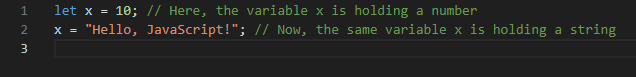
انقیاد نوع در جاوااسکریپت و زمان وقوع آن

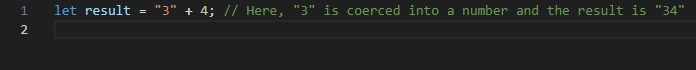
در جاوا اسکریپت، انقیاد نوع(type binding) یک مفهوم مستقیم مانند انقیاد داده نیست. با این حال، جاوا اسکریپت یک زبان تایپ پویا است، به این معنی که متغیرها می توانند مقادیر هر نوع را بدون هیچ گونه اجرای نوع نگه دارند. این در مقایسه با زبان های استاتیک تایپ شده مانند جاوا یا C ++، که در ان متغیرها با یک نوع خاص اعلام می شوند و فقط می توانند مقادیر ان نوع را نگه دارند کاملا متفاوت است.

هنگامی که شما یک مقدار را به یک متغیر در جاوا اسکریپت اختصاص می دهید، نوع متغیر با نوع مقدار اختصاص داده شده تعیین می شود. در اینجا یک مثال ساده است:



در مثال بالا، متغیر x در ابتدا به یک عدد محدود شد، اما بعدا، بدون هیچ مشکلی به یک رشته بازگشت. این انعطاف پذیری یکی از ویژگی های تایپ پویا در جاوا اسکریپت است.

اجبار نوع(Type coercion) نیز یک ویژگی رایج در جاوا اسکریپت است، جایی که نوع یک مقدار به طور خودکار به نوع دیگری بر اساس عملیات انجام شده تبدیل می شود. به عنوان مثال:



در این مورد، رشته "3" به طور خودکار به یک عدد مجبور می شود زمانی که به عدد 4 اضافه می شود.

چه زمانی انقیاد نوع در جاوا اسکریپت انجام می شود؟

انقیاد نوع یا تعیین نوع به صورت پویا در زمان اجرا در جاوا اسکریپت اتفاق می افتد. این به این دلیل است که جاوا اسکریپت یک زبان تفسیری است و تعیین نوع با اجرای کد رخ می دهد.

نوع متغیر می تواند در طول اجرای برنامه تغییر کند، همانطور که در مثال قبلی نشان داده شده است. این ماهیت پویای انقیاد نوع برای انعطاف پذیری جاوا اسکریپت اساسی است.

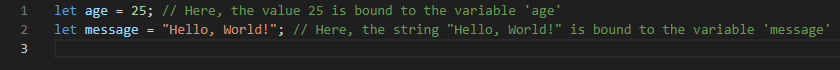
در حالی که تایپ پویا انعطاف پذیری را فراهم می کند و همچنین رخ دادن مسائل غیر منتظره مربوط به نوع را در زمان اجرا را به همراه دارد، به همین دلیل درک انقیاد نوع جاوا اسکریپت و مدیریت کردن نوع‌ها به طور موثر برای نوشتن کد قوی و قابل پیش بینی بسیار مهم است.

بنابراین، به طور خلاصه، انقیاد نوع در جاوا اسکریپت بیشتر در مورد ماهیت پویای سیستم تایپ زبان است و به صورت پویا در زمان اجرا رخ می دهد زیرا مقادیر به متغیرها اختصاص داده می شود و عملیات بر روی ان مقادیر انجام می شود.

انقیاد مقدار در جاوااسکریپت و زمان وقوع آن

در جاوا اسکریپت، مفهوم انقیاد مقدار یک اصطلاح استاندارد در خود زبان نیست، اما می تواند مربوط به ایده "انقیاد یک مقدار به یک متغیر" یا "انقیاد یک مقدار به یک ویژگی یا عنصر" در زمینه چارچوب ها یا کتابخانه هایی باشد که انقیاد داده به UI را اداره می کنند، مانند توسعه وب.

در جاوا اسکریپت، اتصال یک مقدار به یک متغیر از طریق تخصیص متغیر انجام می شود. هنگامی که شما یک مقدار را به یک متغیر اختصاص می دهید، اساسا ان مقدار را به نام متغیر متصل می کنید. به عنوان مثال:



عمل اختصاص یک مقدار به یک متغیر، روش اساسی برای انقیاد مقادیر در جاوا اسکریپت است.

انقیاد مقادیر در زمینه UI

هنگامی که به توسعه وب و چارچوب هایی مانند Angular، React یا Vue.js اشاره می‌شود، مفهوم انقیاد مقدار اغلب به اتصال یک مقدار از مدل داده به یک عنصر UI مانند یک فیلد ورودی، یک منطقه متن یا هر جزء دیگر اشاره دارد.

به عنوان مثال، در React، شما می توانید یک مقدار را به یک عنصر ورودی با استفاده از ویژگی مقدار(value attribute) متصل کنید، مانند این:



در این مثال، inputValue به مقدار عنصر ورودی محدود می شود، به این معنی که هنگامی که inputValue تغییر می کند، عنصر ورودی به روز می شود و هنگامی که کاربر در فیلد ورودی تایپ می کند، inputValue را به روز می کند.

