متغیرها (variables)

یک برنامه جاوا اسکریپت در اکثر اوقات نیاز دارد که با اطلاعات کار کند . در اینجا ۲ نمونه را ذکر می‌کنیم :

۱– فروشگاه اینترنتی : اطلاعات مورد نیاز شامل اطلاعات کالا و سبد خرید می‌شود .

۲– برنامه چت : اطلاعات مورد نیاز شامل اطلاعات کاربران ، پیغام ها و ... است .

متغیرها برای نگهداری اینگونه اطلاعات استفاده می‌شوند .

متغیر

یک متغیر ، یک تکه نام گذاری شده از حافظه ، برای نگهداری داده هاست . از متغیرها برای نگهداری انواع و اقسام داده ها می‌توانیم استفاده کنیم .

برای ساخت متغیر در جاوا اسکریپت از کلمه کلیدی let استفاده می‌کنیم . عبارت زیر متغیری با نام message ایجاد می‌کند .

```
let message;
```

حال بوسیله عملگر انتساب (assignment operator) می‌توانیم داده ای را داخل آن ذخیره کنیم .

```
let message;

message = 'Hello'; // store the string
```

رشته نوشته شده اکنون داخل فضای اختصاص داده شده به متغیر ذخیره می‌شود و بوسیله نام متغیر می‌توانیم به آن دسترسی داشته باشیم .

```
let message;

message = 'Hello!';

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

اگر بخواهیم ساده‌تر و دقیق‌تر عمل کنیم می‌توانیم تعریف و انتساب یک متغیر را در یک خط بنویسیم :

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

هم چنین می‌توانیم چندین متغیر را در یک خط تعریف کنیم :

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

این روش ممکن است کوتاه تر به نظر بیاید ولی توصیه نمی‌شود . برای خوانایی بیشتر ، هر متغیر را در یک خط تعریف کنید .

تعریف در خط مجزا ممکن است طولانی به نظر بیاید ولی خوانایی بهتری دارد .

```
let user = 'John';

let age = 25;

let message = 'Hello';
```

بعضی افراد چند متغیر را به فرم چندخطی نیز تعریف می‌کنند .

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


بعضی نیز کاما را ابتدای خطوط ذکر می‌کنند .

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

همه موارد فوق کار یکسانی را انجام می‌دهند و انتخاب و استفاده از هر یک به سلیقه شخص بستگی دارد .

---

var به جای let

کلمه کلیدی var تقریبا مانند let است . با استفاده از var نیز می‌توان متغیرها را تعریف کرد ولی روش آن قدیمی و با let متفاوت است . تفاوت‌های نامحسوسی بین var و let وجود دارد که فعلا در مورد آن صحبت نمی‌کنم و در فصل های بعد بیشتر توضیح می‌دهیم .

تشبیه به یک مثال واقعی

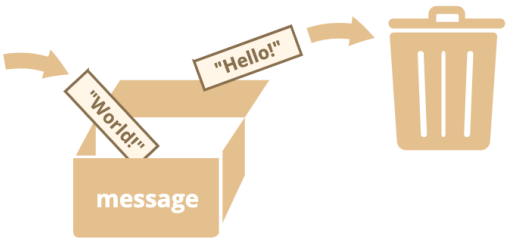
می‌توان متغیر را به یک جعبه تشبیه نمود که می‌توان داده ها را درون آن گذاشت که روی آن برچسبی با یک نام منحصر به فرد زده شده است .

برای مثال متغیر messages را می‌توان به جعبه ای تشبیه نمود که حاوی مقدار "hello" است و برچسب "message" روی آن زده شده است .

هر نوع مقداری را می‌توانیم درون این جعبه قرار دهیم . همچنین می‌توانیم هر موقع که خواستیم این مقدار را تغییر بدهیم :

```
let message;
message = 'Hello!';
message = 'World!'; // value changed
```

هنگامی که مقدار تغییر می‌کند ، داده قبلی حذف و داده جدید افزوده می‌شود .



هم چنین می‌توانیم دو متغیر تعریف کنیم و داده را از یکی کپی و به دیگری اضافه 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!
```

تعریف مجدد متغیر ، باعث بروز خطا می شود . یک متغیر تنها یکبار باید تعریف شود .

