**PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI**

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỀ TÀI

**TẠO GAME CỜ CA-RÔ VỚI JAVASCRIPT**

Giảng viên hướng dẫn: **Th.S Nguyễn Thị Kim Phụng**

Sinh viên nhóm 4:

Phan Trọng Đức\_2051067153

Vũ Trí Minh\_2051067806

Bùi Nam Minh Anh\_1951065199

TP. Hồ Chí Minh, tháng 12/2023

MỤC LỤC

[I. LỜI MỞ ĐẦU 3](#_Toc155448914)

[II. GIỚI THIỆU 4](#_Toc155448915)

[A. TRÍ TUỆ NHÂN TẠO 4](#_Toc155448916)

[1. Trí tuệ nhân tạo là gì 4](#_Toc155448917)

[2. Ưu điểm của trí tuệ nhân tạo 4](#_Toc155448918)

[3. Nhược điểm của trí tuệ nhân tạo 5](#_Toc155448919)

[B. PROJECT GAME CỜ CA-RÔ 7](#_Toc155448920)

[1. Mục tiêu 7](#_Toc155448921)

[2. Ngôn ngữ sử dụng 7](#_Toc155448922)

[3. Các thuật toán 7](#_Toc155448923)

[III. PHÂN TÍCH CODE 9](#_Toc155448924)

[A. CÁC THÀNH PHẦN TRONG WEB CỜ CA-RÔ 10](#_Toc155448925)

[1. Hiển thị thông tin trạng thái của lượt chơi hiện tại là của X hay O: id “status” 10](#_Toc155448926)

[2. Nút bắt đầu game mới: restart-btn 10](#_Toc155448927)

[3. Nút chọn đối thủ: single-player-toggle 10](#_Toc155448928)

[4. Nút chọn kích thước bàn cờ: dimension-button 11](#_Toc155448929)

[5. Bàn cờ: board 11](#_Toc155448930)

[B. BIẾN 11](#_Toc155448931)

[C. SỰ KIỆN 12](#_Toc155448932)

[1. document.addEventListener('DOMContentLoaded', function () 12](#_Toc155448933)

[2. dimensionButton.addEventListener('click', function () 12](#_Toc155448934)

[3. restartButton.addEventListener('click', restartGame) 12](#_Toc155448935)

[4. singlePlayerToggle.addEventListener('click', function () 12](#_Toc155448936)

[5. squareButton.addEventListener('click', () 13](#_Toc155448937)

[D. HÀM 13](#_Toc155448938)

[1. restartGame() 13](#_Toc155448939)

[2. toggleSinglePlayerMode() 14](#_Toc155448940)

[3. makeComputerMove() 14](#_Toc155448941)

[4. handleClick(row, col) 16](#_Toc155448942)

[5. calculateWinner(newSquares, row, col) 17](#_Toc155448943)

[6. findWinningLine(newSquares, row, col, winner) 20](#_Toc155448944)

[7. renderBoard() 23](#_Toc155448945)

[8. updateStatus() 26](#_Toc155448946)

[9. isWinningMove(currentSquares, player) 26](#_Toc155448947)

[IV. TRIỂN KHAI WEB 28](#_Toc155448948)

[A. Triển khai web tĩnh 28](#_Toc155448949)

[1. Chọn hosting 28](#_Toc155448950)

[2. Các bước thực hiện 29](#_Toc155448951)

[3. Kết quả 37](#_Toc155448952)

[B. TRIỂN KHAI WEB ĐỘNG 37](#_Toc155448953)

[1. Mục tiêu 37](#_Toc155448954)

[2. Kỹ thuật sử dụng 37](#_Toc155448955)

[3. Các bước triển khai 37](#_Toc155448956)

[V. TỔNG KẾT 52](#_Toc155448957)

[A. Về môn học 52](#_Toc155448958)

[Thành tựu: 52](#_Toc155448959)

[Điểm yếu: 52](#_Toc155448960)

[B. Về project 52](#_Toc155448961)

[Điểm hài lòng 52](#_Toc155448962)

[Điểm yếu 52](#_Toc155448963)

[C. Lời cảm ơn 52](#_Toc155448964)

[VI. NGUỒN THAM KHẢO 53](#_Toc155448965)

# LỜI MỞ ĐẦU

Có lẽ khi đặt câu hỏi “Bạn biết chơi cờ ca-rô không?” với bất kỳ học sinh, sinh viên nào, câu trả lời nhận được sẽ là có. Bởi đây là trò chơi gắn liền với tuổi học trò của mỗi người. Chỉ với một tờ giấy và cây viết cùng luật chơi đơn giản là có thể khiến học sinh túm tụm chơi quên cả giờ giấc. Cờ ca-rô là trò chơi có tính giải trí cao và nhóm chúng em đã chọn đề tài “Tạo game cờ ca-rô với JavaScript” để làm báo cáo cho học phần Trí tuệ nhân tạo của cô Nguyễn Thị Kim Phụng.

TP. HCM, tháng 12/2023

# GIỚI THIỆU

## TRÍ TUỆ NHÂN TẠO

### Trí tuệ nhân tạo là gì

Trí tuệ nhân tạo là một nhánh của khoa học máy tính, tập trung vào việc tạo ra những cỗ máy thông minh có khả năng bắt chước hành vi của con người.

Khi nói đến trí tuệ nhân tạo là nói đến sự phát triển của hệ thống máy tính có khả năng thực hiện các nhiệm vụ đòi hỏi trí thông minh của con người, chẳng hạn như phân tích dữ liệu, đưa ra quyết định và học hỏi kinh nghiệm.

Trí tuệ nhân tạo được sử dụng trong nhiều lĩnh vực khác nhau, bao gồm chăm sóc sức khỏe, tài chính, giao thông vận tải và giải trí.

### Ưu điểm của trí tuệ nhân tạo

1. Tăng hiệu quả và năng suất

Một trong những lợi thế chính của AI là khả năng nâng cao hiệu quả và năng suất. Các hệ thống được hỗ trợ bởi AI có thể xử lý lượng dữ liệu khổng lồ ở tốc độ cao, cho phép các tổ chức đưa ra quyết định sáng suốt một cách nhanh chóng. Những nhiệm vụ mà con người phải mất hàng giờ hoặc thậm chí vài ngày để hoàn thành có thể được thực hiện bằng thuật toán AI chỉ trong vài giây. Việc tăng hiệu quả này dẫn đến tiết kiệm chi phí và tăng sản lượng.

1. Tự động hóa các nhiệm vụ lặp đi lặp lại

Trí tuệ nhân tạo vượt trội trong việc tự động hóa các nhiệm vụ lặp đi lặp lại và thường lệ. Bằng cách giao các nhiệm vụ này cho hệ thống AI, doanh nghiệp có thể giải phóng nguồn nhân lực để tập trung vào những nỗ lực sáng tạo và phức tạp hơn. Ví dụ, trong sản xuất, robot được trang bị AI có thể thực hiện các nhiệm vụ dây chuyền lắp ráp lặp đi lặp lại với độ chính xác và nhất quán, giúp cải thiện năng suất và giảm sai sót.

1. Nâng cao khả năng ra quyết định

AI có thể phân tích lượng dữ liệu khổng lồ và cung cấp những hiểu biết có giá trị cho việc ra quyết định. Các thuật toán học máy có thể xác định các mô hình, xu hướng và mối tương quan mà con người có thể không chú ý đến. Khả năng trích xuất thông tin hữu ích từ dữ liệu này giúp doanh nghiệp đưa ra quyết định dựa trên dữ liệu, tối ưu hóa quy trình và cải thiện hiệu suất tổng thể.

1. Cải thiện việc chẩn đoán y tế và chăm sóc sức khỏe

Trí tuệ nhân tạo có khả năng cách mạng hóa việc chăm sóc sức khỏe bằng cách cho phép chẩn đoán chính xác hơn và cá nhân hóa điều trị. Các thuật toán AI có thể phân tích các hình ảnh y tế, chẳng hạn như chụp X-quang và chụp MRI, để phát hiện các bất thường và hỗ trợ bác sĩ X quang đưa ra các diễn giải chính xác hơn. Ngoài ra, các chatbot được hỗ trợ bởi AI có thể cung cấp lời khuyên y tế sơ bộ, giảm gánh nặng cho các chuyên gia chăm sóc sức khỏe.

1. Cá nhân hóa trải nghiệm người dùng

AI cho phép trải nghiệm người dùng được cá nhân hóa bằng cách tận dụng dữ liệu để hiểu sở thích và hành vi cá nhân. Các hệ thống đề xuất được hỗ trợ bởi thuật toán AI có thể đề xuất các sản phẩm, dịch vụ hoặc nội dung có liên quan dựa trên các tương tác trước đó của người dùng. Mức độ cá nhân hóa này nâng cao sự hài lòng của người dùng, thúc đẩy sự tương tác và tăng tỷ lệ chuyển đổi cho doanh nghiệp.

### Nhược điểm của trí tuệ nhân tạo

1. Dịch chuyển việc làm và thất nghiệp

Một trong những mối quan tâm chính xung quanh AI là khả năng dịch chuyển công việc và thất nghiệp mà nó có thể gây ra. Khi các hệ thống AI trở nên phức tạp hơn, có nguy cơ một số công việc nhất định sẽ trở nên lỗi thời. Các nhiệm vụ lặp đi lặp lại hoặc dựa trên quy tắc có thể được tự động hóa, dẫn đến giảm nhu cầu về nhân công ở những lĩnh vực đó. Tuy nhiên, điều quan trọng cần lưu ý là AI cũng tạo ra cơ hội việc làm mới trong các lĩnh vực như phát triển, bảo trì và giám sát AI.

1. Những quan ngại về đạo đức

Trí tuệ nhân tạo đặt ra những lo ngại về đạo đức, đặc biệt khi nói đến quá trình ra quyết định. Các thuật toán AI dựa trên dữ liệu và mẫu, có thể gây ra thành kiến hoặc thiếu khả năng xem xét các tác động đạo đức rộng hơn trong các quyết định của chúng. Ví dụ, công nghệ nhận dạng khuôn mặt đã phải đối mặt với những lời chỉ trích vì tiềm ẩn những thành kiến về chủng tộc và giới tính. Điều quan trọng là phải đảm bảo rằng các hệ thống AI được thiết kế với các nguyên tắc đạo đức và khuôn khổ trách nhiệm giải trình chặt chẽ.

1. Phụ thuộc vào công nghệ và dữ liệu

Hiệu quả của hệ thống AI phụ thuộc rất nhiều vào sự sẵn có của dữ liệu chất lượng và công nghệ tiên tiến. Nếu không có bộ dữ liệu chính xác và đa dạng, thuật toán AI có thể tạo ra kết quả sai lệch hoặc không chính xác. Ngoài ra, hệ thống AI yêu cầu quyền truy cập liên tục vào dữ liệu và bảo trì để luôn cập nhật và phù hợp. Sự phụ thuộc quá mức vào công nghệ và dữ liệu có thể đặt ra thách thức khi xảy ra gián đoạn hoặc khi nảy sinh những lo ngại về quyền riêng tư dữ liệu.

1. Rủi ro về bảo mật và quyền riêng tư

Khi AI ngày càng được tích hợp nhiều hơn vào nhiều lĩnh vực khác nhau, các rủi ro về bảo mật và quyền riêng tư liên quan đến nó ngày càng trở nên quan trọng. Các hệ thống được hỗ trợ bởi AI xử lý dữ liệu nhạy cảm là mục tiêu tiềm năng cho các cuộc tấn công mạng. Hơn nữa, các thuật toán AI xử lý thông tin cá nhân làm tăng mối lo ngại về quyền riêng tư và bảo vệ dữ liệu. Việc bảo vệ khỏi những rủi ro này đòi hỏi các biện pháp bảo mật mạnh mẽ và tuân thủ nghiêm ngặt các quy định về quyền riêng tư.

1. Chi phí triển khai và bảo trì cao

Việc triển khai và bảo trì hệ thống AI có thể tốn kém, đặc biệt đối với các doanh nghiệp và tổ chức nhỏ có nguồn lực hạn chế. Việc phát triển các thuật toán AI, mua phần cứng cần thiết và đào tạo nhân viên đòi hỏi phải đầu tư đáng kể. Hơn nữa, hệ thống AI cần cập nhật và bảo trì thường xuyên để đảm bảo hiệu suất tối ưu, điều này làm tăng thêm chi phí chung. Điều cần thiết là phải đánh giá cẩn thận tỷ lệ chi phí-lợi ích trước khi áp dụng các giải pháp AI.

## PROJECT GAME CỜ CA-RÔ

### Mục tiêu

* Tìm hiểu các thuật toán liên quan đến game cờ ca-rô
* Sử dụng code trên mạng để tạo web chơi game cờ ca-rô đơn giản

### Ngôn ngữ sử dụng

HTML, CSS, JavaScript

### Các thuật toán

#### Minimax

* Minimax là một thuật toán ra quyết định được sử dụng trong các trò chơi hai người chơi luân phiên nhau (như Gomoku). Mục tiêu: Tối đa hóa điểm số cho người chơi hiện tại(max) và tối thiểu hóa điểm số cho đối thủ (min).
* Ý tưởng: đánh giá đệ quy các bước di chuyển có thể có trong cây trò chơi, xen kẽ giữa người chơi hiện tại và đối thủ cho đến khi đạt đến độ sâu nhất định sau đó tìm Node có giá trị phù hợp để đi bước tiếp theo.
* Các khái niệm
* Cây trò chơi: Cây trò chơi là sự thể hiện bằng đồ họa của tất cả các nước đi có thể xảy ra và kết quả của chúng trong trò chơi hai người chơi. Nó bao gồm các nút đại diện cho các trạng thái khác nhau trong trò chơi và các cạnh đại diện cho các nước đi có thể xảy ra.
* Nút: Mỗi nút trong cây trò chơi đại diện cho một trạng thái trò chơi cụ thể.
* Nút lá: Các nút lá là các nút cuối cùng của cây trò chơi, nghĩa là chúng nằm ở cuối cây và không có nút con nào. Tại các nút này, thuật toán đánh giá mức độ mong muốn của trạng thái trò chơi.
* Hàm đánh giá: Hàm đánh giá được áp dụng cho các nút lá để gán một giá trị số biểu thị mức độ mong muốn của trạng thái trò chơi tương ứng. Hàm này rất quan trọng để xác định mức độ tốt hay xấu của một trạng thái cụ thể đối với người chơi hiện tại và đối thủ.
* Độ sâu tìm kiếm: Độ sâu tìm kiếm xác định thuật toán Minimax khám phá cây trò chơi đến mức nào. Các nút lá đạt được khi đạt đến độ sâu xác định hoặc khi gặp trạng thái cuối (thắng, thua hoặc hòa).
* Các bước cơ bản:
* Biểu diễn cây trò chơi: biểu diễn trò chơi dưới dạng một cái cây, trong đó mỗi nút tương ứng với một trạng thái trò chơi có thể có và mỗi cạnh thể hiện một nước đi hợp lệ
* Xác định người chơi hiện tại (max) và đối thủ (min)
* Tìm kiếm đệ quy: bắt đầu ở trạng thái trò chơi hiện tại và khám phá các bước di chuyển có thể. Đối với mỗi nước đi, áp dụng thuật toán Minimax một cách đệ quy để đánh giá trạng thái kết quả trò chơi.
* Đánh giá nút lá: tại các nút lá của cây (trạng thái cuối hoặc độ sâu xác định), sử dụng hàm đánh giá để gán điểm số cho trạng thái trò chơi. Điểm này thể hiện mức độ mong muốn về vị trí cho người chơi hiện tại và đối thủ.
* Lan truyền ngược: lan truyền điểm số từ các nút lá sao lưu lên cây. Tại mỗi nút không phải lá, chọn điểm tối đa hoặc tối thiểu, tùy thuộc vào việc nó đại diện tương ứng cho người chơi hiện tại hay đối thủ.
* Quyết định: nút cấp cao nhất của cây nhận được quyết định sau cùng. Nếu người chơi hiện tại là người chơi max, chọn nước đi có điểm cao nhất; nếu người chơi hiện tại là người chơi min, chọn nước đi có điểm thấp nhất.

#### Cắt tỉa Alpha-Beta

* Cắt tỉa Alpha-Beta là một kỹ thuật tối ưu hóa được sử dụng trong thuật toán Minimax để giảm số lượng nút được đánh giá trong cây tìm kiếm. Kỹ thuật này cải thiện đáng kể hiệu quả tìm kiếm, khiến nó trở nên khả thi hơn đối với các trò chơi như Gomoku, cờ vua, … là những game có hệ số phân nhánh cao.
* Mục tiêu: Giảm số lượng nút phân trong cây tìm kiếm mà không ảnh hưởng đến quyết định cuối cùng.
* Ý tưởng: gồm 2 giá trị, alpha và beta, để biểu thị giới hạn về số điểm có thể có của một nước đi. Nếu trong quá trình tìm kiếm, có thể xác định được nước đi tệ hơn cho người chơi hiện tại hoặc tốt hơn cho đối thủ so với giá trị alpha hoặc beta hiện tại, thì phần còn lại của cây con có thể được cắt bỏ bớt.
* Cải thiện hiệu quả: bằng cách cắt bớt những cây con không thể ảnh hưởng đến quyết định cuối cùng, Cắt tỉa Alpha-Beta tránh được những tính toán không cần thiết, giảm đáng kể độ phức tạp về thời gian.
* Các bước cơ bản:
* Khởi tạo Alpha và Beta: Đặt alpha là âm vô cực và beta là dương vô cực.
* Tìm kiếm đệ quy: Thực hiện tìm kiếm đệ quy thông qua cây trò chơi, cập nhật các giá trị alpha và beta khi quá trình tìm kiếm diễn ra.
* Điều kiện cắt tỉa: Nếu tại bất kỳ thời điểm nào, nếu điểm của nút hiện tại được cho là kém hơn alpha hiện tại cho người chơi hiện tại hoặc tốt hơn phiên bản beta hiện tại cho đối thủ thì bỏ bớt phần còn lại của cây con.

# PHÂN TÍCH CODE

## CÁC THÀNH PHẦN TRONG WEB CỜ CA-RÔ

### Hiển thị thông tin trạng thái của lượt chơi hiện tại là của X hay O: id “status”

|  |  |
| --- | --- |
| 1 | **const** statusElement = document.getElementById('status'); |

Dòng 1: khai báo biến hằng statusElement và gán giá trị là phần tử html có id “status” để đọc và điều khiển thuộc tính của phần tử này

### Nút bắt đầu game mới: restart-btn

|  |  |
| --- | --- |
| 1 | **const** restartButton = document.getElementById('restart-btn'); |

Khởi tạo biến hằng restartButton và gán giá trị là phần tử html có id “restart-btn”, để đọc và điều khiển thuộc tính của phần tử này

### Nút chọn đối thủ: single-player-toggle

|  |  |
| --- | --- |
| 1 | **const** singlePlayerToggle = document.getElementById('single-player-toggle'); |

Khai báo biến hằng singlePlayerToggle và gán giá trị là phần tử html có id “single-player-toggle”, để đọc và điều khiển thuộc tính của phần tử này

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | singlePlayerToggle.addEventListener('click', **function** () {  toggleSinglePlayerMode();  restartGame();  **if** (singlePlayerMode && !xIsNext) {  makeComputerMove();  }  }); |

Dòng 1, 2, 3: tạo sự kiện chờ click chuột cho biến singlePlayerToggle, biến này được gán giá trị là phần tử html có id “single-player-toggle”, khi click vào phần tử này, hàm “toggleSinglePlayerMode” sẽ thực thi

Dòng 4: gọi hàm restartGame, để bắt đầu game mới

Dòng 5: kiểm tra biến singlePlayerMode và biến xIsNext, nếu biến singlePlayerMode là true, người chơi đấu với máy và lượt chơi không phải của quân “X” thì thực thi hàm makeComputerMove, để máy tính đi trước

### Nút chọn kích thước bàn cờ: dimension-button

|  |  |
| --- | --- |
| 1 | **const** dimensionButton = document.getElementById('dimension-button'); |

Khai báo biến hằng dimensionButton và gán giá trị phần tử html có id “dimension-button”, để đọc và sử dụng thuộc tính của phần tử này

|  |  |
| --- | --- |
| 1 | dimensionButton.textContent = `${dimension}x${dimension}`; |

Gán giá trị cho thuộc tính “textContent” của đối tượng dimensionButton là kích thước mảng của biến dimension

### Bàn cờ: board

|  |  |
| --- | --- |
| 1 | **const** boardElement = document.getElementById('board'); |

Khai báo biến hằng boardElement và gán giá trị là phần tử html có id “board”, để đọc và sử dụng thuộc tính của phần tử này

## BIẾN

dimension: lưu trữ kích thước hiện tại của bàn cờ, mặc định 10x10

dimensions: lưu trữ mảng các kích thước của bàn cờ (10, 12, 16, 20)

dimensionElement: gồm các kích thước 10x10, 12x12, 16x16, 20x20

dimensionIndex: Theo dõi chỉ mục hiện tại trong mảng dimensions.

squares: mảng 2D đại diện cho bảng, ban đầu chứa giá trị null.

xIsNext: biến boolean cho biết đến lượt của người chơi nào (true là X, false là O).

theWinner: người chiến thắng khi trò chơi kết thúc (X, O hoặc null nếu không có người chiến thắng).

winningLine: Lưu tọa độ của 5 ô liên tiếp để làm nổi bật

singlePlayerMode: biến boolean chuyển đổi giữa chế độ một người chơi (đấu với máy) và hai người chơi đấu với nhau

## SỰ KIỆN

### document.addEventListener('DOMContentLoaded', function ()

Sự kiện 'DOMContentLoaded' được trình duyệt kích hoạt khi tài liệu HTML đã được tải và phân tích cú pháp đầy đủ, nghĩa là tất cả nội dung HTML, bao gồm cây DOM (Mô hình đối tượng tài liệu), đã sẵn sàng để thao tác bằng JavaScript.

