انواع داده ها

یک مقدار در جاوا اسکریپت همیشه از نوع خاصی است. به عنوان مثال، یک رشته یا یک عدد. هشت نوع داده اصلی در جاوا اسکریپت وجود دارد. در اینجا به طور کلی به آنها می پردازیم و در فصل های بعدی به تفصیل در مورد هر یک صحبت خواهیم کرد.

ما می توانیم هر نوع را در یک متغیر قرار دهیم. به عنوان مثال، یک متغیر می تواند در یک لحظه یک رشته باشد و سپس یک عدد را ذخیره کند:

```
// no error
let message = "hello";
message = 123456;
```

به زبان‌های برنامه‌نویسی که اجازه چنین چیزهایی را می‌دهند، مانند جاوا اسکریپت، تایپ پویا (dynamically typed) گفته می‌شود، به این معنی که انواع داده‌ها وجود دارد، اما متغیرها به هیچ یک از آنها محدود نمی‌شوند.

عدد Number

```
let n = 123;
n = 12.345;
```

نوع Number هم اعداد صحیح و هم اعداد ممیز شناور را نشان می دهد. عملیات زیادی برای اعداد وجود دارد، به عنوان مثال. ضرب *، تقسیم /، جمع +، تفریق - و غیره. علاوه بر اعداد منظم، به اصطلاح "مقادیر عددی ویژه" نیز وجود دارد که به این نوع داده تعلق دارند: Infinity، Infinity- و NaN. بی نهایت بیانگر بی نهایت ریاضی ∞ است. این یک مقدار ویژه است که از هر عددی بزرگتر است. ما می توانیم آن را در نتیجه تقسیم بر صفر بدست آوریم:

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

یا فقط به آن اشاره مستقیم کنید:

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

NaN نشان دهنده یک خطای محاسباتی است. این نتیجه یک عملیات ریاضی نادرست یا تعریف نشده است، به عنوان مثال:

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

NaN خوب نیست. هر عملیات ریاضی دیگری روی NaN،NaN را برمی‌گرداند:

```
alert( NaN + 1 ); // NaN
alert( 3 * NaN ); // NaN
alert( "not a number" / 2 - 1 ); // NaN
```

بنابراین، اگر یک NaN در جایی در یک عبارت ریاضی وجود داشته باشد، به کل نتیجه انتشار می یابد (فقط یک استثنا برای آن وجود دارد: 0 ** NaN برابر 1 است).

عملیات ریاضی ایمن هستند

انجام ریاضیات در جاوا اسکریپت "ایمن" است. ما می توانیم هر کاری انجام دهیم: تقسیم بر صفر، رشته های غیر عددی به عنوان اعداد و غیره. اسکریپت هرگز با یک خطای مهلک ("die") متوقف نمی شود. در بدترین حالت، ما NaN را به عنوان نتیجه دریافت خواهیم کرد.

مقادیر عددی ویژه به طور رسمی به نوع number تعلق دارند. البته اعداد به معنای رایج این کلمه نیستند. در بخش اعداد بیشتر در مورد کار با اعداد خواهیم دید.

BigInt

در جاوا اسکریپت، نوع number نمی تواند مقادیر صحیح بزرگتر از $(2^{53} - 1)$ (یعنی 9007199254740991) یا کمتر از $-(2^{53} - 1)$ را برای منفی ها نشان دهد. این یک محدودیت فنی ناشی از نمایش داخلی آنهاست. برای اکثر اهداف این کاملاً کافی است، اما گاهی اوقات ما به اعداد واقعاً بزرگ نیاز داریم، به عنوان مثال. برای رمزنگاری یا زمان با دقت میکروثانیه. نوع BigInt اخیراً برای نمایش اعداد صحیح با طول دلخواه به زبان اضافه شده است. یک مقدار BigInt با الحاق n به انتهای یک عدد صحیح ایجاد می شود:

```
// the "n" at the end means it's a BigInt
const bigInt = 1234567890123456789012345678901234567890n;
```

از آنجایی که به ندرت به اعداد BigInt نیاز است، ما آنها را در اینجا پوشش نمی دهیم، اما فصل جداگانه ای را به BigInt اختصاص داده ایم. زمانی که به چنین اعداد بزرگی نیاز دارید، آن را بخوانید.