چه زمانی انقیاد داده در جاوا اسکریپت انجام می شود؟

انقیاد داده در زمینه UI یا اتصال داده ها معمولا زمانی انجام می شود که شما می خواهید تغییرات در مدل داده خود را به UI منعکس کنید و بالعکس. در چارچوب هایی مانند Angular، React یا Vue.js، این اغلب به عنوان بخشی از فرایند رندر اتفاق می افتد. هر زمان که مقدار محدود تغییر می کند، باعث به روز رسانی عنصر UI مرتبط می شود و هنگامی که کاربر با UI تعامل می کند، می تواند مقدار محدود را در مدل داده به روز کند.

زمان بندی انقیاد داده بستگی به چارچوب خاص و چرخه عمر ان دارد. به عنوان مثال، در React، به روز رسانی وضعیت یک جزء باعث رندر مجدد جزء و عناصر محدود ان می شود و UI را برای منعکس کردن حالت جدید به روز می کند. در Angular، تغییرات در مدل باعث ایجاد یک چرخه تشخیص تغییر می شود و دیدگاه را با مقادیر جدید به روز می کند.

به طور خلاصه، انقیاد داده در جاوا اسکریپت می تواند به اتصال مقادیر به متغیرها یا در زمینه توسعه وب، مقادیر متصل شده به عناصر یا اجزای UI به منظور حفظ UI در همگام سازی با مدل داده های اساسی اشاره کند. این یک جنبه اصلی ساخت رابط های کاربری پویا و تعاملی است و نقش کلیدی در توسعه وب مدرن دارد.

آیا تعاریف متغیرها ضمنی است یا صریح و یا هر دو نوع تعریف وجود دارد؟ با ذکر مثال توضیح داده شود.

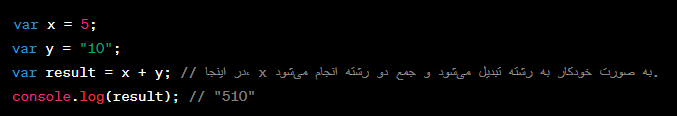
در جاوا اسکریپت، تعاریف متغیر می تواند هم ضمنی و هم صریح باشد و تمایز بین این دو ریشه در نحوه تعریف و مقداردهی اولیه متغیرها است.

در جاوااسکریپت، انقیاد نوع (Type coercion) به تبدیل یک نوع داده به نوع دیگر اشاره دارد. این انقیاد ممکن است به صورت خودکار (Implicit) یا اختیاری (Explicit) انجام شود.

انقیاد نوع خودکار (Implicit Type Coercion):

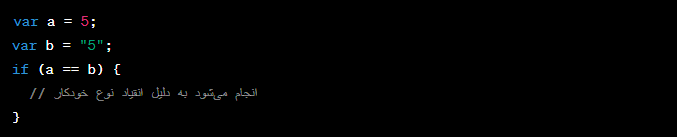
در عملیات ریاضی:

در جاوااسکریپت، اگر یک عملیات ریاضی بین دو نوع داده مختلف انجام شود، موتور جاوااسکریپت تلاش می‌کند تا به صورت خودکار نوع داده‌ها را تبدیل کند. برای مثال:



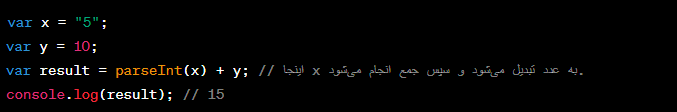
در مقایسه:

در بعضی مواقع، مقایسه اشیاء با انواع مختلف باعث انجام انقیاد نوع خودکار می‌شود. برای مثال:



انقیاد نوع اختیاری (Explicit Type Coercion):

برنامه‌نویس می‌تواند به صورت اختیاری نیز انقیاد نوع انجام دهد. این کار با استفاده از توابعی مانند parseInt، parseFloat، String و ... انجام می‌شود. برای مثال:

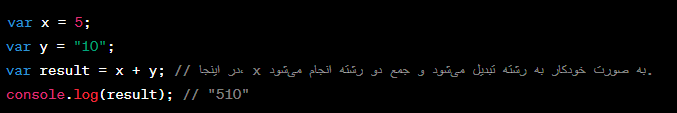


در کل، انقیاد نوع در جاوااسکریپت به صورت پویا و در زمان اجرا انجام می‌شود، و بستگی به عملیاتی که انجام می‌شود، ممکن است انقیاد نوع خودکار رخ دهد یا برنامه‌نویس مستقیماً انقیاد نوع انجام دهد.

انقیاد نوع در جاوااسکریپت به دلیل این که یک زبان تایپ دینامیک و از نوع خیلی ضعیف است، ممکن است در برخی مواقع به مشکلاتی منجر شود که برنامه‌نویسان باید مواظب آن باشند.

