Bài: Câu lệnh điều kiện if-else và switch–case trong JavaScript

Xem bài học trên website để ủng hộ Kteam: Câu lệnh điều kiện if-else và switch–case trong JavaScript

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Ở các bài trước, Kteam đã giới thiệu đến các bạn về các kiểu dữ liệu cơ bản trong JavaScript

Trong bài này, chúng ta sẽ cùng tìm hiểu về một cấu trúc mới trong JavaScript: **câu lệnh điều kiện**

Nội dung

Để nắm được bài này, các bạn cần có kiến thức về:

- Biến trong JavaScript
- Các kiểu dữ liệu cơ bản trong JavaScript: boolean, number, string...
- Khối lệnh trong JavaScript

Nội dung mà chúng ta sẽ cùng tìm hiểu:

- Đặt vấn đề #1
- Cấu trúc điều kiện if
- Cấu trúc điều kiện if - else
- Ternary operator (?:)
- Cấu trúc điều kiện if - else if – else
- Đặt vấn đề #2
- Cấu trúc điều kiện switch – case
- Lệnh break trong JavaScript

Đặt vấn đề #1

Hôm nay, Long mở heo đất và lấy tiền trong đó đi mua đồ.

Nếu như Long có 10 nghìn, thì Long sẽ mua một cái kẹo, và 3 cây bút

Nếu Long có 20 nghìn, thì Long sẽ mua một cái kẹo, 3 cây bút và 2 cuốn vở.

Nếu...Long có một số tiền là n, thì Long sẽ mua gì ?

Nó là một câu hỏi liên quan đến một cấu trúc mới: **cấu trúc điều kiện.**

Tức là, **với một điều kiện cho trước** (số tiền)**, thì một công việc nào đó sẽ được thực hiện** (mua đồ).

Câu lệnh điều kiện với if

Nếu các bạn để ý, thì các mệnh đề trên, sẽ có cấu trúc theo dạng:

> Nếu....(A)...thì...(B)

Trong đó, (A) là một điều kiện, còn (B) là những thứ sẽ diễn ra nếu điều kiện đó được đáp ứng.

Trong JavaScript, mọi thứ diễn ra y chang như thế. Cụ thể, cấu trúc như trên khi được "dịch" sang ngôn ngữ JavaScriptthì nó sẽ có dạng:

> **if(<condition>) <statement>**

Trong đó:

- **<condition>** là một điều kiện, tức là, nó sẽ mang một trong 2 giá trị: **truthy** hoặc **falsy**.
- **<statement>**: có thể là một câu lệnh hoặc một khối lệnh, là những thứ sẽ xảy ra nếu **<condition>** là **truthy**.

Ví dụ:

Javascript:

```
a = 123
// 123
if(a == 123) console.log("a bằng 123")
// a bằng 123
// undefined
b = ""
// ''
if(b == false) console.log("b bằng false")
// b bằng false
// undefined
```

Trên thực tế, **<condition>** chỉ là một điều kiện, nhưng JavaScript không cấm việc bạn "nhét" cho nó thêm vài điều kiện nữa, ví dụ như sau:

Javascript:

```
a = 1, b = 2, c = 'Kteam'
// 'Kteam'
if((a == 1) && (b == 2)) console.log("Yes")
// Yes
// undefined
if((c == "Kteam") || (b == 3)) console.log("Yes")
// Yes
// undefined
```

> **Lưu ý:** Việc nhét nhiều điều kiện vào sau if thực chất là sử dụng các biểu thức quan hệ cho nhiều điều kiện khác nhau. Các điều kiện này sẽ được "gom" lại thành một điều kiện duy nhất

Bên cạnh đó, ta cũng có thể sử dụng các toán tử ở bên trong một lệnh if:

Javascript:

```
a = 0
// 0
if(a += 5) console.log("a = 5")
// a = 5
// undefined
a
// 5
if(a-=5) console.log("Kterr")
// undefined
```

Tuy nhiên

Javascript:

```
x = 1
// 1
y = 2
// 2
if(x = y) y++;
// 2
x
// 2
y
// 3
```

Đừng bao giờ tự phức tạp hóa code của bạn, vì trên thực tế, khi làm việc, code của bạn sẽ qua tay rất nhiều người. Bạn sẽ không muốn "tặng" cho họ một câu đố đâu nhỉ ?

