



JONAS.IO  
SCHMEDTMANN

# Thử thách mã hóa cho tất cả các phần

Hoàn chỉnh  
Khóa học JavaScript



@jonasschmedtman

JS

# Mục lục

Hướng dẫn.....	4
Nguyên tắc cơ bản về JavaScript - Phần 1 .....	5
Thử thách viết mã số 1 .....	5
Thử thách viết mã số 2 .....	6
Thử thách viết mã số 3 .....	7
Thử thách viết mã số 4 .....	8
Nguyên tắc cơ bản về JavaScript - Phần 2 .....	9
Thử thách viết mã số 1 .....	9
Thử thách viết mã số 2 .....	10
Thử thách viết mã số 3 .....	11
Thử thách viết mã số 4 .....	12
Kỹ năng dành cho nhà phát triển & Cài đặt trình chỉnh sửa .....	13
Thử thách viết mã số 1 .....	13
JavaScript trong Trình duyệt: DOM và Sự kiện .....	14
Thử thách viết mã số 1 .....	14
Cấu trúc dữ liệu, toán tử hiện đại và chuỗi .....	15
Thử thách viết mã số 1 .....	15
Thử thách viết mã số 2 .....	17
Thử thách viết mã số 3 .....	18
Thử thách viết mã số 4 .....	19
Xem xét kỹ hơn về các chức năng.....	20
Thử thách viết mã số 1 .....	20
Thử thách viết mã số 2 .....	22

Làm việc với mảng.....	23
Thử thách viết mã số 1 .....	23
Thử thách viết mã số 2 .....	24
Thử thách viết mã số 3 .....	24
Thử thách viết mã số 4 .....	25
 Lập trình hướng đối tượng (OOP) .....	27
Thử thách viết mã số 1 .....	27
Thử thách viết mã số 2 .....	28
Thử thách viết mã số 3 .....	28
Thử thách viết mã số 4 .....	29
 JavaScript không đồng bộ .....	30
Thử thách viết mã số 1 .....	30
Thử thách viết mã số 2 .....	32
Thử thách viết mã số 3 .....	33

## Hướng dẫn

§ Khóa học JavaScript hoàn chỉnh bao gồm hơn 25 thử thách mã hóa để bạn tự mình thực hành các khái niệm chúng ta học được trong mỗi phần. Đây là một phần cơ bản trong hành trình học tập của bạn 🧐

§ Tài liệu này bao gồm tất cả các thử thách lập trình trong khóa học này, để bạn có thể đọc văn bản thử thách mà không cần xem video (Tôi vẫn khuyên bạn nên xem video vì tôi có thể cung cấp thêm một số manh mối)

§ Một số thử thách yêu cầu một số mã bắt đầu, bạn có thể lấy mã này từ tài liệu này thay vì sao chép từ video

§ Cách giải của mỗi thử thách đều có trong video nên các bạn cứ yên tâm nhé đi xem chúng 😊

§ Một số thử thách sau này có mục đích khá khó khăn (dù sao thì chúng cũng được gọi là thử thách phải không? ). Nếu 🤔 một trong số chúng quá khó, đừng lo lắng, điều này là hoàn toàn bình thường! Chỉ cần bắt đầu xem giải pháp, sau đó tạm dừng video và tiếp tục tự làm việc. Đấu tranh là một phần của cuộc hành trình, nhưng bạn có thể làm được 🚀

## Nguyên tắc cơ bản về JavaScript - Phần 1

### Thử thách viết mã số 1

Mark và John đang cố gắng so sánh chỉ số BMI (Chỉ số khối cơ thể) của họ, được tính theo công thức:  $BMI = \text{khối lượng} / \text{chiều cao}^2 = \text{khối lượng} / (\text{chiều cao} * \text{chiều cao})$  (khối lượng tính bằng kg và chiều cao tính bằng mét).

Nhiệm vụ của bạn:

1. Lưu trữ khối lượng và chiều cao của Mark và John ở dạng biến số
2. Tính chỉ số BMI của cả hai bằng công thức (thậm chí bạn có thể thực hiện cả hai phiên bản)
3. Tạo biến Boolean 'markHigherBMI' chứa thông tin về liệu Mark có chỉ số BMI cao hơn John hay không.

dữ liệu thử nghiệm:

§ Số liệu 1: Minh nặng 78kg, cao 1m69. John nặng 92 kg và cao 1,95 m.

§ Số liệu 2: Mark nặng 95 kg, cao 1,88 m. John nặng 85 kg và cao 1,76 m.

CHÚC MAY MẮN 😊

## Thử thách mã hóa #2

Sử dụng ví dụ về chỉ số BMI từ Thử thách số 1, mã bạn đã viết và cải thiện nó.

Nhiệm vụ của bạn:

1. In một đầu ra đẹp mắt ra bàn điều khiển, cho biết ai có chỉ số BMI cao hơn. Thông báo là "Chỉ số BMI của Mark cao hơn của John!" hoặc "Chỉ số BMI của John cao hơn của Mark!"
2. Sử dụng một mẫu chữ để bao gồm các giá trị BMI trong kết quả đầu ra. Ví dụ: "Chỉ số BMI của Mark (28,3) cao hơn của John (23,9)!"

Gợi ý: Sử dụng câu lệnh if/else




CHÚC MAY MẮN



## Thử thách viết mã số 3

Có hai đội thể dục dụng cụ, Cá heo và Gấu túi. Họ thi đấu với nhau 3 lần. Người chiến thắng với số điểm trung bình cao nhất sẽ giành được một chiếc cúp!