Phương thức addEventListener được sử dụng để đính kèm hàm xử lý sự kiện vào sự kiện đã chỉ định ('DOMContentLoaded' trong trường hợp này). Tham số thứ hai của addEventListener là một hàm ẩn danh (còn được gọi là hàm gọi lại) sẽ được thực thi khi sự kiện xảy ra.

### dimensionButton.addEventListener('click', function ()

Khi người dùng nhấp vào nút có id " dimension-button", chức năng bên trong khối lệnh sẽ được thực thi: xử lý sự kiện nhấp chuột bằng cách cập nhật kích thước của bảng trò chơi và khởi động lại trò chơi với kích thước mới.

### restartButton.addEventListener('click', restartGame)

Khi người dùng nhấp vào nút có id "restart-btn", hàm restartGame sẽ được gọi. Hàm restartGame có nhiệm vụ tải lại trò chơi, xóa bảng và cập nhật trạng thái trò chơi để bắt đầu trò chơi mới

### singlePlayerToggle.addEventListener('click', function ()

Khi người dùng nhấp vào nút có id "single-player-toggle", chức năng bên trong khối lệnh sẽ được thực thi: xử lý sự kiện nhấp chuột bằng cách chuyển đổi giữa chế độ một người chơi và chế độ hai người chơi, đồng thời cập nhật trạng thái trò chơi. Hàm chuyển đổiSinglePlayerMode() được gọi trong khối lệnh này để chuyển đổi giữa các chế độ chơi

### squareButton.addEventListener('click', ()

Khi người dùng nhấp vào một trong các ô vuông được biểu thị bằng SquareButton, sự kiện nhấp chuột này sẽ kích hoạt hàm handClick, hàm này xử lý nước đi của người chơi trong trò chơi, kiểm tra điều kiện thắng và hiển thị bảng trò chơi đã cập nhật.

## HÀM

### restartGame()

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | restartGame() {  squares = Array(dimension).fill(Array(dimension).fill(**null**));  xIsNext = **true**;  theWinner = **null**;  winningLine = [];  renderBoard();  updateStatus();  document.body.style.backgroundImage = "url()";  } |

Dòng 2, 3: khai báo biến squares là mảng 2 chiều gồm các giá trị null

Dòng 4: khai báo biến xIsNext kiểu boolean có giá trị true, giúp lưu thông tin lượt chơi

Dòng 5: khai báo biến theWinner là null, chưa có người thắng

Dòng 6: khai báo biến winningLine là mảng rỗng, mảng sẽ chứa tọa độ các ô tạo thành 5 quân liên tiếp theo chiều dọc, hoặc ngang hoặc chéo

Dòng 7: hàm tái hiện trạng thái bàn cờ dựa vào biến squares

Dòng 8: hàm cập nhật trạng thái của game, gồm lượt chơi đang của ai, có người thắng hay chưa

Dòng 9: vô hiệu hóa hình nền bằng html DOM

### toggleSinglePlayerMode()

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **function** toggleSinglePlayerMode() {  singlePlayerMode = !singlePlayerMode;  **if** (singlePlayerMode) {  singlePlayerToggle.innerHTML = '&#x1F4BB;';  } **else** {  singlePlayerToggle.innerHTML = '&#x1F477; ';  }  } |

singlePlayerMode = !singlePlayerMode;

Chuyển đổi giá trị của biến singlePlayerMode. Nếu true thì nó trở thành false và ngược lại. Biến này được sử dụng để theo dõi xem trò chơi đang ở chế độ một người chơi hay chế độ hai người chơi.

if (singlePlayerMode) { ... } else { ... }: kiểm tra giá trị của singlePlayerMode. Nếu true (có nghĩa là trò chơi ở chế độ một người chơi), thì đặt nội dung HTML của phần tử singlePlayerToggle thành biểu tượng máy tính. Nếu false (có nghĩa là trò chơi ở chế độ hai người chơi), thì đặt nội dung HTML thành biểu tượng người.

### makeComputerMove()

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | **function** makeComputerMove() {  **if** (!singlePlayerMode || theWinner) {  **return**;  }  **const** availableMoves = [];  **const** humanPlayer = xIsNext ? 'X' : 'O';  **const** computerPlayer = xIsNext ? 'O' : 'X';  squares.forEach((row, rowIndex) => {  row.forEach((col, colIndex) => {  **if** (!squares[rowIndex][colIndex]) {  availableMoves.push([rowIndex, colIndex]);  }  });  });  **if** (availableMoves.length > 0) {  *// Check if the computer can win in the next move*  *// ... (omitted for brevity)*  *// Random move for normal difficulty*  **const** randomIndex = Math.floor(Math.random() \* availableMoves.length);  **const** [row, col] = availableMoves[randomIndex];  handleClick(row, col);  }  } |

if (!singlePlayerMode || theWinner) { return; }: Kiểm tra xem trò chơi không ở chế độ chơi đơn hay đã có người chiến thắng. Nếu một trong hai điều kiện là đúng thì hàm sẽ thoát sớm và máy tính không thực hiện nước đi nào.

const availableMoves = [];

Khởi tạo mảng availableMoves lưu trữ các vị trí có sẵn (trống) trên bảng trò chơi nơi máy tính có thể thực hiện nước đi.

const humanPlayer = xIsNext? 'X' : 'O';

const ComputerPlayer = xIsNext? 'O' : 'X';

Xác định các ký hiệu cho người chơi và máy tính dựa trên lượt hiện tại (xIsNext).

Squares.forEach((row, rowIndex) => { ... });

Lặp qua từng hàng và cột trong bảng trò chơi (hình vuông) để điền vào mảng availableMoves với tọa độ của các vị trí trống nơi máy tính có thể thực hiện nước đi.

if (availableMoves.length > 0) { ... }: Điều kiện này kiểm tra xem máy tính có thực hiện được các bước di chuyển hay không.

Bên trong khối điều kiện:

const RandomIndex = Math.floor(Math.random() \* availableMoves.length);

Tạo ra một chỉ mục ngẫu nhiên trong phạm vi các nước đi có sẵn.

const [row, col] = availableMoves[randomIndex];

Trích xuất tọa độ hàng và cột từ nước đi được chọn ngẫu nhiên.

handClick(row, col);

Gọi hàm handClick, mô phỏng một cú nhấp chuột trên bảng trò chơi ở tọa độ đã chỉ định. Hàm này có nhiệm vụ cập nhật trạng thái trò chơi, kiểm tra điều kiện thắng và hiển thị bảng trò chơi đã cập nhật.

### handleClick(row, col)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | **function** handleClick(row, col) {  **if** (theWinner || squares[row][col]) {  **return**;  }  **const** newSquares = squares.map((row) => [...row]);  newSquares[row][col] = xIsNext ? 'X' : 'O';  squares = newSquares;  xIsNext = !xIsNext;  **const** winner = calculateWinner(newSquares, row, col);  **if** (winner) {  theWinner = winner;  winningLine = findWinningLine(newSquares, row, col, winner);  }  renderBoard();  updateStatus();  **if** (singlePlayerMode && !theWinner && !xIsNext) {  makeComputerMove();  }  } |

if (theWinner || Squares[row][col]) { return; }: Điều kiện này kiểm tra xem đã có người chiến thắng chưa hoặc ô vuông được nhấp vào đã được lấp đầy chưa. Nếu một trong hai điều kiện là đúng thì hàm sẽ thoát sớm và không thực hiện thêm hành động nào.

const newSquares = Squares.map((row) => [...row]);

Tạo một bản sao mới của mảng hình vuông bằng hàm map, để tránh sửa đổi trực tiếp mảng ô vuông ban đầu.

newSquares[row][col] = xIsNext ? 'X' : 'O';

Dòng này cập nhật giá trị của hình vuông được nhấp vào trong mảng mới dựa trên biểu tượng của người chơi hiện tại ('X' hoặc 'O').

Squares = newSquares;

Dòng này cập nhật mảng hình vuông ban đầu với mảng newSquares đã sửa đổi.

xIsNext = !xIsNext;

Dòng này chuyển đổi lượt giữa 'X' và 'O' cho nước đi tiếp theo.

const Winner = CalculateWinner(newSquares, row, col);

Kiểm tra xem nước đi hiện tại có dẫn đến người chiến thắng hay không bằng cách gọi hàm CalculateWinner.

if (người chiến thắng) { ... }: Nếu có người chiến thắng:

theWinner = Winner;

Đặt biến theWinner thành biểu tượng của người chơi chiến thắng.

winningLine = findWinningLine(newSquares, row, col, Winner);

Gọi hàm findWinningLine để gán giá trị trả về cho biến winningLine .

renderBoard();

Gọi hàm renderBoard để cập nhật hình ảnh trực quan của bảng trò chơi.

updateStatus();

Gọi hàm updateStatus để hiển thị trạng thái trò chơi hiện tại.

### calculateWinner(newSquares, row, col)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81 | **function** calculateWinner(newSquares, row, col) {  **const** currentPlayer = newSquares[row][col];  *// Check horizontally*  **let** count = 1;  **let** leftCol = col - 1;  **while** (leftCol >= 0 && newSquares[row][leftCol] === currentPlayer) {  count++;  leftCol--;  }  **let** rightCol = col + 1;  **while** (rightCol < dimension && newSquares[row][rightCol] === currentPlayer) {  count++;  rightCol++;  }  **if** (count >= 5) {  **return** currentPlayer;  }  *// Check vertically*  count = 1;  **let** topRow = row - 1;  **while** (topRow >= 0 && newSquares[topRow][col] === currentPlayer) {  count++;  topRow--;  }  **let** bottomRow = row + 1;  **while** (bottomRow < dimension && newSquares[bottomRow][col] === currentPlayer) {  count++;  bottomRow++;  }  **if** (count >= 5) {  **return** currentPlayer;  }  *// Check diagonally (top-left to bottom-right)*  count = 1;  **let** topLeftRow = row - 1;  **let** topLeftCol = col - 1;  **while** (topLeftRow >= 0 && topLeftCol >= 0 && newSquares[topLeftRow][topLeftCol] === currentPlayer) {  count++;  topLeftRow--;  topLeftCol--;  }  **let** bottomRightRow = row + 1;  **let** bottomRightCol = col + 1;  **while** (bottomRightRow < dimension && bottomRightCol < dimension && newSquares[bottomRightRow][bottomRightCol] === currentPlayer) {  count++;  bottomRightRow++;  bottomRightCol++;  }  **if** (count >= 5) {  **return** currentPlayer;  }  *// Check diagonally (top-right to bottom-left)*  count = 1;  **let** topRightRow = row - 1;  **let** topRightCol = col + 1;  **while** (topRightRow >= 0 && topRightCol < dimension && newSquares[topRightRow][topRightCol] === currentPlayer) {  count++;  topRightRow--;  topRightCol++;  }  **let** bottomLeftRow = row + 1;  **let** bottomLeftCol = col - 1;  **while** (bottomLeftRow < dimension && bottomLeftCol >= 0 &&  newSquares[bottomLeftRow][bottomLeftCol] === currentPlayer) {  count++;  bottomLeftRow++;  bottomLeftCol--;  }  **if** (count >= 5) {  **return** currentPlayer;  }  **return** **null**;  } |

const currentPlayer = newSquares[row][col];

Truy xuất ký hiệu ('X' hoặc 'O') của người chơi đã thực hiện nước đi hiện tại. Kiểm tra ngang: kiểm tra điều kiện chiến thắng theo chiều ngang ở bên trái và bên phải nước vừa đi.

Kiểm tra dọc: kiểm tra điều kiện chiến thắng theo chiều dọc bên trên và bên dưới nước vừa đi.

Kiểm tra đường chéo (trên cùng bên trái đến dưới cùng bên phải và trên cùng bên phải đến dưới cùng bên trái): kiểm tra điều kiện thắng theo đường chéo theo cả hai hướng.

if (count >= 5) { return currentPlayer; }: Nếu số quân liên tiếp theo bất kỳ hướng nào lớn hơn hoặc bằng 5 (biểu thị điều kiện thắng) thì sẽ trả về ký hiệu của người chơi thắng (X hoặc O). Nếu không tìm thấy điều kiện chiến thắng theo bất kỳ hướng nào thì trả về null để cho biết rằng không có người chiến thắng.

### findWinningLine(newSquares, row, col, winner)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83 | **function** findWinningLine(newSquares, row, col, winner) {  **const** currentPlayer = winner;  **const** lines = [];  *// Check horizontally*  **let** leftCol = col - 1;  **while** (leftCol >= 0 && newSquares[row][leftCol] === currentPlayer) {  lines.push([row, leftCol]);  leftCol--;  }  lines.push([row, col]);  **let** rightCol = col + 1;  **while** (rightCol < dimension && newSquares[row][rightCol] === currentPlayer) {  lines.push([row, rightCol]);  rightCol++;  }  **if** (lines.length >= 5) {  **return** lines;  }  *// Check vertically*  **let** topRow = row - 1;  **while** (topRow >= 0 && newSquares[topRow][col] === currentPlayer) {  lines.push([topRow, col]);  topRow--;  }  lines.push([row, col]);  **let** bottomRow = row + 1;  **while** (bottomRow < dimension && newSquares[bottomRow][col] === currentPlayer) {  lines.push([bottomRow, col]);  bottomRow++;  }  **if** (lines.length >= 5) {  **return** lines;  }  *// Check diagonally (top-left to bottom-right)*  **let** topLeftRow = row - 1;  **let** topLeftCol = col - 1;  **while** (topLeftRow >= 0 && topLeftCol >= 0 && newSquares[topLeftRow][topLeftCol] === currentPlayer) {  lines.push([topLeftRow, topLeftCol]);  topLeftRow--;  topLeftCol--;  }  lines.push([row, col]);  **let** bottomRightRow = row + 1;  **let** bottomRightCol = col + 1;  **while** (bottomRightRow < dimension && bottomRightCol < dimension && newSquares[bottomRightRow][bottomRightCol] === currentPlayer) {  lines.push([bottomRightRow, bottomRightCol]);  bottomRightRow++;  bottomRightCol++;  }  **if** (lines.length >= 5) {  **return** lines;  }  *// Check diagonally (top-right to bottom-left)*  **let** topRightRow = row - 1;  **let** topRightCol = col + 1;  **while** (topRightRow >= 0 && topRightCol < dimension && newSquares[topRightRow][topRightCol] === currentPlayer) {  lines.push([topRightRow, topRightCol]);  topRightRow--;  topRightCol++;  }  lines.push([row, col]);  **let** bottomLeftRow = row + 1;  **let** bottomLeftCol = col - 1;  **while** (bottomLeftRow < dimension && bottomLeftCol >= 0 && newSquares[bottomLeftRow][bottomLeftCol] === currentPlayer) {  lines.push([bottomLeftRow, bottomLeftCol]);  bottomLeftRow++;  bottomLeftCol--;  }  **if** (lines.length >= 5) {  **return** lines;  }  **return** [];  } |

const currentPlayer = winner;

Truy xuất ký hiệu ('X' hoặc 'O') của người chơi đã thắng

constlines = [];

Khởi tạo một mảng để lưu trữ dòng ô vuông dẫn đến chiến thắng.

Kiểm tra ngang, dọc và chéo: kiểm tra theo chiều ngang, chiều dọc và đường chéo để tìm đường vuông dẫn đến chiến thắng, tiếp tục mở rộng đường theo cả hai hướng cho đến khi chạm đến mép bàn cờ hoặc gặp ô vuông không thuộc về người chơi chiến thắng.

if (lines.length >= 5) { trả về các dòng; }: Nếu độ dài của mảng dòng lớn hơn hoặc bằng 5 (biểu thị điều kiện thắng) thì trả về mảng đại diện cho dòng thắng. Nếu không tìm thấy dòng chiến thắng, nó sẽ trả về một mảng trống.

### renderBoard()

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | **function** renderBoard() {  boardElement.innerHTML = '';  **for** (**let** row = 0; row < dimension; row++) {  **const** rowElement = document.createElement('div');  rowElement.className = 'board-row';  **for** (**let** col = 0; col < dimension; col++) {  **const** value = squares[row][col];  **const** isWinningSquare = winningLine.some(([winRow, winCol]) => winRow === row && winCol === col);  **const** squareButton = document.createElement('button');  squareButton.className = 'square';  squareButton.style.backgroundColor = isWinningSquare ? 'yellow' : 'white';  squareButton.style.color = value === 'X' ? 'blue' : 'red';  squareButton.style.fontWeight = isWinningSquare ? 'bold' : 'normal';  squareButton.textContent = value;  squareButton.addEventListener('click', () => {  **if** (!singlePlayerMode || (singlePlayerMode && xIsNext)) {  handleClick(row, col);  }  });  rowElement.appendChild(squareButton);  }  boardElement.appendChild(rowElement);  }  } |

boardElement.innerHTML = '';

Xóa nội dung hiện có bên trong boardElement. Bước này là cần thiết để cập nhật bảng với trạng thái mới.

Vòng ngoài (Hàng): Lặp lại từng hàng của bảng trò chơi.

const rowElement = document.createElement('div');: Tạo phần tử div mới để thể hiện một hàng trong bảng trò chơi.

Vòng lặp bên trong (Cột): Lặp lại từng cột trong hàng hiện tại.

const value = Squares[row][col];: Lấy giá trị ('X', 'O' hoặc null) của hình vuông ở hàng và cột hiện tại.

const isWinningSquare = chiến thắngLine.some(([winRow, winCol]) => winRow === row && winCol === col);

Kiểm tra xem hình vuông hiện tại có phải là một phần của dòng chiến thắng hay không bằng cách so sánh tọa độ của nó với các tọa độ trong dòng chiến thắng mảng.

const SquareButton = document.createElement('button');

Tạo thành phần nút mới để thể hiện hình vuông trong bảng trò chơi.

Tạo kiểu cho hình vuông:SquareButton.className = 'square';

Đặt lớp CSS cho mục đích tạo kiểu.

SquareButton.style.backgroundColor = isWinningSquare ? 'vàng' : 'trắng';

Đặt màu nền của hình vuông thành màu vàng nếu nó là một phần của đường chiến thắng, nếu không, nó sẽ đặt màu nền thành màu trắng.

SquareButton.style.color = giá trị === 'X' ? 'blue' : 'red';

Đặt màu văn bản dựa trên biểu tượng của người chơi ('X' màu xanh lam, 'O' màu đỏ).

SquareButton.style.fontWeight = isWinningSquare ? 'bold' : 'normal';

Đặt độ đậm của phông chữ thành đậm cho các ô vuông trong dòng chiến thắng, nếu không, nó sẽ đặt thành bình thường.

Đặt nội dung văn bản và sự kiện nhấp chuột:

SquareButton.textContent = value;

Đặt nội dung văn bản của nút hình vuông thành ký hiệu của người chơi hoặc một chuỗi trống (chữ thường).

SquareButton.addEventListener('click', () => { ... });: Thêm trình xử lý sự kiện nhấp chuột vào nút vuông. Nếu nó không ở chế độ chơi đơn hoặc đến lượt người chơi, nó sẽ gọi hàm handClick với tọa độ của hình vuông được nhấp vào.

Gắn nút vuông vào hàng: rowElement.appendChild(squareButton);: Nối nút hình vuông vào hàng hiện tại.

Nối hàng vào bảng trò chơi: boardElement.appendChild(rowElement);

### updateStatus()

function updateStatus() {

if (theWinner) {

statusElement.textContent = `Chiến thắng: ${theWinner}`;

document.body.style.backgroundImage = "url(l03b0s.d57ca31e.gif)";

} else {

statusElement.textContent = `Người chơi: ${xIsNext ? 'X' : 'O'}`;

}

}

if (theWinner) { ... }: Kiểm tra xem có người chiến thắng hay không (theWinner không phải là null).

Trường hợp người chiến thắng:

statusElement.textContent = Chiến thắng: ${theWinner};

Đặt nội dung văn bản của thành phần trạng thái để chỉ ra người chiến thắng. Ví dụ: nếu 'X' là người chiến thắng thì trạng thái sẽ là "Chiến thắng: X".

document.body.style.backgroundImage = "url(l03b0s.d57ca31e.gif)";

Thay đổi hình nền của trang web (tạo hiệu ứng chiến thắng)

Không có trường hợp người chiến thắng:

else { ... }: Thực thi khối này nếu không có người chiến thắng.

statusElement.textContent = Người chơi: ${xIsNext ? 'X' : 'O'};

Đặt nội dung văn bản của thành phần trạng thái để cho biết lượt của người chơi hiện tại. Nếu xIsNext đúng thì có nghĩa đó là người chơi 'X' (nếu không thì đó là người chơi 'O').

