مقایسه ها

از ریاضیات ، عملگرهای مقایسه ای زیادی را می‌شناسیم .

در جاوا اسکریپت آنها به شکل زیر نوشته می شوند :

- بزرگتر / کوچکتر از : a>b و a<b
- بزرگتر مساوی / کوچکتر مساوی : a>=b و a<=b
- برابری : a==b توجه داشته باشید که علامت == ، برای مقایسه برابری استفاده شده در صورتی که = برای انتساب استفاده می‌شود .
- نابرابری : a!=b

در این آموزش راجع به انواع عملگرهای مقایسه ای و خصوصیات آنها و همچنین نحوه پیاده سازی آنها در جاوا اسکریپت صحبت خواهیم نمود .

نتیجه مقایسه ، یک مقدار بولین است .

نتیجه تمام عملگرهای مقایسه ای یک مقدار بولین است :

- true : به معنای "بله" ، "صحیح" و یا "درست" است .
- false : به معنای "خیر" ، "اشتباه" و یا "ناصحیح" است .

برای مثال :

```
alert( 2 > 1 );  // true (correct)
alert( 2 == 1 ); // false (wrong)
alert( 2 != 1 ); // true (correct)
```

نتیجه یک مقایسه مانند هر مقدار دیگری می‌تواند به یک متغیر انتساب داده شود :

```
let result = 5 > 4; // assign the result of the comparison
alert( result ); // true
```

مقایسه رشته ای

برای مقایسه دو رشته با یکدیگر ، جاوا اسکریپت از مفهوم dictionary یا lexicographical استفاده می‌کند . به عبارت دیگر ، رشته ها کاراکتر به کاراکتر با یکدیگر مقایسه می شوند . برای مثال :

```
alert( 'Z' > 'A' ); // true
alert( 'Glow' > 'Glee' ); // true
alert( 'Bee' > 'Be' ); // true
```

الگوریتم مقایسه دو رشته ساده است :

I. اولین کاراکتر دو رشته با یکدیگر مقایسه می‌شود .

II. اگر اولین کاراکتر رشته اول ، بزرگتر (کوچکتر) از کاراکتر اول رشته دوم بود ، پس رشته اول بزرگتر (کوچکتر) از رشته دوم است .

III. اگر کاراکتر اول دو رشته برابر بود ، دو کاراکتر بعدی را به همین شیوه مقایسه می کند .

IV. تا به پایان رسیدن یکی از رشته ها ، مراحل فوق را انجام می دهد .

۷. اگر دو رشته طول یکسانی داشتند آن دو رشته برابرند . در غیر اینصورت رشته بلندتر ، بزرگتر است .
در مثال فوق ، نتیجه مقایسه "A"<"Z" در اولین مقایسه بدست می آید چرا که هر دو یک کاراکتر دارند ولی مقایسه دو عبارت "Glow" و "Gloe" کاراکتر به کاراکتر انجام می شود .

ترتیب مقایسه کلمات ، مانند یک دیکشنری واقعی نیست و براساس unicode است . به عنوان مثال ، کوچکی و بزرگی حروف مهم هستند و A با a برابر نیست . چرا؟ چون حروف کوچک ایندکس بالاتری نسبت به حروف بزرگ در جدول یونیکد دارند . در فصل Strings بیشتر در این مورد صحبت خواهیم کرد .

مقایسه نوع های مختلف

وقتی مقادیری که قرار است مقایسه شوند از نوع های مختلف باشند ، جاوا اسکریپت این مقادیر را به عدد تبدیل می‌کند . برای مثال :

```
alert( '2' > 1 ); // true, string '2' becomes a number 2
```

```
alert( '01' == 1 ); // true, string '01' becomes a number 1
```

برای مقادیر بولین ، true تبدیل به 1 و false تبدیل به 0 می شود . برای مثال :

```
alert( true == 1 ); // true
```

```
alert( false == 0 ); // true
```

یک سوژه جالب!

در جاوا اسکریپت این امکان وجود دارد که دو مقدار برابر باشند ولی مقدار بولین متفاوتی داشته باشند!

مثال زیر را در نظر داشته باشید :

```
let a = 0;
alert( Boolean(a) ); // false

let b = "0";

alert( Boolean(b) ); // true

alert(a == b); // true!
```


از نظر جاوا اسکریپت این نتیجه بسیار هم عادی است!

دو مقدار بالا از دو نوع متفاوت می باشد ، پس هنگام مقایسه با یکدیگر ، جاوا اسکریپت آنها را به عدد تبدیل می کند ، بنابراین "0" به 0 و 0 نیز به همان شکل می‌ماند و این در صورتی است که در تبدیل بولی ، "0" به true و 0 به false تبدیل می شود .