Nhiệm vụ của bạn:

1. Tính điểm trung bình cho mỗi đội, sử dụng dữ liệu kiểm tra bên dưới 2. So sánh điểm trung bình của các đội để xác định đội chiến thắng trong cuộc thi và in ra bảng điều khiển.  
Đừng quên rằng có thể có một trận hòa, vì vậy hãy kiểm tra điều đó (hòa có nghĩa là họ có cùng điểm trung bình)
3. Phần thưởng 1: Bao gồm yêu cầu về số điểm tối thiểu là 100. Với quy tắc này, một đội chỉ thắng nếu có số điểm cao hơn đội kia, đồng thời số điểm ít nhất là 100 điểm. Gợi ý: Sử dụng toán tử logic để kiểm tra điểm tối thiểu, cũng như nhiều khối khác-nếu  

4. Tiền thưởng 2: Điểm tối thiểu cũng áp dụng cho một trận hòa! Vì vậy, một trận hòa chỉ xảy ra khi cả hai đội có cùng số điểm và cả hai đều có số điểm lớn hơn hoặc bằng 100 điểm. Nếu không, không có đội nào giành cúp

dữ liệu thử nghiệm:

§ Dữ liệu 1: Cá heo đạt điểm 96, 108 và 89. Gấu túi đạt điểm 88, 91 và 110 § Dữ liệu  
thư ờng 1: Cá heo đạt điểm 97, 112 và 101. Gấu túi đạt điểm 109, 95 và 123 § Dữ liệu thư ờng 2:  
Cá heo đạt điểm 97, 112 và 101. Gấu túi đạt điểm 109, 95 và 106

CHÚC MAY MẮN 

## Thử thách viết mã số 4

Steven muốn xây dựng một máy tính tiền boa rất đơn giản cho bất cứ khi nào anh ấy đi ăn ở nhà hàng. Ở đất nước của anh ấy, người ta thường boa 15% nếu giá trị hóa đơn từ 50 đến 300. Nếu giá trị khác, tiền boa là 20%.

Nhiệm vụ của bạn:

1. Tính tiền tip, tùy theo giá trị hóa đơn. Tạo một biến gọi là 'tip' cho việc này. Không được phép sử dụng câu lệnh if/else (Nếu dễ dàng hơn đối với bạn, bạn 🤖 thể bắt đầu bằng câu lệnh if/else, sau đó thử chuyển đổi nó thành toán tử bậc ba!)
2. In một chuỗi ra bảng điều khiển chứa giá trị hóa đơn, tiền boa và giá trị cuối cùng (hóa đơn + tiền boa). Ví dụ: "Hóa đơn là 275, tiền boa là 41,25 và tổng giá trị là 316,25"

dữ liệu thử nghiệm:

§ Dữ liệu 1: Kiểm tra các giá trị hóa đơn 275, 40 và 430

gợi ý:

§ Để tính 20% của một giá trị, chỉ cần nhân nó với  $20/100 = 0,2$  § Giá trị X nằm trong khoảng từ 50 đến 300, nếu nó  $\geq 50$  &  $\leq 300$  🤔

CHÚC MAY MẮN 😊



## Nguyên tắc cơ bản về JavaScript – Phần 2

### Thử thách viết mã số 1

Trở lại với hai đội thể dục dụng cụ Dolphins và Koalas! Có một bộ môn thể dục dụng cụ mới hoạt động khác hẳn.

Mỗi đội thi đấu 3 lượt, sau đó tính điểm trung bình của 3 lượt (tức là mỗi đội tính 1 điểm trung bình).

Một đội chỉ thắng nếu có ít nhất gấp đôi số điểm trung bình của đội kia.

Nếu không, không có đội chiến thắng!

Nhiệm vụ của bạn:

1. Tạo hàm mũi tên 'calcAverage' để tính điểm trung bình của 3 điểm
2. Sử dụng hàm để tính điểm trung bình cho cả hai đội
3. Tạo hàm 'checkWinner' lấy điểm trung bình của mỗi đội làm tham số ('avgDolphins' và 'avgKoalas'), sau đó ghi người chiến thắng vào bảng điều khiển, cùng với điểm chiến thắng, theo quy tắc ở trên.

Ví dụ: "Koalas thắng (30 so với 13)"

4. Sử dụng chức năng 'checkWinner' để xác định người chiến thắng cho cả Dữ liệu 1 và Dữ liệu 2
5. Bỏ qua rút thăm lần này

dữ liệu thử nghiệm:

§ Dữ liệu 1: Cá heo đạt điểm 44, 23 và 71. Gấu túi đạt điểm 65, 54 và 49 § Dữ liệu

2: Cá heo đạt điểm 85, 54 và 41. Gấu túi đạt điểm 23, 34 và 27

gợi ý:

§ Để tính trung bình cộng của 3 giá trị, hãy cộng tất cả chúng lại với nhau và chia cho 3 §

Để kiểm tra xem số A có ít nhất gấp đôi số B hay không, hãy kiểm tra  $A \geq 2 * B$ .

Áp dụng điều này cho điểm trung bình của nhóm 🤔

CHÚC MAY MẮN 😊

## Thử thách mã hóa #2

Steven vẫn đang xây dựng máy tính tiền boa của mình, sử dụng các quy tắc giống như trước đây: Tiền boa 15% cho hóa đơn nếu giá trị hóa đơn nằm trong khoảng từ 50 đến 300 và nếu giá trị khác thì tiền boa là 20%.

Nhiệm vụ của bạn:

- Viết hàm 'calcTip' lấy bất kỳ giá trị hóa đơn nào làm đầu vào và trả về tiền boa tương ứng, được tính toán dựa trên các quy tắc ở trên (bạn có thể kiểm tra mã từ thử thách máy tính tiền boa đầu tiên nếu cần). Sử dụng loại chức năng mà bạn thích nhất. Kiểm tra chức năng bằng cách sử dụng giá trị hóa đơn là 100. Và bây giờ hãy sử dụng mảng! Vì vậy, hãy tạo một mảng 'hóa đơn' chứa dữ liệu thử nghiệm phía dưới.
- Tạo mảng 'tips' chứa giá trị tiền boa cho mỗi hóa đơn, được tính từ hàm bạn đã tạo trước đó.
- Bonus: Tạo mảng 'total' chứa tổng giá trị hóa đơn + tiền boa

Dữ liệu thử nghiệm: 125, 555 và 44

Gợi ý: Hãy nhớ rằng một mảng cần một giá trị ở mỗi vị trí và giá trị đó thực sự có thể là giá trị được trả về của một hàm! Vì vậy, bạn chỉ có thể gọi một hàm dưới dạng các giá trị mảng (vì vậy trước tiên đừng lưu trữ các giá trị tiền boa trong các biến riêng biệt mà ngay trong mảng mới)