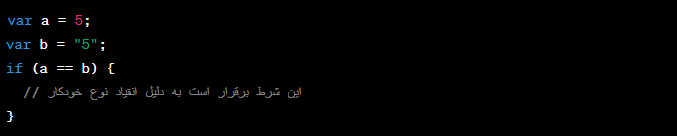
انقیاد نوع در عملیات ریاضی:

در عملیات ریاضی، انقیاد نوع خودکار می‌تواند به نتایج غیر منتظره منجر شود:



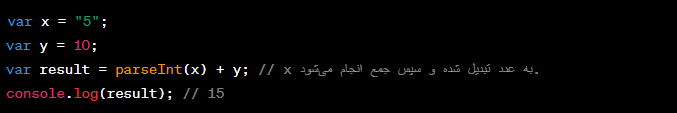
انقیاد نوع در مقایسه:

مقایسه اعداد با رشته‌ها ممکن است منجر به نتایج غیر منتظره شود:



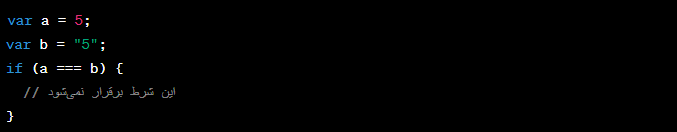
انقیاد نوع اختیاری:

برنامه‌نویسان می‌توانند از توابعی مانند parseInt یا parseFloat برای انجام انقیاد نوع اختیاری استفاده کنند:



تجنب انقیاد نوع ناخواسته:

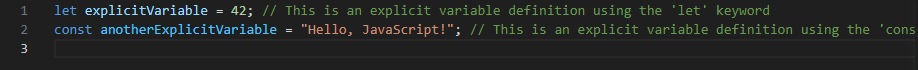
برای جلوگیری از انقیاد نوع ناخواسته، برنامه‌نویسان معمولاً از اپراتور === (برابری به همه جهات و نوع داده) به جای == (برابری به همه جهات) در مقایسه‌ها استفاده می‌کنند. این کار باعث می‌شود که نوع داده‌ها هم بررسی شود و فقط در صورت برابری نوع‌ها، مقایسه به درستی انجام شود:



در کل، انقیاد نوع در جاوااسکریپت باید با دقت مدیریت شود تا اشکالات مربوط به نوع‌ها به وجود نیاید و کد به درستی اجرا شود.

تعریف صریح متغیر

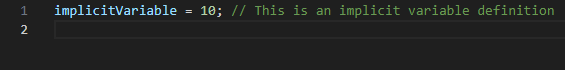
تعریف صریح متغیر زمانی اتفاق می افتد که یک توسعه دهنده، یک متغیر را با استفاده از کلمات کلیدی let، const یا var اعلام می کند و یک مقدار اولیه را در زمان اعلان اختصاص می دهد. به عنوان مثال:



در مثال بالا، هر دو متغیر به شکل صریح تعریف شده‌اند. انواع آن‌ها بر اساس نوع مقادیر اختصاص داده شده به آن‌ها (عدد و رشته و ...) تعیین می شوند.

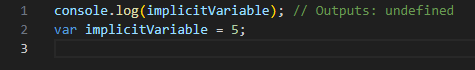
تعریف ضمنی متغیر

تعریف ضمنی متغیر زمانی اتفاق می افتد که یک مقدار به یک متغیر اختصاص داده می شود بدون اینکه به صراحت آن را با استفاده از let، const یا var اعلام کند. هنگامی که یک متغیر به طور ضمنی تعریف می شود، زمانی ایجاد می شود که شما یک مقدار را بدون اعلام صریح به ان اختصاص می دهید. به عنوان مثال:



در این مثال، implicitVariable به طور ضمنی تعریف می شود زمانی که مقدار 10 به ان اختصاص داده می شود. جاوا اسکریپت به طور خودکار نوع متغیر را بر اساس مقدار اختصاص داده شده تعیین می کند. این یک مثال از تایپ پویا است و قابلیت تعریف متغیر ضمنی جاوا اسکریپت را نشان می دهد.

مهم است که توجه داشته باشید که تمام تعریف متغیرها در جاوا اسکریپت به بالای تابع یا اسکریپت حاوی آن‌ها محدود میشوند. این بدان معنی است که تخصیص متغیرها در جایی که انتظار دارید اتفاق می افتد، تعریف واقعی به بالا محدود می رود. به عنوان مثال:



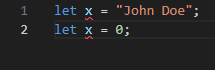
در این مورد، متغیر implicitVariable به بالای دامنه خود محدود میشود، که منجر به کنسول می شود و دستور console.log خطا را پرتاب نمی کند، بلکه خروجی تعریف نشده است.

نقش let، const و var

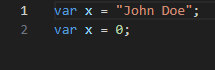
* کلمه کلیدی let در ES6 معرفی شد (2015).
* متغیرهای تعریف شده با let نمی توانند دوباره تعریف شوند.
* متغیرهای تعریف شده با let باید قبل از استفاده تعریف شوند.
* متغیرهای تعریف شده با let دارای محدوده بلوک هستند.

نمیتوان دوباره تعریف کرد

متغیرهای تعریف شده با let نمی توانند دوباره تعریف شوند. شما نمی توانید به طور تصادفی یک متغیر تعریف شده با let را دوباره تعریف کنید. به طور مثال تعریف متغیر به شکل زیر اشتباه است :

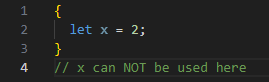


اما تعریف متغیر به شکل زیر صحیح است :

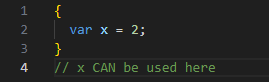


محدوده بلوک

قبل از ES6 (2015)، جاوا اسکریپت دارای محدوده جهانی و محدوده تابع بود. در ES6 دو کلمه کلیدی مهم جدید جاوا اسکریپت را معرفی شد: let و cons. این دو کلمه کلیدی محدوده بلوک را در جاوااسکریپت معرفی میکنند. متغیرهای اعلام شده در داخل یک بلوک { } از خارج از بلوک قابل دسترسی نیستند:

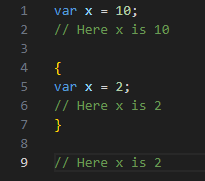


متغیرهای اعلام شده با کلمه کلیدی var محدوده بلوک ندارند. متغیرهای اعلام شده در داخل یک بلوک { } می توانند از خارج از بلوک قابل دسترسی باشند.