### isWinningMove(currentSquares, player)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | **function** isWinningMove(currentSquares, player) {  **for** (**let** row = 0; row < dimension; row++) {  **for** (**let** col = 0; col < dimension; col++) {  **if** (!currentSquares[row][col]) {  **const** newSquares = currentSquares.map((row) => [...row]);  newSquares[row][col] = player;  **if** (calculateWinner(newSquares, row, col) === player) {  **return** **true**;  }  }  }  }  **return** **false**;  } |

Lặp qua các hình vuông: sử dụng các vòng lặp lồng nhau để lặp qua từng ô vuông trên bảng trò chơi.

Kiểm tra các ô trống: if (!currentSquares[row][col]) { ... }: Kiểm tra xem ô vuông hiện tại có trống không.

Mô phỏng di chuyển:

const newSquares = currentSquares.map((row) => [...row]);

Tạo một bản sao của bảng trò chơi hiện tại (currentSquares) để mô phỏng việc thực hiện một nước đi mà không sửa đổi bảng gốc.

newSquares[row][col] = player;

Mô phỏng việc thực hiện nước đi của người chơi (người chơi) được chỉ định trong ô trống hiện tại.

Kiểm tra điều kiện chiến thắng:

if (tínhWinner(newSquares, row, col) === player) { return true; }

Sử dụng hàm CalculateWinner() để kiểm tra xem việc di chuyển trong ô hiện tại có mang lại chiến thắng cho người chơi được chỉ định hay không. Nếu vậy, hàm trả về true, cho biết nước đi này là nước đi thắng.

Trả về sai: nếu không tìm thấy nước đi chiến thắng nào sau khi lặp qua tất cả các ô trống, hàm sẽ trả về sai.

# TRIỂN KHAI WEB

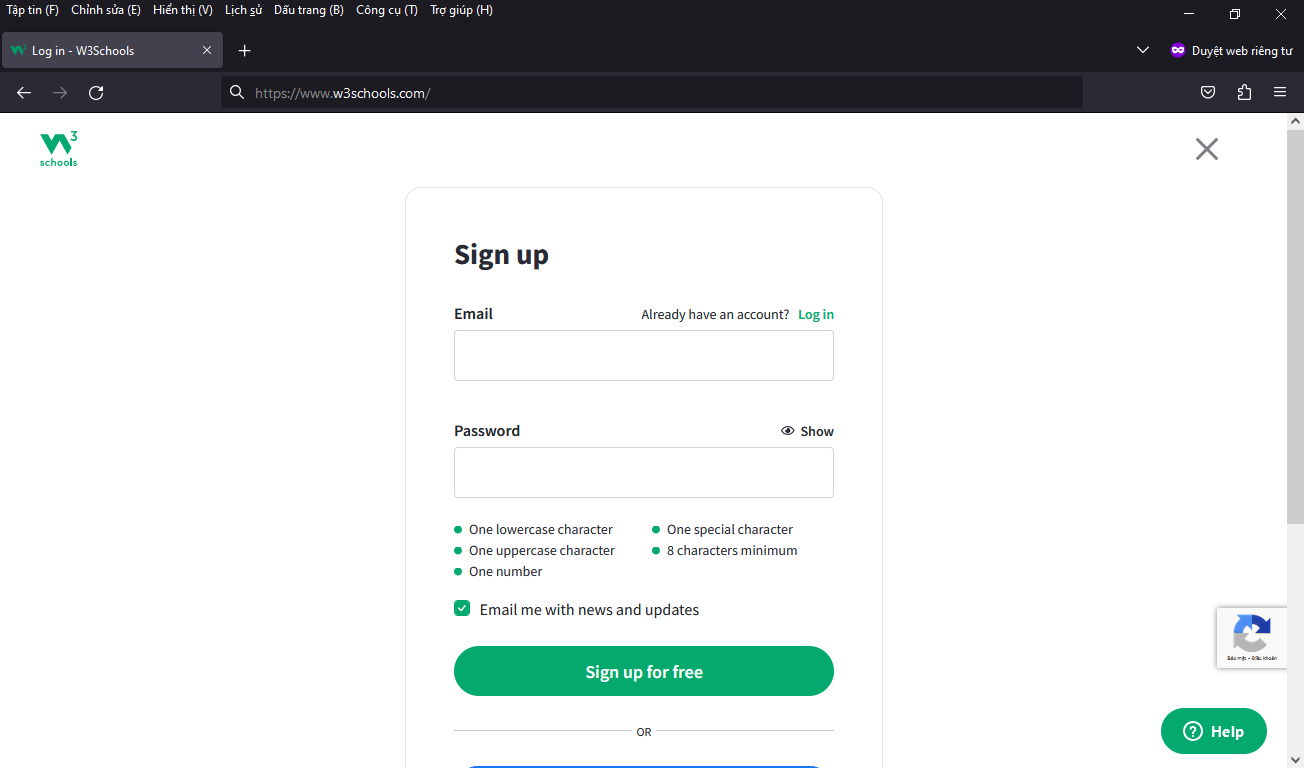
## Triển khai web tĩnh

### Chọn hosting

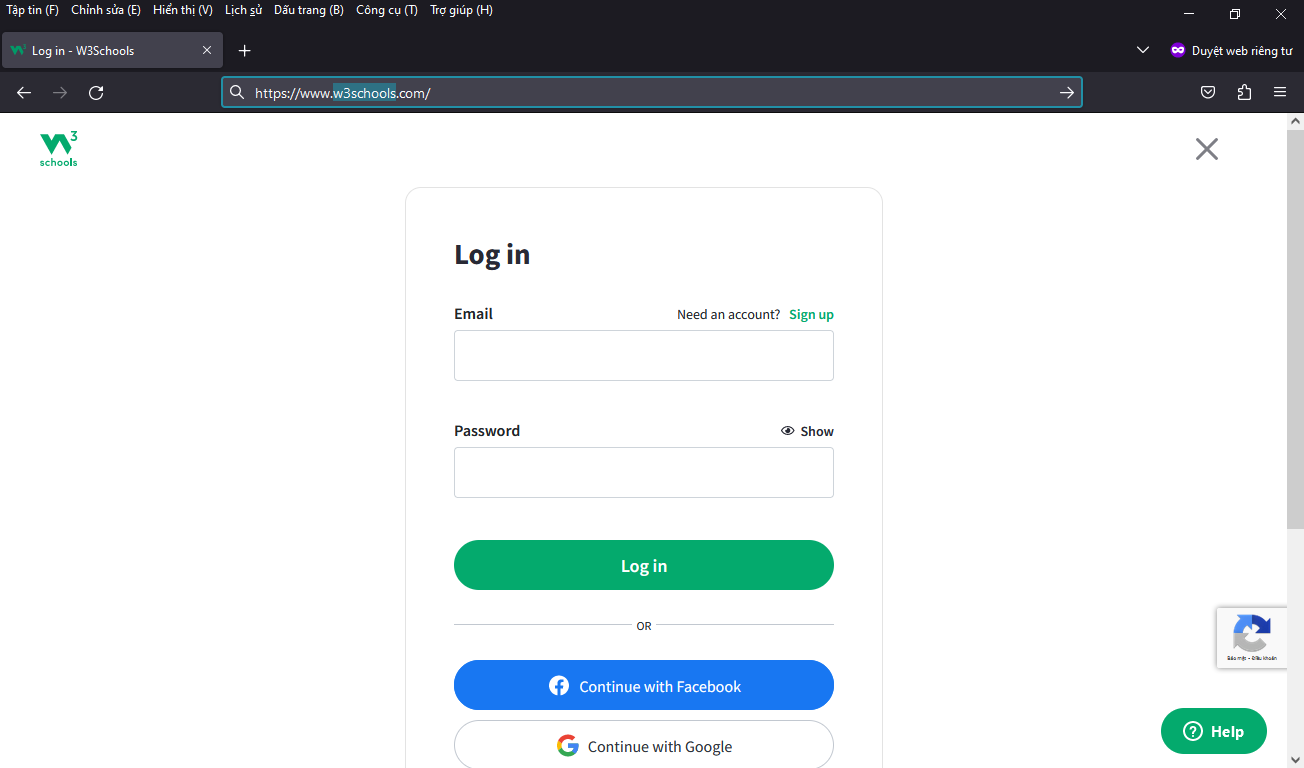
Chọn hosting miễn phí của w3schools.

### Các bước thực hiện

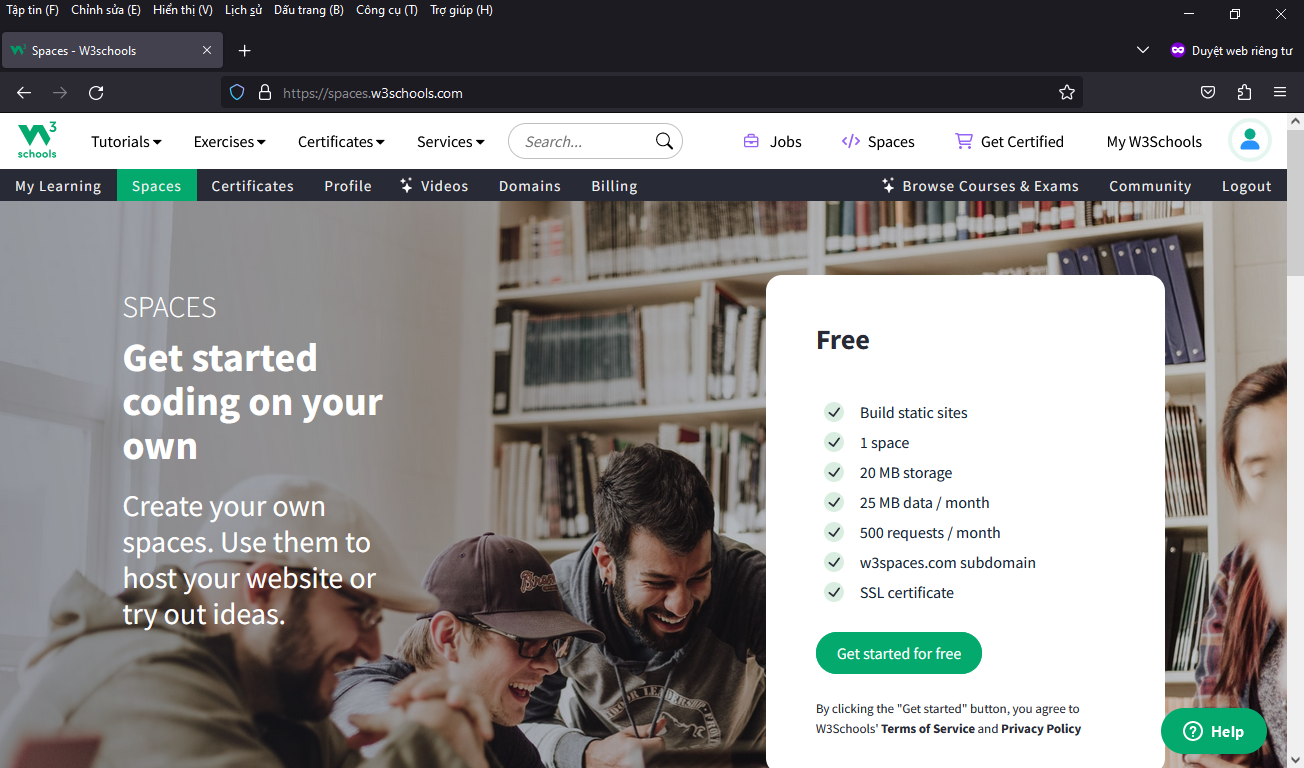
Tạo tài khoản



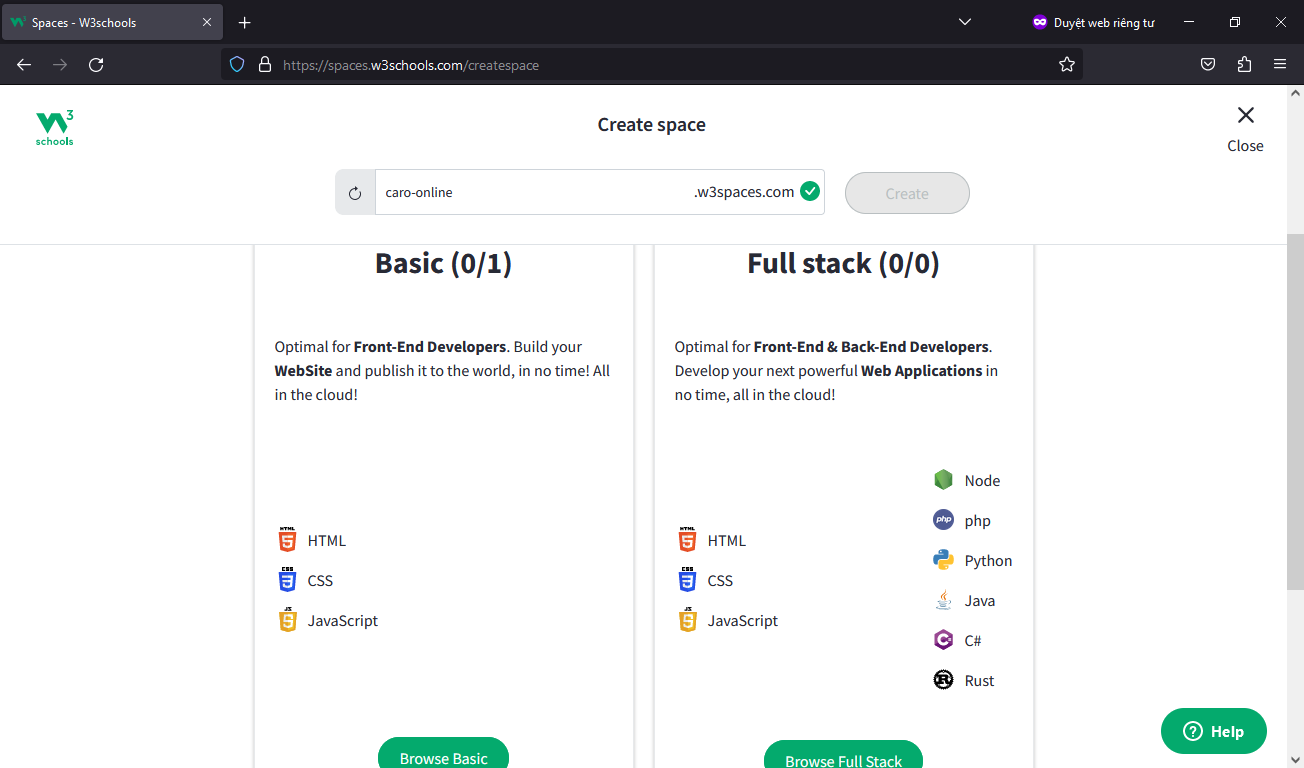
Đăng nhập



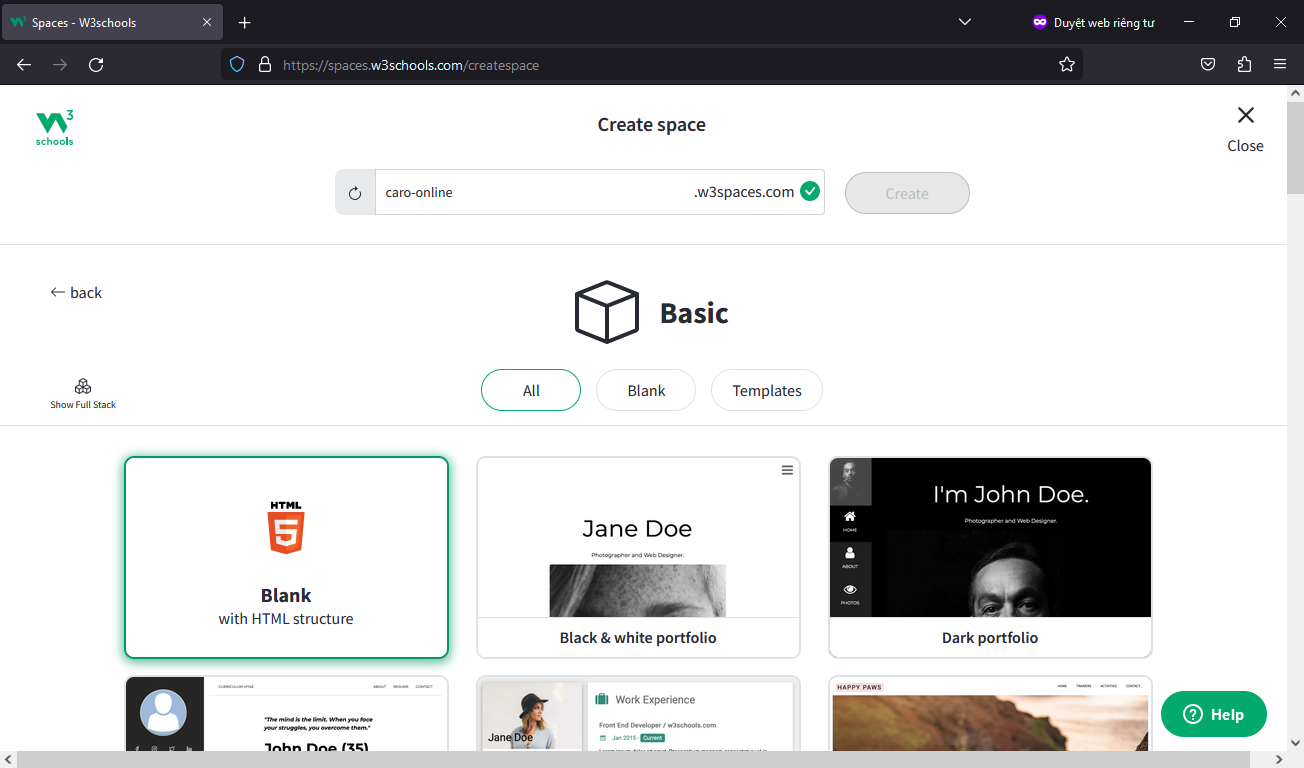
Vào Spaces, click “Get started for free”



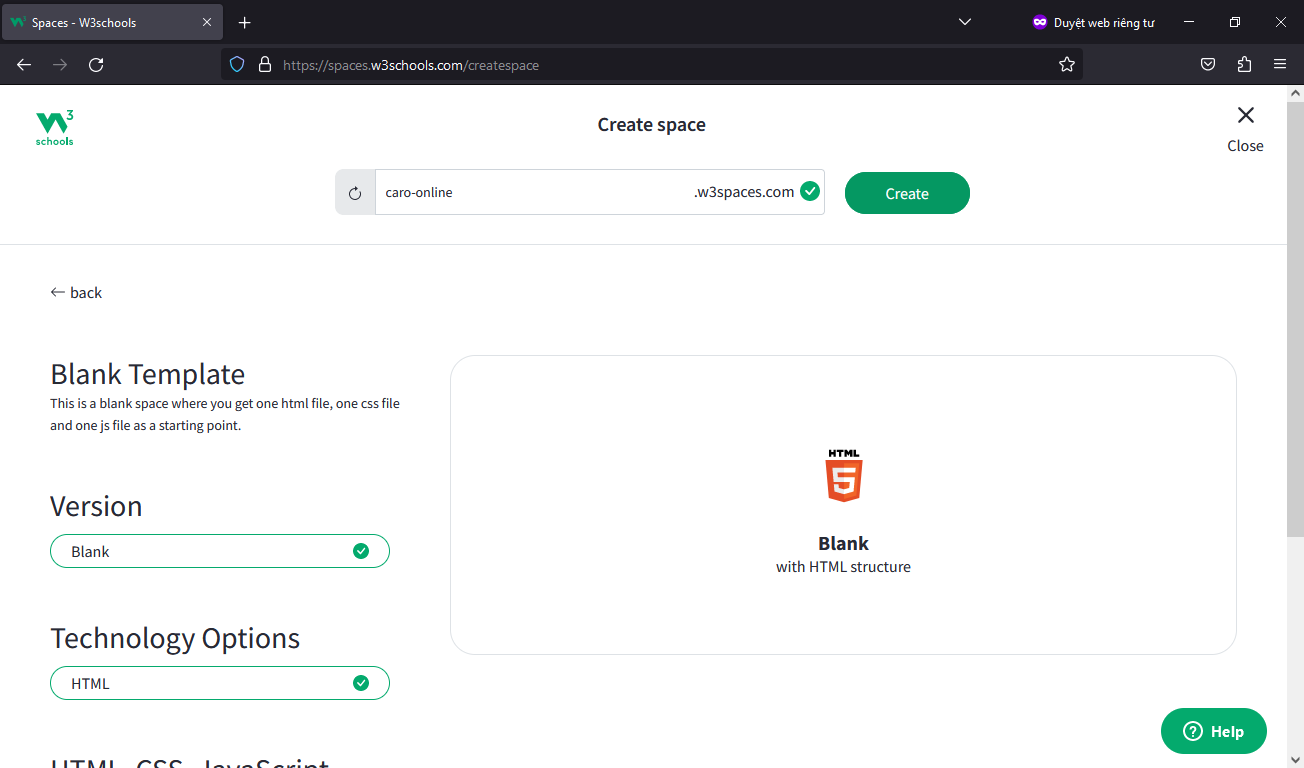
Đặt tên cho trang web, chọn “Browse Basic”



Chọn “Blank” (trang trống)