CHÚC MAY MẮN



## Thử thách viết mã số 3

Hãy quay lại cảnh Mark và John so sánh chỉ số BMI của họ! Lần này, hãy sử dụng các đồ vật để thực hiện các phép tính! Ghi nhớ:  $BMI = \text{khối lượng} / \text{chiều cao}^2 = \text{khối lượng} / (\text{chiều cao} * \text{chiều cao})$  (khối lượng tính bằng kg và chiều cao tính bằng mét)

Nhiệm vụ của bạn:

1. Đối với mỗi người trong số họ, hãy tạo một đối tượng có các thuộc tính về tên đầy đủ, khối lượng và chiều cao của họ (Mark Miller và John Smith)
2. Tạo một phương thức 'calcBMI' trên mỗi đối tượng để tính chỉ số BMI (cùng một phương thức trên cả hai đối tượng). Lưu trữ giá trị BMI cho một thuộc tính và cũng trả lại giá trị đó từ phương thức
3. Đăng nhập vào bảng điều khiển của người có chỉ số BMI cao hơn, cùng với tên đầy đủ và chỉ số BMI tương ứng. Ví dụ: "Chỉ số BMI của John (28,3) cao hơn của Mark (23,9)!"

Dữ liệu kiểm tra: Mark nặng 78 kg và cao 1,69 m. John nặng 92 kg và cao 1,95 m.

CHÚC MAY MẮN 😊

## Thử thách viết mã số 4

Hãy cải thiện máy tính tiền boa của Steven hơn nữa, lần này là sử dụng các vòng lặp!

Nhiệm vụ của bạn:

1. Tạo mảng 'hóa đơn' chứa tất cả 10 giá trị hóa đơn kiểm tra
2. Tạo mảng trống cho tiền boa và tổng ('tips' và 'totals')
3. Sử dụng hàm 'calcTip' mà chúng ta đã viết trước đó (không cần lặp lại) để tính tiền boa và tổng giá trị (hóa đơn + tiền boa) cho mọi giá trị hóa đơn trong mảng hóa đơn. Sử dụng vòng lặp for để thực hiện 10 phép tính!

Dữ liệu thử nghiệm: 22, 295, 176, 440, 37, 105, 10, 1100, 86 và 52

Gợi ý: Gọi 'calcTip' trong vòng lặp và sử dụng phương thức đẩy để thêm giá trị vào mảng mẹo và mảng tổng 😊

Thứ ờng:

4. Phần thứ ờng: Viết một hàm 'calcAverage' nhận một mảng có tên 'arr' làm đối số. Hàm này tính trung bình cộng của tất cả các số trong mảng đã cho. Đây là một thử thách khó khăn (chúng tôi chưa từng làm điều này trước đây)! Đây là cách giải quyết nó:

4.1. Trước tiên, bạn sẽ cần cộng tất cả các giá trị trong mảng. Để thực hiện phép cộng, hãy bắt đầu bằng cách tạo một biến 'tổng' bắt đầu từ 0. Sau đó lặp qua mảng bằng vòng lặp for. Trong mỗi lần lặp lại, hãy thêm giá trị hiện tại vào biến 'tổng'. Bằng cách này, vào cuối vòng lặp, bạn có tất cả các giá trị được cộng lại với nhau

4.2. Để tính trung bình cộng, hãy chia tổng bạn đã tính trước đó cho độ dài của mảng (vì đó là số phần tử)

4.3. Gọi hàm với mảng 'tổng'

CHÚC MAY MẮN 😊

Kỹ năng dành cho nhà phát triển & Cài đặt trình chỉnh sửa

## Thử thách viết mã số 1

Đưa ra một mảng nhiệt độ tối đa được dự báo, nhiệt kế sẽ hiển thị một chuỗi với các nhiệt độ đã cho. Ví dụ: [17, 21, 23] sẽ in ra "...17°C trong 1 ngày... 21°C trong 2 ngày... 23°C trong 3 ngày..."

Nhiệm vụ của bạn:

1. Tạo một hàm 'printForecast' nhận vào một mảng 'arr' và ghi lại một chuỗi như trên vào bàn điều khiển. Hãy thử nó với cả hai bộ dữ liệu thử nghiệm.
2. Sử dụng khuôn khổ giải quyết vấn đề: Hiểu vấn đề và chia nhỏ vấn đề thành các vấn đề phụ!

dữ liệu thử nghiệm:

§ Dữ liệu 1: [17, 21, 23] § Dữ

liệu 2: [12, 5, -5, 0, 4]

CHÚC MAY MẮN 😊

## JavaScript trong Trình duyệt: DOM và Sự kiện

### Thử thách viết mã số 1

Triển khai chức năng nghỉ trò chơi để người chơi có thể đoán mới!

Nhiệm vụ của bạn:

1. Chọn phần tử có lớp 'again' và đính kèm trình xử lý sự kiện nhấp chuột 2. Trong hàm xử lý, khôi phục các giá trị ban đầu của 'score' và biến 'số bí mật'
3. Khôi phục các điều kiện ban đầu của thông báo, số, điểm và đầu vào đoán  
lĩnh vực
4. Đồng thời khôi phục màu nền ban đầu (#222) và chiều rộng số (15rem)

CHÚC MAY MẮN 😊

# Cấu trúc dữ liệu, toán tử hiện đại và chuỗi

## Thử thách viết mã số 1

Chúng tôi đang xây dựng một ứng dụng cá cược bóng đá (bóng đá cho những người bạn Mỹ của tôi 😊)!

Giả sử chúng tôi lấy dữ liệu từ một dịch vụ web về một trò chơi nhất định (biến 'trò chơi' ở trang tiếp theo). Trong thử thách này, chúng ta sẽ làm việc với dữ liệu đó.

