

نوع های داده (Data types)

@alithecodeguy

یک مقدار در جاوا اسکریپت همیشه نوع مشخصی دارد. مثلاً رشته است یا عدد است یا ...
هشت نوع داده پایه ای در جاوا اسکریپت وجود دارد. در اینجا ، ما به طور گذرا آنها را بررسی می کنیم و در فصل های بعد در مورد هر کدام به طور مفصل صحبت خواهیم کرد. هر نوع داده را در یک متغیر می توان قرار داد. به عنوان مثال ، یک متغیر می تواند از نوع رشته باشد ولی بعداً روی آن مقدار عددی ذخیره کرد.

```
// no error
```

```
let message = "hello";
```

```
message = 123456;
```

زبان هایی که چنین اجازه ای را می دهند (مانند جاوا اسکریپت) ، زبان های پویا (dynamic) نامیده می شوند.

نوع عددی (Number)

```
let n = 123;
```

```
n = 12.345;
```

نوع عددی ، نمایشگر دو نوع اعداد صحیح و اعداد اعشاری می باشد.

عملگرهای متفاوتی برای اعداد وجود دارند مانند ضرب (*) ، تقسیم (/) ، جمع (+) و تفریق (-) و ...

در کنار اعداد معمولی ، مقادیر عددی مخصوصی نیز وجود دارند که به این دسته تعلق دارند : Infinity , -Infinity , NaN

Infinity : نشان دهنده بینهایت در ریاضیات ∞ است و یک مقدار مخصوص است که از هر عددی بزرگتر است. با تقسیم هر عددی به صفر این عدد حاصل می شود :

```
alert( 1 / 0 ); // Infinity
```

یا به طور مستقیم می شود آن را صدا زد :

```
alert( Infinity ); // Infinity
```

NaN : نشان دهنده یک خطای محاسباتی است و نتیجه عملگر ریاضی غیر درست و یا undefined است. برای مثال :

```
alert( "not a number" / 2 ); // NaN, such division is erroneous
```

NaN چسبناک است. هر عملگری که با NaN ترکیب شود ، NaN می شود.

```
alert( "not a number" / 2 + 5 ); // NaN
```

پس اگر قسمتی از یک عبارت محاسباتی NaN باشد ، کل آن عبارت را تحت تاثیر قرار می دهد.

عملگرهای ریاضی ، مطمئن عمل می کنند. محاسبات ریاضی در جاوا اسکریپت safe یا مطمئن هستند. ما هر کاری می توانیم انجام دهیم : تقسیم بر صفر ، استفاده از داده های غیر عددی در محاسبات و ...

اسکرپت هرگز با خطا متوقف نخواهد شد. در بدترین حالت ، مقدار NaN را باز می گرداند.

مقادیر عددی خاص متعلق به نوع number هستند. اگرچه در دنیای عادی ، عدد محسوب نمی شوند.

در فصل Numbers در مورد داده های عددی بیشتر صحبت خواهیم کرد.

@alithecodeguy

نوع BigInt

در جاوا اسکرپت ، نوع عددی نمی تواند مقادیر بیشتر از (9007199254740991) یا کمتر از -9007199254740991 را نمایش دهد. این محدودیتی است که با توجه به ساختار داخلی آن اعمال شده است.

در بیشتر مواقع این بازه ، کفایت می کند ولی گاهی اوقات ما به اعداد خیلی بزرگ نیاز داریم. برای مثال : رمزنگاری یا مقادیر زمان به فرم میکرو ثانیه.

BigInt اخیرا به جاوا اسکرپت اضافه شده است تا مقادیر صحیح با طول دلخواه را نمایش دهد.

یک مقدار BigInt با اضافه کردن n به انتهای یک عدد صحیح ایجاد می شود.

// the "n" at the end means it's a BigInt

```
const bigInt = 1234567890123456789012345678901234567890n;
```

مقادیر BigInt به ندرت نیاز می شوند. در فصل BigInt به صورت مفصل در مورد آن صحبت خواهیم کرد.

در حال حاضر BigInt توسط Firefox/Chrome/Edge پشتیبانی می شود ولی توسط IE/Safari پشتیبانی نمی شود.