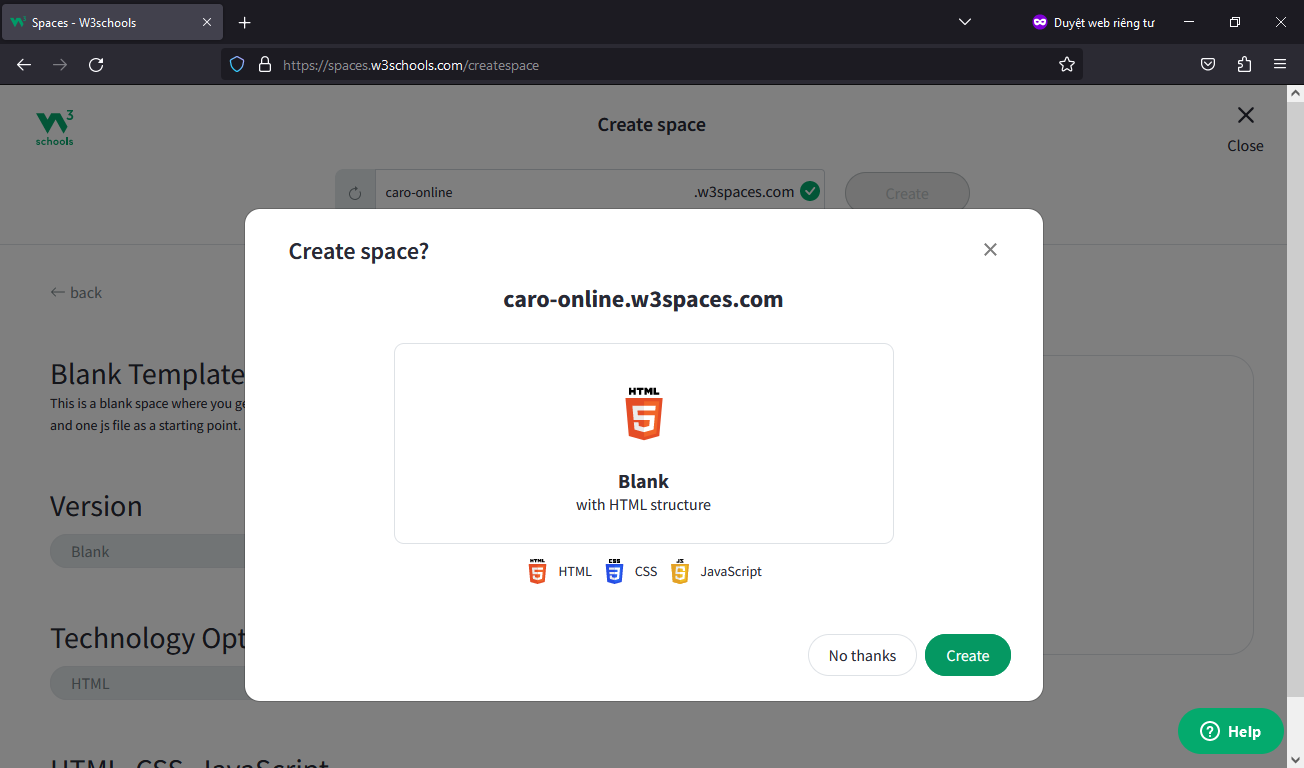


Click “Create” để tạo

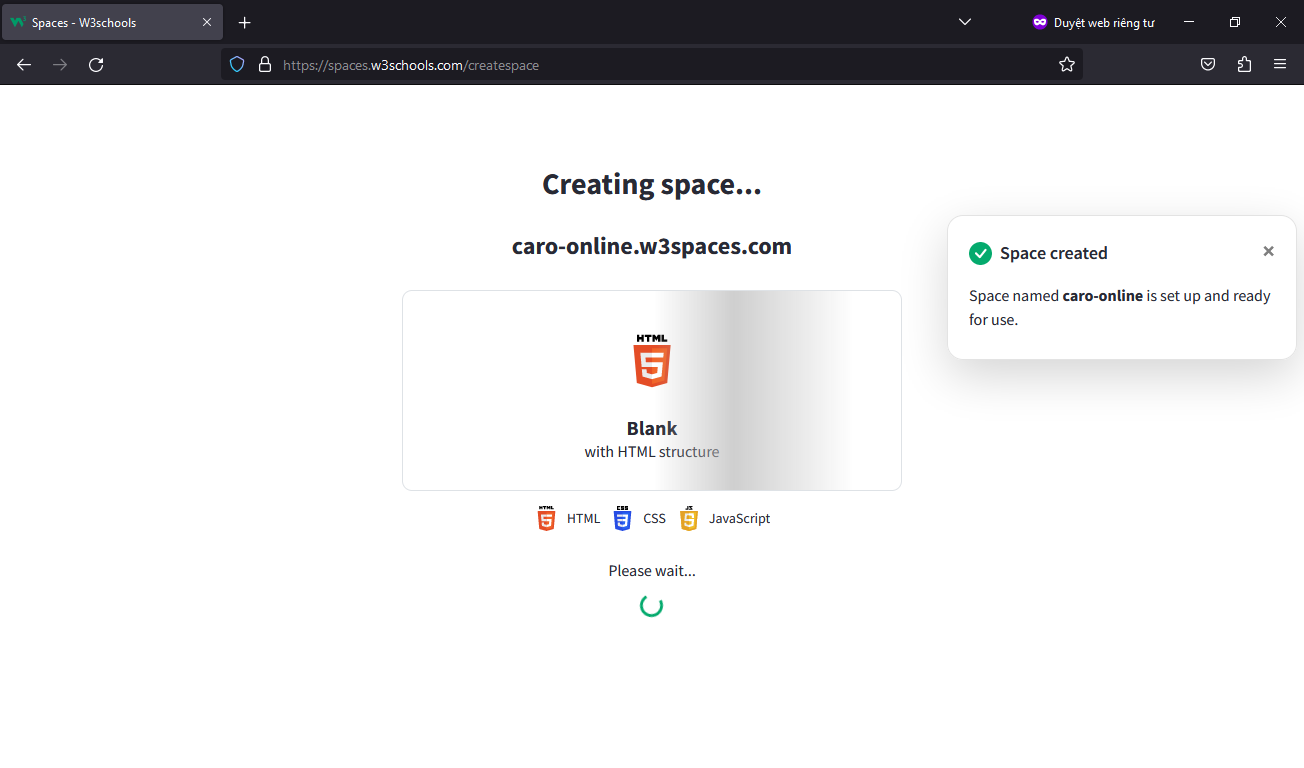


Nếu muốn tạo trang với tên đã chọn, click “Create”, nếu không, click “No

thanks” để chọn tên khác (tên đã chọn để tạo trang không thể thay đổi)