Nhiệm vụ của bạn:

1. Tạo một mảng người chơi cho mỗi đội (các biến 'players1' và 'players2')
2. Người chơi đầu tiên trong bất kỳ hàng người chơi nào là thủ môn và những người khác là người chơi trên sân. Đối với Bayern Munich (đội 1), hãy tạo một biến ('gk') với tên của thủ môn và một mảng ('fieldPlayers') với tất cả 10 cầu thủ còn lại
3. Tạo một mảng 'allPlayers' chứa tất cả người chơi của cả hai đội (22 người chơi)
4. Trong trận đấu, Bayern Munich (đội 1) sử dụng 3 cầu thủ dự bị. Vì vậy, hãy tạo một mảng mới ('players1Final') chứa tất cả các cầu thủ ban đầu của đội 1 cộng với 'Thiago', 'Coutinho' và 'Perisic'
5. Dựa trên đối tượng game.odds, tạo một biến cho mỗi lẻ (được gọi là 'team1', 'draw' và 'team2')
6. Viết một hàm ('printGoals') nhận một số tên cầu thủ tùy ý (không phải một mảng) và in từng tên đó ra bàn điều khiển, cùng với tổng số bàn thắng được ghi (số tên cầu thủ được chuyển vào)
7. Đội có tỷ lệ cược thấp hơn có nhiều khả năng giành chiến thắng. In ra bàn điều khiển mà đội có nhiều khả năng giành chiến thắng hơn mà không cần sử dụng câu lệnh if/else hoặc toán tử bậc ba.

Dữ liệu thử nghiệm cho 6.: Đầu tiên, hãy sử dụng các cầu thủ 'Davies', 'Muller', 'Lwandowski' và 'Kimmich'.

Sau đó, gọi lại chức năng với người chơi từ game.scored

CHÚC MAY MẮN 😊

```

const game =
  { team1: 'Bayern Munich',
    team2: 'Borussia Dortmund', cầu
    thủ: [ [
      'Neuer',
      'Pavard',
      'Martinez',
      'Alaba',
      'Davies',
      'Kimmich',
      'Goretzka',
      'Coman',
      'Muller',
      'Gnarby',
      'Lewandowski', ],
    [
      'Burki',
      'Schulz',
      'Hummels',
      'Akanji',
      'Hakimi',
      'Weigl',
      'Witsel',
      'Hazard',
      'Brandt',
      'Sancho',
      'Gotze', ], ],
    ghi bàn: '4:0',
    ghi bàn:
    ['Lewandowski', 'Gnarby', 'Lewandowski', 'Hummels'], ngày:
    '9/11/2037', tỷ lệ cược: { team1: 1.33, x: 3.25, team2:
    6.5, }, };

```



## Thử thách mã hóa #2

Hãy tiếp tục với ứng dụng cá cược bóng đá của chúng tôi! Tiếp tục sử dụng biến 'trò chơi' từ trước.

Nhiệm vụ của bạn:

1. Lặp lại mảng `game.scored` và in tên từng người chơi ra bàn điều khiển, cùng với số bàn thắng (Ví dụ: "Bàn thắng 1: Lewandowski")
2. Dùng vòng lặp để tính trung bình lẻ và log vào console (Chúng ta đã học cách tính trung bình cộng, bạn nào không nhớ có thể vào xem lại)
3. In 3 tỷ lệ cược ra bảng điều khiển, nhưng ở định dạng đẹp, chính xác như sau:  
Tỷ lệ thắng của Bayern Munich: 1,33 Tỷ lệ hòa: 3,25

Tỷ lệ kèo trận thắng Borussia Dortmund: 6.5

Lấy tên nhóm trực tiếp từ đối tượng trò chơi, không mã hóa chúng (ngoại trừ "vẽ"). Gợi ý: Lưu ý cách tỷ lệ cược và các đối tượng trò chơi có cùng tên thuộc tính 4. 🤔

Phần thứ 2: Tạo một đối tượng gọi là 'người ghi bàn' chứa tên của những cầu thủ đã ghi bàn là thuộc tính và số bàn thắng là giá trị. Trong trò chơi này, nó sẽ trông như thế này:

```
{
  Gnarby: 1,
  Hummels: 1,
  Lewandowski: 2
}
```

CHÚC MAY MẮN 😊

## Thử thách viết mã số 3

Hãy tiếp tục với ứng dụng cá cược bóng đá của chúng tôi! Lần này, chúng tôi có một bản đồ gọi là 'gameEvents' (xem bên dưới) với nhật ký các sự kiện đã xảy ra trong trò chơi. Các giá trị chính là các sự kiện và các khóa là số phút mà mỗi sự kiện xảy ra (một trận bóng đá có 90 phút cộng thêm thời gian bù giờ).

Nhiệm vụ của bạn:

1. Tạo một mảng 'sự kiện' của các sự kiện trò chơi khác nhau đã xảy ra (không trùng lặp)
2. Sau khi trận đấu kết thúc, người ta nhận thấy thẻ vàng từ phút 64 không công bằng. Vì vậy, hãy xóa sự kiện này khỏi nhật ký sự kiện trò chơi.
3. Tính toán và ghi chuỗi sau vào bảng điều khiển: "Trung bình cứ 9 phút lại có một sự kiện xảy ra" (hãy nhớ rằng một trò chơi có 90 phút)
4. Lặp lại 'gameEvents' và ghi từng phần tử vào bảng điều khiển, đánh dấu cho dù đó là hiệp một hay hiệp hai (sau 45 phút) của trận đấu, như sau:

[NỬA ĐẦU] TẬP 17:  MỤC TIÊU

CHÚC MAY MẮN 

```
const gameEvents = bản đồ mới([ [17,
  MỤC TIÊU ,
  [36, '  ngư ời'], [47,
  MỤC TIÊU ,
  [61, '  ngư ời'],
  [64, ' Thẻ vàng'], [69, '
   Thẻ đỏ'],
  [70, '  ngư ời'],
  [72, '  ngư ời'], [76,
  MỤC TIÊU , [80, MỤC TIÊU'],
  ,
  [92, ' Thẻ vàng'], ]]);
```

## Thử thách viết mã số 4

Viết chương trình nhận danh sách tên biến được viết bằng `underscore_case` và chuyển đổi chúng thành `camelCase`.

Đầu vào sẽ đến từ một vùng văn bản được chèn vào DOM (xem mã bên dưới để chèn các phần tử) và chuyển đổi sẽ xảy ra khi nhấn nút.