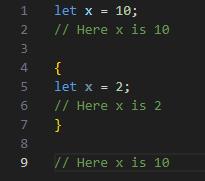


بازتعریف متغیرها

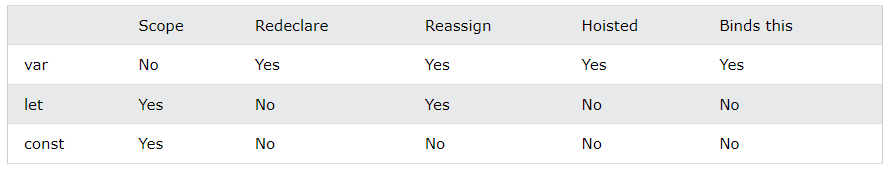
بازتعریف یک متغیر با استفاده از کلمه کلیدی var می تواند مشکلاتی را تحمیل کند. بازتعریف یک متغیر در داخل یک بلوک همچنین متغیر خارج از بلوک را دوباره تعریف خواهد کرد:



بازتعریف یک متغیر با استفاده از کلمه کلیدی let می تواند این مشکل را حل کند. بازتعریف یک متغیر در داخل یک بلوک، متغیر خارج از بلوک را دوباره تعریف نخواهد کرد:

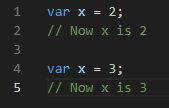


تفاوت بین var، let و const

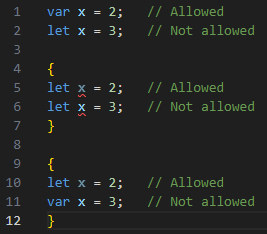


تعریف مجدد

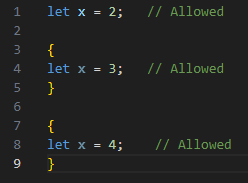
بازتعریف یک متغیر جاوا اسکریپت با var در هر جایی از یک برنامه مجاز است:



با let، بازتعریف یک متغیر در همان بلوک مجاز نیست:



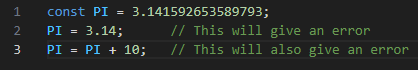
تعریف مجدد یک متغیر با let، در یک بلوک دیگر، مجاز است:



* کلمه کلیدی const در ES6 (2015) معرفی شد.
* متغیرهای تعریف شده با const نمی توانند مجددا تعریف شوند.
* متغیرهای تعریف شده با const نمی توانند دوباره اختصاص داده شوند.
* متغیرهای تعریف شده با const دارای محدوده بلوک هستند.

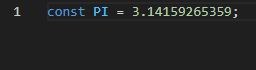
نمیتوانند دوباره تخصیص داده شوند

یک متغیر const را نمیتوان دوباره اختصاص داد:

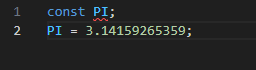


باید تخصیص داده شوند

به متغیرهای const جاوا اسکریپت باید در زمان تعریف یک مقدار اختصاص داده شود. به طور مثال تعریف متغیر به شکل زیر صحیح است:

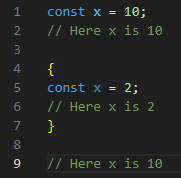


اما تعریف متغیر به شکل زیر غلط است:



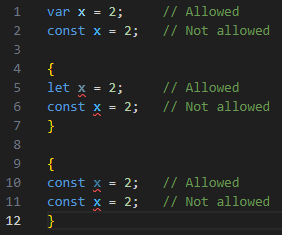
محدوده بلوک

رفتار const در بلوک مانند رفتار let است.

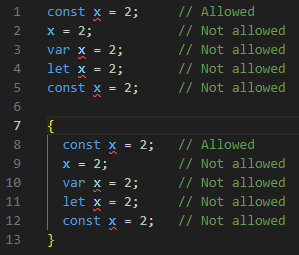


بازتعریف

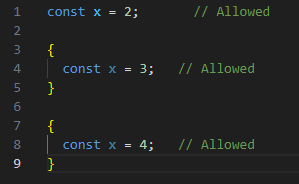
تعریف مجدد var موجود یا متغیر let برای const، در همان محدوده، مجاز نیست:



تخصیص مجدد یک متغیر const موجود، در همان محدوده، مجاز نیست:



بازتعریف یک متغیر با const، در محدوده دیگری، و یا در یک بلوک دیگر، مجاز است:



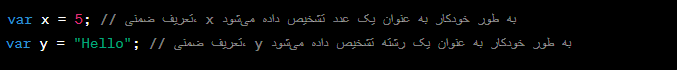
استفاده از let و const متغیر را به شکل صریح تعریف میکنند که اغلب به عنوان بهترین روش در نظر گرفته می شود زیرا به جلوگیری از انتقال مجدد تصادفی مقادیر متغیر ها کمک می کند و محدوده سطح بلوک را فراهم می کند و به افزایش پیش بینی کد کمک می کند.