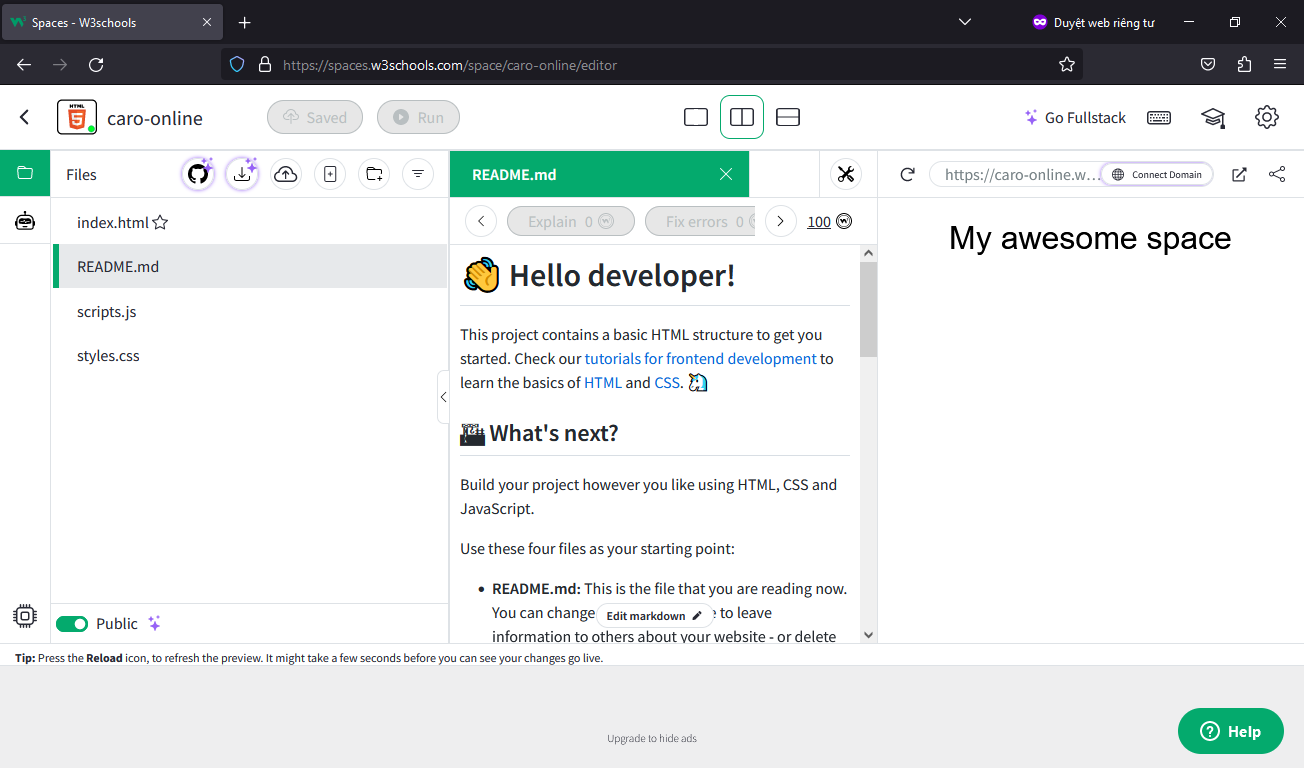


Tạo trang thành công

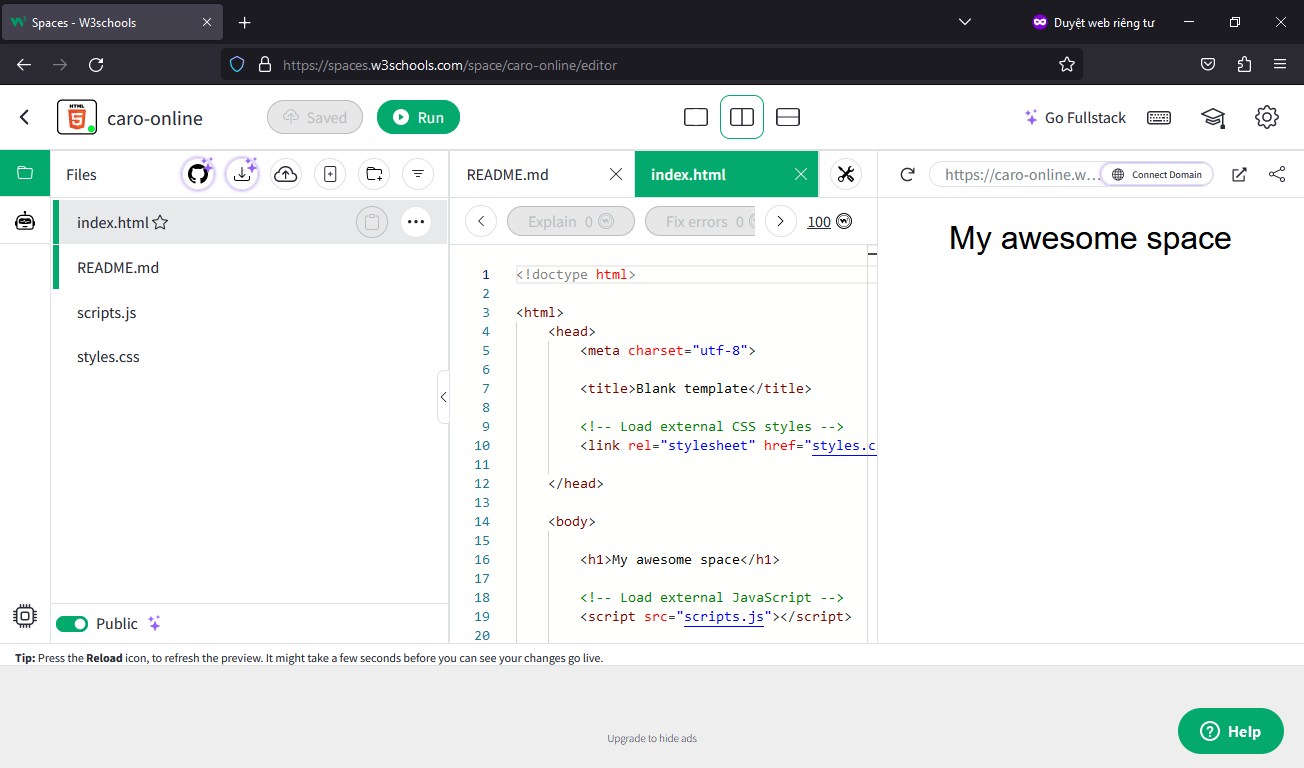


Giao diện lần đầu: bên trái là cây thư mục, ở giữa là trình soạn code,

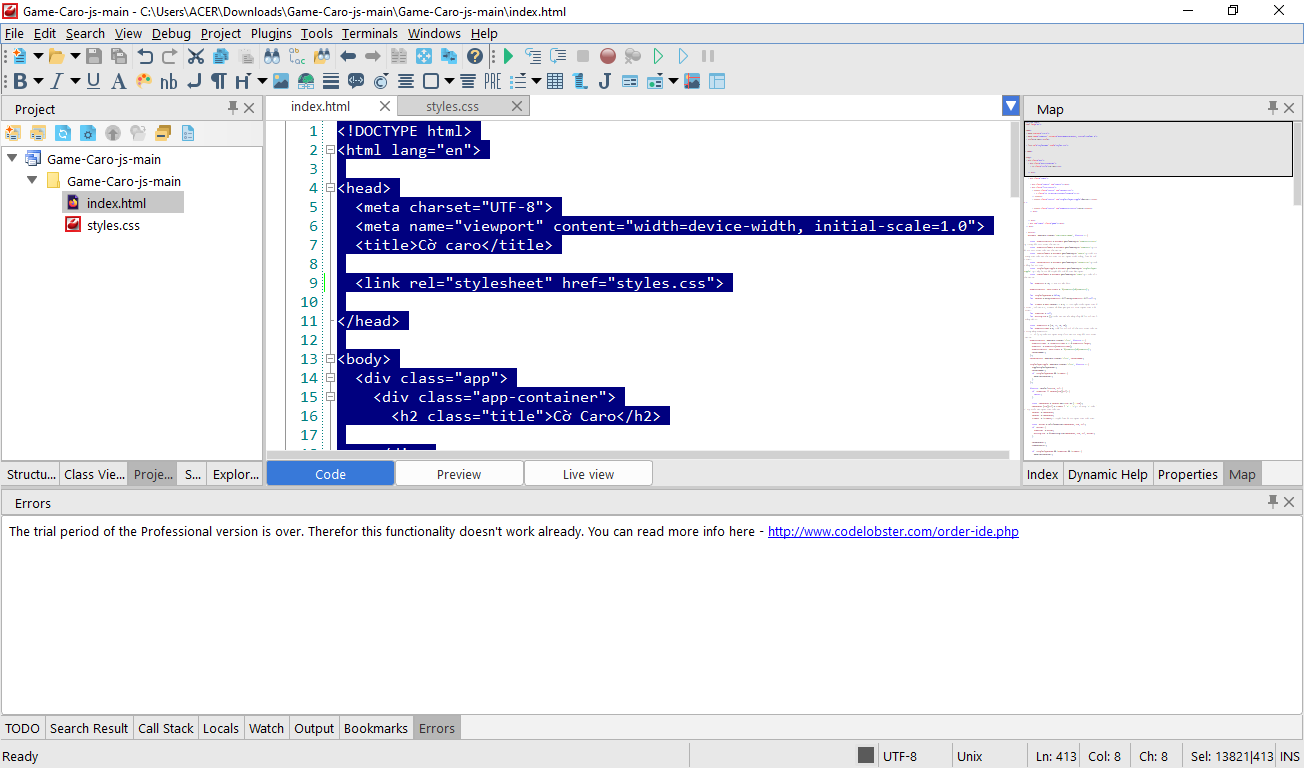
bên phải là xem trước trang web



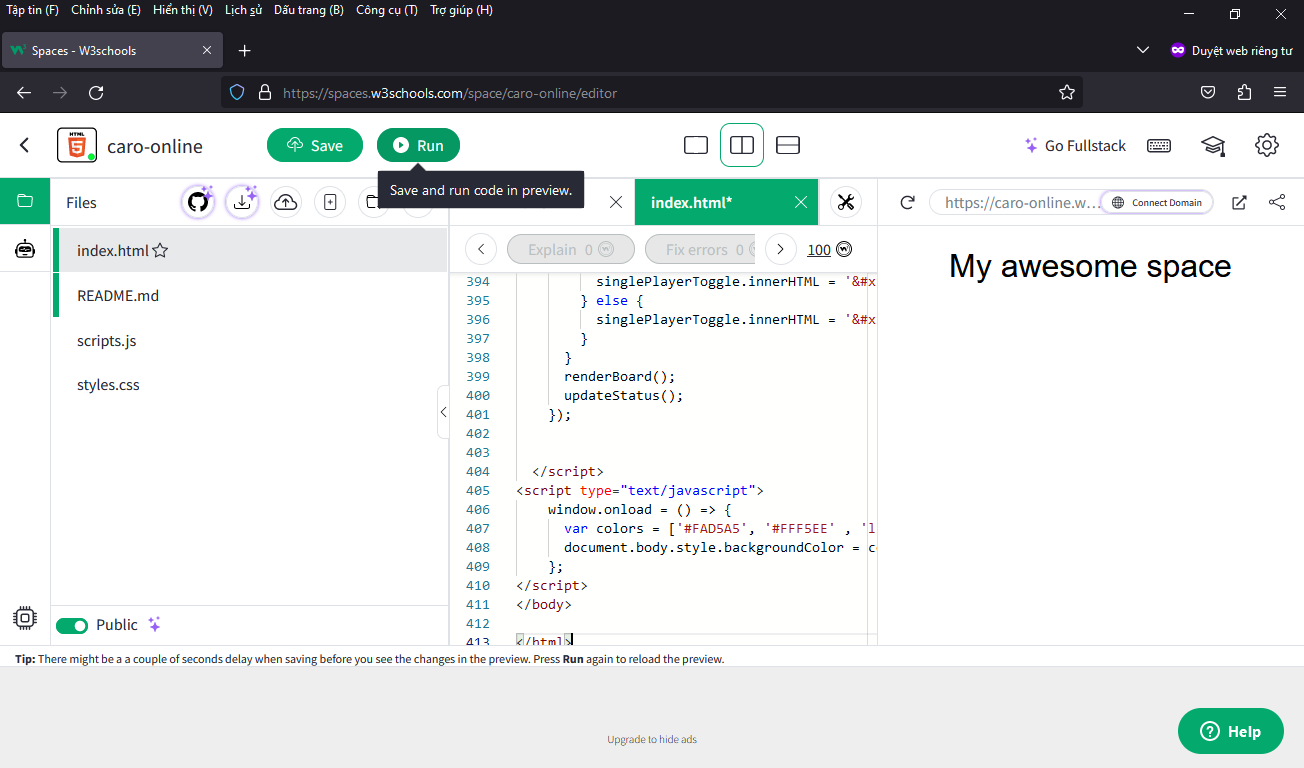
Chọn mở trang index.html



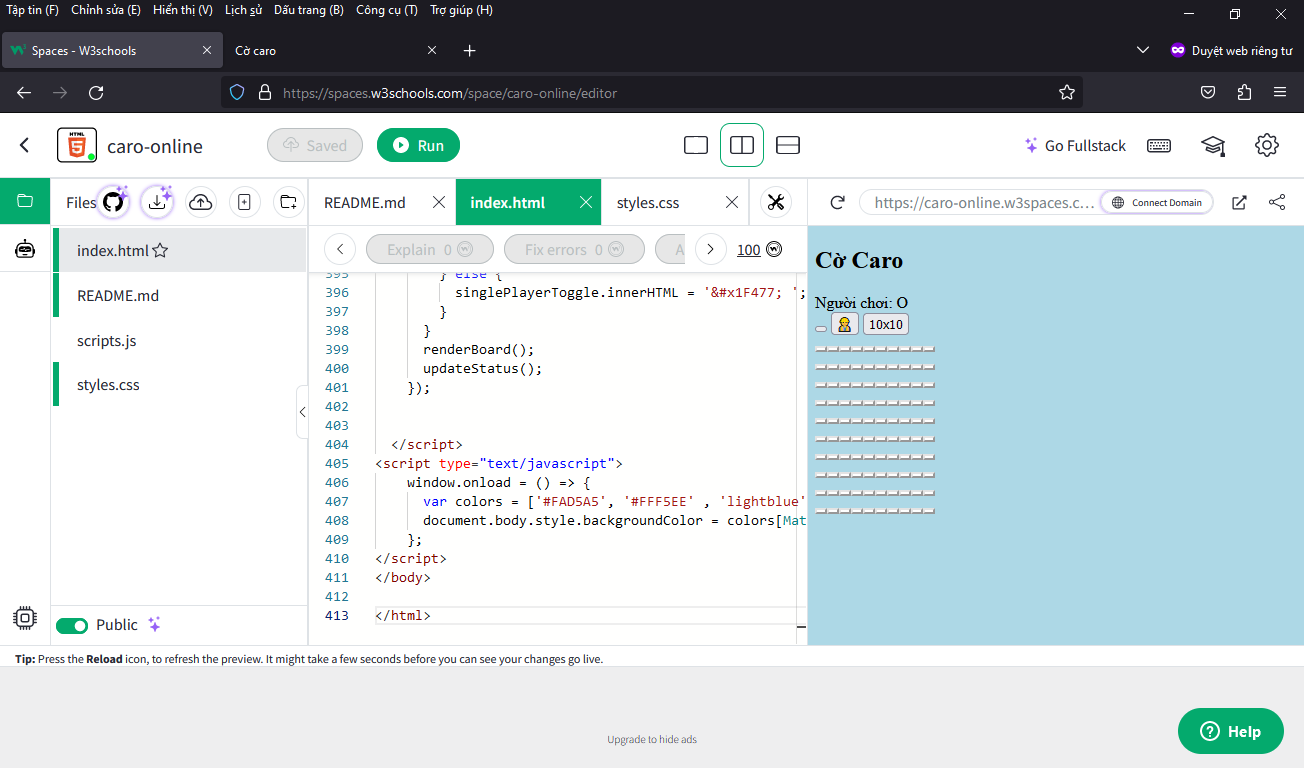
Copy code của web đã tạo trên máy tính



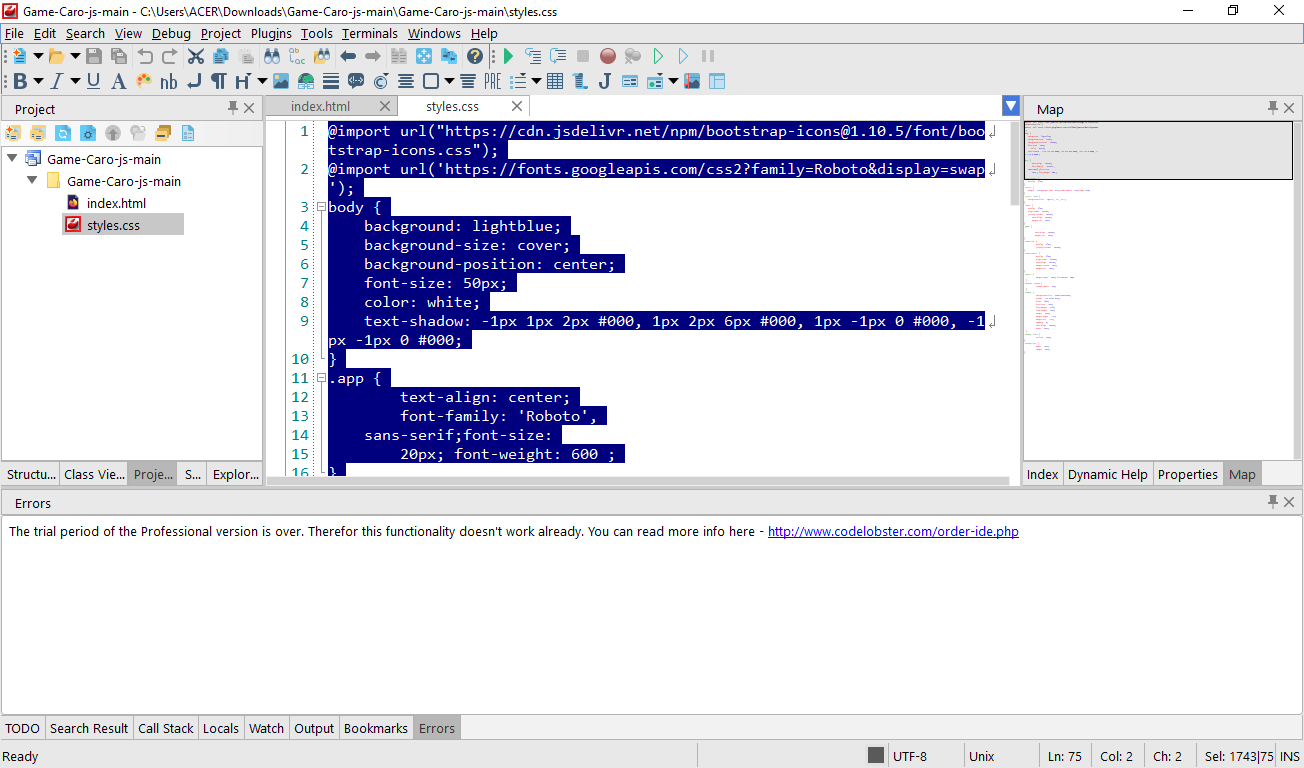
Dán vào file index.html của trang web vừa tạo



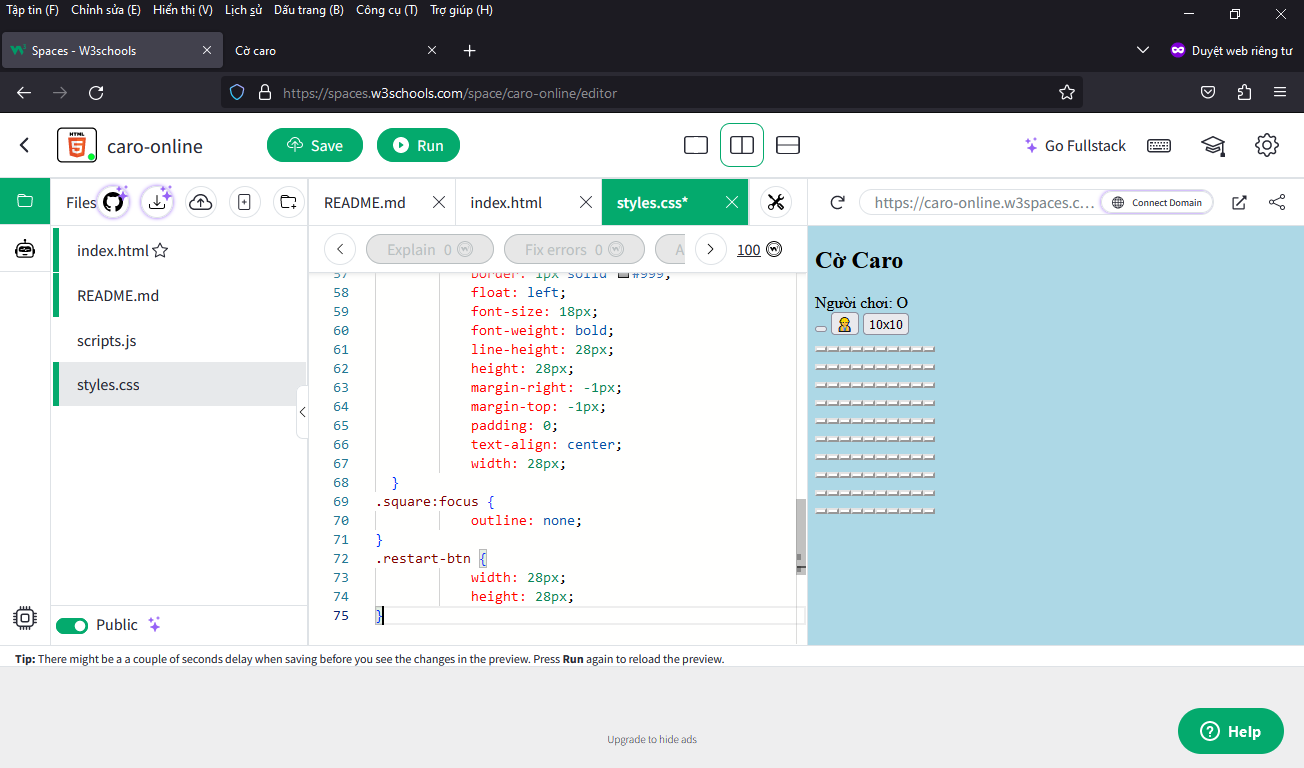
Click “Run” để lưu và hiển thị trang web bên phải



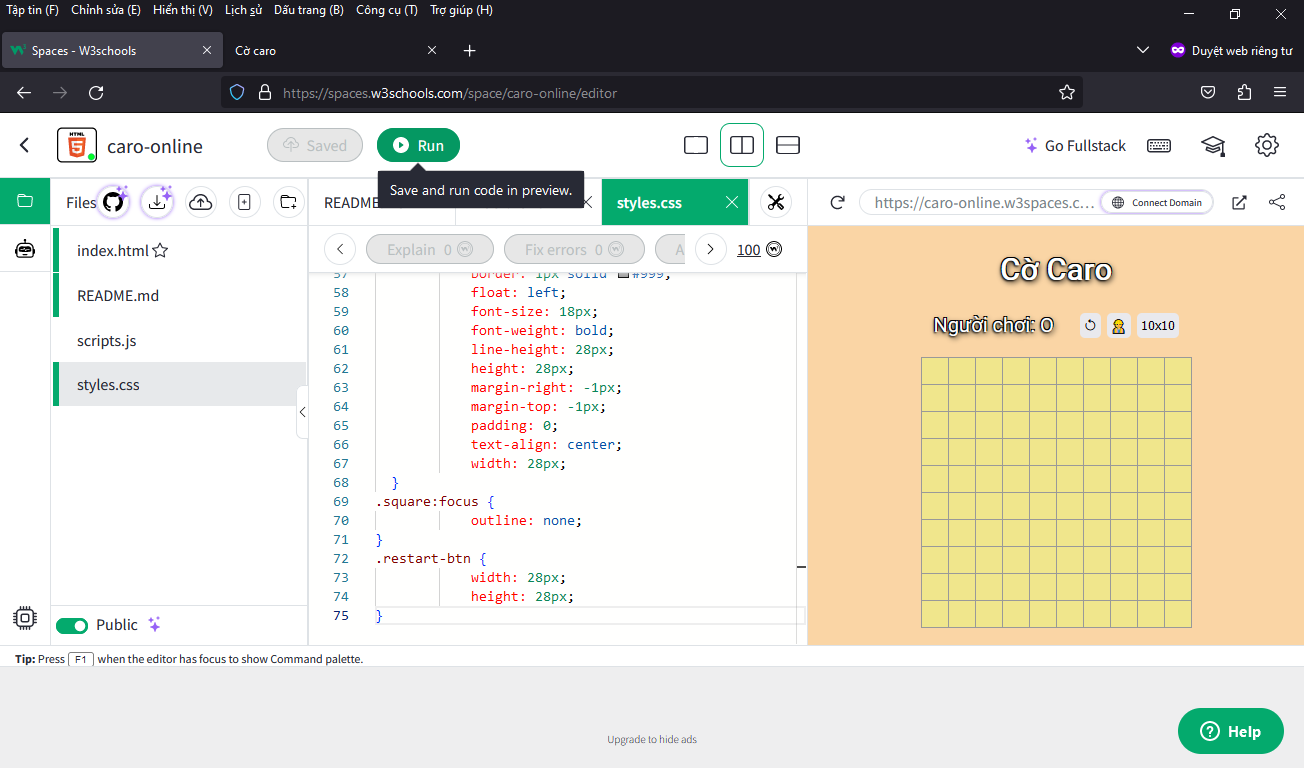
Copy code trong trang styles.css trên máy tính



Dán vào file styles.css của trang web



Click “Run” để lưu, trang web sau khi có css có giao diện hoàn chỉnh



### Kết quả

Truy cập địa chỉ: <https://caro-online.w3spaces.com/>



## TRIỂN KHAI WEB ĐỘNG

### Mục tiêu

Nâng cấp từ phiên bản 2 người chơi đấu với nhau trên cùng 1 thiết bị thành 2 người chơi đấu với nhau trên 2 thiết bị riêng biệt

### Kỹ thuật sử dụng

Frontend: HTML, CSS, JavaScript

Backend:

* Nền tảng: node.js
* Thư viện/framework: express, http, socket.io

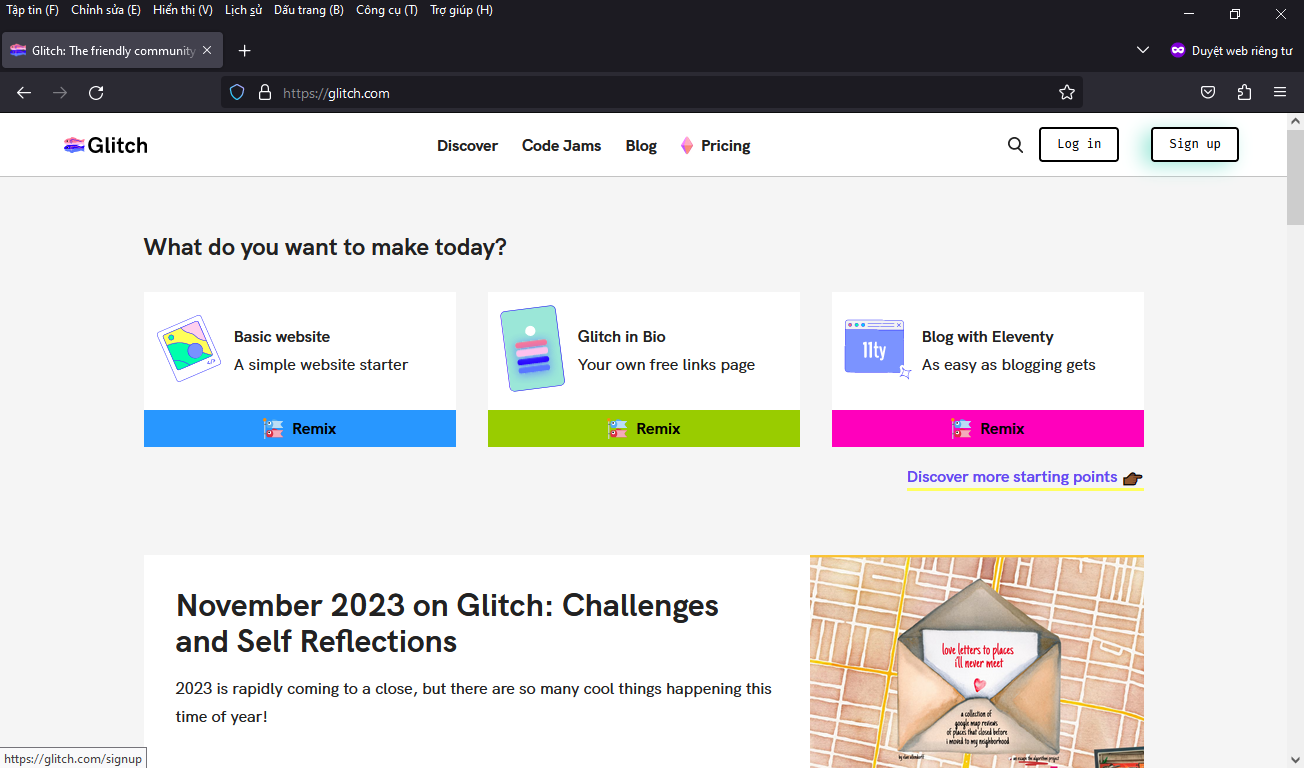
### Các bước triển khai

#### Chọn nhà cung cấp dịch vụ backend server

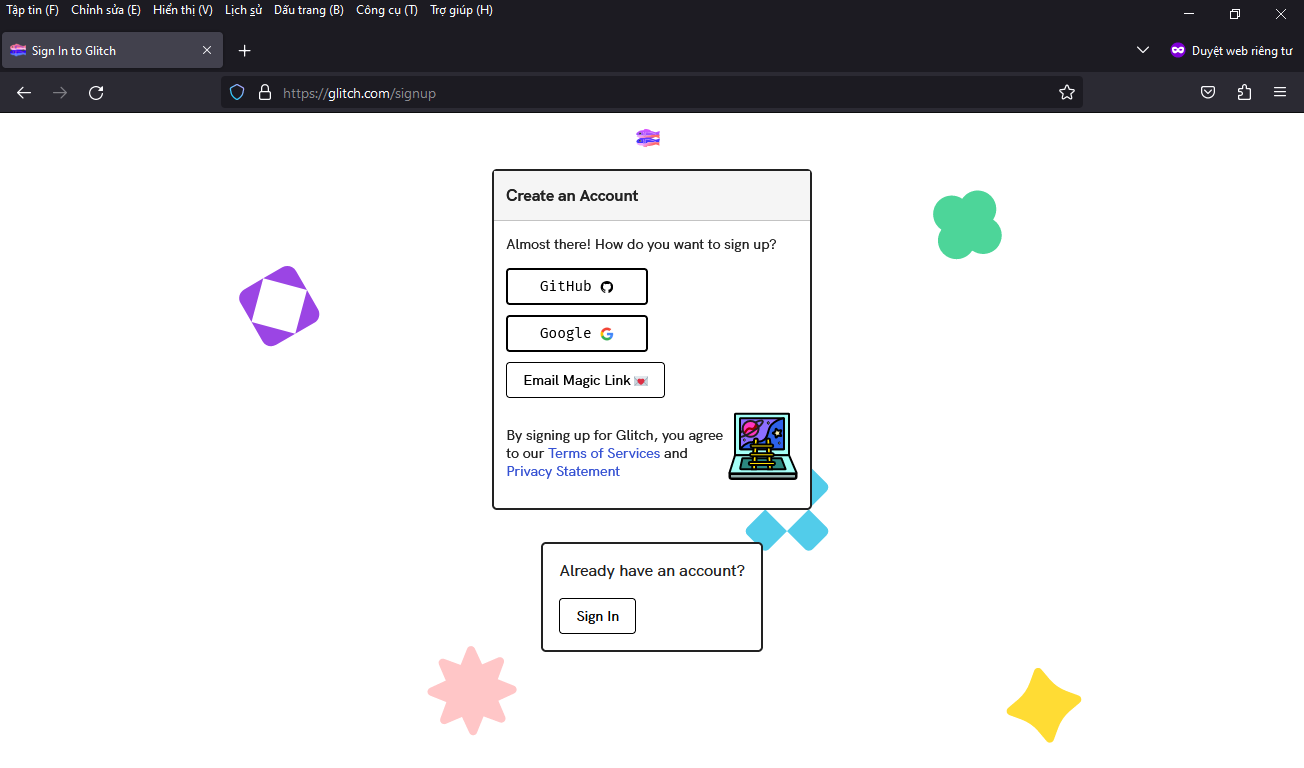
Chọn dịch vụ của <https://glitch.com> vì Glitch cho phép triển khai server node.js nhanh chóng và miễn phí 1000 giờ online mỗi tháng không giới hạn số project.

