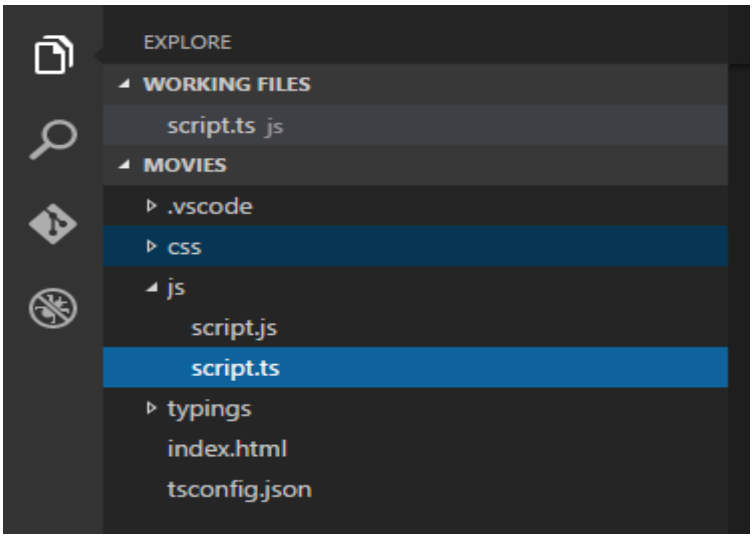
Hướng dẫn dùng Visual Studio Code (VSC)

Download: https://code.visualstudio.com/

1. Thanh sidebar

Nhìn về phía bên trái, ta thấy 4 biểu tượng cửa sổ, chức năng:

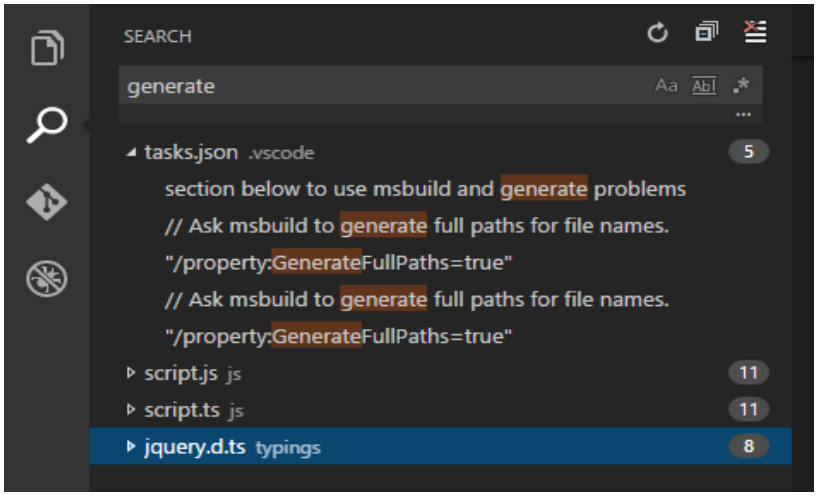


Visual Studio Code Explore

1.1. Explore

Đầu tiên là biểu tượng  .

- ✓ Với cửa sổ VSC Explore. Khung cửa sổ này chia làm 2 phần. Phần trên cùng là: Working Files, chứa những file ta đang làm việc. Còn phần bên dưới là cây thư mục, chứa toàn bộ các *file và folder có trong dự án*.
- ✓ Các icon nhỏ xuất hiện bên phải. Nếu muốn biết các biểu tượng này làm gì, drag chuột lên nó thì sẽ có một tooltip thông báo. Ta cũng có thể truy cập các chức năng này bằng cách R_click .



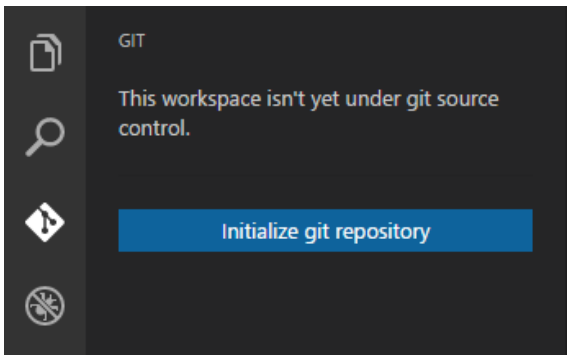
Search

1.2. Công cụ tìm kiếm

Kế đến là biểu tượng  công cụ tìm kiếm. Click vào nó, một khung cửa sổ tìm kiếm hiện ra.