نوع رشته ای (string)

یک رشته در جاوا اسکرپت باید توسط کوتیشن (") ، دابل کوتیشن (") ، و یا بک تیک (`) احاطه شود.

```
let str = "Hello";
```

```
let str2 = 'Single quotes are ok too';
```

```
let phrase = `can embed another ${str}`;
```

@alithecodeguy

کوتیشن یا دابل کوتیشن ، کوتیشن های ساده شناخته می شوند.

عملا جاوا اسکرپت هیچ تفاوتی بین آن دو قایل نمی شود.

بک تیک ها ، کوتیشن هایی با قابلیت های اضافه اند . آنها به ما اجازه می دهند عبارات و مقادیر مختلف را بوسیله `${...}` داخل عبارات رشته ای بگنجانیم . برای مثال :

```
let name = "John";
// embed a variable
alert(`Hello, ${name}!`); // Hello, John!
// embed an expression
alert(`the result is ${1 + 2}`); // the result is 3
```

عبارت داخل `${...}` محاسبه شده و نتیجه ، بخشی از رشته می شود . ما هر چیزی را می توانیم در آن قرار دهیم : یک متغیر به نام `name` یا یک عبارت ریاضی مانند `1+2` یا هر عبارت پیچیده دیگری .

توجه داشته باشید که این قابلیت فقط در بک تیک امکان پذیر است . باقی کوتیشن ها این قابلیت را ندارند .

```
alert( "the result is ${1 + 2}" ); // the result is ${1 + 2} (double quotes do nothing)
```

در بخش `strings` در مورد رشته ها بیشتر صحبت خواهیم کرد .

در جاوا اسکریپت ، نوع `character` وجود ندارد . در بعضی از زبان ها ، نوع خاصی به نام `character` وجود دارد که می توان حروف را در آن ذخیره کرد . برای مثال در زبان جاوا و `C` ، این نوع داده ای `char` نامیده می شود . در جاوا اسکریپت همچنین نوعی وجود ندارد . تنها یک نوع رشته ای وجود ندارد : `string` یک رشته می تواند شامل یک کاراکتر یا تعدادی بیشتری از آنها باشد .

نوع بولین (Boolean – logical type)

مقادیر منطقی دو مقدار دارند : `true` یا `false`

این نوع معمولا برای نگهداری مقادیر بلی / خیر استفاده می شود . `true` یعنی بله و `false` یعنی خیر .

برای مثال :

```
let nameFieldChecked = true; // yes, name field is checked
let ageFieldChecked = false; // no, age field is not checked
```

مقادیر `Boolean` همچنین از عبارات مقایسه ای نیز حاصل می شود .

```
let isGreater = 4 > 1;
alert( isGreater ); // true (the comparison result is "yes")
```

در فصل `Logical operators` به طور مفصل راجع به آنها صحبت خواهیم کرد .

مقدار null

مقدار مخصوص null متعلق به هیچ یک از انواع توضیح داده شده نیست. خودش به تنهایی، نوع مخصوص به خودش را دارد که فقط شامل مقدار null می شود.

```
let age = null;
```

در جاوا اسکریپت، null اشاره کننده به یک شی ناموجود یا مانند سایر زبان ها null pointer نیست. Null فقط مقدار مخصوصی است که نشانگر "هیچی"، "خالی" یا "مقدار ناشناخته" است. عبارت بالا یعنی مقدار age ناشناخته است.

@alithecodeguy

مقدار undefined

مقدار undefined نیز از سایر نوع ها مجزا است. این نیز مانند null نوع مخصوص به خودش را ایجاد می کند. undefined یعنی "مقداری منتسب نشده است".

اگر تغییری تعریف شود ولی مقداری منتسب نشود مقدار آن undefined می شود :

```
let age;
```

```
alert(age); // shows "undefined"
```

همچنین می شود به طور صریح مقدار undefined را منتسب کرد :

```
let age = 100;
```

```
// change the value to undefined
```

```
age = undefined;
```

```
alert(age); // "undefined"
```

... ولی توصیه نمی شود که این کار را انجام دهید. Null برای انتساب مقدار "empty" یا "unknown" به متغیرها استفاده می شود در حالی که undefined به عنوان مقدار پیش فرض برای متغیرهای مقدار دهی نشده شناخته می شود.

