.
* Async, Await, Promise, CallBack dùng để kiểm soát luồng chạy
* So sánh Protractor và Selenium:

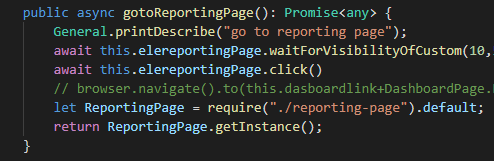
|  |  |
| --- | --- |
| **Selenium** | **Protractor** |
| Binding Language:   * + Java, Python,Ruby,Javacript, C#, PHP, Perl ... (đa số các ngôn ngữ thông dụng trên thế giới) | Binding Language:   * + Javascript , Typescript |
| Application Under Test (AUT):   * + Non-Angular : tốt vì hỗ trợ nhiều ngôn ngữ, dễ học , dễ dùng   + Angular : không tốt như protractor vì không hỗ trợ đồng bộ hóa giữa test và app phải dùng các loại wait khác nhau để có thể chạy đồng bộ hóa trên angular | Application Under Test (AUT):   * + Non-Angular : không tốt như selenium vì sử dụng phức tạp hơn, chỉ dùng được Javascript hay Typescript không có hỗ trợ thêm như Angular app   + Angular : tốt do có hỗ trợ automation wait |
| Unit Testing Framework support:   * + JUnit, TestNG, NUnit, XUnit, MSTest, PyTest … | Unit Testing Framework support:   * + Jasmine, Mocha, CucumberJs |

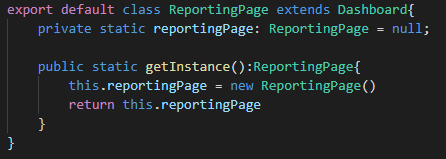
* Một số Update của Selenium 4
  + Selenium 4 giờ đây đã theo W3C Standard Webdriver Protocol nên webdriver có thể tương tác trực tiếp với trình duyệt mà không cần phải encoding, decoding API .
  + Selenium grid(selenium 4) rất linh hoạt cho phép test trên các trình duyệt khác nhau, trình duyệt với các version khác nhau và các hệ điều hành khác nhau, không cần phải setup một cách riêng lẽ hub và node khi người dùng khởi động máy chủ, grid sẽ tự khởi động trên cả hub và node
  + Một số tính năng của selinium 4:
* Capture screenshot trên web element cụ thể
* Mở một tab mới trên trình duyệt cùng với tab cũ
* Mở một cửa sổ mới trên trình duyệt
* Có thể lấy được coordinates, dimension, width, height của element
* Thêm năm relative locators: below(),toLeftOf(),toRightOf(),above(), near() , tất cả đều hổ trợ việc sử dụng withTagName() method
* Một số thông tin về Typescript:
* Typescript là static type checker , công dụng là thêm type và check type cho javascript, khi nói code Typescript là nói code Javascript với bộ extension là typescript. Cho nên dù là Typescript có các tính năng mà javascript không có nhưng sau khi compile thì sẽ không còn các tính năng này.
* Có 2 phong cách gán biến trong Javascript – Typescript
  + Một là đa số dùng let hết cần mới dùng const
  + Hai là đa số dùng const hết cần mới dùng let

“There are two schools of thought about the use of the const key‐ word. One approach is to use const only for values that are funda‐ mentally unchanging, like the physical constants shown, or program version numbers, or byte sequences used to identify file types, for example. Another approach recognizes that many of the so-called variables in our program don’t actually ever change as our program runs. In this approach, we declare everything with const, and then if we find that we do actually want to allow the value to vary, we switch the declaration to let. This may help prevent bugs by ruling out accidental changes to variables that we did not intend.

In one approach, we use const only for values that must not change. In the other, we use const for any value that does not hap‐ pen to change. I prefer the former approach in my own code.”

Trích : JavaScript: The Definitive Guide 7e, p54.

* Cách giải quyết cicular import của typescript/javascript phải dùng cách trộn giữa typescript và javascript , sử dụng require() và cast về đúng type mong muốn khi viết testcase (<https://www.logigear.com/blog/test-automation/15-best-practices-for-building-an-awesome-protractor-framework/>) . Một cách giải quyết tốt hơn là tạo ra một method tại page mà ta muốn trả về giá trị là instance của page, vd:
  + Ta có dashboard là parent class mà ta muốn một method nào đó của class trả về instance của dashboard’s child-class 
  + ReportingPage là dashboard’s child-class



* Khi ta throw một Error ta muốn chỗ nào mà chương trình chạy tới chỗ đó thì chương trình sẽ dừng lại và báo lỗi >> dùng để kiểu soát , throw Error chỗ ta muốn.