1.3. Code Git

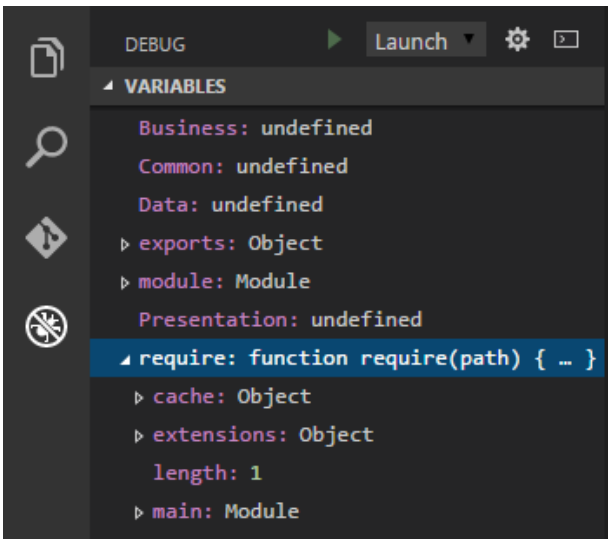
Tiếp biểu tượng  .



Git

Version Control System (VCS) là công cụ không thể thiếu trong lập trình nhóm và Git.

- ✓ Nếu dự án chưa *được Git quản lý*, VSC hiển thị một cái nút to khởi tạo.
- ✓ Ta có thể dùng trực tiếp khung cửa sổ này để commit mà không cần nhảy qua cửa sổ dòng lệnh.

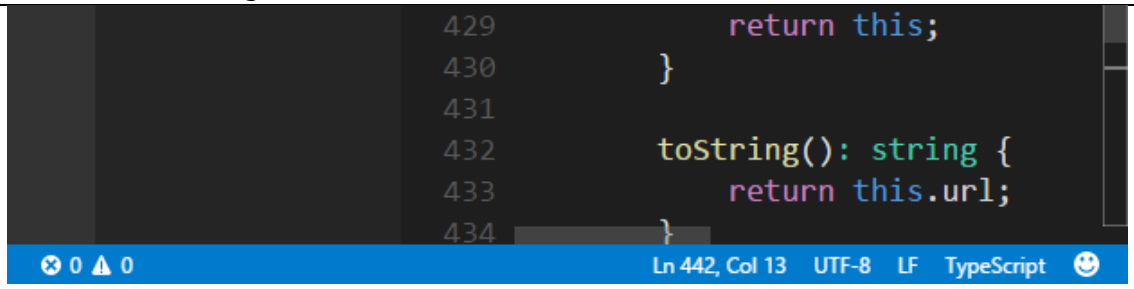


Debug

- ✓ Biểu tượng cuối cùng là con bọ nằm trong vòng tròn gạch chéo. Đây là chức năng debugger.
- ✓ Thao tác cơ bản khi dùng debugger là như nhau. Đầu tiên ta đặt breakpoint, sau đó chạy chương trình cho tới khi đụng breakpoint.
- ✓ Kế đến, kiểm tra giá trị của biến, hoặc chạy từng dòng code để kiểm tra các thay đổi.

2. Thanh trạng thái

- ✓ Nhìn xuống góc dưới bên trái, ta thấy hai biểu tượng với con số phía trước. Đây là mục cho ta biết số error và warning hiện có trong code.
- ✓ Thông thường, lờ đi các *warning*, nhưng để code sạch, ta không nên chừa lại bất kỳ warning nào.
- ✓ Nếu thấy con số trước error hay warning > 0, ta phải tìm cách sửa nó. Khi click vào mục này, VSC hiển thị thông tin về số dòng gây ra lỗi, từ đó, ta có thể nhảy ngay đến dòng có lỗi để sửa.