#### Các bước thực hiện

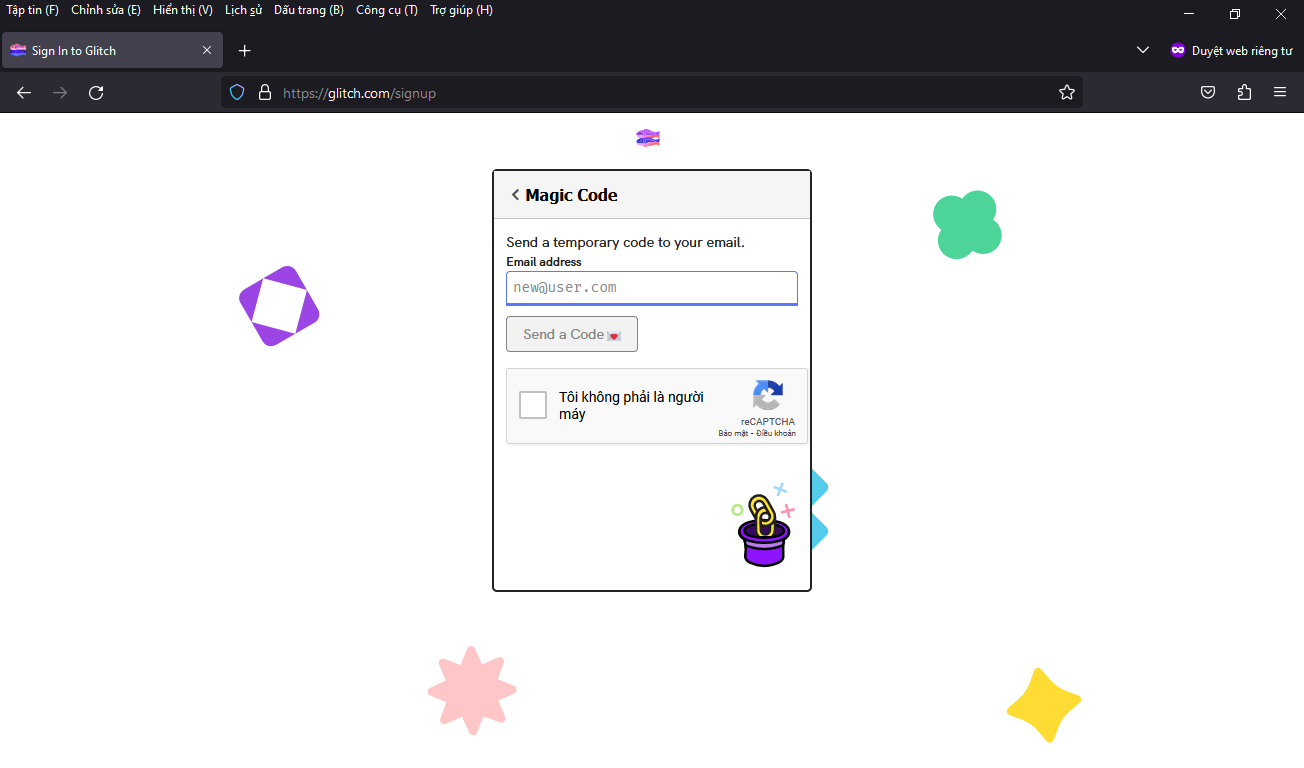
Trường hợp chưa có tài khoản: truy cập địa chỉ [https://glitch.com](https://glitch.com/), click nút “Sign up”



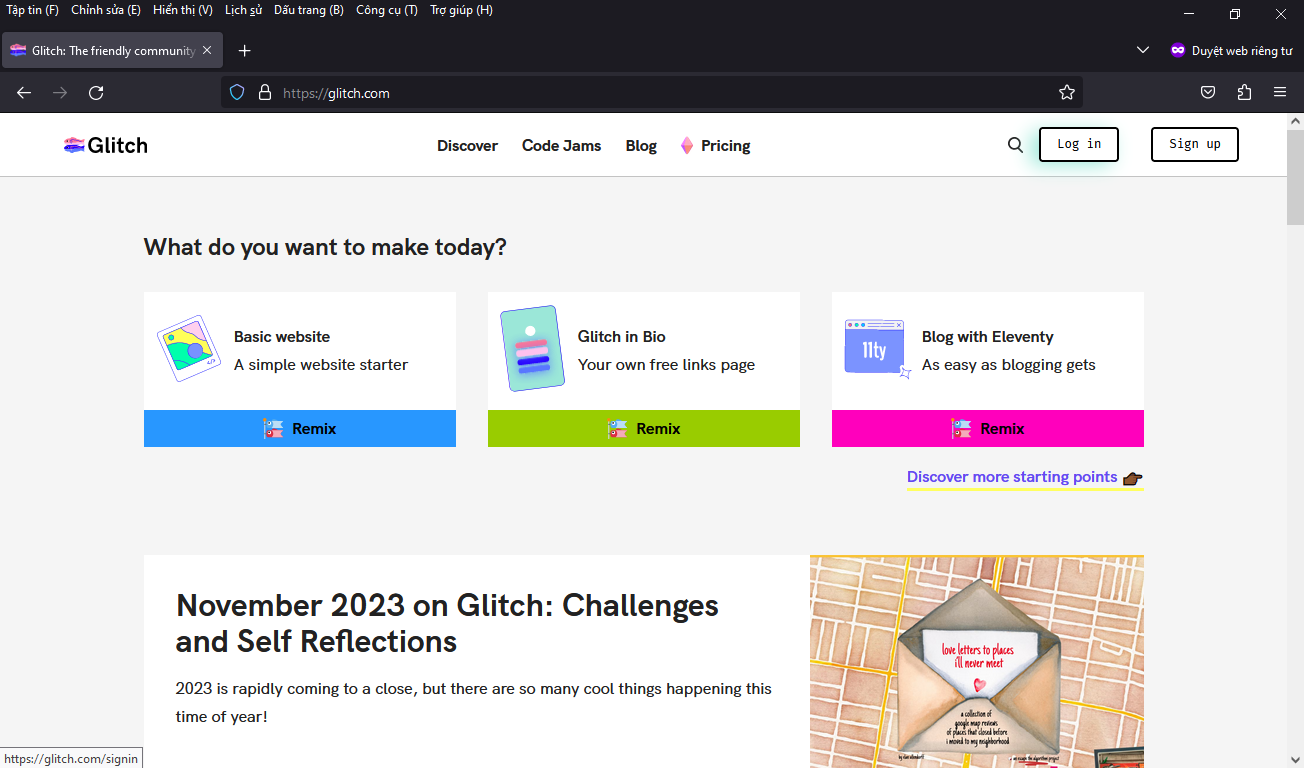
Nếu muốn tạo tài khoản bằng email, chọn “Email Magic Link”



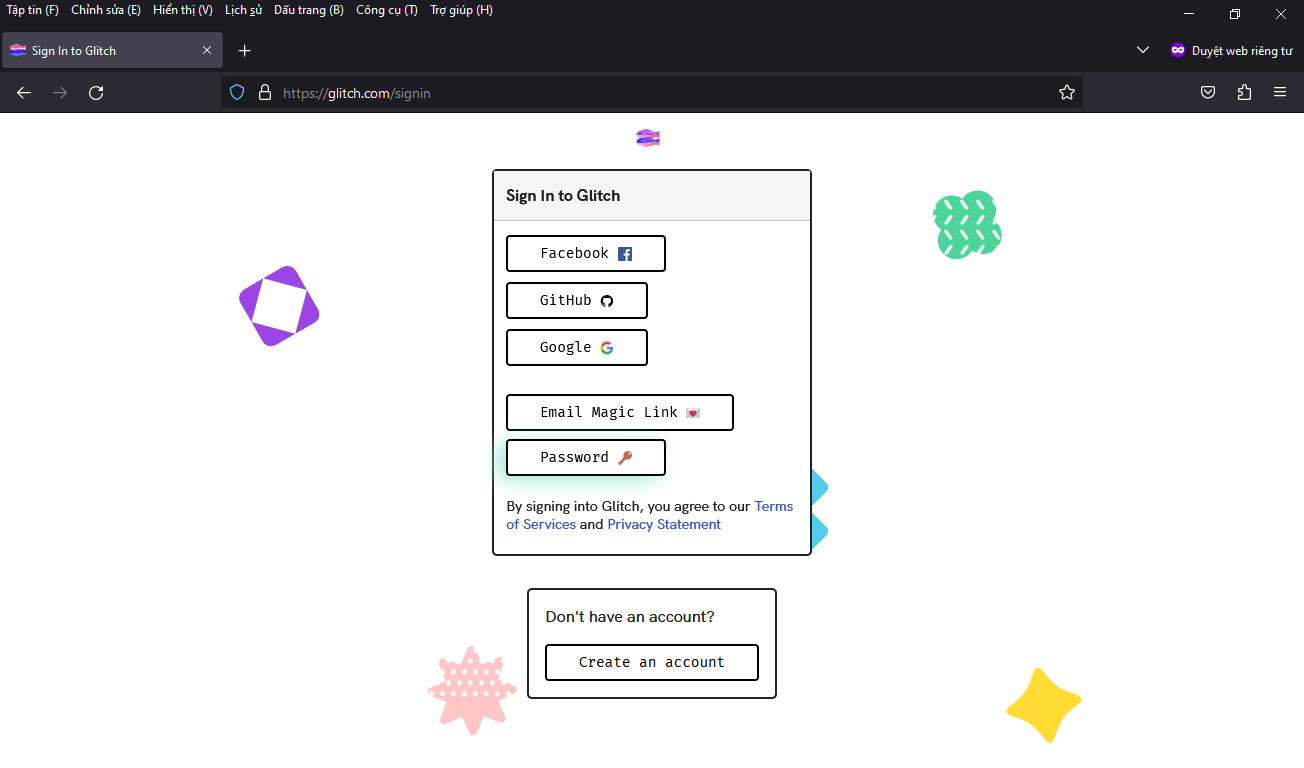
Điền địa chỉ email, click vào “Tôi không phải là người máy”, click nút “Send a Code”



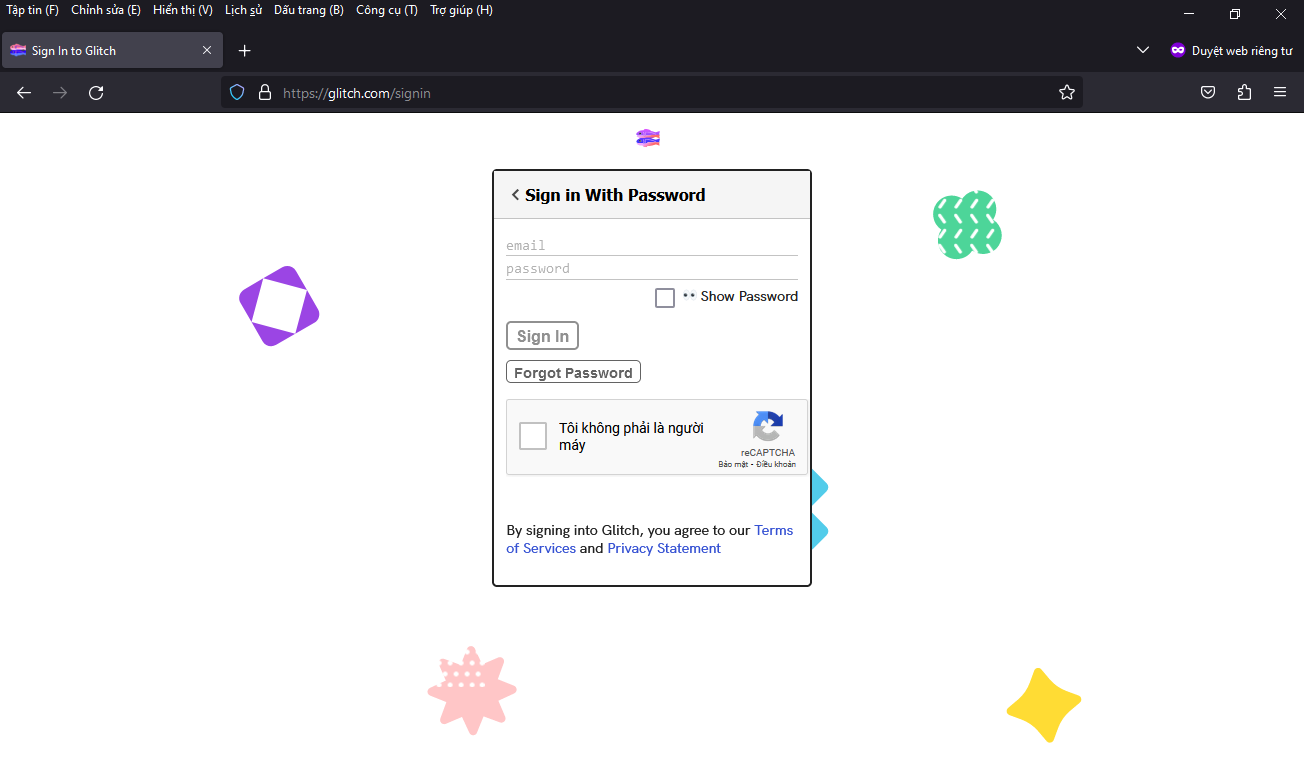
Trường hợp đã có tài khoản: truy cập địa chỉ <https://glitch.com>, click nút “Log in”



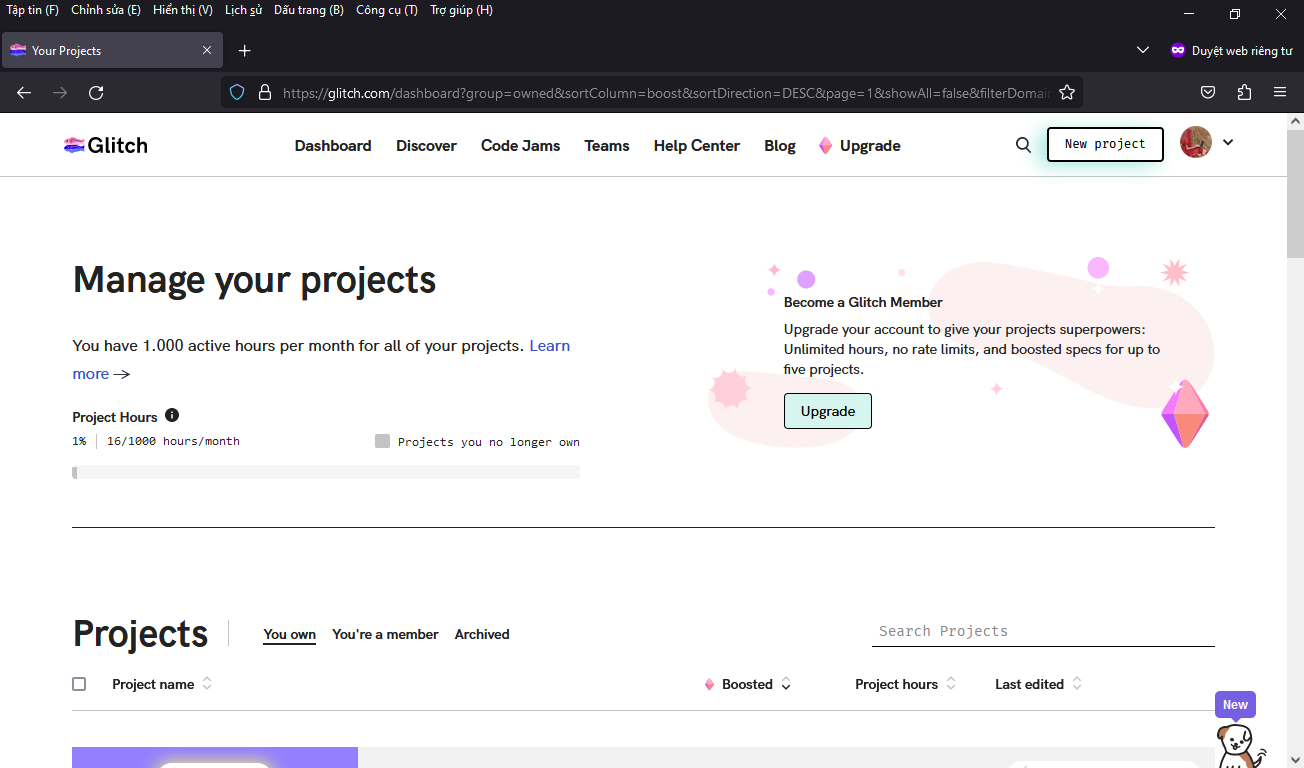
Click nút “Password”



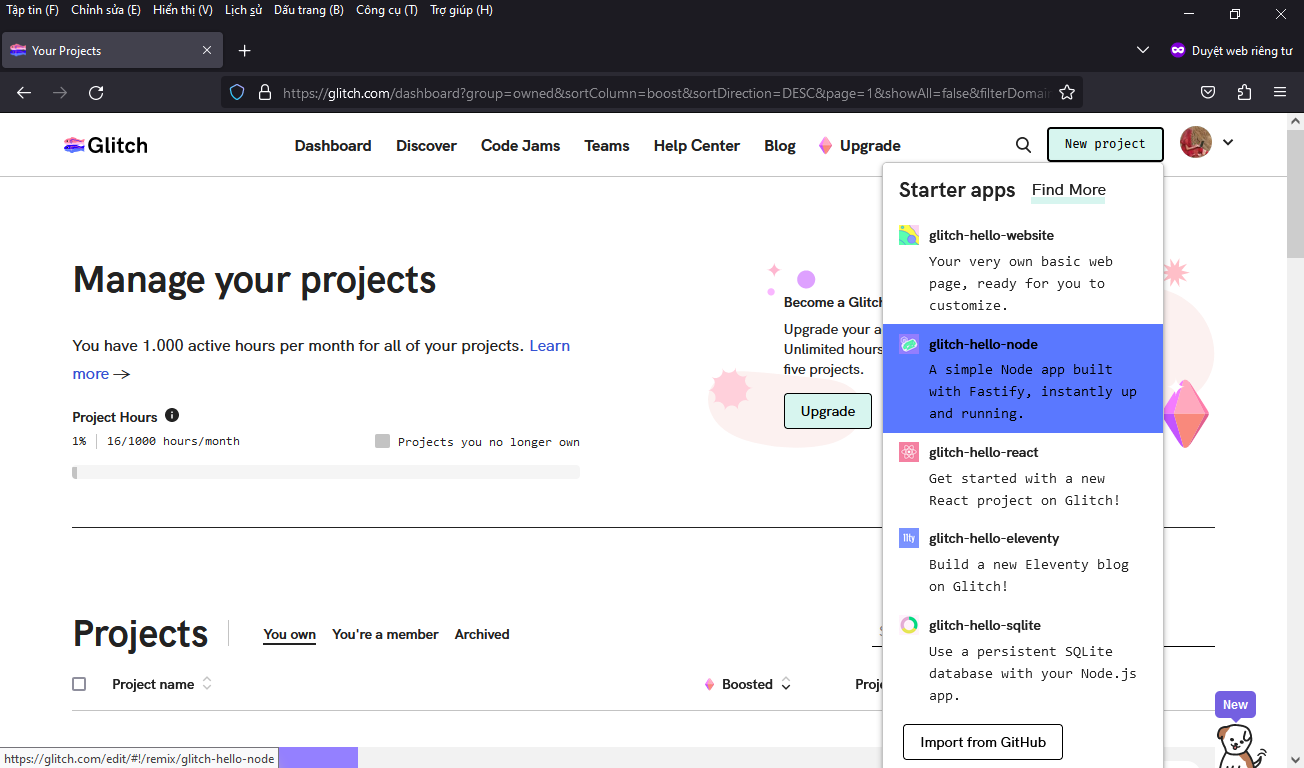
Nhập email, password, click vào “Tôi không phải là người máy”, click nút “Sign in”



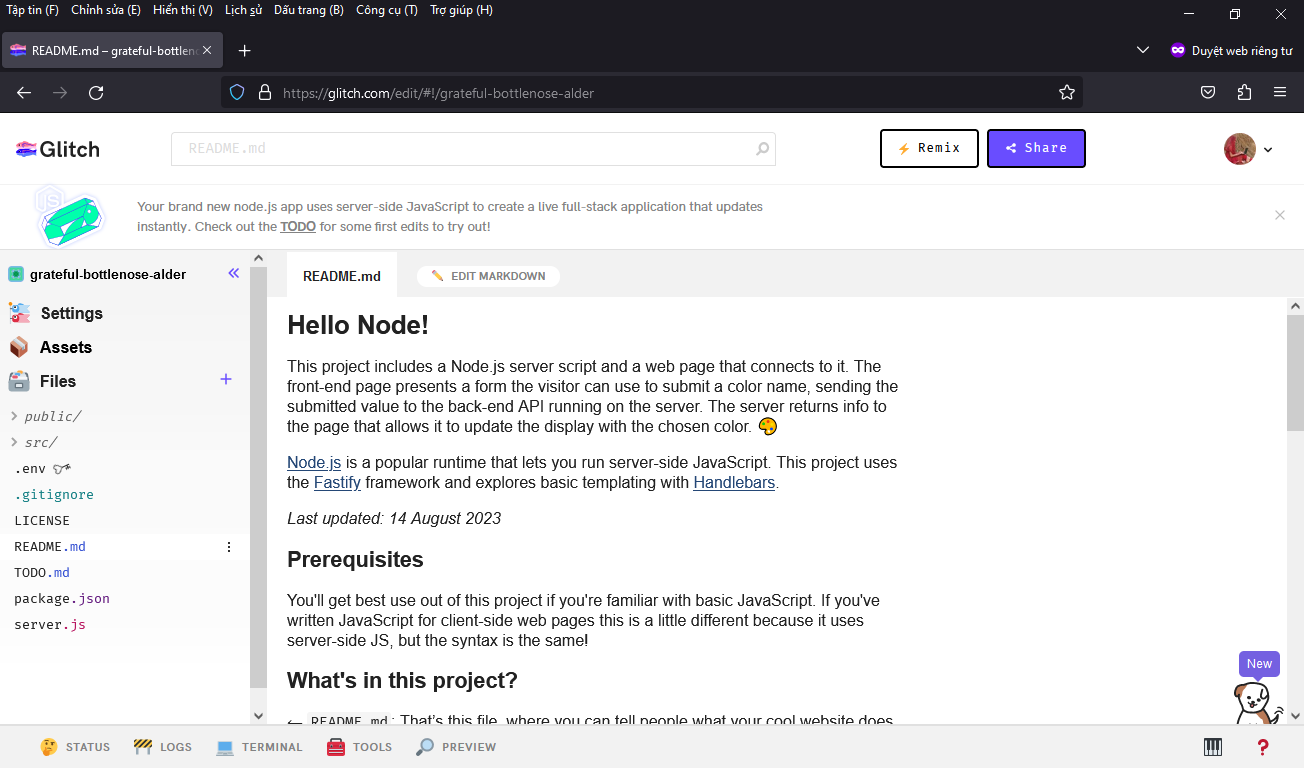
Click “New Project”



