متغیرها

اغلب اوقات، یک برنامه جاوا اسکریپت باید با اطلاعات کار کند. در اینجا دو نمونه وجود دارد: یک فروشگاه آنلاین - اطلاعات ممکن است شامل کالاهای فروخته شده و یک سبد خرید باشد. یک برنامه چت - اطلاعات ممکن است شامل کاربران، پیام ها و موارد دیگر باشد. برای ذخیره این اطلاعات از متغیرها استفاده می شود.

یک متغیر

یک متغیر یک قسمتی از حافظه برای ذخیره سازی داده ها است. ما می‌توانیم از متغیرها برای ذخیره موارد مختلف، بازدیدکنندگان و سایر داده‌ها استفاده کنیم.

برای ایجاد متغیر در جاوا اسکریپت، از کلمه کلیدی let استفاده میکنیم.

عبارت زیر یک متغیر با نام message ایجاد می کند (به عبارت دیگر: اعلام می کند):

```
let message;
```

اکنون می‌توانیم با استفاده از عملگر انتساب داده را در آن قرار دهیم:

```
let message;

message = 'Hello'; // store the string 'Hello' in the variable named message
```

رشته اکنون در ناحیه حافظه مرتبط با متغیر ذخیره می شود. ما می توانیم با استفاده از نام متغیر به آن دسترسی پیدا کنیم:

```
let message;
message = 'Hello!';

alert(message); // shows the variable content
```

برای مختصر، می‌توانیم اعلان متغیر و انتساب را در یک خط ترکیب کنیم:

```
let message = 'Hello!'; // define the variable and assign the value

alert(message); // Hello!
```

همچنین می توانیم چندین متغیر را در یک خط اعلام کنیم:

```
let user = 'John', age = 25, message = 'Hello';
```

ممکن است کوتاه‌تر به نظر برسد، اما ما آن را توصیه نمی‌کنیم. برای خوانایی بهتر، لطفاً از یک خط برای هر متغیر استفاده کنید.

اعلان چند خطی کمی طولانی تر است، اما خواندن آن آسان تر است:

```
let user = 'John';
let age = 25;
let message = 'Hello';
```

برخی افراد نیز چندین متغیر را در این سبک چند خطی تعریف می کنند:

```
let user = 'John',
  age = 25,
  message = 'Hello';
```

... یا حتی به سبک "اول کاما":

```
let user = 'John'
  , age = 25
  , message = 'Hello';
```

از نظر فنی، همه این اعلان ها یک کار را انجام می دهند. بنابراین، این یک موضوع سلیقه و زیبایی شناسی شخصی است.

var به جای let

در اسکریپت های قدیمی، ممکن است کلمه کلیدی دیگری نیز پیدا کنید: var به جای :let

کلمه کلیدی var تقریباً همان let است. همچنین یک متغیر را اعلام می‌کند، اما به روشی «قدیمی» کمی متفاوت است.

```
var message = 'Hello';
```

تفاوت های ظریفی بین let و var وجود دارد، اما آنها هنوز برای ما مهم نیستند. ما آنها را به طور مفصل در فصل old var پوشش خواهیم داد.

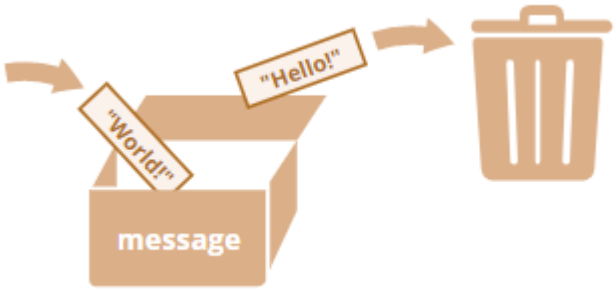
یک قیاس واقعی

ما می‌توانیم به راحتی مفهوم «متغیر» را درک کنیم اگر آن را به‌عنوان «جعبه‌ای» برای داده‌ها با برچسبی با نام منحصربه‌فرد روی آن تصور کنیم. به عنوان مثال، متغیر message را می توان به عنوان جعبه‌ای با برچسب "پیام" با مقدار Hello! تصور کرد. در آن:

ما می توانیم هر مقداری را در جعبه قرار دهیم. ما همچنین می توانیم آن را هر چند بار که بخواهیم تغییر دهیم:

```
1  let message;
2
3  message = 'Hello!';
4
5  message = 'World!'; // value changed
6
7  alert(message);
```

هنگامی که مقدار تغییر می کند، داده های قدیمی از متغیر حذف می شوند:



ما همچنین می توانیم دو متغیر را اعلام کنیم و داده ها را از یکی به دیگری کپی کنیم.

```
let hello = 'Hello world!';

let message;

// copy 'Hello world' from hello into message
message = hello;

// now two variables hold the same data
alert(hello); // Hello world!
alert(message); // Hello world!
```

دوبار اعلام خطا باعث ایجاد خطا می شود

یک متغیر باید فقط یک بار اعلام شود. اعلان مکرر همان متغیر یک خطا است:

```
let message = "This";

// repeated 'let' leads to an error
let message = "That"; // SyntaxError: 'message' has already been declared
```

بنابراین، باید یک متغیر را یک بار اعلام کنیم و سپس بدون let به آن مراجعه کنیم.

زبان های تابعی

جالب است بدانید که زبان های برنامه نویسی تابعی مانند Scala یا Erlang وجود دارند که تغییر مقادیر متغیر را ممنوع می کنند. در چنین زبان‌هایی، زمانی که مقدار «در جعبه» ذخیره می‌شود، برای همیشه وجود دارد. اگر بخواهیم چیز دیگری را ذخیره کنیم، زبان ما را مجبور می‌کند یک جعبه جدید ایجاد کنیم (یک متغیر جدید را اعلام کنیم). ما نمی توانیم از قدیمی استفاده مجدد کنیم.

اگرچه ممکن است در نگاه اول کمی عجیب به نظر برسد، اما این زبان ها کاملاً قادر به توسعه هستند. بیشتر از آن، حوزه هایی مانند محاسبات موازی وجود دارد که این محدودیت مزایای خاصی را به همراه دارد. مطالعه چنین زبانی (حتی اگر قصد ندارید به زودی از آن استفاده کنید) برای وسعت بخشیدن به ذهنتان توصیه می شود.

نامگذاری متغیر

دو محدودیت برای نام متغیرها در جاوا اسکریپت وجود دارد:

1. نام باید فقط شامل حروف، اعداد یا نمادهای $ و _ باشد.
2. کاراکتر اول نباید یک رقم باشد.

نمونه هایی از نام های معتبر:

```
let userName;
let test123;
```

هنگامی که نام شامل چندین کلمه باشد، CamelCase معمولا استفاده می شود. یعنی: کلمات یکی پس از دیگری می روند، هر کلمه به جز ابتدا با یک حرف بزرگ شروع می شود: myVeryLongName.
جالب اینجاست که علامت دلار '$' و زیرخط '_' نیز می توانند در نام ها استفاده شوند. آنها نمادهای منظمی هستند، درست مانند حروف، بدون هیچ معنای خاصی. این اسامی معتبر است:

```
let $ = 1; // declared a variable with the name "$"
let _ = 2; // and now a variable with the name "_"

alert($ + _); // 3
```

نمونه هایی از نام متغیرهای نادرست:

```
let 1a; // cannot start with a digit

let my-name; // hyphens '-' aren't allowed in the name
```

نکته مهم

متغیرهایی به نام apple و APPLE دو متغیر متفاوت هستند.

حروف غیر لاتین مجاز است، اما توصیه نمی شود

استفاده از هر زبانی، از جمله حروف سیریلیک یا حتی هیروگلیف، مانند این امکان پذیر است:

```
let имя = '...';
let 我 = '...';
```

از نظر فنی، هیچ خطایی در اینجا وجود ندارد. چنین نام هایی مجاز هستند، اما یک قرارداد بین المللی برای استفاده از انگلیسی در نام های متغیر وجود دارد. حتی اگر در حال نوشتن یک اسکریپت کوچک باشیم، ممکن است عمر طولانی در پیش داشته باشد. افراد کشورهای دیگر ممکن است نیاز داشته باشند که آن را بخوانند.