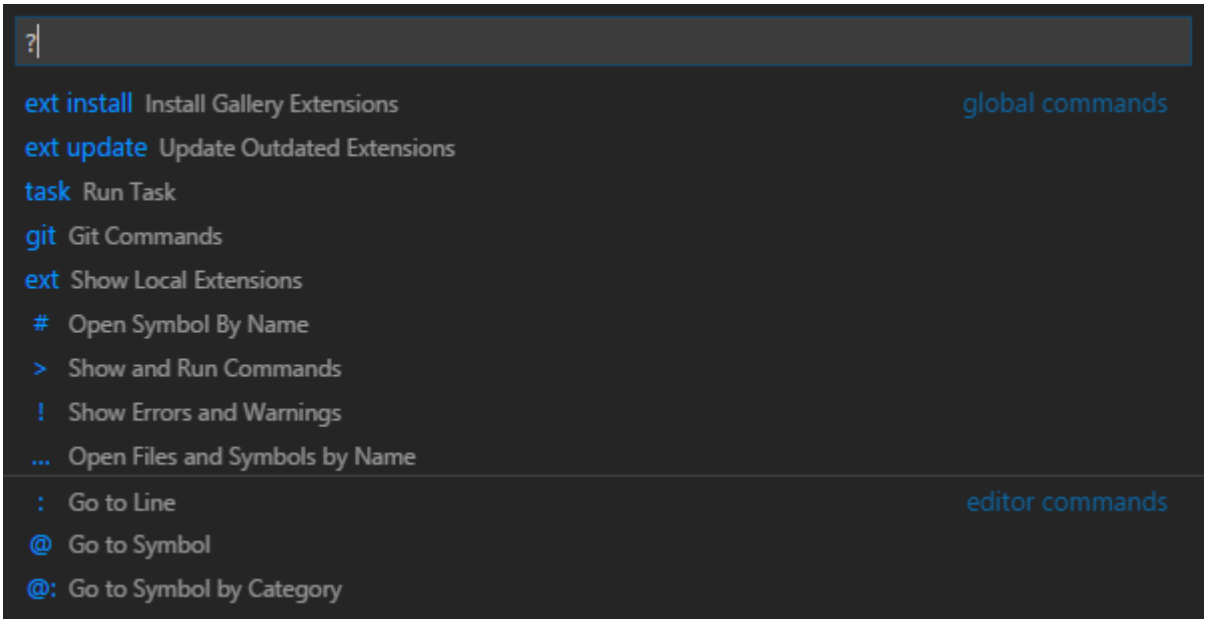
Status Bar

Tiếp theo, ta chuyển sang góc dưới bên phải. 

- ✓ Mục đầu tiên là **số dòng và số cột** trong soạn thảo.
- ✓ Mục thứ hai là định dạng **encoding**, mặc định là UTF-8. Nếu muốn đổi, ta click vào và chọn encoding phù hợp.
- ✓ Thứ ba là mục ***ký tự kết thúc dòng***.
- ✓ Trong Windows, ký tự bao gồm *Carriage Return và Line Feed* (CRLF). Nếu để mặc định là LF.
- ✓ Ngôn **ngữ lập trình**: VSC sẽ tự động phát hiện ngôn ngữ dựa vào đuôi file. Nếu nó chọn không đúng, ta dùng mục này để chỉnh lại.
- ✓ Cuối cùng là biểu tượng mặt cười.

3. Khung nhập lệnh

Khung nhập lệnh (*Command Palette*) là tính năng nổi tiếng của Sublime Text. Các trình soạn thảo code ra đời sau đều tích hợp tính năng này.



Command Palette

3.1. Giả sử cần nhảy đến file tên **script.js** trong dự án, nhấn tổ hợp phím Ctrl + P, khung nhập lệnh hiện ra giữa màn hình.

- ✓ Hãy nhập vào vài ký tự đầu tiên của tên file. Trong quá trình nhập, VSC sẽ hiển thị kết quả gần đúng nhất với các ký tự đang nhập.
- ✓ Nếu thấy file cần mở nằm trong danh sách này, *dùng mũi tên* lên xuống để di chuyển vùng chọn đến nó rồi nhấn Enter.

- ✓ Bằng cách này, không phải rời tay khỏi bàn phím để dùng chuột click vào file trong cửa sổ Explore. Nhờ vậy, ta có thể di chuyển rất nhanh giữa các file.

3.2. *Để chạy một lệnh VSC*

- ✓ Dùng tổ hợp phím **Ctrl + Shift + P**. Lúc này, khung nhập lệnh tương tự hiện ra.
- ✓ Tuy nhiên, ta thấy nó có một dấu > phía trước. Nếu dùng **Ctrl + P** để mở khung nhập lệnh, sau đó gõ thêm dấu > thì nhận được kết quả tương tự.

3.3. *Giả sử muốn thay đổi theme cho VSC*

- ✓ Thay vì phải mò trong đống menu để tìm ra mục theme, dùng **Ctrl + Shift + P**, sau đó gõ từ khóa **theme.** VSC liệt kê các chức năng liên quan tới theme, dùng mũi tên lên xuống để chọn theme phù hợp và nhấn Enter.
- ✓ *Khi muốn di chuyển nhanh đến dòng cụ thể*, dùng Ctrl + G. Lúc này, khung nhập lệnh hiện ra và điền sẵn dấu**:**
  - Nhập số dòng và nhấn Enter để nhảy ngay đến dòng đó.
- ✓ *Ngoài* ra, khi muốn nhảy đến một class hay method, ta dùng Ctrl + Shift + O,
  - Sau đó nhập tên class hay method muốn đến.
  - Các tên này được gọi là *symbol* và được biểu thị bằng ký tự **@** phía trước.

3.4. Bên *trong* khung nhập lệnh, VSC gợi ý ta nhấn phím **?** để xem thông tin trợ giúp. VSC sẽ liệt kê một loạt các ký tự đặc biệt và các lệnh có thể dùng trong khung nhập lệnh. Trường hợp quên phím tắt, ta vào menu Goto và sẽ thấy ngay các phím tắt cần thiết.