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

زبان های Functional

شاید جالب به نظر برسد که زبان هایی functional مانند Scala و Erlang وجود دارد که اجازه تغییر مقادیر متغیرها را نمی‌دهند . در این زبان ها ، هنگامی که مقدار درون "جعبه" ذخیره شد ، برای همیشه آنجا می‌ماند . اگر نیاز داشته باشیم تا مقدار دیگری را ذخیره کنیم ، زبان ما را وادار به ساخت یک جعبه جدید و در واقع تعریف متغیر جدید می‌کند . از متغیر قبلی نمی‌توانیم استفاده کنیم .

این امر ممکن است که در ابتدا کمی عجیب به نظر برسد ولی این زبان ها کاملا قابلیت این را دارند که برنامه های پیچیده بنویسند . همچنین مواقعی مانند محاسبات موازی وجود د ارد که این محدودیت ها مفید واقع می‌شوند .

مطالعه چنین زبان‌هایی حتی اگر قصد استفاده از آنها را ندارید می‌تواند واقع شود .

نامگذاری متغیرها

دو محدودیت در نامگذاری متغیرها در جاوا اسکریپت وجود دارد .

۱– اسامی فقط می‌توانند شامل حروف ، اعداد و کاراکترهای $ و _ باشند .

۲– اولین کاراکتر نباید عدد باشد .

برای مثال ، نام‌های زیر معتبر هستند :

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

هنگامی که نام مورد نظر چند کلمه ای باشد ، پیشنهاد می‌شود که به صورت camelCase نوشته شود یعنی همه حروف کلمه اول ، به صورت کوچک و فقط حرف اول کلمات بعدی به صورت بزرگ نوشته شود . مانند myVeryLongName

نکته جالب اینکه از کاراکترهای $ و _ می‌شود در اسامی نیز استفاده نمود . آنها کاراکترهای معمولی مانند سایر کاراکترها هستند و معنی خاص دیگری ندارند . اسامی زیر معتبر هستند :

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

اسامی زیر نامعتبر هستند :

```
let 1a; // cannot start with a digit
let my-name; // hyphens '-' aren't allowed in the name
```

اسامی به بزرگی و کوچکی حروف حساس هستند .

از حروف غیر لاتین هم می‌شود در اسامی استفاده کرد ولی پیشنهاد نمی‌شود .

برای اسامی از کلمات کلیدی و کلمات رزرو شده نمی‌توانید استفاده کنید . به عنوان مثال نامگذاری زیر نامعتبر است :

```
let let = 5;
```

انتساب بدون use strict

در حالت عادی قبل از استفاده از یک متغیر باید آن را تعریف کنیم ولی در گذشته امکان این وجود داشت که بدون استفاده از let نیز یک متغیر را ایجاد کرد. برای مدیریت و انطباق پذیری کدهای قدیمی ، بدون استفاده از use strict می‌توانیم از این قابلیت استفاده کنیم.

```
// note: no "use strict" in this example
num = 5; // the variable "num" is created if it didn't exist
alert(num); // 5
```


این روش استفاده نامناسب است و در صورت وجود no strict خطا خواهد داد :

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

ثابت ها (constants)

برای تعریف constant ها به جای let از const استفاده می‌کنند :

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

متغیرهایی که بوسیله const تعریف شوند "constant" نامیده می‌شوند. constant ها نمی‌توانند مقداردهی مجدد شوند به عبارت دیگر پس از مقداردهی دیگر نمی‌توان مقدار آنها را تغییر داد. تلاش برای این امر منجر به خطا می‌شود :

```
const myBirthday = '18.04.1982';
myBirthday = '01.01.2001'; // error, can't reassign the constant!
```

هنگامی که یک برنامه نویس مطمن است که مقدار یک متغیر هیچ گاه تغییر نخواهد کرد می تواند آن را با const تعریف نماید.

constantهایی با حروف بزرگ

روش معمول در نامگذاری constant ها این است که از حروف بزرگ استفاده کنیم و برای جداسازی کلمات از _ استفاده نماییم.

برای مثال :

```
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
```


این روش یادگیری مزایایی دارد از جمله : به خاطر سپردن سریع تر اسامی ، کاربرد راحت‌تر ، خوانایی بیشتر

چه هنگامی اسامی constantها را به صورت حروف بزرگ و چه هنگامی به صورت عادی بنویسیم؟

constant یعنی متغیری که مقدارش هیچگاه تغییر نکند ولی constantهایی هستند که مقادیرشان قبل از اجرا مشخص است و constantهایی هستند که در در زمان اجرا محاسبه می‌شوند ولی بعد از انتساب دیگر تغییر نمی‌کنند. مثال :

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

مقدار pageLoadTime قبل از بارگذاری صفحه مشخص نیست بنابراین به صورت عادی نامگذاری می‌شود ولی همچنان constant است و مقدارش پس از انتساب تغییر نمی‌کند .