مسائل مربوط به سازگاری

در حال حاضر، BigInt در Firefox/Chrome/Edge/Safari پشتیبانی می شود، اما در IE پشتیبانی نمی شود.

می توانید جدول سازگاری MDN BigInt را بررسی کنید تا بدانید کدام نسخه از یک مرورگر پشتیبانی می شود.

رشته

یک رشته در جاوا اسکریپت باید با نقل قول احاطه شود.

```
let str = "Hello";
let str2 = 'Single quotes are ok too';
let phrase = `can embed another ${str}`;
```

در جاوا اسکریپت 3 نوع نقل قول وجود دارد.

نقل قول های دوگانه: "Hello".

نقل قول تک: Hello.

بکتیک:`Hello`.

نقل قول های دوتایی و تکی نقل قول های "ساده" هستند. عملاً هیچ تفاوتی بین آنها در جاوا اسکریپت وجود ندارد. بک‌تیک‌ها نقل قول‌های «عملکرد گسترده» هستند. آنها به ما این امکان را می دهند که متغیرها و عبارات را با قرار دادن آنها در ${...} در یک رشته جاسازی کنیم، به عنوان 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 ${...} ارزیابی می شود و نتیجه بخشی از رشته می شود. ما می توانیم هر چیزی را در آنجا قرار دهیم: یک متغیر مانند نام یا یک عبارت حسابی مانند 1 + 2 یا چیز پیچیده تر. لطفاً توجه داشته باشید که این کار فقط در بکتیک قابل انجام است. نقل قول های دیگر این قابلیت جاسازی را ندارند!

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

ما رشته ها را به طور کامل در فصل رشته ها پوشش خواهیم داد.

هیچ نوع character وجود ندارد.

در برخی از زبان‌ها، برای یک کاراکتر یک نوع « character » خاص وجود دارد. به عنوان مثال در زبان C و در جاوا char نامیده می شود. در جاوا اسکریپت، این نوع وجود ندارد. فقط یک نوع وجود

دارد: رشته. یک رشته ممکن است از صفر کاراکتر (خالی باشد)، یک کاراکتر یا تعداد زیادی از آنها تشکیل شده باشد.

بولی (نوع منطقی)

نوع بولی تنها دو مقدار دارد: true و false. این نوع معمولاً برای ذخیره مقادیر yes/no استفاده می‌شود: true به معنای "بله، صحیح" و false به معنای "نه، نادرست" است. برای مثال:

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

مقادیر بولی نیز در نتیجه مقایسه ها به دست می آیند:

```
let isGreater = 4 > 1;

alert( isGreater ); // true (the comparison result is "yes")
```

مقدار Null

مقدار null ویژه به هیچ یک از انواع توصیف شده در بالا تعلق ندارد. یک نوع جداگانه برای خود تشکیل می دهد که فقط حاوی مقدار null است:

```
let age = null;
```

در جاوا اسکریپت، null مانند برخی از زبان‌های دیگر «اشاره به یک شی غیر موجود» یا «اشاره‌گر تهی» نیست. این فقط یک مقدار ویژه است که نشان دهنده "هیچ چیز"، "خالی" یا "مقدار ناشناخته" است. کد بالا بیان می کند که سن نامشخص است.

مقدار undefined

مقدار ویژه undefined نیز جدا است. این یک نوع خود را درست می کند، درست مانند null. معنای تعریف نشده «ارزش تعیین نشده» است. اگر متغیری اعلان شود، اما تخصیص داده نشود، مقدار آن undefined است:

```
let age;

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

از نظر فنی، می توان به طور صریح undefined را به یک متغیر اختصاص داد:

```
let age = 100;

// change the value to undefined
age = undefined;

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

... اما ما انجام این کار را توصیه نمی کنیم. به طور معمول، از null برای اختصاص یک مقدار خالی یا ناشناخته به یک متغیر استفاده می‌شود، در حالی که undefined به عنوان مقدار اولیه پیش‌فرض برای چیزهای تخصیص‌نخورده رزرو شده است.

اشیاء و نمادها

نوع شی خاص است. همه نوع داده ها، اولیه (نوع داده اولیه) نامیده می شوند زیرا مقادیر آنها فقط می تواند حاوی یک چیز باشد (خواه یک رشته یا یک عدد یا هر چیز دیگری). در مقابل، اشیاء برای ذخیره مجموعه ای از داده ها و موجودیت های پیچیده تر استفاده می شوند. از آنجایی که اشیاء بسیار مهم هستند، مستحق برخورد ویژه هستند. ما بعداً در فصل اشیاء، پس از یادگیری بیشتر در مورد نوع داده اولیه‌ها، به آنها خواهیم پرداخت.