به طور خلاصه، جاوا اسکریپت از تعاریف متغیر ضمنی و صریح پشتیبانی می کند. با تعاریف صریح، شما از کلمات کلیدی let یا const برای اعلام یک متغیر استفاده می کنید. در تعاریف ضمنی، یک متغیر زمانی ایجاد می شود که یک مقدار به ان اختصاص داده شود. درک تفاوت های ظریف تعریف متغیر در جاوا اسکریپت می تواند در نوشتن کد قابل پیش بینی و قابل نگهداری کمک کند.

در جاوااسکریپت، تعریف متغیرها می‌تواند به صورت ضمنی (Implicit) یا صریح (Explicit) انجام شود. همچنین، اغلب متغیرها به صورت دینامیک (Dynamic) تعریف می‌شوند، که به این معناست که نوع متغیر ممکن است در طول زمان تغییر کند.

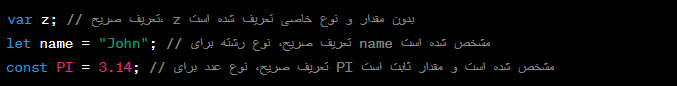
تعریف ضمنی:

تعریف ضمنی متغیرها به معنای این است که موتور جاوااسکریپت خودکار نوع متغیر را تشخیص داده و مقدار مناسب را تعیین می‌کند. این موضوع بیشتر در زمان انتساب (assignment) اتفاق می‌افتد.



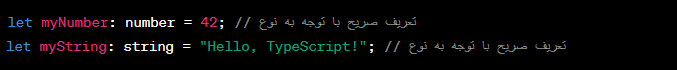
تعریف صریح:

تعریف صریح متغیرها به معنای این است که برنامه‌نویس به صورت صریح نوع متغیر را مشخص می‌کند. این کار با استفاده از کلمه‌های کلیدی مانند var، let، یا const انجام می‌شود.



تعریف صریح با توجه به نوع:

تا ES6، تعریف صریح نوع در جاوااسکریپت امکان‌پذیر نبود. اما با معرفی let و const در ES6، برنامه‌نویسان می‌توانند به صورت صریح نوع متغیر را مشخص کنند.

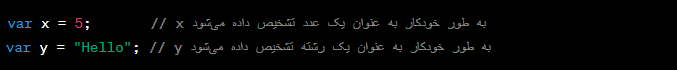


در مثال بالا، myNumber به عنوان یک عدد و myString به عنوان یک رشته تعریف شده‌اند. این قابلیت اغلب در سیستم‌های جاوااسکریپت توسعه یافته‌تر، مانند TypeScript، مورد استفاده قرار می‌گیرد.

تعریف متغیرها در جاوااسکریپت:

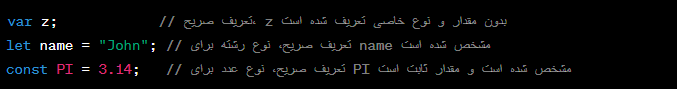
1. تعریف ضمنی:

در تعریف ضمنی، موتور جاوااسکریپت خودکار نوع متغیر را تشخیص داده و مقدار مناسب را تعیین می‌کند. این به این معناست که شما نیازی به صراحتًا مشخص کردن نوع متغیر ندارید.



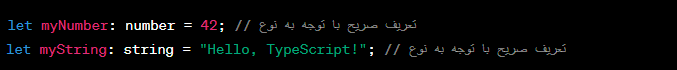
2. تعریف صریح:

تعریف صریح به این معناست که برنامه‌نویس نوع متغیر را به صورت صریح مشخص می‌کند. این عمل با استفاده از کلمات کلیدی مانند var، let یا const انجام می‌شود.



3. تعریف صریح با توجه به نوع:

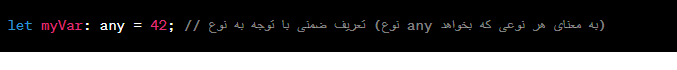
در برخی از توسعه‌های جدید جاوااسکریپت، مانند TypeScript، این امکان وجود دارد که برنامه‌نویس نوع متغیر را صریحاً مشخص کند.



در اینجا، myNumber به عنوان یک عدد و myString به عنوان یک رشته با توجه به نوع مشخص شده‌اند.

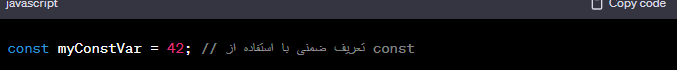
4. تعریف ضمنی با توجه به نوع:

از ES6 به بعد، با معرفی let و const، تعریف ضمنی نیز می‌تواند با توجه به نوع انجام شود.



5. تعریف ضمنی با استفاده از: const

از آنجایی که const متغیرها را ثابت می‌کند، تعریف ضمنی با استفاده از const ممکن است در برخی موارد باعث ایجاد کد کمتری با خطا شود.



با این توضیحات، شما می‌توانید با استفاده از تعریف ضمنی یا صریح، به وفور با متغیرها در جاوااسکریپت کار کنید و از آن‌ها به عنوان داده‌های دینامیک و قابل تغییر بهره‌مند شوید.