اسامی رزرو شده

لیستی از کلمات رزرو شده وجود دارد که نمی توان از آنها به عنوان نام متغیر استفاده کرد زیرا توسط خود زبان استفاده می شود. به عنوان مثال: let، class، return و function رزرو شده اند. کد زیر یک خطای نحوی می دهد:

```
let let = 5; // can't name a variable "let", error!
let return = 5; // also can't name it "return", error!
```

یک تمرین بدون `use strict`

معمولاً قبل از استفاده از متغیر باید آن را تعریف کنیم. اما در زمان های قدیم، از نظر فنی امکان ایجاد یک متغیر صرفاً با انتساب مقدار بدون استفاده از let وجود داشت. اگر از اسکریپت های خود برای حفظ سازگاری با اسکریپت های قدیمی استفاده نکنیم، همچنان کار می کند.

```
// note: no "use strict" in this example

num = 5; // the variable "num" is created if it didn't exist

alert(num); // 5
```

این یک عمل بد است و باعث خطا در حالت `use strict` می شود:

```
"use strict";

num = 5; // error: num is not defined
```

ثابت ها

برای اعلام یک متغیر ثابت (غیرقابل تغییر)، به جای let از const استفاده کنید:

```
const myBirthday = '18.04.1982';
```

متغیرهای اعلام شده با استفاده ازconst ، ثابت نامیده می شوند. آنها را نمی توان دوباره تغییر داد. تلاش برای انجام این کار باعث خطا می شود:

```
const myBirthday = '18.04.1982';

myBirthday = '01.01.2001'; // error, can't reassign the constant!
```

هنگامی که یک برنامه نویس مطمئن است که متغیری هرگز تغییر نخواهد کرد، می تواند آن را با const اعلام کند تا تضمین کند و به وضوح این واقعیت را به همه منتقل کند.

ثابت های با حروف بزرگ

به خاطر سپردن مقادیری که قبل از اجرا شناخته شده اند دشوار است بخاطر همین روش گسترده ای برای استفاده از ثابت ها به عنوان نام مستعار وجود دارد. چنین ثابت هایی با استفاده از حروف بزرگ و زیرخط نامگذاری می شوند. برای مثال، بیایید برای رنگ‌ها در قالب‌های به اصطلاح «وب» (هگزادسیمال) ثابت بسازیم:

```
const COLOR_RED = "#F00";
const COLOR_GREEN = "#0F0";
const COLOR_BLUE = "#00F";
const COLOR_ORANGE = "#FF7F00";

// ...when we need to pick a color
let color = COLOR_ORANGE;
alert(color); // #FF7F00
```

مزایایی این روش:

- یادآوری COLOR_ORANGE بسیار ساده‌تر از "#FF7F00" است.
- اشتباه تایپ "#FF7F00" بسیار ساده تر از COLOR_ORANGE است.
- هنگام خواندن کد، COLOR_ORANGE بسیار معنادارتر از #FF7F00 است.

چه زمانی باید برای یک ثابت از حروف بزرگ استفاده کنیم و چه زمانی آن را به طور معمول نام گذاری کنیم؟ بیایید آن را بشکافیم. ثابت بودن فقط به این معنی است که مقدار یک متغیر هرگز تغییر نمی کند. اما ثابت هایی هستند که قبل از اجرا شناخته می شوند (مانند مقدار هگزادسیمال برای قرمز) و ثابت هایی هستند که در زمان اجرا، در طول اجرا محاسبه می شوند، اما پس از تخصیص اولیه تغییر نمی کنند. برای مثال:

```
const pageLoadTime = /* time taken by a webpage to load */;
```

مقدار pageLoadTime قبل از بارگذاری صفحه مشخص نیست، بنابراین به طور معمول نامگذاری می شود. اما همچنان ثابت است زیرا پس از انتساب تغییر نمی کند. به عبارت دیگر، ثابت‌های با نام سرمایه فقط به عنوان نام مستعار برای مقادیر «hard-coded» استفاده می‌شوند.

