

عملگرهای منطقی

@alithecodeguy

سه عملگر منطقی در جاوا اسکریپت وجود دارد: `||` (OR یا)، `&&` (AND و)، `!` (NOT قرینه)

اگرچه این عملگرها، عملگرهای منطقی نامیده می‌شوند، ولی در واقع روی هر مقدار از هر نوعی می‌توانند اعمال شوند. نتیجه آنها نیز می‌تواند از هر نوعی باشد.

`||` (OR)

عملگر OR با دو خط عمود نشان داده می‌شود:

```
result = a || b;
```

در برنامه نویسی کلاسیک، عملگر OR فقط برای مقادیر منطقی استفاده می‌شد. اگر یکی از آرگومان‌ها `true` بود، نتیجه عبارت `true` می‌شد، در غیر اینصورت نتیجه عبارت `false` می‌شد. این عملگر، در جاوا اسکریپت قابلیت‌های بیشتری دارد و کمی قویتر است. ولی ابتدا، ببینیم در محاسبه مقادیر منطقی با این عملگر چه اتفاقی رخ می‌دهد:

```
alert( true || true ); // true
alert( false || true ); // true
alert( true || false ); // true
alert( false || false ); // false
```

همانطور که می‌بینیم، نتیجه عبارت همیشه `true` بوده مگر اینکه هر دو عملگر `false` باشد. اگر عملوند مقدار بولین نباشد، برای محاسبه نتیجه، به مقدار بولین تبدیل می‌شود. برای مثال عدد 1 به عنوان `true` و عدد 0 به عنوان `false` در نظر گرفته می‌شود.

```
if (1 || 0) { // works just like if( true || false )
  alert( 'truthy!' );
}
```

اکثر اوقات، عملگر `||` به همراه `if` برای محاسبه شرط به کار می‌رود. برای مثال:

```
let hour = 9;
if (hour < 10 || hour > 18) {
  alert( 'The office is closed.' );
}
```

می‌توانیم تعداد بیشتری شرط به مثال فوق اضافه کنیم:

```
let hour = 12;
let isWeekend = true;
if (hour < 10 || hour > 18 || isWeekend) {
  alert( 'The office is closed.' ); // it is the weekend
}
```

@alithecodeguy

عملگر || (OR) به دنبال اولین مقدار truthy است.

منطقی که در بالا توضیح داده شد ، کاربرد کلاسیک آن است . حالا می‌خواهیم در باره ویژگی های اضافه آن در جاوا اسکریپت صحبت کنیم . مثال زیر را ببینید :

```
result = value1 || value2 || value3;
```

عملگر || به ترتیب مراحل زیر را انجام می‌دهد :

- عملوند ها را از چپ به راست محاسبه می‌کند .
 - هر عملوند را به مقدار بولین تبدیل می‌کند . اگر نتیجه true بود ، محاسبه را متوقف می‌کند و مقدار اصلی عملوند را بر می‌گرداند .
 - اگر همه عملوند ها محاسبه شدند و مقدار همگی آنها false بود ، آخرین عملوند را بر می‌گرداند .
- مقداری که بر می‌گردد به شکل اصلی خودش قبل از تبدیل شدن است .
- به عبارت دیگر ، عملگر || اولین مقدار truthy یا آخرین مقدار (در صورت که همگی false باشند) را بر می‌گرداند .
- مثال :

```
alert( 1 || 0 ); // 1 (1 is truthy)
alert( null || 1 ); // 1 (1 is the first truthy value)
alert( null || 0 || 1 ); // 1 (the first truthy value)
alert( undefined || null || 0 ); // 0 (all falsy, returns the last value)
```

ویژگی فوق منجر به کاربرد جالبی در مقایسه با استفاده کلاسیک از آن می‌شود .

● پیدا کردن اولین مقدار truthy از لیستی از متغیرها یا عبارات

برای مثال ، سه متغیر firstname ، lastname و nickname با مقادیر دلخواه داریم .

می‌خواهیم از عملگر || به گونه ای استفاده کنیم که اولین متغیر حاوی دیتا را نشان دهد و در صورتی که هیچ یک شامل دیتا نبودند ، عبارت Anonymous را نشان دهد .

```
let firstName = "";
let lastName = "";
let nickName = "SuperCoder";
alert( firstName || lastName || nickName || "Anonymous"); // SuperCoder
```

اگر همه متغیرها یا عبارات falsy بودند ، رشته Anonymous نمایش داده می‌شد .

• ارزیابی اتصال کوتاه (Short-circuit evaluation)

ویژگی دیگر `||` ، ارزیابی اتصال کوتاه نامیده می‌شود و معنای آن این است عملگر `||` تا جایی محاسبات را انجام می‌دهد که اولین مقدار `truthy` برسد ، سپس مقدار یافت شده فوراً برگردانده شده و باقی عبارات ، دست نخورده رها می‌گردد. اهمیت این ویژگی زمانی مشخص می‌گردد که عملوند مورد نظر صرفاً یک مقدار خالی نبوده بلکه یک عبارت با ویژگی‌های دیگر باشد ، مانند انتساب یا فراخوانی یک `function` . به عنوان مثال :

```
true || alert("not printed");
```

```
false || alert("printed");
```

در خط اول ، عملگر `||` ، پس از رسیدن به مقدار `true` محاسبات را متوقف کرده پس `alert` فراخوانی نمی‌شود .