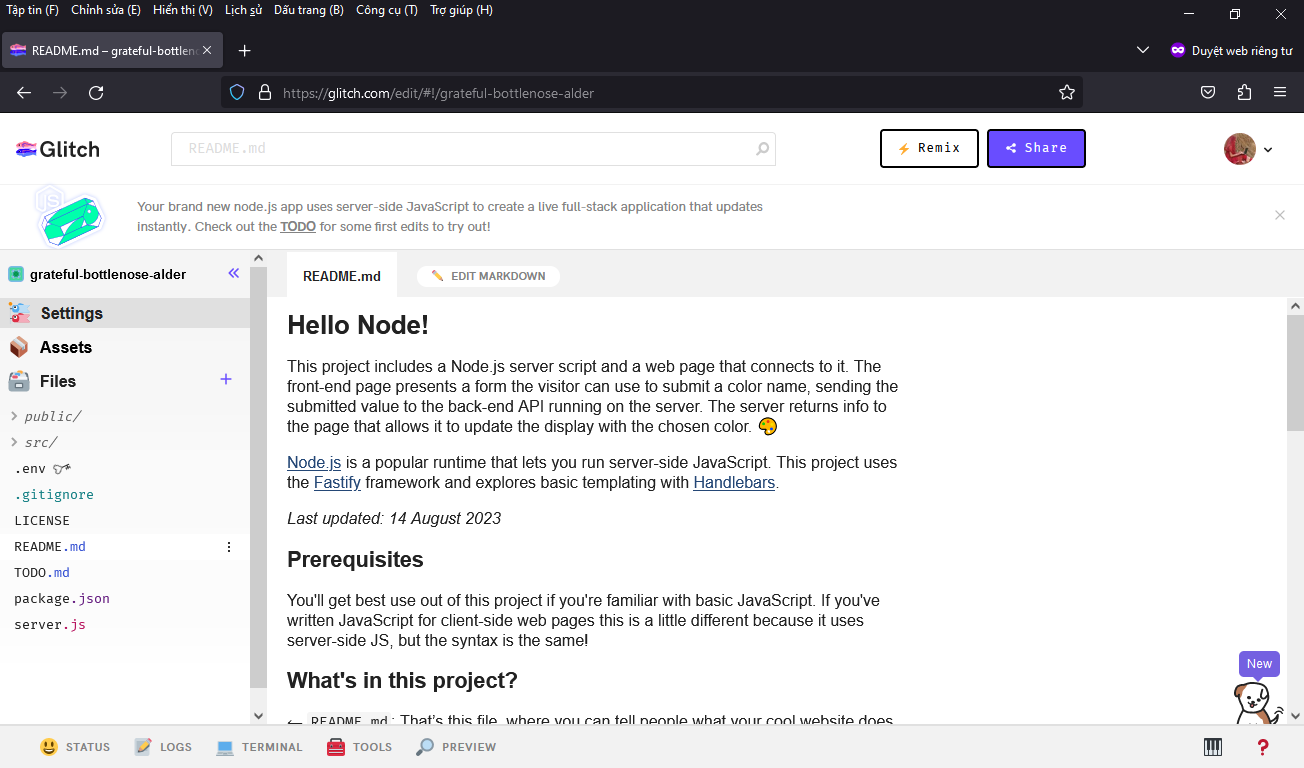
Chọn glitch-hello-node



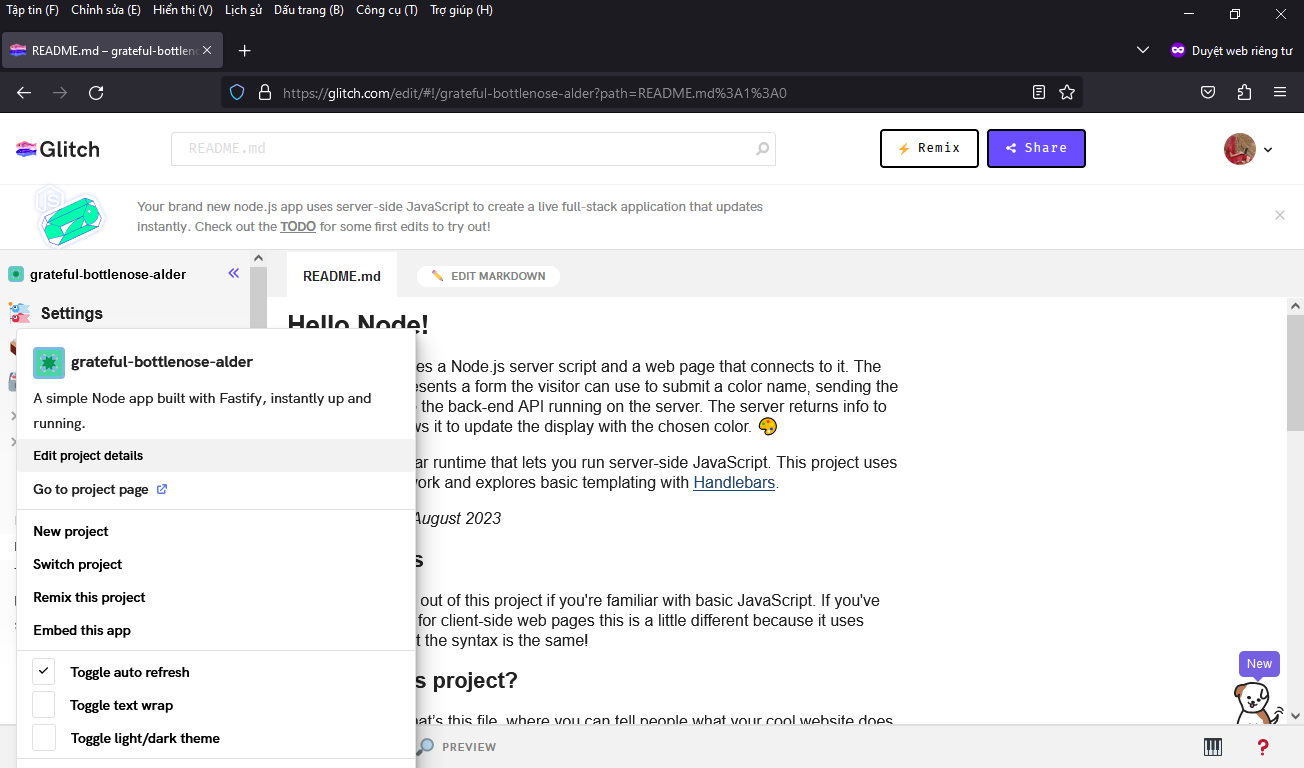
Glitch mặc định tạo project với tên ngẫu nhiên



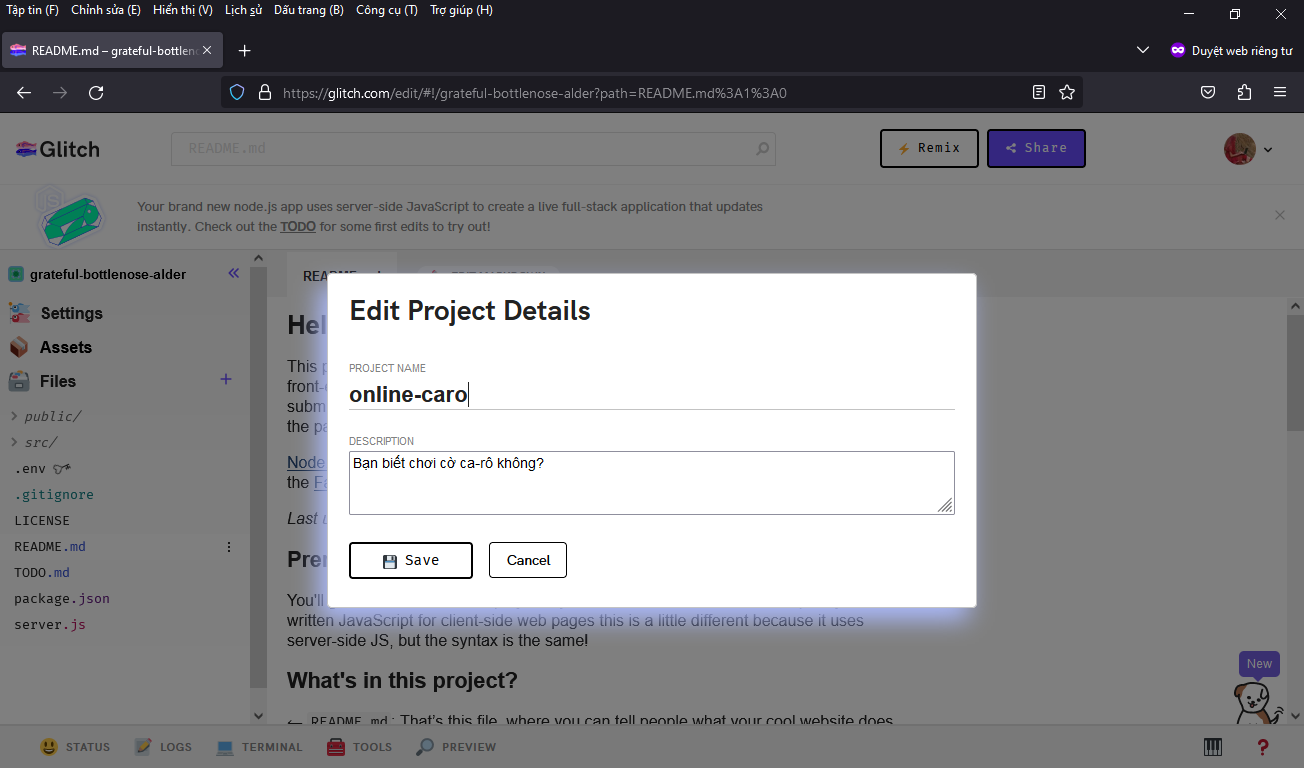
Click vào “Settings”



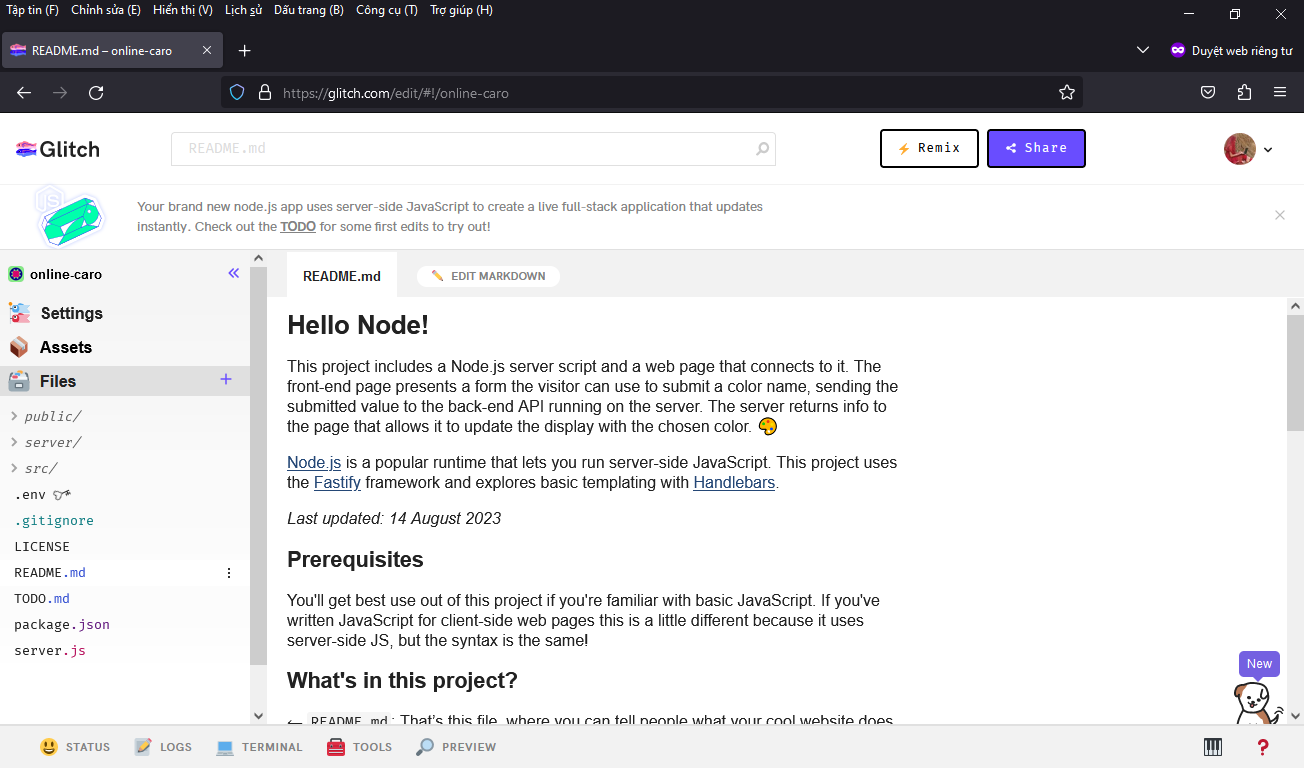
Click “Edit project details”



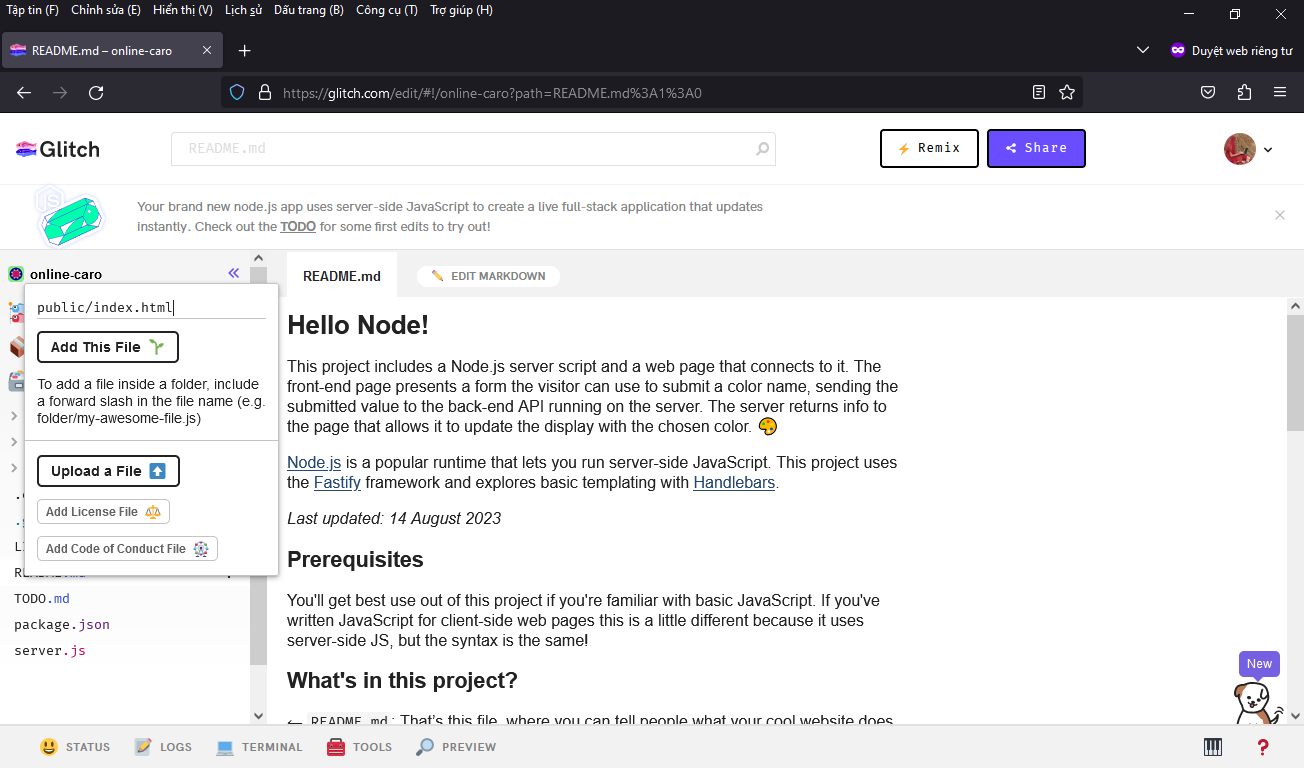
Đặt lại tên và mô tả project, click “Save”



Click vào dấu cộng bên phải “Files”



Vì file index.html sẽ nằm trong thư mục public, nhập vào “public/index.html”



Mở file package.json, thêm vào các dòng

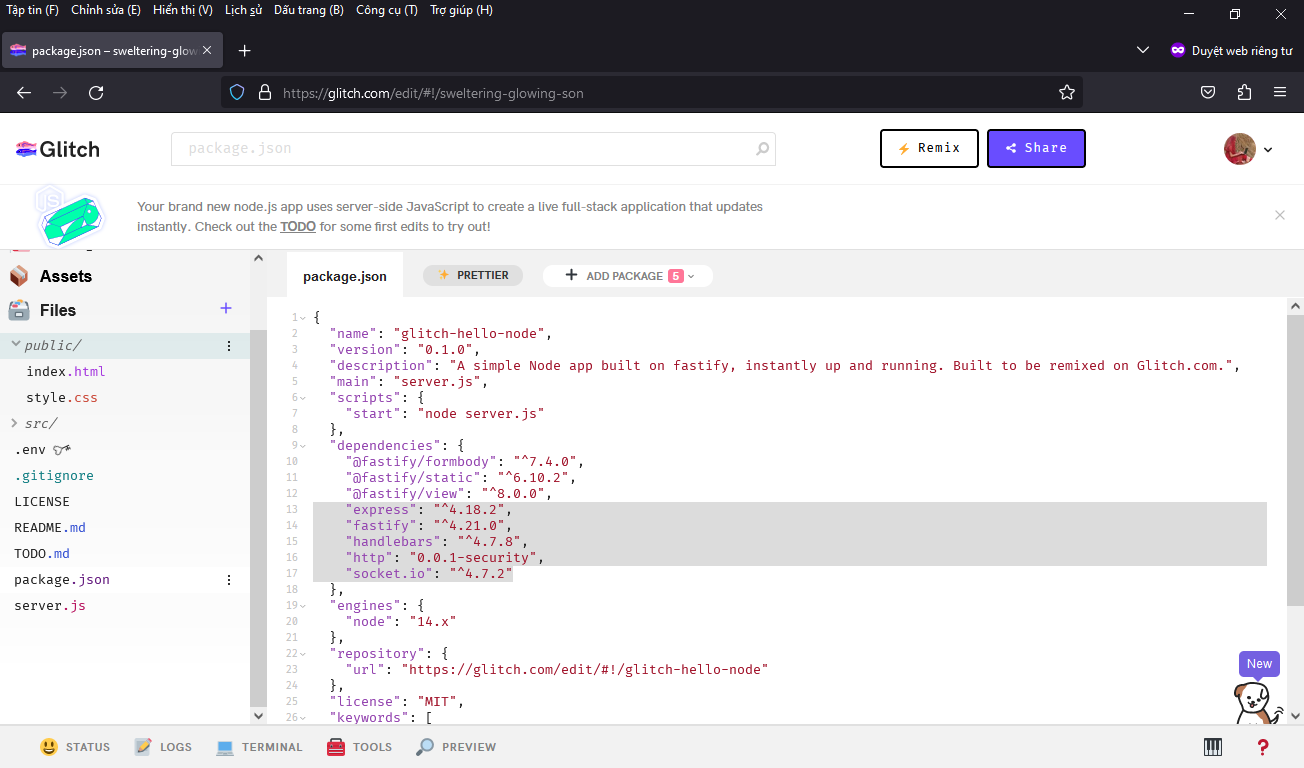
"express": "^4.18.2",

"fastify": "^4.21.0",

"handlebars": "^4.7.8",

"http": "0.0.1-security",

"socket.io": "^4.7.2"

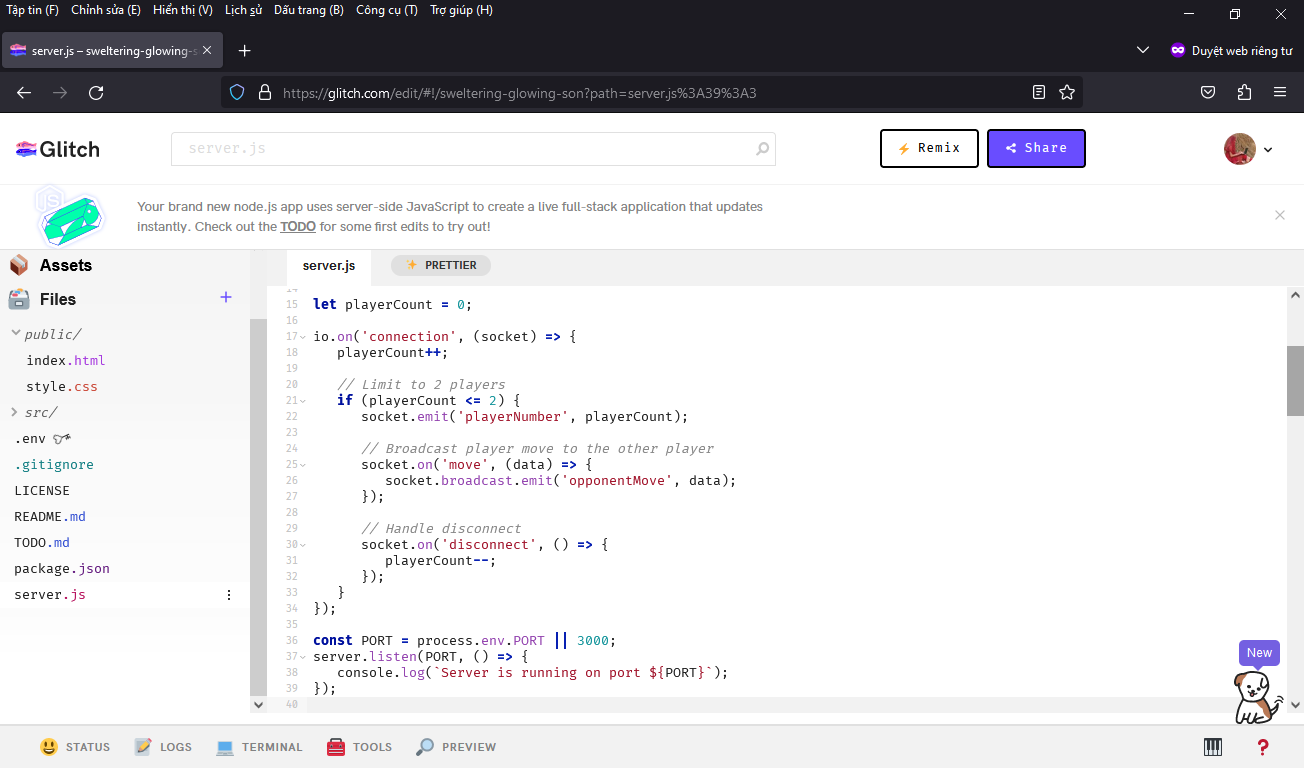


Mở terminal, gõ câu lệnh “npm install”