Dữ liệu kiểm tra (dán vào vùng văn bản, bao gồm cả khoảng trắng):

```
underscore_case
first_name
Một số_Variable
tính toán_AGE bị
trì hoãn_khởi hành
```

Nên tạo đầu ra này (5 đầu ra `console.log` riêng biệt): `underscoreCase`

```
someVariable   ✓   Tính toán Tuổi bị trì
tên đầu tiên   ✓✓
               ✓✓✓
               ✓✓✓✓
               ✓✓✓✓✓
```

gợi ý:

- § Nhớ ký tự nào xác định một dòng mới trong vùng văn bản 😊
- § Giải pháp chỉ cần hoạt động đối với một biến được tạo thành từ 2 từ, như `a_b` § Bắt đầu mà không cần lo lắng về . Giải quyết vấn đề đó chỉ sau khi bạn có biến chuyển đổi tên hoạt động 😊
- § Thử thách này có mục đích khó, vì vậy hãy bắt đầu xem giải pháp trong trường hợp bạn đang bị mắc kẹt. Sau đó tạm dừng và tiếp tục!

Sau đó, kiểm tra với dữ liệu thử nghiệm của riêng bạn!

CHÚC MAY MẮN 😊

```
document.body.append(document.createElement('textarea'));
document.body.append(document.createElement('button'));
```

## Xem xét kỹ hơn về chức năng

### Thử thách viết mã số 1

Hãy xây dựng một ứng dụng thăm dò ý kiến đơn giản!

Một cuộc thăm dò có một câu hỏi, một loạt tùy chọn mà mọi người có thể chọn và một mảng có số câu trả lời cho mỗi tùy chọn. Dữ liệu này được lưu trữ trong đối tượng 'thăm dò ý kiến' khởi động bên dưới.

Nhiệm vụ của bạn:

1. Tạo một phương thức gọi là 'registerNewAnswer' trên đối tượng 'thăm dò ý kiến'. Phương pháp này thực hiện 2 việc: 1.1. Hiển thị cửa sổ nhắc người dùng nhập số tùy chọn đã chọn. Lời nhắc sẽ như sau: Ngôn ngữ lập trình yêu thích của bạn là gì?

0: JavaScript

1: Python

2: Rỉ sét

3: C++

(Ghi số tùy chọn)

- 1.2. Dựa trên số đầu vào, hãy cập nhật thuộc tính mảng 'câu trả lời'. Vì

ví dụ: nếu tùy chọn là 3, hãy tăng giá trị ở vị trí 3 của mảng lên 1. Đảm bảo kiểm tra xem đầu vào có phải là một số không và liệu số đó có hợp lý không (ví dụ: câu trả lời 52 sẽ không hợp lý, phải không?)

2. Gọi phương thức này bất cứ khi nào người dùng nhấp vào nút "Trả lời thăm dò ý kiến" .
3. Tạo phương thức 'displayResults' hiển thị kết quả thăm dò ý kiến. Các phương thức lấy một chuỗi làm đầu vào (được gọi là 'kiểu'), có thể là 'chuỗi' hoặc 'mảng'. Nếu loại là 'mảng', chỉ cần hiển thị mảng kết quả như hiện tại, sử dụng console.log(). Đây phải là tùy chọn mặc định. Nếu loại là 'chuỗi', hãy hiển thị chuỗi như "Kết quả bình chọn là 13, 2, 4, 1".
4. Chạy phương thức 'displayResults' ở cuối mỗi gọi phương thức 'registerNewAnswer'.
5. Phần thử nghiệm: Sử dụng phương thức 'displayResults' để hiển thị 2 mảng trong dữ liệu thử nghiệm. Sử dụng cả tùy chọn 'mảng' và 'chuỗi' . Không đặt các mảng trong đối tượng thăm dò ý kiến! Vì vậy, từ khóa this sẽ trông như thế nào trong tình huống này?

Dữ liệu thử nghiệm cho tiền thư ởng:

§ Dữ liệu 1: [5, 2,  
3] § Dữ liệu 2: [1, 5, 3, 9, 6, 1]

Gợi ý: Sử dụng nhiều công cụ bạn đã học trong phần này và phần trước



CHÚC MAY MẮN 😊

```
thăm dò const = {  
  câu hỏi: "Ngôn ngữ lập trình yêu thích của bạn là gì?", tùy chọn: ["0:  
  JavaScript", "1: Python", "2: Rust", "3: C++"], // Điều này tạo ra [0, 0,  
  0, 0]. Thêm trong phần tiếp theo! câu trả lời: new Array(4).fill(0), };
```

## Thử thách mã hóa #2

Đây là một thử thách về tư duy hơn là thử thách viết mã 🤖

Nhiệm vụ của bạn:

1. Lấy IIFE bên dưới và ở cuối chức năng, đính kèm một trình xử lý sự kiện thay đổi màu của phần tử h1 đã chọn ('tiêu đề') thành màu xanh lam, mỗi khi phần tử nội dung được nhấp. Không chọn lại phần tử h1!
2. Và bây giờ hãy giải thích cho chính bạn (hoặc ai đó xung quanh bạn) tại sao điều này lại hiệu quả! Dành tất cả thời gian bạn cần. Hãy suy nghĩ về thời điểm chính xác hàm gọi lại được thực thi và điều đó có ý nghĩa gì đối với các biến liên quan trong ví dụ này.

```
(hàm () {  
  tiêu đề const = document.querySelector('h1');  
  header.style.color = 'đỏ'; })();
```

CHÚC MAY MẮN 😊

# Làm việc với mảng

## Thử thách viết mã số 1

Julia và Kate đang nghiên cứu về loài chó. Vì vậy, mỗi người trong số họ đã hỏi 5 chủ sở hữu chó về tuổi của con chó của họ và lưu trữ dữ liệu vào một mảng (một mảng cho mỗi người). Hiện tại, họ chỉ quan tâm đến việc biết một con chó là trưởng thành hay chó con.

Chó được coi là trưởng thành nếu ít nhất 3 tuổi và là chó con nếu chưa đầy 3 tuổi.