آیا در جاوااسکریپت متغیرهای ایستا در پشته وجود دارد؟ توضیح دهید.

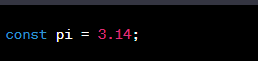
در جاوااسکریپت، متغیرهای ایستا به مفهوم متغیرهایی که در پشته (stack) ذخیره می‌شوند، معنا نمی‌شود. پشته اصطلاحا برای ذخیره داده‌ها و مراجع به توابع در زمان اجرا استفاده می‌شود، اما متغیرها اصولا در حافظه (heap) ذخیره می‌شوند.

در جاوااسکریپت، متغیرها در حافظه ذخیره می‌شوند و به طور مستقیم از پشته استفاده نمی‌کنند. پشته بیشتر برای مدیریت فرآیندهای توابع و استفاده از حافظه موقتی برای ذخیره مقادیر محلی توابع استفاده می‌شود.

در جاوااسکریپت، مفهوم "متغیرهای ایستا" به صورت صریح وجود ندارد. جاوااسکریپت یک زبان برنامه‌نویسی دینامیک است و متغیرهای آن به صورت دینامیک در حافظه ذخیره می‌شوند.

متغیرهای ایستا به طور کلی به متغیرهایی اشاره دارند که مقدار آن‌ها در طول زمان تغییر نمی‌کند. در جاوااسکریپت، این مفهوم به صورت صریح وجود ندارد. با این حال، می‌توانید از کلمه کلیدی `const` برای تعریف متغیرهای ثابت استفاده کنید که مقدار آن‌ها پس از تخصیص قابل تغییر نباشد.

مثال:



در این مثال، `pi` به عنوان یک متغیر ثابت تعریف شده است و مقدار آن پس از اولین تخصیص قابل تغییر نیست. این متغیر به نوعی به مفهوم متغیرهای ایستا نزدیکتر است، اما همچنان به عنوان یک متغیر دینامیک در جاوااسکریپت در نظر گرفته می‌شود.

بنابراین، مفهوم متغیرهای ایستا که به معنای ذخیره در پشته باشد، در جاوااسکریپت صحیح نیست. متغیرها در این زبان به صورت دینامیک در حافظه ذخیره می‌شوند و ممکن است در طول اجرا تغییر کنند.

آیا در جاوااسکریپت متغیرهای پویا در پشته وجود دارد؟ توضیح دهید.

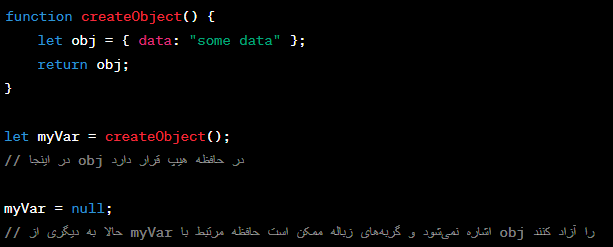
در جاوااسکریپت، مفهوم "متغیرهای پویا در پشته" به طور کلی درست نیست. در زبان‌های برنامه‌نویسی مانند جاوا و C++، مفهوم پویا بودن (dynamic) به ارتباط با تخصیص حافظه در زمان اجرا مرتبط است. این زبان‌ها می‌توانند به صورت پویا حافظه را تخصیص دهند یا آن را آزاد کنند.

در جاوااسکریپت، متغیرها به صورت دینامیک در حافظه ذخیره می‌شوند، اما در جاوااسکریپت، ما نیازی به مدیریت مستقیم حافظه نداریم و تمام برنامه ها به شکل پویا عمل میکنند.

در جاوااسکریپت، متغیرهای اصطلاحاً در حافظه هیپ (Heap) قرار می‌گیرند و مفهوم "متغیرهای پویا در پشته" به صورت مستقیم در این زبان وجود ندارد. در جاوااسکریپت، حافظه پویا به صورت خودکار توسط سطل زباله (garbage collectors) مدیریت می‌شود.

وقتی یک متغیر در جاوااسکریپت تعریف می‌شود و به آن مقداری اختصاص داده می‌شود، متغیر و مقدارش در حافظه هیپ قرار می‌گیرند. اگر دیگر به آن متغیر ارجاعی وجود نداشته باشد، گربه‌های زباله پیش می‌روند و متغیرها و اشیاء غیرقابل دسترس را از حافظه حذف می‌کنند.

مثال:



در این مثال، `obj` یک متغیر است که در حافظه هیپ قرار دارد. هنگامی که `myVar` به `null` تنظیم می‌شود، دیگر به `obj` اشاره نمی‌شود و گربه‌های زباله ممکن است حافظه مرتبط با `obj` را آزاد کنند. این فرآیند مدیریت حافظه به صورت خودکار و توسط محیط اجرایی جاوااسکریپت انجام می‌شود، بنابراین نیاز به دخالت دستی بر روی پشته وجود ندارد.

بنابراین، می‌توان گفت که در جاوااسکریپت، متغیرها به صورت پویا در حافظه ذخیره می‌شوند، اما نه به صورت مستقیم در پشته. اطلاعات مربوط به توابع و محیط اجرایی در پشته قرار دارند، اما این با مفهوم متغیرهای پویا به طور مستقیم مرتبط نیست.

آیا در جاوااسکریپت متغیرهای پویادر هیپ به طور صریح وجود دارد؟ توضیح دهید.