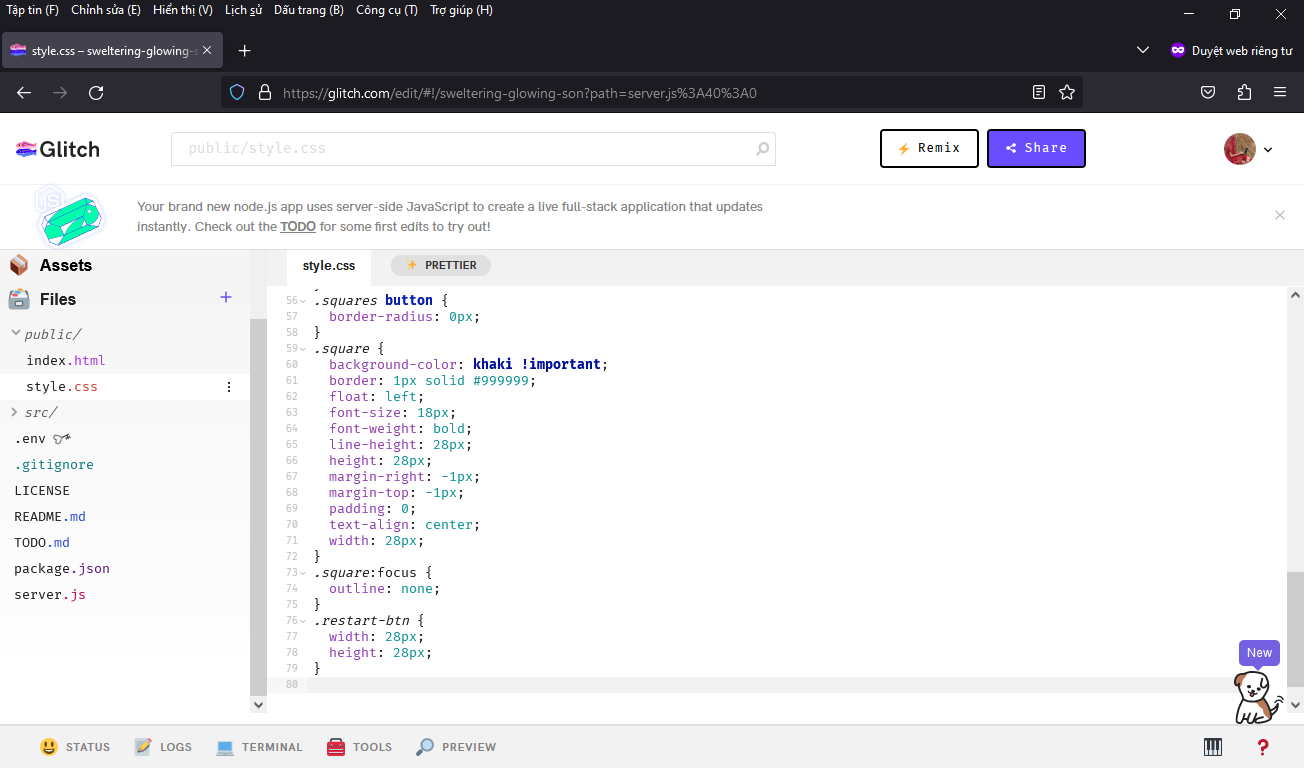


Mở file server.js, xóa code sẵn của Glitch, thay bằng đoạn code:

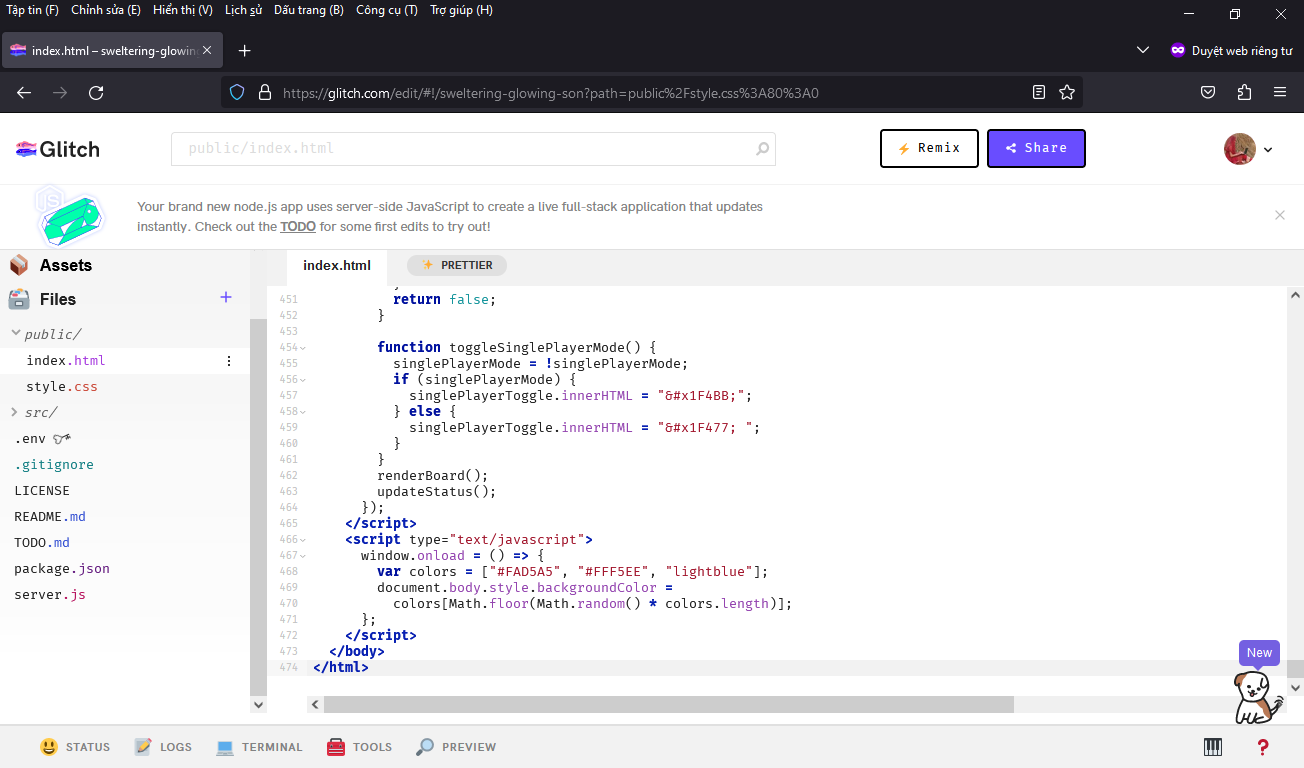
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | **const** express = require('express');  **const** http = require('http');  **const** socketIO = require('socket.io');  **const** app = express();  **const** server = http.createServer(app);  **const** io = socketIO(server);  app.use(express.**static**('public'));  app.get('/', (req, res) => {  res.sendFile(\_\_dirname + '/public/index.html');  });  **let** playerCount = 0;  io.on('connection', (socket) => {  playerCount++;  *// Limit to 2 players*  **if** (playerCount <= 2) {  socket.emit('playerNumber', playerCount);  *// Broadcast player move to the other player*  socket.on('move', (data) => {  socket.broadcast.emit('opponentMove', data);  });  *// Handle disconnect*  socket.on('disconnect', () => {  playerCount--;  });  }  });  **const** PORT = process.env.PORT || 3000;  server.listen(PORT, () => {  console.log(`Server is running on port ${PORT}`);  }); |



Copy code trong file style.css trên máy dán vào file style.css của project trên Glitch

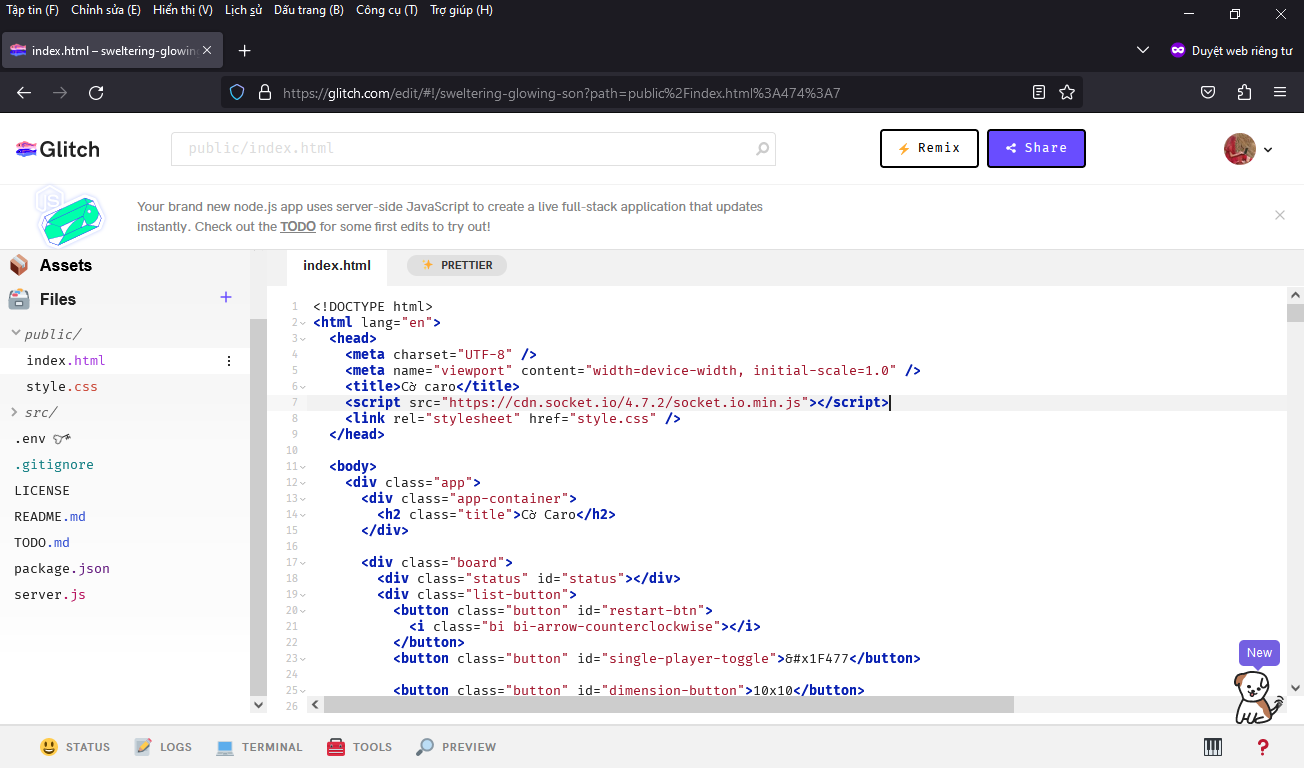


Copy code trong file index.html trên máy dán vào file index.html của project trên Glitch



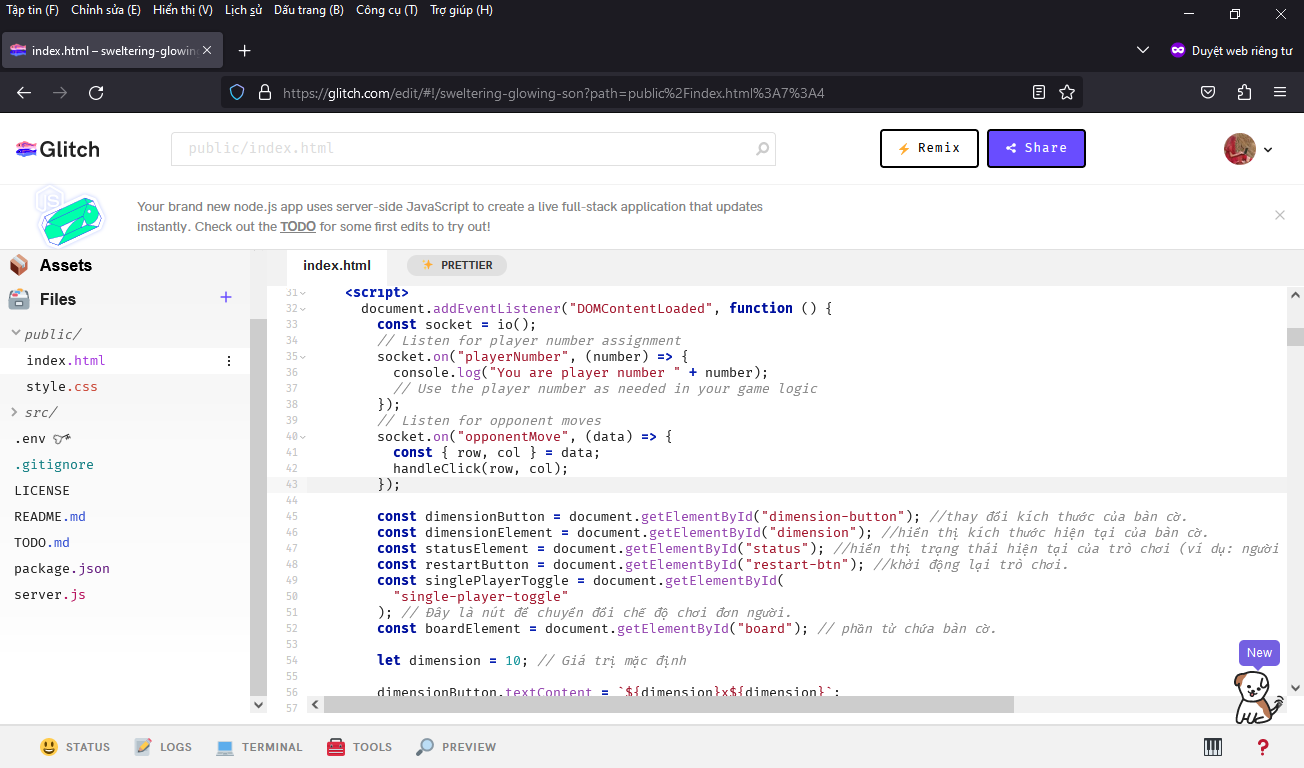
Thêm vào head của file index.html đoạn script khai báo thư viện/framework socket.io

|  |  |
| --- | --- |
| 1  2 | <script src="https://cdn.socket.io/4.7.2/socket.io.min.js"></script> |



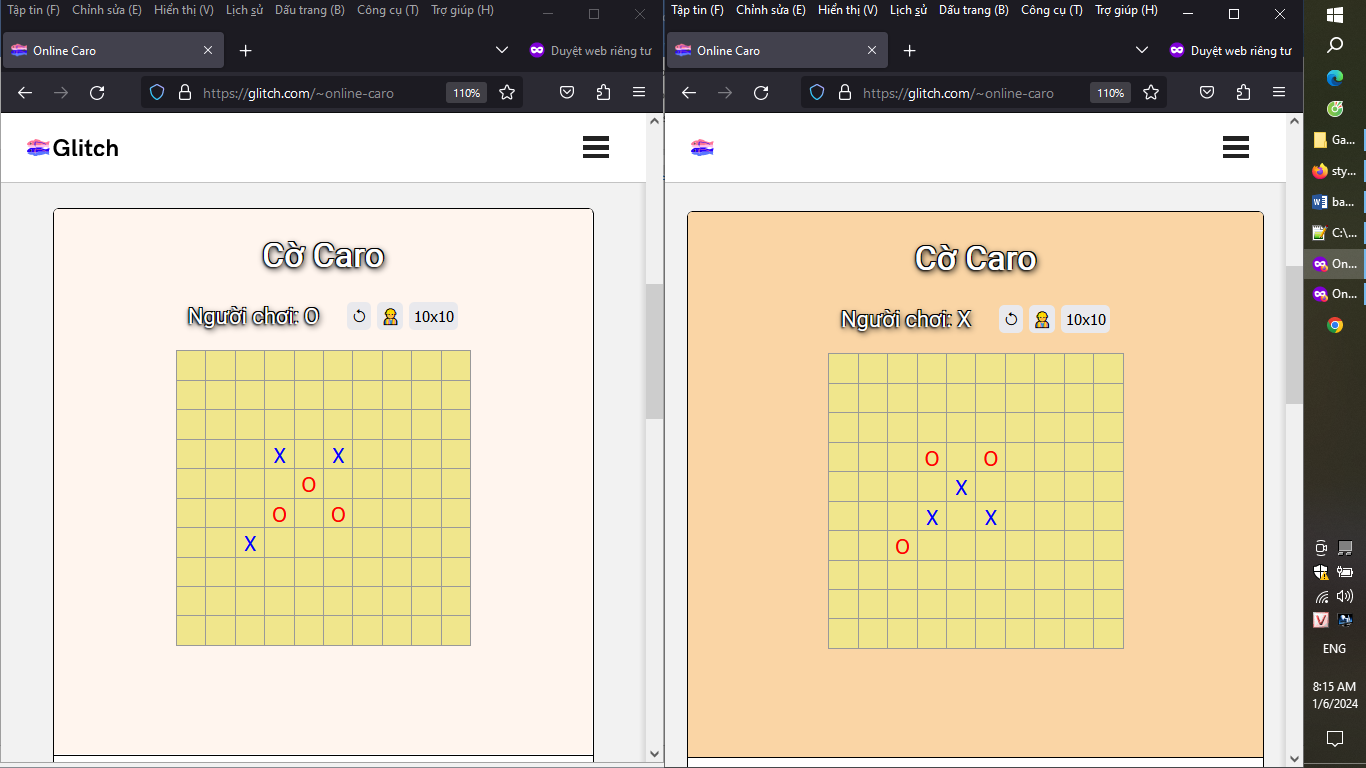
Thêm đoạn code khởi tạo socket vào trang index.html

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | **const** socket = io();  *// Listen for player number assignment*  socket.on("playerNumber", (number) => {  console.log("You are player number " + number);  *// Use the player number as needed in your game logic*  });  *// Listen for opponent moves*  socket.on("opponentMove", (data) => {  **const** { row, col } = data;  handleClick(row, col);  }); |



#### Kết quả

Truy cập địa chỉ: <https://glitch.com/~online-caro>



# TỔNG KẾT

## Về môn học

### Thành tựu:

* Hiểu được các thuật toán cơ bản của AI
* Hiểu được các logic nền tảng
* Biết thêm về ngôn ngữ Prolog
* Bắt tay vào project thực tế

### Điểm yếu:

* Chưa tự code được project hoàn chỉnh

## Về project

### Điểm hài lòng

* Hiểu cách thức hoạt động của thuật toán minimax, cắt tỉa alpha-beta
* Triển khai thành công trang web lên internet

### Điểm yếu

* Logic của code: cần cải thiện. Chức năng 2 người chơi đấu với nhau trên cùng 1 thiết bị hoạt động tốt nhưng khi đấu với nhau trên 2 thiết bị riêng biệt, đã xảy ra kết quả không như mong đợi
* Hàm đánh giá: chỉ ở mức tối thiểu nên chức năng đấu với máy tính chưa đạt yêu cầu

## Lời cảm ơn

Nhóm 4 chúng em xin gửi lời cảm ơn chân thành đến cô Nguyễn Thị Kim Phụng đã truyền đạt kiến thức nền tảng, từ giải thuật đến logic, trong môn học Trí tuệ nhân tạo. Việc này giúp ích rất nhiều cho chúng em trong việc hiểu và vận dụng lý thuyết vào project. Dù chỉ là copy code trên mạng nhưng nếu không có sự tận tâm và thúc đốc của cô, chúng em đã không thực sự bắt tay vào project và sẽ không đạt được tiến bộ sau môn học. Chúng em kính chúc cô và gia đình năm mới nhiều sức khỏe và luôn bình an.

# NGUỒN THAM KHẢO

<https://techmaster.vn/posts/37756/huong-dan-tao-game-co-caro-bang-javascript>

<https://viblo.asia/p/thuat-toan-minimax-ai-trong-game-APqzeaVVzVe>

<https://bard.google.com/chat/>

[**https://chat.openai.com/c/**](https://chat.openai.com/c/)

<https://gamevh.net/news/3449>

<https://tooabstractive.com/engineering-contents/advantages-and-disadvantages-of-artificial-intelligence/>

<https://viblo.asia/p/thuat-toan-minimax-ai-trong-game-APqzeaVVzVe>