Cấu trúc điều kiện if-else

Giờ Long lại nảy sinh một suy nghĩ khác: Nếu trong heo có trên 20 nghìn thì Long sẽ mua truyện tranh, còn nếu ít hơn thì sẽ mua kẹo chia cho các anh.

Rõ ràng là một tình huống hay gặp: nếu một điều kiện được thỏa mãn, thì một hành động nào đó sẽ xảy ra, và trong trường hợp ngược lại, thì cũng vẫn diễn ra một hành động khác.

Để đáp ứng được yêu cầu đó, Js hỗ trợ cấu trúc điều kiện: if-else;

> **if (<condition>) <statement-1>;**
>
> **else <statement-2>**

Trong đó:

- **<condition>** thì vẫn giống như trên, chỉ khác là: nếu **<condition>** là **falsy** thì sẽ thực hiện **<statement-2>**
- **<statement-1>**: là lệnh (hoặc khối lệnh) sẽ được thực thi khi **<condition>** là **truthy**
- **<statement-2>**: là lệnh (khối lệnh) được thực hiện nếu **<condition>** là **falsy**.

Ví dụ:

Javascript:

```
if(tien_cua_long < "20 nghin") console.log("Long mua keo");
else console.log("Long mua truyen tranh")
// Long mua truyen tranh
// undefined
```

Và đến giờ này, Long lại nghĩ khác (quả thật là một cậu bạn thiếu kiên nhẫn 😟): nếu Long có đủ 20 nghìn, thì Long sẽ mua xe máy múc, còn nếu chỉ có một lượng tiền trong khoảng 10-20 nghìn, thì Long mua truyện, còn nếu ít hơn 10 nghìn thì mua kẹo.

Đây là một trường hợp phức tạp hơn của điều kiện: **các lệnh lồng nhau**

Javascript:

```
tien_cua_long = 12000
//12000
if(tien_cua_long >= 20000)
 console.log("Long mua may muc"); else if(tien_cua_Long >= 10000)
 console.log("Long mua truyen"); else
 console.log("Long mua keo")
// Long mua truyen
// undefined
```

Ternary operator

Nó là một toán tử 3 ngôi cho phép thực hiện các lệnh điều kiện một cách nhanh chóng hơn.

Cú pháp:

> **<condition> ? <value-1> : <value-2>**

Nếu **<condition>** là **truthy**, thì lấy **<value-1>**, ngược lại lấy **<value-2>**

Chúng ta cùng xem xét các trường hợp của một ví dụ:

Trường hợp 1:

Javascript:

```
f = 0
// 0
t = ""
// ''
if(f > 0)
 t = "kteam"; else
 t = "kter"
// 'kter'
t
// 'kter'
```

Trường hợp 2:

Javascript:

```
f = 0
// 0
t = (f > 0 ? "kteam" : "kter")
// 'kter'
t
// 'kter'
```

Ta có thể thấy rõ ràng rằng: trong nhiều trường hợp, sử dụng **ternary operator** có thể giảm được thời gian code, và giúp cho code của chúng ta trông "thon gọn" hơn.

Nhưng đừng bao giờ dùng các ternary operator lồng nhau, vì chúng ảo lắm.

Lấy một ví dụ đơn giản: Giả sử tất cả các biến trong đây đều có giá trị (ngoại trừ k) thì sẽ thế nào? Lại một câu đố hóc búa! Nếu bạn có thể đưa ra đáp án, vui lòng comment bên dưới BÌNH LUẬN

Javascript:

```
var k = a ? (b ? (c ? d : e) : (d ? e : f)) : f ? (g ? h : i) : j;
```

Đặt vấn đề #2

Toàn là một con người lập dị (:v). Những yêu cầu của cậu ta thật là khó hiểu.

Khác với Long, Toàn quyết định thảy 1 con súc sắc, và quy định sẽ ăn số kẹo tương ứng với số nút thảy được.

Như vậy, về tổng thể, sẽ có 6 trường hợp có thể xảy ra với số kẹo mà Toàn ăn.

Trong JavaScript, cũng có một cấu trúc điều kiện để giúp chúng ta kiểm tra các trường hợp có thể xảy ra của một điều kiện nào đó, và với khả năng đó, sẽ có một lệnh hay một khối lệnh được thực hiện.