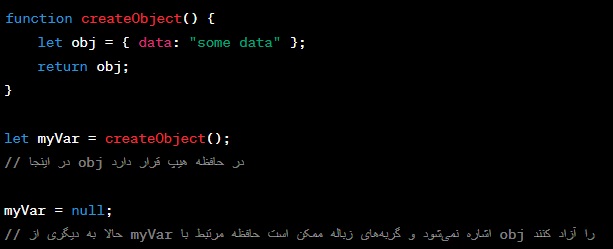
در جاوااسکریپت، تا حد زیادی مدیریت حافظه به صورت خودکار انجام می‌شود و توسعه‌دهندگان نیازی به مستقیم تخصیص یا آزاد کردن حافظه ندارند. در واقع، مفهوم "متغیرهای پویا در هیپ" به طور مستقیم در جاوااسکریپت وجود ندارد.

متغیرها در جاوااسکریپت به صورت دینامیک در حافظه ذخیره می‌شوند و به صورت خودکار توسط گربه‌های زباله مدیریت می‌شوند. گربه‌های زباله به صورت دوره‌ای اجرا می‌شوند و اشیاءی که دیگر به آن‌ها ارجاعی وجود ندارد، حذف می‌شوند.

به عبارت دیگر، توسعه‌دهندگان در جاوااسکریپت نیازی به تخصیص یا آزاد کردن حافظه به صورت صریح ندارند و این عملیات‌ها به صورت اتوماتیک انجام می‌شوند. در نتیجه، متغیرها در هیپ (heap) ذخیره می‌شوند و مدیریت حافظه به صورت متقابل با سطل زباله(Garbage Collector) انجام می‌شود.

در جاوااسکریپت، مفهوم "متغیرهای پویا در هیپ" به صورت صریح وجود ندارد. در این زبان، متغیرها به صورت دینامیک در حافظه ذخیره می‌شوند و تخصیص و آزادسازی حافظه به صورت خودکار توسط سطل زباله (garbage collectors) انجام می‌شود.

مثال:



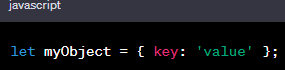
در این مثال، `obj` یک شیء است که در حافظه هیپ قرار دارد. هنگامی که `myVar` به `null` تنظیم می‌شود، دیگر به `obj` اشاره نمی‌شود و گربه‌های زباله ممکن است حافظه مرتبط با `obj` را آزاد کنند. این فرآیند توسط محیط اجرایی جاوااسکریپت به صورت خودکار انجام می‌شود و نیازی به مدیریت صریح حافظه در هیپ نداریم.

بنابراین، متغیرها در جاوااسکریپت به صورت پویا در حافظه ذخیره می‌شوند و مفهوم متغیرهای پویا در هیپ به صورت صریح وجود ندارد.

آیا در جاوااسکریپت متغیرهای پویادر هیپ به طور ضمنی وجود دارد؟ توضیح دهید.

بله، در جاوااسکریپت متغیرهای پویا (dynamic variables) در حافظه هیپ (heap) به طور ضمنی وجود دارند. زبان‌هایی که از مدیریت حافظه خودکار استفاده می‌کنند، مانند جاوااسکریپت، اشیاء (objects) به صورت پویا در حافظه هیپ ایجاد می‌شوند.

وقتی یک متغیر از نوع شیء (object) در جاوااسکریپت تعریف می‌شود، این شیء در حافظه هیپ قرار می‌گیرد. به عنوان مثال:

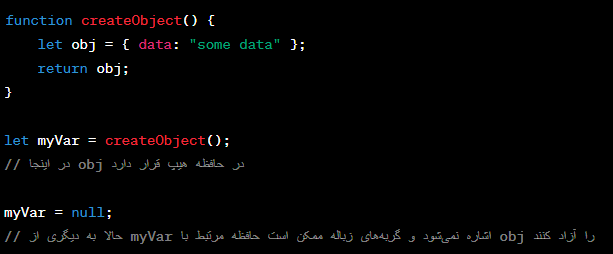


در اینجا، `myObject` یک متغیر است که یک شیء را نمایندگی می‌کند و این شیء در حافظه هیپ ذخیره می‌شود. گربه‌های زباله جاوااسکریپت به طور دوره‌ای حافظه را پیمایش کرده و اشیاءی که دیگر به آن‌ها ارجاعی وجود ندارد را تشخیص داده و حذف می‌کنند.

از آنجا که تخصیص و آزادسازی حافظه به صورت خودکار انجام می‌شود، توسعه‌دهندگان نیازی به نگرانی در مورد مدیریت حافظه به صورت صریح ندارند و متغیرهایشان به صورت ضمنی در حافظه هیپ ذخیره می‌شوند.

بله، در جاوااسکریپت متغیرها و اشیاء در حافظه هیپ به طور ضمنی وجود دارند. زمانی که شیء‌هایی ایجاد می‌شوند و به متغیرها اختصاص داده می‌شوند، این شیء‌ها در حافظه هیپ قرار می‌گیرند و مدیریت حافظه توسط گربه‌های زباله انجام می‌شود.

مثال:



در این مثال، `obj` یک شیء است که به آن اختصاص داده شده و در حافظه هیپ قرار دارد. وقتی `myVar` به `null` تنظیم می‌شود، دیگر به `obj` اشاره نمی‌شود و گربه‌های زباله ممکن است حافظه مرتبط با `obj` را آزاد کنند.