به عبارت ساده‌تر ، constantهای حروف بزرگ ، constantهایی هستند که مقادیرشان hard-code شده است .

اسامی را درست انتخاب کنید !

نکته بسیار مهمی در مورد متغیرها وجود دارد و آن انتخاب درست اسامی است .

نام یک متغیر باید مرتب ، واضح و گویا باشد . نامگذاری متغیرها یکی از مهارت‌های پیچیده در برنامه‌نویسی می‌باشد . نگاهی گذرا به اسامی انتخاب شده برای متغیرها مشخص می‌کند که چه کدی توسط افراد مبتدی و چه کدی توسط برنامه نویسان حرفه‌ای نوشته شده است .

در یک پروژه واقعی ، بیشتر زمان صرف اصلاح ، تغییر و بروزرسانی کد موجود می‌شود تا اینکه بخواهیم مورد جداگانه ای را از صفر بنویسیم . هنگامی که بعد از مدتی سراغ یک کد نوشته شده میاییم ، اگر در آن کد اسامی مناسبی انتخاب شده باشد ، فهم آن بسیار راحت تر خواهد بود .

لطفا قبل از انتخاب اسامی ، زمانی را به فکر کردن به آنها اختصاص بدهید . مطمن باشید که ضرر نخواهید کرد .

بعضی از قوانین مفید در انتخاب اسامی :

۱– اسامی قابل فهم برای انسان انتخاب کنید مانند userName و shoppingCart

۲– از اسامی مخفف مانند مانند a پرهیز کنید مگر اینکه واقعا بدانید در حال انجام چه کاری هستید .

۳– تا حداکثر مقدار ممکن اسامی را را دقیق و گویا انتخاب کنید . نمونه های از اسامی نامناسب data و value هستند .

اینچنین اسم هایی هیچ چیز به ما نمی‌گویند . این اسامی تنها در صورتی مفید هستند که خود کد گویای دلیل و چرایی استفاده از انها باشد .

۴– با هم تیمی های خود برای نامگذاری به توافق برسید . اگر بازدید کننده سایت user نامیده می‌شود ، استفاده از اسامی مانند curretnUser یا newUser بهتر از currentVisitor و یا newMainInTown است .

ساده به نظر می رسد؟ شاید اینگونه باشد ولی انتخاب اسامی مناسب در واقعیت اینگونه نیست . خواهید دید !

استفاده دوباره از متغیر یا ساخت متغیر جدید؟

برنامه نویسان شیرین عقلی نیز وجود دارند که به جای تعریف متغیر جدید ، ترجیح می‌دهند که از متغیرهای موجود استفاده کنند .

نتیجه؟ متغیر آنها مانند جعبه ای است که افراد متفاوت چیزهای مختلفی درون آن می‌اندازند بدون اینکه برچسب آن را تغییر بدهند . الآن چه چیزی داخل جعبه است؟ چه کسی می داند؟

چنین برنامه نویس‌هایی زمان اندکی را صرفه جویی می‌کنند ولی زمان بیشتری را برای debug کردن از دست می‌دهند .

تعداد متغیر زیاد ،چیز خوبی است .

minifier های جاوا اسکریپت مدرن و مرورگرها ، کد را به گونه ای بهینه می‌کنند که مشکلات performance ایجاد نکند . استفاده از متغیرهای متفاوت برای مقادیر متفاوت به engin ها برای بهینه سازی کد ، کمک نیز خواهد کرد .

خلاصه

با استفاده از var ، let و یا const می توانیم متغیرهایی بسازیم و داده ها را در انها ذخیره کنیم.

- let : راه مدرن تعریف متغیر
- var : راه قدیمی تعریف متغیر. در حالت عادی از این روش استفاده نمی‌کنیم ولی در فصل های بعد راجع به آن توضیح می‌دهیم.
- const : مانند let است ولی مقدار متغیر نمی‌تواند تغییر کند.

متغیرها باید به گونه ای نامگذاری شوند که به راحتی بتوان متوجه شد داخل آن چه چیزی ذخیره شده است.