Nhiệm vụ của bạn:

Tạo một hàm 'checkDogs', hàm này chấp nhận 2 mảng tuổi của chó ('dogsJulia' và 'dogsKate') và thực hiện những việc sau: 1. Julia phát hiện ra rằng chủ của hai con chó đầu tiên và cuối cùng thực sự có mèo, không phải chó! Vì vậy, hãy tạo một bản sao nông của mảng của Julia và xóa tuổi mèo khỏi mảng đã sao chép đó (vì đó là một cách làm không tốt để thay đổi các tham số hàm)

2. Tạo một mảng có cả dữ liệu của Julia (đã sửa) và của Kate 3. Đối với mỗi con chó còn lại, hãy đăng nhập vào bảng điều khiển xem đó là con trưởng thành ("Con chó số 1 là con trưởng thành và 5 tuổi") hay con chó con ("Con chó số 2 vẫn là một con chó con ") 🐶
4. Chạy hàm cho cả hai bộ dữ liệu thử nghiệm

dữ liệu thử nghiệm:

§ Dữ liệu 1: Dữ liệu của Julia [3, 5, 2, 12, 7], Dữ liệu của Kate [4, 1, 15, 8, 3] § Dữ liệu 2: Dữ liệu của Julia [9, 16, 6, 8, 3], của Kate dữ liệu [10, 5, 6, 1, 4]

Gợi ý: Sử dụng các công cụ từ tất cả các bài giảng trong phần này cho đến nay 😊

CHÚC MAY MẮN 😊

## Thử thách mã hóa #2

Hãy quay lại nghiên cứu của Julia và Kate về chó. Lần này, họ muốn chuyển đổi tuổi của chó sang tuổi của con người và tính tuổi trung bình của những con chó trong nghiên cứu của họ.

Nhiệm vụ của bạn:

Tạo một hàm 'calcAverageHumanAge', hàm này chấp nhận một mảng tuổi của chó ('ages') và thực hiện các việc sau theo thứ tự: 1. Tính tuổi của chó theo năm của con người bằng công thức sau: nếu chó  $\leq 2$  tuổi già, tuổi người =  $2 * \text{tuổi chó}$ . Nếu con chó  $> 2$  tuổi,  $\text{humanAge} = 16 + \text{dogAge} * 4$  2. Loại trừ tất cả những con chó dưới 18 tuổi của con người (giống như

nuôi chó ít nhất 18 tuổi)

3. Tính tuổi con người trung bình của tất cả các con chó trư ởng thành (bạn nên biết từ những thách thức khác về cách chúng tôi tính trung bình) 🤔
4. Chạy hàm cho cả hai bộ dữ liệu thử nghiệm

dữ liệu thử nghiệm:

§ Dữ liệu 1: [5, 2, 4, 1, 15, 8, 3] § Dữ

liệu 2: [16, 6, 10, 5, 6, 1, 4]

CHÚC MAY MẮN 😊

## Thử thách viết mã số 3

Viết lại hàm 'calcAverageHumanAge' từ Thử thách #2, như ng lần này là hàm mũi tên và sử dụng chuỗi!

dữ liệu thử nghiệm:

§ Dữ liệu 1: [5, 2, 4, 1, 15, 8, 3] § Dữ

liệu 2: [16, 6, 10, 5, 6, 1, 4]

CHÚC MAY MẮN 😊



## Thử thách viết mã số 4

Julia và Kate vẫn đang nghiên cứu về chó, và lần này họ đang nghiên cứu xem chó ăn quá nhiều hay quá ít.

Ăn quá nhiều có nghĩa là khẩu phần thức ăn hiện tại của chó lớn hơn khẩu phần khuyến nghị, còn ăn quá ít thì ngược lại.

Ăn một lượng vừa phải có nghĩa là khẩu phần thức ăn hiện tại của chó nằm trong phạm vi trên 10% và dưới 10% so với khẩu phần khuyến nghị (xem gợi ý).

Nhiệm vụ của bạn:

- Lập lại mảng 'dogs' chứa các đối tượng chó và đối với mỗi con chó, hãy tính toán phần thức ăn được khuyến nghị và thêm nó vào đối tượng dư thừa dạng một thuộc tính mới. Không tạo một mảng mới, chỉ cần lặp qua mảng. Dẫn đầu: khuyến nghịThức phẩm = trọng lượng \*\* 0,75 \* 28. (Kết quả tính bằng gam thực phẩm và trọng lượng cần tính bằng kg)
- Tìm con chó của Sarah và đăng nhập vào bảng điều khiển xem nó có ăn quá nhiều hay quá nhiều không nhỏ bé. Gợi ý: Một số con chó có nhiều chủ, vì vậy trước tiên bạn cần tìm Sarah trong mảng chủ sở hữu, vì vậy phần này hơi phức tạp (có mục đích) 🤔
- Tạo một mảng chứa tất cả những người nuôi chó ăn quá nhiều ('chủ sở hữuEatTooMuch') và một mảng có tất cả chủ sở hữu của những con chó ăn quá ít ('chủ sở hữuEatTooLittle').
- Ghi một chuỗi vào bảng điều khiển cho mỗi mảng được tạo trong 3., như sau: "Matilda and Alice and Bob's dogs eat too much!" và "Chó của Sarah, John và Michael ăn quá ít!"
- Đăng nhập vào bảng điều khiển xem có con chó nào ăn chính xác lượng thức ăn không điều đó được khuyến nghị (chỉ đúng hoặc sai)
- Đăng nhập vào bảng điều khiển xem có con chó nào ăn đủ lượng thức ăn không (chỉ đúng hoặc sai)
- Tạo một mảng chứa những con chó đang ăn một lượng thức ăn vừa phải (cố gắng sử dụng lại điều kiện đã sử dụng trong 6.)
- Tạo một bản sao nông của mảng 'chó' và sắp xếp nó theo phần thức ăn được đề xuất theo thứ tự tăng dần (hãy nhớ rằng các phần nằm bên trong các đối tượng của mảng)



gợi ý:

§ Sử dụng nhiều công cụ khác nhau để giải quyết những thách thức này, bạn có thể sử dụng bản tóm tắt bài giảng để lựa chọn giữa chúng 😊