@alithecodeguy

اشیا و نمادها (Objects and Symbols)

نوع object، نوع مخصوصی است. باقی نوع ها primitive نامیده می شوند چرا که مقدار آنها تنها می تواند شامل یک چیز باشد (یک عدد، یک رشته یا هر چیز دیگری). در مقابل، از object برای ذخیره مجموعه ای از داده ها و موجودیت های پیچیده استفاده می شود.

با توجه به این موضوع، object ها رفتار مخصوصی دارند. در فصل objects در مورد آنها بیشتر صحبت خواهیم کرد.

نوع symbol برای ساخت شناسه های منحصر به فرد برای object ها استفاده می شود. بعد از object در مورد آن بیشتر صحبت می کنیم.

عملگر operator

عملگر typeof نوع داده را بر می گرداند. این عملگر مواقعی مفید خواهد بود که بخواهیم مقادیری از نوع های مختلف را پردازش کنیم یا بخواهیم سریعاً متوجه نوع داده بشویم.

به دو صورت می توان از این عملگر استفاده کرد :

• به عنوان یک عملگر : `typeof x`

• به عنوان یک `function` : `typeof(x)`

به عبارت دیگر ، چه با پرانتز چه بی پرانتز کار می کند و نتیجه یکسانی بر می گرداند.

نتیجه این عملگر رشته ای شامل نام نوع داده است :

```
typeof undefined // "undefined"
```

```
typeof 0 // "number"
```

```
typeof 10n // "bigint"
```

```
typeof true // "boolean"
```

```
typeof "foo" // "string"
```

```
typeof Symbol("id") // "symbol"
```

```
typeof Math // "object" (1)
```

```
typeof null // "object" (2)
```

```
typeof alert // "function" (3)
```

سه خط پایانی ، ممکن است نیاز به کمی توضیح داشته باشد :

• `Math` یک `object` داخلی است که شامل عملگرهای ریاضی می شود.

• نتیجه عبارت `typeof null` ، `object` می شود. این یک خطای شناخته شده در جاوا اسکریپت است که در روزهای اولیه بوجود آمده و برای حفظ سازگاری نگه داشته شده است.

• نتیجه عبارت `typeof alert` ، `function` می شود زیرا `alert` یک `function` است. در فصل های بعد راجع به `function` صحبت خواهیم کرد و همچنین خواهیم آموخت نوع مخصوص "function" وجود ندارد. `Function` ها متعلق به نوع `object` هستند ولی عبارت `typeof` رفتار متفاوتی دارد و مقدار `function` را برمی گرداند. این مورد نیز از روزهای اول در جاوا اسکریپت وجود داشت.

خلاصه

- 8 نوع پایه ای برای داده ها در جاوا اسکریپت وجود دارد :

@alithecodeguy

- number : برای اعدادی از هر نوع. هم صحیح هم اعشاری
- bigint : برای اعداد صحیحی با طول دلخواه
- string : برای رشته ها. یک رشته ممکن است شامل یک یا تعداد بیشتری کاراکتر باشد. نوع مجزایی برای کاراکترهای تکی وجود ندارد.
- boolean : برای مقادیر true یا false
- null : برای مقادیر ناشناخته. نوع مجزایی که فقط شامل خودش است.
- undefined : برای متغیرهای مقدار دهی نشده. نوع مجزایی که فقط شامل خودش است.
- object : برای ساختارهای داده ای پیچیده
- symbol : برای شناسه های منحصر به فرد
- عملگر typeof اجازه می دهد متوجه بشویم که چه نوع داده ای در متغیر ذخیره شده است.
- به ۲ حالت نوشته می شود : typeof x و typeof(x)
- رشته ای بر می گرداند شامل نام نوع داده ذخیره شده در متغیر
- برای null رشته object را بر می گرداند. این یک خطا در جاوا اسکریپت است چرا که null واقعا object نیست.

@alithecodeguy

Spechial Thanks to :

- mehdico