نام گذاری صحیح متغییرها

در مورد متغیرها، یک چیز بسیار مهم دیگر وجود دارد. نام متغیر باید معنای واضحی داشته باشد و داده‌هایی را که ذخیره می‌کند توصیف کند. نامگذاری متغیرها یکی از مهم ترین و پیچیده ترین مهارت های برنامه نویسی است. یک نگاه سریع به نام متغیرها می تواند نشان دهد که کدام کد توسط یک مبتدی در مقابل یک توسعه دهنده با تجربه نوشته شده است. در یک پروژه واقعی، بیشتر وقت صرف اصلاح و گسترش یک پایه کد موجود می شود تا اینکه چیزی کاملاً از ابتدا بنویسند. هنگامی که پس از مدتی انجام کار دیگری به کدی باز می گردیم، پیدا کردن اطلاعاتی که به خوبی برچسب گذاری شده اند بسیار آسان تر است. یا به عبارت دیگر زمانی که متغیرها نام خوبی دارند.

لطفاً قبل از اعلام یک متغیر، زمانی را صرف فکر کردن به نام مناسب آن کنید. انجام این کار به نفع شما خواهد بود. برخی از قوانینی که باید رعایت شوند عبارتند از:

- از نام‌های قابل خواندن برای انسان مانند userName یا shoppingCart استفاده کنید.
- از اختصارات یا نام های کوتاه مانند a، b، c خودداری کنید، مگر اینکه واقعاً بدانید که چه می کنید.
- نام ها را حداکثر تشریح و مختصر کنید. نمونه هایی از نام های بد data و value هستند. چنین نام هایی اطلاعاتی به توسعه دهنده نمیدهد. استفاده از آنها فقط در صورتی اشکالی ندارد که زمینه کد به طور استثنایی مشخص کند که متغیر به کدام data یا value ارجاع می دهد.
- در تیم خود و در ذهن خود بر روی شرایط توافق کنید. اگر یک بازدیدکننده سایت user نامیده می شود، باید متغیرهای مرتبط را به جای currentVisitor یا newManInTown، currentUser یا newUser نامگذاری کنیم.

ساده به نظر می رسد؟ در واقع اینطور است، اما ایجاد نام های توصیفی و مختصر متغیر در عمل چنین نیست.

استفاده مجدد یا ایجاد؟

و آخرین نکته. برخی از برنامه نویسان تنبل هستند که به جای اعلام متغیرهای جدید، تمایل به استفاده مجدد از متغیرهای موجود دارند. در نتیجه، متغیرهای آن‌ها مانند جعبه‌هایی هستند که افراد بدون تغییر استیکر، چیزهای مختلفی را درون آن می‌اندازند. الان داخل جعبه چیه؟ چه کسی می داند؟ باید نزدیکتر بیاییم و بررسی کنیم. چنین برنامه نویسانی در اعلان متغیر اندکی صرفه جویی می کنند اما ده برابر بیشتر در اشکال زدایی ضرر می کنند.

یک متغیر اضافی خوب است، نه بد. مینی‌فایرها و مرورگرهای جاوا اسکریپت به اندازه کافی کد را بهینه می‌کنند، بنابراین مشکلاتی در عملکرد ایجاد نمی‌کنند. استفاده از متغیرهای مختلف برای مقادیر مختلف حتی می تواند به موتور کمک کند تا کد شما را بهینه کند.

خلاصه

با استفاده از کلمات کلیدی var، let یا const می توانیم متغیرها را برای ذخیره داده ها اعلام کنیم.

- Let - یک اعلان متغیر مدرن است.
- Var - یک اعلان متغیر قدیمی است. معمولاً ما اصلاً از آن استفاده نمی‌کنیم، اما تفاوت‌های ظریف مربوط به اجازه در فصل قدیمی « «varرا پوشش می‌دهیم، فقط در صورت نیاز.
- Const - مانند let است، اما مقدار متغیر را نمی توان تغییر داد.

متغیرها باید به گونه‌ای نام‌گذاری شوند که به ما امکان دهد به راحتی بفهمیم چه چیزی در داخل آنها وجود دارد.

Working with variables

importance: 2

1. Declare two variables: `admin` and `name`.
2. Assign the value `"John"` to `name`.
3. Copy the value from `name` to `admin`.
4. Show the value of `admin` using `alert` (must output “John”).