نوع نماد برای ایجاد شناسه های منحصر به فرد برای اشیا استفاده می شود. ما باید به خاطر کامل بودن آن را در اینجا ذکر کنیم، اما جزئیات را نیز تا شناخت اشیاء به تعویق بیندازیم.

عملگر `typeof`

عملگر typeof نوع آرگومان را برمی گرداند. زمانی مفید است که بخواهیم مقادیر انواع مختلف را متفاوت پردازش کنیم یا فقط بخواهیم یک بررسی سریع انجام دهیم. فراخوانی به typeof x رشته ای را با نام نوع باز می گرداند:

```
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 آخر ممکن است نیاز به توضیح بیشتری داشته باشد:

- Math یک شی داخلی است که عملیات ریاضی را ارائه می دهد. آن را در فصل اعداد خواهیم آموخت. در اینجا، فقط به عنوان نمونه ای از یک شی عمل می کند.
- نتیجه typeof null، object است. این یک خطای رسمی شناخته شده در typeof است که از روزهای اولیه جاوا اسکریپت می آید و برای سازگاری نگهداری می شود. قطعا null یک شی نیست. این یک مقدار ویژه با نوع جداگانه خود است. رفتار typeof در اینجا اشتباه است.
- نتیجه typeof alert، function است، زیرا alert یک تابع است. ما توابع را در فصل های بعدی مطالعه خواهیم کرد، جایی که همچنین خواهیم دید که نوع خاصی از "تابع" در جاوا اسکریپت وجود ندارد. توابع متعلق به نوع شی هستند. اما typeof با آنها رفتار متفاوتی دارد و "عملکرد" را برمی‌گرداند. این نیز از روزهای اولیه جاوا اسکریپت می آید. از نظر فنی، چنین رفتاری صحیح نیست، اما در عمل می تواند درست باشد.

نحو `typeof(x)`

همچنین ممکن است با نحو دیگری برخورد کنید: typeof(x). این همان `typeof x` است.

برای روشن کردن: typeof یک عملگر است، نه یک تابع. پرانتزهای اینجا بخشی از typeof نیستند. این نوعی پرانتز است که برای گروه بندی ریاضی استفاده می شود.

معمولاً چنین پرانتزهایی حاوی یک عبارت ریاضی مانند (2 + 2) هستند، اما در اینجا فقط یک آرگومان x دارند. از نظر نحوی، آنها اجازه می دهند از فاصله بین نوع عملگر و آرگومان آن جلوگیری شود و برخی افراد آن را دوست دارند. برخی از افراد typeof(x) را ترجیح می دهند، اگرچه نحو `typeof x` بسیار رایج تر است.

خلاصه

8 نوع داده اصلی در جاوا اسکریپت وجود دارد.

- number برای اعداد از هر نوع: عدد صحیح یا ممیز شناور، اعداد صحیح با $\pm 2^{53} - 1$ محدود می شوند.
- bigint برای اعداد صحیح با طول دلخواه است.
- `String` برای رشته یک رشته ممکن است صفر یا چند کاراکتر داشته باشد، هیچ نوع تک نویسه جداگانه ای وجود ندارد.
- `Boolean` برای درست/نادرست.
- null برای مقادیر ناشناخته - یک نوع مستقل که یک مقدار واحد دارد null.
- undefined برای مقادیر تخصیص نیافته - یک نوع مستقل که دارای یک مقدار واحد تعریف نشده است.
- `Object` برای ساختارهای داده پیچیده تر.
- `Symbol` برای شناسه های منحصر به فرد.

عملگر typeof به ما اجازه می دهد تا ببینیم کدام نوع در یک متغیر ذخیره می شود.

- معمولاً به عنوان typeof x استفاده می شود، اما typeof(x) نیز امکان پذیر است.
- رشته ای را با نام نوع، مانند "رشته" برمی گرداند.
- برای `null` مقدار "object"- این یک خطا در زبان است، در واقع یک شی نیست.

در فصل‌های بعدی، ما روی مقادیر اولیه تمرکز می‌کنیم و زمانی که با آنها آشنا شدیم، به سراغ اشیا می‌رویم.