برابری سخت گیرانه! (strict)

عملگر "برابری" معمولی ، یک مشکل دارد و آن این است که نمی‌تواند بین 0 و false تمایز قایل شود .

```
alert( 0 == false ); // true
```

مشابه این اتفاق برای رشته های خالی نیز رخ می دهد :

```
alert( '' == false ); // true
```

دلیل موارد بالا این است که هنگامی که مقادیر از دو نوع متفاوت را بوسیله عملگر برابری معمولی (==) مقایسه می کنیم ، مقادیر مورد نظر به نوع عددی تبدیل می شوند . بنابراین رشته خالی و false ، هر دو به صفر تبدیل می شوند .
اگر بخواهیم بین 0 و false تمایز قایل شویم ،چکار باید کنیم؟
عملگر برابری سختگیرانه (strict equality) که با علامت === نشان داده می شود ، برابری دو مقدار متفاوت را بدون تبدیل نوع انجام می‌دهد . برای مثال :

```
alert( 0 === false ); // false, because the types are different
```

همچنین عملگری به نام نابرابری سختگیرانه نیز وجود دارد که با علامت !== نمایش داده می‌شود . نوشتن عملگرهای سختگیرانه شاید طولانی تر به نظر بیاید ولی در واقع از بروز خطاهای احتمالی در آینده جلوگیری می کند .

مقایسه با null و undefined

هنگام مقایسه null و undefined با سایر مقادیر ، شاهد رفتار دور از انتظاری خواهیم بود .

مقایسه بوسیله عملگر برابری سختگیرانه :

```
alert( null === undefined ); // false
```

مقایسه بوسیله عملگر برابری معمولی :

```
alert( null == undefined ); // true
```

در ریاضیات و سایر عملگرهای مقایسه ای =< => < > :

null و undefined به عدد تبدیل می شوند . null به صفر و undefined به NaN تبدیل می شود .

نتیجه غیر منتظره : مقایسه null با صفر

```
alert( null > 0 );  // (1) false
alert( null == 0 ); // (2) false
alert( null >= 0 ); // (3) true
```

از نظر ریاضی ، نتیجه فوق عجیب است .

بنا بر خط آخر ، null بزرگتر یا مساوی صفر است بنابراین باید حداقل در یکی از خطوط بالاتر از آن ، true باشد ولی در هر دو false است .

دلیل رفتار فوق این است که عملگر برابری معمولی (==) رفتاری متفاوت از سایر عملگرهای مقایسه ای دارد . مقایسه ها null را به عدد تبدیل می کنند و با آن به شکل 0 برخورد می کنند به همین خاطر نتیجه null >= 0 ، true شده و نتیجه null > 0 ، false می شود . از طرف دیگر ، در برابری معمولی ، null و undefined فقط برابر یکدیگر بوده و با هیچ مقداری دیگه برابر نیستند . به همین خاطر null == 0 ، false می شود .

undefined غیر قابل قیاس است .

```
alert( undefined > 0 ); // false (1)
alert( undefined < 0 ); // false (2)
alert( undefined == 0 ); // false (3)
```

نتایج فوق به دلایل زیر بدست آمده است :

- نتیجه عبارت 1 و 2 ، false می شود چرا که undefined به NaN تبدیل شده و NaN مقدار مخصوصی است که در هر عبارتی false بر می گرداند .
- عبارت سوم نیز false بر می گرداند چرا که undefined فقط با null و undefined برابر است .

پرهیز از بروز مشکل :

برای چه از مثال های فوق استفاده کردیم؟ آیا حتما لازم است که همه رفتارها در هر حالتی را بدانیم؟ حقیقتا خیر! ولی در طول زمان به تدریج با آنها آشنا خوهید شد ، ولی برای جلوگیری از بروز مشکل ،دو قانون کلی وجود دارد :

- در همه عملگرهای مقایسه ای با undefined و null به جز === ، دقت بیشتری به خرج دهید .
- عملگرهای مقایسه ای را همراه با متغیری که مقدار آن ممکن است undefined و یا 0 شود استفاده نکنید مگر اینکه کاملا مطمن باشید چه اتفاقی ممکن است رخ دهد .اگر متغیری این مقادیر را داشته باشد ، جداگانه آنها را بررسی کنید .

خلاصه :

- عملگرهای مقایسه ای مقدار بولین بر می گردانند .
- رشته ها ، حرف به حرف ، مشابه دیکشنری ها مقایسه می شوند .
- هنگامی که نوع مقادیر متفاوت داشته باشند ، قبل از مقایسه به عدد تبدیل می شوند (به جز ===) .
- مقدار null و مقدار undefined تنها برابر یکدیگر هستند و با هیچ مقدار دیگری برابر نیستند .
- هنگام استفاده از عملگر های مقایسه ای مراقب مقدارهای null/undefined باشید .