§ Nằm trong phạm vi 10% trên và dư ới phần dư ợc khuyến nghị có nghĩa là: hiện tại > (dư ợc khuyến nghị \* 0,90) && hiện tại < (dư ợc khuyến nghị \* 1,10). Về cơ bản, phần hiện tại phải nằm trong khoảng từ 90% đến 110% so với phần khuyến nghị.

dữ liệu thử nghiệm:

```
const chó = [  
  { trọng lượng: 22, curFood: 250, chủ sở hữu: ['Alice', 'Bob'] }, { trọng lượng: 8,  
    curFood: 200, chủ sở hữu: ['Matilda'] }, { trọng lượng: 13, curFood: 275, chủ sở hữu :  
    ['Sarah', 'John'] }, { cân nặng: 32, curFood: 340, chủ sở hữu: ['Michael'] }, ];
```

CHÚC MAY MẮN 😊

# Lập trình hướng đối tượng (OOP)

## Thử thách viết mã số 1

Nhiệm vụ của bạn:

1. Sử dụng hàm tạo để triển khai 'Xe hơi'. Một chiếc ô tô có đặc tính 'làm' và 'tốc độ'. Thuộc tính 'tốc độ' là tốc độ hiện tại của ô tô tính bằng km/h
2. Thực hiện phương pháp 'tăng tốc' để tăng tốc độ của ô tô lên 10 và ghi tốc độ mới vào bảng điều khiển
3. Thực hiện phương pháp 'phanh' để giảm tốc độ của ô tô xuống 5 và ghi lại

tốc độ mới cho bảng điều khiển

Tạo 2 đối tượng 'Xe hơi' và thử nghiệm gọi 'tăng tốc' và 'phanh' nhiều lần trên mỗi ngữ cảnh trong số họ

dữ liệu thử nghiệm:

§ Xe dữ liệu 1: 'BMW' đi với tốc độ 120 km/h §

Xe dữ liệu 2: 'Mercedes' đi với tốc độ 95 km/h

CHÚC MAY MẮN 😊

## Thử thách mã hóa #2

Nhiệm vụ của bạn:

1. Tạo lại Thử thách số 1, nhưng lần này sử dụng lớp ES6 (gọi nó là 'CarCl')
2. Thêm một getter gọi là 'speedUS' để trả về tốc độ hiện tại tính bằng mi/h (chia bằng 1.6)
3. Thêm một trình thiết lập có tên là 'speedUS' để đặt tốc độ hiện tại tính bằng mi/h (nhưng chuyển đổi nó thành km/h trước khi lưu trữ giá trị, bằng cách nhân đầu vào với 1,6)
4. Tạo một chiếc ô tô mới và thử nghiệm với 'tăng tốc' và 'phanh' các phương thức và với getter và setter.

dữ liệu thử nghiệm:

§ Xe dữ liệu 1: 'Ford' đi với tốc độ 120 km/h

CHÚC MAY MẮN



## Thử thách viết mã số 3

Nhiệm vụ của bạn:

1. Sử dụng hàm tạo để triển khai Ô tô điện (được gọi là 'EV') dưới dạng "lớp" con của 'Ô tô'. Bên cạnh nhãn hiệu và tốc độ hiện tại, 'EV' còn có mức sạc pin hiện tại tính bằng % (thuộc tính 'sạc')
2. Triển khai phương thức 'chargeBattery' lấy đối số 'chargeTo' và đặt mức sạc pin thành 'chargeTo'
3. Thực hiện phương pháp 'tăng tốc' để tăng tốc độ của ô tô lên 20, và giảm phí 1%. Sau đó đăng nhập một tin nhắn như thế này: 'Tesla đi với tốc độ 140 km/h, với mức sạc 22%'
4. Tạo một đối tượng ô tô điện và thử nghiệm gọi 'tăng tốc', 'phanh' và 'chargeBattery' (sạc đến 90%). Chú ý điều gì sẽ xảy ra khi bạn 'tăng tốc'! Gợi ý: Xem lại định nghĩa đa hình



dữ liệu thử nghiệm:

§ Ô tô dữ liệu 1: 'Tesla' đi với tốc độ 120 km/h, với mức sạc 23%

CHÚC MAY MẮN



## Thử thách viết mã số 4

Nhiệm vụ của bạn:

1. Tạo lại Thử thách #3, như lần này sử dụng các lớp ES6: tạo lớp con 'EVC1' của lớp 'CarCl'

2. Đặt thuộc tính 'sạc' ở chế độ riêng tư 3.

Triển khai khả năng xâu chuỗi 'tăng tốc' và 'sạc Pin'

các phương thức của lớp này và cũng cập nhật phương thức 'phanh' trong lớp 'CarCl'. Sau đó thử nghiệm với chuỗi!

dữ liệu thử nghiệm:

5 Xe dữ liệu 1: 'Rivian' chạy với tốc độ 120 km/h, với mức sạc 23%

CHÚC MAY MẮN 😊

# JavaScript không đồng bộ

## Thử thách viết mã số 1

Trong thử thách này, bạn sẽ xây dựng một hàm 'whereAmI' để hiển thị một quốc gia chỉ dựa trên tọa độ GPS. Để làm được điều đó, bạn sẽ sử dụng API thứ hai để mã hóa tọa độ địa lý. Vì vậy, trong thử thách này, lần đầu tiên bạn sẽ sử dụng một API của riêng mình 😊

Nhiệm vụ của bạn:

### PHẦN 1

1. Tạo hàm 'whereAmI' lấy giá trị vĩ độ ('lat') và giá trị kinh độ ('lng') làm giá trị đầu vào (đây là các tọa độ GPS, ví dụ trong dữ liệu thử nghiệm bên dưới).

2. Thực hiện "mã hóa địa lý ngược" của tọa độ được cung cấp. Mã hóa địa lý ngược có nghĩa là chuyển đổi tọa độ thành một vị trí có ý nghĩa, chẳng hạn như tên thành phố và quốc gia.