گاهی اوقات برنامه نویسان از این ویژگی استفاده می‌کنند تا دستورات فقط زمانی اجرا شود که قسمت چپ عملگر `||` مقدار `falsy` داشته باشد .

@alithecodeguy

(AND) &&

عملگر `AND` با دو آمپرساند به شکل `&&` نمایش داده می‌شود .

```
result = a && b;
```

در برنامه نویسی کلاسیک ، `AND` تنها در صورتی مقدار `true` را برمی‌گرداند که هر دو عملوند `true` باشند در غیر اینصورت `false` برمی‌گرداند .

```
alert( true && true ); // true
```

```
alert( false && true ); // false
```

```
alert( true && false ); // false
```

```
alert( false && false ); // false
```

مثال :

```
let hour = 12;
```

```
let minute = 30;
```

```
if (hour == 12 && minute == 30) {
```

```
    alert( 'The time is 12:30' );
```

```
}
```

مانند عملگر `||` ، هر مقداری را می‌شود به عنوان عملوند `AND` استفاده کرد .

```
if (1 && 0) {...}
```

عملگر && (AND) به دنبال اولین مقدار falsy است.

مثال زیر را ببینید :

```
result = value1 && value2 && value3;
```

عملگر && به ترتیب مراحل زیر را انجام می دهد :

- عملوند ها را از چپ به راست محاسبه می کند .
- هر عملوند را به مقدار بولین تبدیل می کند . اگر نتیجه false بود ، محاسبه را متوقف می کند و مقدار اصلی عملوند را بر می گرداند .

- اگر همه عملوند ها محاسبه شدند و مقدار همگی آنها true بود ، آخرین عملوند را بر می گرداند .

به عبارت دیگر ، عملگر && اولین مقدار falsy یا آخرین مقدار (در صورت که همگی true باشند) را بر می گرداند .

قوانین فوق مشابه قوانین || هستند . تفاوت آنها این است که AND اولین مقدار falsy و OR اولین مقدار truthy را بر می گرداند . مثال :

```
// if the first operand is truthy,  
// AND returns the second operand:  
alert( 1 && 0 ); // 0  
alert( 1 && 5 ); // 5  
// if the first operand is falsy,  
// AND returns it. The second operand is ignored  
alert( null && 5 ); // null  
alert( 0 && "no matter what" ); // 0
```

همچنین می توانیم چندین مقدار را در یک خط وارد کنیم :

```
alert( 1 && 2 && null && 3 ); // null
```

هنگامی که همه مقادیر truthy باشند ، مقدار آخر برگردانده می شود :

```
alert( 1 && 2 && 3 ); // 3, the last one
```

اولویت عملگر && بالاتر از || است .

پس در واقع عبارت c && d || (a && b) یعنی (c && d) || (a && b)

|| یا && را جایگزین if نکنید.

مثال :

```
let x = 1;
```

```
(x > 0) && alert( 'Greater than zero!');
```

عبارت سمت راست && تنها هنگامی اجرا می شود که مقدار $x > 0$ ، true بشود. بنابراین عبارت فوق را به این شکل نیز می شود نوشت :

```
let x = 1;
```

```
if (x > 0) alert( 'Greater than zero!');
```

اگر چه استفاده از && کوتاه تر است ولی خوانایی کد را کاهش می دهد. بنابراین پیشنهاد می شود هر عملگر را تنها برای منظوری که برای آن ساخته شده است استفاده کنید : if هنگامی که به if نیاز داریم و && هنگامی که به AND نیاز داریم.

@alithecodeguy

! (NOT)

عملگر منطقی NOT با علامت ! نشان داده می شود.

```
result = !value;
```

این عملگر تنها یک عملوند می گیرد و به ترتیب مراحل زیر را انجام می دهد :

- عملوند را به مقدار منطقی آن یعنی true یا false تبدیل می کند.

- قرینه آن را بر می گرداند.

برای مثال :

@alithecodeguy

```
alert( !true ); // false
```

```
alert( !0 ); // true
```

از دو علامت !! برای تبدیل یک مقدار به مقدار بولین آن استفاده می کنند :

```
alert( !! "non-empty string" ); // true
```

```
alert( !! null ); // false
```

اولین ! قرینه مقدار بولین مقدار داخا پرانتز را برگردانده و ! دوم ، مقدار بازگردانده شده را مجددا قرینه می کند. در نهایت با استفاده از دو علامت !! یک مقدار به مقدار منطقی آن تبدیل می شود. اولویت عملگر ! بالاتر از سایر عملگرهای منطقی است. برای تبدیل یک مقدار به مقدار منطقی آن استفاده از روش زیر خواناتر بوده و پیشنهاد می گردد :

```
alert( Boolean("non-empty string") ); // true
```

```
alert( Boolean(null) ); // false
```