4. Trình soạn thảo code

Khung soạn thảo của VSC đơn giản, nổi bật nhất là tính năng tách *khung soạn thảo ra làm 2 hay 3 phần*. Nếu ta viết code với màn hình kích thước lớn. VSC cho phép tách ra nhiều khung soạn thảo để tận dụng diện tích màn hình.

- ✓ Để mở thêm khung thứ hai, ta bấm **Ctrl + 2**.

Khung mới hiện ra bên phải khung hiện tại. Nếu vẫn còn chỗ trống, ta mở thêm khung thứ ba bằng tổ hợp phím **Ctrl + \.** Để đóng khung, ta dùng **Ctrl + W**.

- ✓ *VSC cho mở tối đa 3 khung soạn thảo.*

Để di chuyển giữa các khung, ta dùng Ctrl và phím số tương ứng với thứ tự khung từ trái sang phải.

5. Emmet

Khi dùng **Sublime Text**, phụ thuộc rất nhiều vào Emmet để viết code với ít phím gõ nhất. Mỗi khi cài mới **Sublime Text**, VSC tích hợp Emmet từ đầu nên không cần cài đặt thêm gì cả.

- ✓ Khi tạo một file HTML mới, ta thường phải gõ đi gõ lại cái khung của nó. Nào là thẻ <html>, thẻ <head> rồi thẻ <body>, sau đó thêm <title>.
  - Với Emmet,chỉ cần gõ dấu **!** rồi nhấn Tab. Ngay lập tức, cái sườn HTML hiện ra. Để tạo thẻ <link> tham chiếu đến file CSS, gõ **link** rồi Tab.
  - Emmet sẽ viết sẵn các thuộc tính cần thiết của thẻ **<link>** và đặt con trỏ ngay tại **href** để chờ,nhập đường dẫn, nhanh chóng và tiện lợi.

6. Đa con trỏ

Với tính năng *đa con trỏ*, ta có thể nhập liệu ở nhiều vị trí cùng lúc. Khi click 1 từ, VSC sẽ đánh dấu những từ tương tự có trong file.

- ✓ Nếu nhấn Ctrl + F2, trình soạn thảo sẽ bôi đen những từ này. Thao tác này rất hữu ích khi ta cần thay đổi tên của một biến hay method trong toàn bộ file mà không cần mở hộp thoại *Find and Replace*.
- ✓ Ta cũng có thể dùng Ctrl + D để chọn từ ở vị trí con trỏ soạn thảo. Nếu tiếp tục nhấn Ctrl + D, các từ tương tự bên dưới cũng sẽ lần lượt được bôi đen.
- ✓ Ngoài ra, ta có thể chọn vị trí đặt các con trỏ soạn thảo bằng cách giữ phím Alt và click vào vị trí mong muốn.
- ✓ Ta có thể dùng Ctrl + Alt + Mũi tên lên xuống để tạo thêm con trỏ soạn thảo ở dòng trên hoặc dưới dòng hiện tại. Tính năng này rất hữu ích khi có những từ khóa nằm cùng vị trí cột trên nhiều dòng liên tiếp.