Sử dụng API này để thực hiện mã hóa địa lý ngược: <https://geocode.xyz/api>. Cuộc gọi AJAX sẽ được thực hiện với một URL có định dạng sau:

<https://geocode.xyz/52.508,13.381?geoit=json>. Sử dụng API tìm nạp và hứa sẽ lấy dữ liệu.

Không sử dụng chức năng 'getJSON' mà chúng tôi đã tạo, đó là hành vi gian lận



3. Sau khi bạn có dữ liệu, hãy xem dữ liệu đó trong bảng điều khiển để xem tất cả các thuộc tính mà bạn nhận được về vị trí được cung cấp. Sau đó, sử dụng dữ liệu này, ghi một thông báo như thế này vào bảng điều khiển: "Bạn đang ở Berlin, Đức"

4. Chuỗi một phương thức .catch đến cuối chuỗi lời hứa và ghi lỗi vào

bảng điều khiển

5. API này cho phép bạn chỉ thực hiện 3 yêu cầu mỗi giây. Nếu bạn tải lại nhanh, bạn sẽ gặp lỗi này với mã 403. Đây là lỗi với yêu cầu. Hãy nhớ rằng, hàm fetch() không từ chối lời hứa trong trường hợp này. Vì vậy, hãy tự tạo một lỗi để từ chối lời hứa, với một thông báo lỗi có ý nghĩa

### PHẦN 2

6. Bây giờ là lúc sử dụng dữ liệu nhận được để hiển thị một quốc gia. Vì vậy, hãy lấy thuộc tính có liên quan từ kết quả API mã hóa địa lý và cắm nó vào API quốc gia mà chúng tôi đang sử dụng.

7. Kết xuất quốc gia và bất kỳ lỗi nào, giống như chúng ta đã làm trong bài giảng trước (bạn thậm chí có thể sao chép mã này, không cần phải nhập mã tự động)

dữ liệu thử nghiệm:

§ Tọa độ 1: 52.508, 13.381 (Vĩ độ, Kinh độ) § Tọa độ 2:  
19.037, 72.873 § Tọa độ 3: -33.933, 18.474

CHÚC MAY MẮN 😊

## Thử thách mã hóa #2

Đối với thử thách này, bạn sẽ thực sự phải xem video! Sau đó, xây dựng chức năng tải hình ảnh mà tôi vừa chỉ cho bạn trên màn hình.

Nhiệm vụ của bạn:

Nhiệm vụ lần này không phải là siêu mô tả, vì vậy bạn có thể tự mình tìm ra một số nội dung. Giả vờ như bạn đang làm việc một mình 🤔

### PHẦN 1

1. Tạo hàm 'createImage' nhận 'imgPath' làm đầu vào.

Hàm này trả về một lời hứa tạo một hình ảnh mới (sử dụng `document.createElement('img')`) và đặt thuộc tính `.src` cho đường dẫn hình ảnh được cung cấp

2. Khi hình ảnh tải xong, hãy nối nó vào phần tử DOM bằng lệnh

lớp 'hình ảnh' và giải quyết lời hứa. Giá trị hoàn thành phải là chính phần tử hình ảnh.

Trong trường hợp có lỗi khi tải hình ảnh (nghe sự kiện 'lỗi'), từ chối lời hứa

3. Nếu phần này quá khó đối với bạn, hãy xem phần đầu tiên của giải pháp

### PHẦN 2

4. Sử dụng lời hứa bằng cách sử dụng `.then` và cũng thêm trình xử lý lỗi 5. Sau khi

hình ảnh được tải, hãy tạm dừng thực thi trong 2 giây bằng cách sử dụng `'chờ'`

chức năng chúng tôi đã tạo trước đó

6. Sau khi 2 giây trôi qua, hãy ẩn hình ảnh hiện tại (đặt thuộc tính CSS hiển thị thành 'none')

và tải hình ảnh thứ hai (Gợi ý: Sử dụng phần tử hình ảnh được trả về bởi lời hứa 'createImage' để ẩn hình ảnh hiện tại. Bạn sẽ cần một biến toàn cầu cho điều đó)



7. Sau khi tải xong hình ảnh thứ hai, hãy tạm dừng thực hiện lại trong 2 giây 8. Sau khi 2

giây trôi qua, hãy ẩn hình ảnh hiện tại

Dữ liệu kiểm tra: Hình ảnh trong thư mục `img`. Kiểm tra trình xử lý lỗi bằng cách chuyển sai đường dẫn hình ảnh. Đặt tốc độ mạng thành "3G nhanh" trong tab Mạng của công cụ dành cho nhà phát triển, nếu không thì hình ảnh tải quá nhanh

CHÚC MAY MẮN 😊



## Thử thách viết mã số 3

Nhiệm vụ của bạn:

### PHẦN 1

1. Viết hàm async `'loadNPause'` để tạo lại Thử thách #2, lần này sử dụng `async/await` (chỉ phần sử dụng lời hứa, sử dụng lại hàm `'createImage'` trước đó)
2. So sánh hai phiên bản, suy nghĩ về những điểm khác biệt lớn và xem phiên bản nào bạn thích hơn
3. Đừng quên kiểm tra trình xử lý lỗi và cài đặt tốc độ mạng thành "3G nhanh" trong tab Mạng công cụ dành cho nhà phát triển

### PHẦN 2

1. Tạo chức năng không đồng bộ `'loadAll'` nhận một mảng dữ liệu dẫn hình ảnh `'imgArr'`
2. Sử dụng `.map` để lặp qua mảng, để tải tất cả các hình ảnh bằng chức năng `'createImage'` (gọi mảng kết quả là `'imgs'`)
3. Kiểm tra mảng `'imgs'` trong bảng điều khiển! Nó có giống như bạn mong đợi không?
4. Sử dụng chức năng kết hợp lời hứa để thực sự lấy hình ảnh từ mảng 5. Thêm lớp `'song song'` cho tất cả các hình ảnh (nó có một số kiểu CSS)

Dữ liệu thử nghiệm Phần 2: `['img/img-1.jpg', 'img/img-2.jpg', 'img/img 3.jpg']`. Để kiểm tra, tắt chức năng `'loadNPause'`

CHÚC MAY MẮN 😊