Đó là cấu trúc **switch-case**

Cấu trúc switch-case

Cú pháp:

Javascript:

```
switch(<expression>) {
        case <value-1>:
                <statement-1>
                break;
        case <value-2>:
                <statement-2>
                break;
        case <value-3>:
                <statement-3>
                break;
        ......................
        case <value-n>:
                <statement-n>
        default:
                <default-statement>
```

Trong đó:

- **<expression>**: là biểu thức, giá trị này sẽ được so sánh lần lượt với **<value>** của từng **case**, nếu như **<value>** mà bằng với **<expression>**, thì **<statement>** tương ứng được thực hiện
- **<value>**: là các giá trị được mang đi so sánh, lệnh (khối lệnh) bên trong nó sẽ được thực hiện nếu**<value>** bằng với **<expression>**.
- **<statement>**: là một lệnh hoặc một khối lệnh, sẽ được thực hiện nếu **<value>** tương ứng bằng với **<expression>**
- **<default-statement>**: là lệnh (khối lệnh) sẽ được thực hiện nếu **<expression>** khác với tất cả các **<value>**.

Ví dụ:

Javascript:

```
a = "This" + ' ' + "is" + "howKteam"
// 'This ishowKteam'
> switch(a) {
     case "This is howKteam":
         console.log("True")
        break;
     case "This ishowKteam":
         console.log("False")
        break;
}
// False
// undefined
```

Nếu có nhiều case trùng nhau, chương trình sẽ chỉ thực hiện các lệnh trong <statement> đầu tiên

Javascript:

```
candies = 2
// 2
switch (candies) {
   case 2:
     console.log("Two");
     break;
   case 2:
     console.log("Hai");
     break
}
// Two
// undefined
```

Lưu ý: Nếu ta không đặt lệnh break sau các **<statement>**, thì một điều không mong muốn sẽ xảy ra:

Ví dụ: Cho đoạn code sau:

Javascript:

```
candies = 2;
// undefined
switch(candies) {
 case 1:
     console.log("One candy");
 case 2:
     console.log("Two candies");
 case 3:
     console.log("Three candies");
 default:
     console.log("More than 3 candies");
}
// Two candies
// Three candies
// More than 3 candies
// undefined
```

Tại sao nó lại in ra một mớ bòng bong thế kia ? Đáp án, một cách đơn giản, là do ta thiếu lệnh **break,** và nó được gọi là fall through.

Theo mặc định, chương trình sẽ thực hiện các lệnh (khối lệnh) từ nơi có **<value>** bằng với **<expression>** cho đến cuối mà không xét các điều kiện tiếp theo nữa.

Để "fix" nó, ta sẽ cần đến lệnh break.

Lệnh break trong JavaScript

Trong JavaScript và hầu hết ngôn ngữ lập trình khác, lệnh ***break*** được dùng để thoát khỏi một vòng lặp (khái niệm mà chúng ta sẽ cùng tìm hiểu trong bài tiếp theo).

Bên cạnh đó, với JavaScript, ***break*** còn được dùng để thoát khỏi **cấu trúc switch-case**. Cụ thể, nếu ta đặt đúng như cấu trúc, thì nó sẽ thoát ngay khi đã thực hiện một (khối) lệnh, mà không để tràn sang các (khối) lệnh khác.

Ví dụ:

Javascript:

```
candies = 1
// 1
switch(candies) {
     case 1:
         console.log("One candy");
         break;
     case 2:
         console.log("Two candies");
     default:
         console.log("More than 2 candies");
}
// One candy
// undefined
```

Bên cạnh đó, các bạn có thể lồng nhiều trường hợp vào chung một nhóm:

Javascript:

```
switch (candies) {
   case 1:
   case 2:
     console.log("One candy or two candies");
     break;
   default:
     console.log("More than 2 candies");
     break;
}
// One candy or two candies
// undefined
```

Hoặc

Javascript:

```
candies = 1
// 1
switch (candies) {
   case 1:
   case 2:
     console.log("One candy or two candies");
     break;
   default:
     console.log("More than 2 candies");
     break;
}
// One candy or two candies
// undefined
```

Kết luận

Trong bài này, chúng ta đã được tìm hiểu về câu lệnh điều kiện trong JavaScript

Ở bài sau, chúng ta sẽ cùng nhau tìm hiểu về Bài tập về câu lệnh điều kiện trong JavaScript

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".