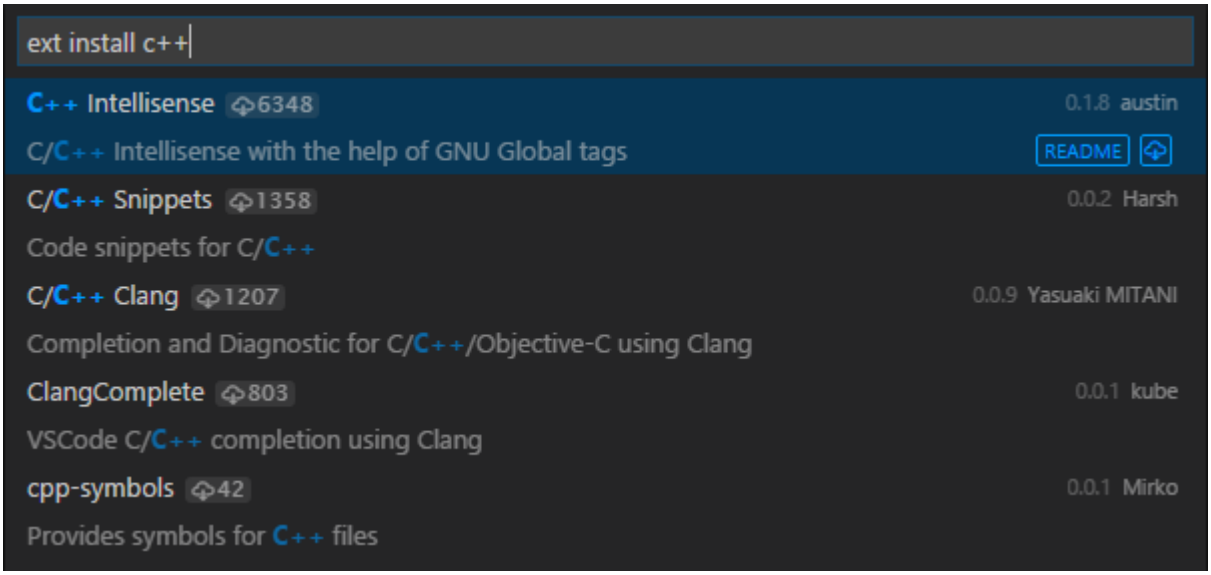
7. Intellisense

VSC không có **Intellisense**.

- Khi viết code, **Intellisense sẽ** liên tục xuất hiện để gợi ý.

Trong trường hợp ta cần được gợi ý mà lại không thấy Intellisense, ta gọi nó bằng **Ctrl + Space**.

- Intellisense không những cung cấp tên biến hay method, mà còn thêm một dòng mô tả chức năng ngay bên dưới.



Intellisense

*Các ngôn ngữ phổ biến đều được **Intellisense** hỗ trợ.*

- ✓ Tải về extension bổ sung. Đầu tiên, ta mở khung nhập lệnh (Ctrl + Shift + P), rồi nhập vào **ext install**.
- ✓ Chờ để VSC lấy danh sách extension từ server. Sau đó, ta chọn extension muốn cài trong danh sách.

Giả sử cài extension *C++ Intellisense* để VSC nhắc mã khi viết code C++. Mở khung nhập lệnh, gõ vào **ext install c++.** Nằm ngay đầu danh sách là C++ **Intellisense,** chọn nó và nhấn Enter, Intellisense sẽ cung cấp gợi ý cho tôi khi viết code C++.

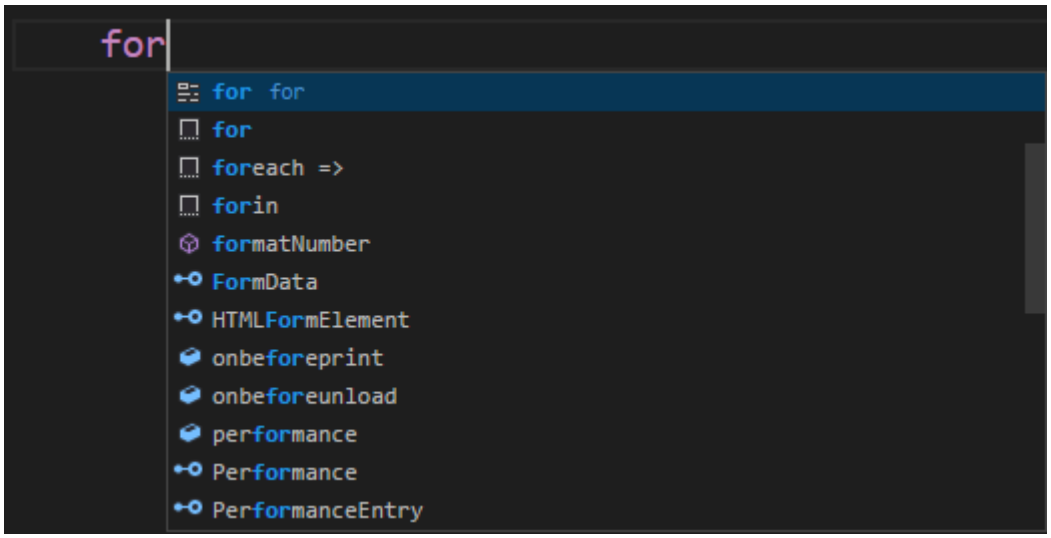
8. Code Snippet

Code snippet là đoạn code nhỏ thường được dùng ở nhiều chỗ khác nhau. Các s*nippet* đều có một cái tên, và khi gõ cái tên rồi nhấn Tab, toàn bộ đoạn code sẽ được chèn vào vị trí.

Mặc định: VSC cung cấp **snippe**t cho các tính năng thường dùng trong ngôn ngữ. Ngoài ra, ta cũng có thể tạo snippet riêng để dùng sau này.

- ✓ Giả sử đang viết JavaScript và cần dùng vòng lặp **for.**
- ✓ Gõ **for** và **Intellisense** hiện ra nhắc là có một snippet tên for, chọn nó rồi nhấn Enter.

- ✓ VSC chèn toàn bộ khung của vòng lặp for và bôi đen những vị trí các biến để tôi thay thế cho dễ.



Code Snippet

Có những đoạn code tuy không khó nhưng do ít dùng nên ta thường hay quên. Do vậy, cách hiệu quả nhất là lưu chúng thành các **snippet.**

Để làm việc này,

- ✓ Chọn: ***File* > *Preferences* > *User Snippets*,** sau đó khung nhập lệnh hiện ra yêu cầu chọn ngôn ngữ.
- ✓ VSC lưu snippet cho mỗi ngôn ngữ trong một file riêng biệt dưới dạng JSON. Sau khi chọn ngôn ngữ, file JSON tương ứng sẽ được mở ra.
- ✓ Trong này, VSC hướng dẫn cách thêm snippet bằng một ví dụ trong phần ghi chú. Ta chỉ cần bắt chước cấu trúc trong ví dụ mẫu để thêm snippet mới.

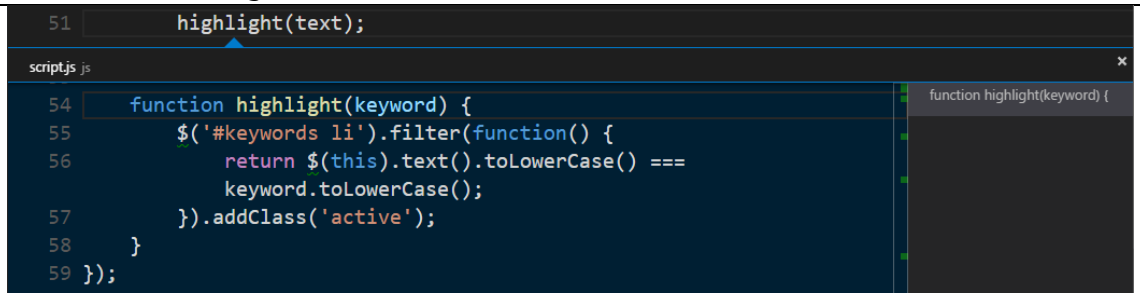
9. Di chuyển trong code

Khi viết những ứng dụng lớn, di chuyển qua lại trong code trở nên phức tạp. Trong Visual Studio, ta được trang bị hai tính năng **Go to Definition và Peek Definition** để nhảy ngay đến phần định nghĩa của một class hay method. Vì sự hữu ích không thể chối cãi, hai tính năng tuyệt vời này được mang sang VSC.

Để dùng *Definition* tính năng này, ta click phải chuột vào tên class hoặc method, sau đó chọn *Go to Definition*, hoặc dùng phím F12.

- ✓ Cửa sổ code chuyển sang phần định nghĩa. Để quay lại vị trí cũ, ta nhấn **Alt + Mũi tên trái**.
- ✓ Tính năng ***Peek Definition*** được tích hợp vào VSC.
- ♦ Đây là tính năng tương tự Go to *Definition* nhưng cửa sổ code không nhảy sang vị trí khác. Thay vào đó, một cửa sổ con sẽ được mở ra ngay bên dưới. Ta có thể truy cập tính năng này bằng menu ngữ cảnh hoặc bằng Alt + F12.
- ✓ Ngoài ra, ta cũng có thể tìm tất cả các tham chiếu đến class hay method bằng tính năng Find All References với tổ hợp Shift + F12.

  Lúc này, VSC hiển thị khung cửa sổ con như tính năng Peek Definition nhưng ở cột bên phải còn có thêm danh sách các tham chiếu.



Peek Definition

- ✓ Ngoài ra, Visual Studo Code có một cách để xem nhanh định nghĩa bằng cách giữ phím Ctrl rồi rê chuột lên tên class hay method. Dùng tính năng này để xem nhanh code trước khi quyết định có nhảy đến phần định nghĩa hay không.

Một tính năng hữu ích khác đó là di chuyển ngay đến chỗ code có lỗi.

- Khi lỗi xảy ra, thanh trạng thái sẽ hiển thị số lỗi ở góc dưới bên trái màn hình.
- Click vào biểu tượng này sẽ mở ra khung nhập lệnh với dấu !.
- Bên canh đó, ta có thể nhảy ngay đến lỗi đầu tiên bằng phím F8.
- Lúc này, VSC sẽ hiển thị thông tin ngay tại dòng có lỗi.
- Nếu nhấn F8 lần nữa, ta sẽ được đưa đến vị trí có lỗi tiếp theo, và cứ như thế cho đến khi lặp qua hết các lỗi.

10. Cấu hình

VSC cung cấp một loạt các theme cho ta tha hồ lựa chọn, và ta cũng có thể cài thêm theme nếu muốn. **Theme** được phân ra làm hai mảng sáng (light) và tối (dark).

- ✓ Để đổi **theme**, ta vào ***File* > *Preferences* > *Color Theme***, rồi chọn theme phù hợp.

*Ngoài **theme,** VSC cung cấp nhiều tùy chọn khác để ta cấu hình theo ý mình.*

- ✓ Tương tự **Sublime Text**, file cấu hình chỉ là một file JSON và ta truy cập nó bằng cách vào *File* > *Preferences* > *User Settings*. Lúc này, trình soạn thảo code chia làm 2 phần, phần bên trái chứa các cấu hình mặc định để tham khảo, bên phải là một file trắng tên là **settings.json**.
- ✓ Để thông số cấu hình *không bị ghi đè* khi update VSC, ta sẽ ghi thông tin ở nửa bên phải.
  Đầu tiên, tìm cấu hình cần thay đổi ở nửa bên trái. Sau đó ta copy và dán vào nửa bên phải. Cuối cùng, ta thay đổi thông số cho phù hợp. Sau khi lưu lại file này, mọi thay đổi sẽ có hiệu lực.

*Khi chỉnh sửa cấu hình trong **User Settings**, áp dụng những thay đổi này cho tất cả dự án.*

- ✓ VSC còn cho phép ta cấu hình theo từng dự án khác nhau. Ví dụ muốn khi mở dự án A thì code editor sẽ có kích cỡ font là 16, còn khi mở dự án B thì kích cỡ là 18.
- Để làm việc này, ta vào ***File* > *Preferences* > *Workspace Settings***. Lúc này, trình soạn thảo code cũng chia làm hai nửa tương tự như cấu hình User Settings.
- Tuy nhiên, khi nhìn vào cây thư mục trong cửa sổ Explore, ta thấy VSC đã tạo thêm một thư mục mới tên**.settings** và bên trong chứa file **settings.json.**
- Đây chính là nơi lưu cấu hình cho từng dự án. Theo quy tắc, cấu hình của dự án sẽ được ưu tiên hơn cấu hình của hệ thống.
- Nếu ký tự muốn chia sẻ thông số cấu hình kèm theo dự án cho người khác, Workspace Settings là cách để thực hiện điều này.

Ngoài ra, ta cũng có thể cấu hình phím tắt.

- ✓ Để làm việc này, ta vào *File* > *Preferences* > *Keyboard Shortcuts*.

- ✓ Trình soạn thảo code chia làm hai nửa. Phía bên trái là những phím tắt mặc định.
- ✓ Để thay đổi, ta copy và dán vào nửa phải rồi thay đổi theo ý muốn.
  Khác với User hay Workspace Settings, file cấu hình phím tắt sử dụng một mảng các đối tượng, và mỗi đối tượng chứa hai thuộc tính bắt buộc là key và **command** cùng với một thuộc tính không bắt buộc là **when**.
- ✓ Thuộc tính key quy định các phím tắt. Thuộc tính command chỉ định lệnh sẽ chạy khi bấm phím tắt này. Còn thuộc tính when chỉ ra phạm vi hiệu lực của phím tắt.

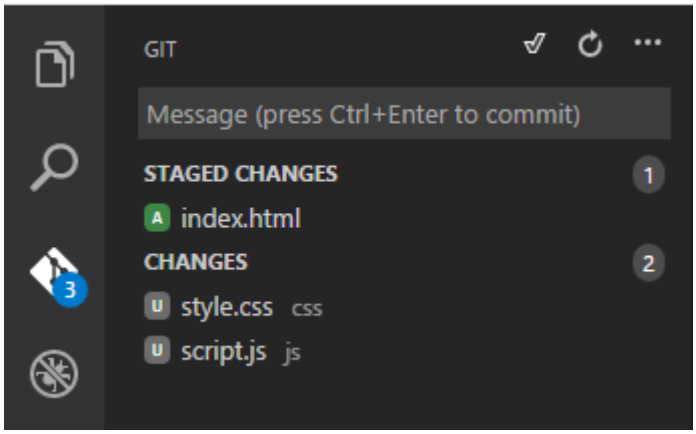
11. Quản lý code với Git

Git là thành phần không thể thiếu khi viết ứng dụng lớn, hoặc khi làm việc nhóm. Mọi thay đổi trong code cần được lưu lại, kể cả thông tin về người thực hiện thay đổi này. Khi có lỗi xảy ra, ta có thể quay về trạng thái cũ của code. *Để dùng Git trong* VSC, ta phải **cài Git** sẵn trong máy.

Cửa sổ Git chia làm hai phần: ***Staged Changes và Change***s

- ✓ Changes chứa file bị thay đổi kể từ lần commit cuối cùng. Còn Staged Changes chứa file sẽ được commit trong lần tiếp theo.
- ✓ Khi drag chuột lên tên file, ta thấy phía bên phải xuất hiện biểu tượng. Để biết tính năng của chúng, drag chuột và chờ tooltip xuất hiện.
- ✓ Để commit các file trong Staged Changes, ta ghi thông tin vào khung Message, sau đó nhấn Ctrl + Enter.
- ✓ Phía trên khung Message là một loạt các biểu tượng khác.

Trong đó, dấu ba chấm cung cấp một menu gồm khá nhiều tính năng của Git. Nếu chưa quen, ký tự có thể tham khảo bài viết của tôi hướng dẫn sử dụng tính năng cơ bản của Git.



Git

Phía trước tên file trong cửa sổ **Git** có một ký tự nhỏ. Đây là kí hiệu thông báo trạng thái của file. Bên dưới là danh sách các ký tự này:

- U (Untracked): file chưa được theo dõi.
- A (Added): file được thêm vào index để theo dõi.
- M (Modified): file đã được thay đổi.
- D (Deleted): file đã bị xóa.

- ✓ Khi click lên file có kí hiệu M (Modified) trong cửa sổ Git, trình soạn thảo hiển thị những thay đổi trong file kể từ lần *commit* cuối cùng.
- ✓ Những thay đổi này được đánh dấu bằng hai màu xanh và đỏ. Màu xanh nghĩa là thêm mới, còn màu đỏ nghĩa là bị xóa.

- ✓ Mặc định, VSC sẽ hiển thị dưới dạng *Side by Side View*.
- ✓ Để thay đổi cách hiển thị sang *Inline View*, ta chọn mục *Switch to Inline View* nằm trong menu dấu ba chấm (…) phía trên khung soạn thảo.

12. Debug

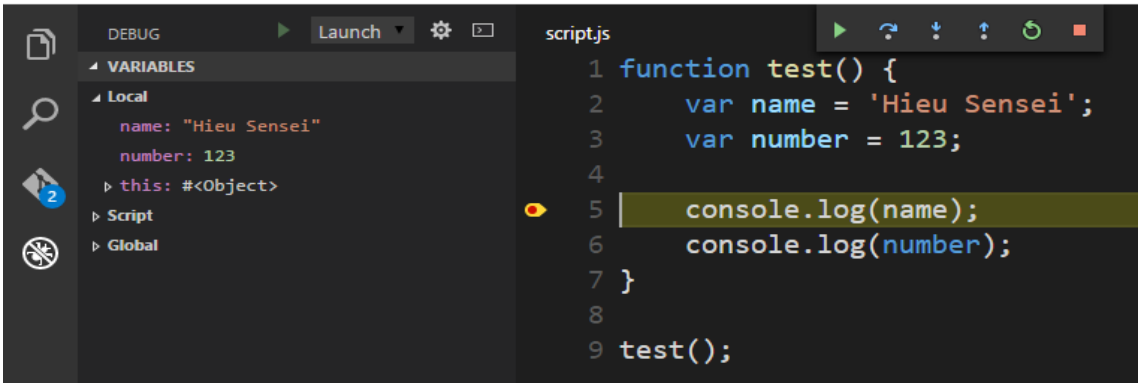
Mắc lỗi và sửa lỗi với VSC cung cấp một trình debug đủ thỏa mãn nhu cầu debug căn bản.

Cửa sổ debug chứa những thành phần căn bản ta thường thấy trong bất kỳ debugger:

- Khung Variables chứa thông tin về các biến.
- Khung Watch để thêm các biến cần theo dõi.
- Khung Call Stack là danh sách các hàm được gọi theo thứ tự thời gian ngược, nghĩa là hàm gọi sau sẽ nằm đầu danh sách.
- Khung Breakpoints chứa các tùy chọn cho breakpoint trong quá trình debug.

Hiện tại, VSC chỉ hỗ trợ debug ***file JavaScript hoặc TypeScript***.

- ✓ Trước khi debug, ta phải cấu hình nó bằng cách click vào biểu tượng bánh răng để tạo file **launch.json**.
- ✓ VSC hiện ra khung nhập lệnh để hỏi ta muốn debug trong môi trường nào. *Cũng trong file cấu hình,* thay đổi đường dẫn đến file cần debug ở thuộc tính **program**. Cấu hình xong, có thể bắt đầu debug.



Debug

- ♦ Để debug, vào trong file cần kiểm tra và click vào đầu dòng để đặt breakpoint.
- ♦ Một chấm đỏ xuất hiện đánh dấu chỗ dừng khi code chạy tới đó.
- ♦ Tiếp theo, chọn chế độ Launch và nhấn nút Start.
- ♦ Debugger chạy code và dừng ngay tại breakpoint.
- ♦ Các thông tin ở khung bên trái cũng được cập nhật. Phía trên, ta thấy 5 nút điều khiển: Continue, Step Over, Step Into, Step Out, Restart, và Stop.