این روند بهبود کارایی و مدیریت حافظه را ساده‌تر می‌کند و توسعه‌دهندگان نیازی به مدیریت صریح حافظه ندارند. حافظه‌هایی که دیگر به آن‌ها اشاره ایجاد نشده باشد، توسط گربه‌های زباله حذف می‌شوند.

متغیر های ایستا در جاوااسکریپت چگونه پیاده سازی شده اند؟

در جاوااسکریپت، متغیرهای ایستا با استفاده از const پیاده‌سازی می‌شوند. کلمه‌ی const برای تعریف متغیرها به کار می‌رود که مقدار آن‌ها در طول اجرای برنامه تغییر نخواهد کرد. این کلمه کلیدی در ES6 (ECMAScript 2015) معرفی شد و تا کنون به عنوان روش اصلی برای تعریف متغیرهای ایستا در جاوااسکریپت مورد استفاده قرار می‌گیرد.



در این مثال، pi به عنوان یک متغیر ایستا با استفاده از const تعریف شده است. این متغیر نمی‌تواند در طول اجرای برنامه تغییر کند و اگر تلاش کنید مقدار آن را تغییر دهید، با یک خطا مواجه خواهید شد.

مزایا و کاربردها:

ثبات کد (Code Stability): استفاده از متغیرهای ایستا باعث می‌شود که مقدار متغیرها در طول زمان تغییر نکند و کد ثابت‌تر و قابل‌ترکیب‌تری ارائه شود.

پیش‌بینی مقدار (Value Predictability): اگر یک متغیر با const تعریف شود، برنامه‌نویسان و تا اندازه‌ای حتی خوانندگان کد می‌توانند انتظار داشته باشند که مقدار آن تا زمان اجرای برنامه تغییر نخواهد کرد.

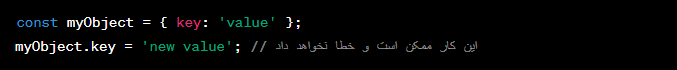
کد خواناتر (Readability): تعریف متغیرهای ایستا با const می‌تواند کد را خواناتر کند و افراد دیگری که کد شما را مشاهده می‌کنند را در درک مفهوم و مقصود کمک کند.

نکات مهم:

در صورتی که متغیری با const تعریف شود، باید به یکبار مقداردهی شود. این یعنی شما نمی‌توانید یک const بدون مقداردهی اولیه اعلام کنید.



متغیرهای ایستا با const می‌توانند اشیاء (Objects) و آرایه‌ها را نیز در خود نگه‌داری کنند. با این حال، تغییر ویژگی‌های داخلی یک شیء یا آرایه با const ممکن است باعث شود خطا رخ دهد، اما شناخته‌شده‌اند.



در کل، متغیرهای ایستا با استفاده از const یکی از ابزارهای مهم برای بهبود قابلیت خواندن و پیش‌بینی کد در جاوااسکریپت هستند.

حتما، با کمال میل. دعوت می‌کنم تا به چند نکته بیشتر در مورد متغیرهای ایستا (const) در جاوااسکریپت بپردازیم:

1. تفاوت با let و var:

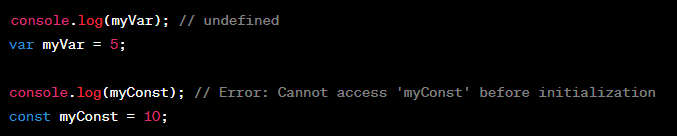
let برای تعریف متغیرهای محلی استفاده می‌شود که می‌توانند در یک بلوک مشخص (مثل یک حلقه یا یک شرط) تغییر کنند.

const نیز برای تعریف متغیرهای محلی استفاده می‌شود، اما مقدار آن‌ها نمی‌تواند تغییر کند.

var که در ES5 معرفی شد، نحوه‌ی تعریف متغیرها را مشخص می‌کند و می‌تواند دچار مشکلاتی مانند hoisting شود.

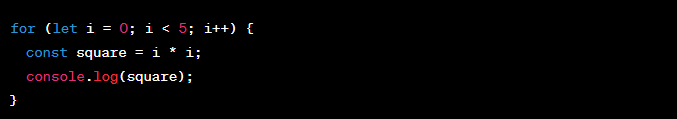
2. Hoisting در متغیرهای const:

متغیرهای const همچنان تحت hoisting هستند. این بدان معناست که می‌توانید یک متغیر const را پیش از تعریف آن استفاده کنید، اما به دلیل خصوصیات const که یک بار مقداردهی شده‌اند نمی‌توانند مقدار دیگری را دریافت کنند.



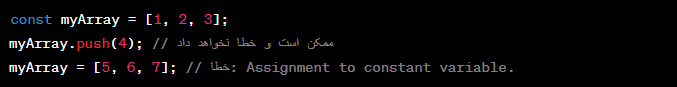
3. استفاده در حلقه‌ها:

استفاده از const در حلقه‌ها می‌تواند کمک کند تا از تغییر ناخواسته مقادیر جلوگیری شود.



4. آرایه‌ها و: const

در صورتی که یک آرایه با const تعریف شود، می‌توانید عناصر آرایه را تغییر دهید، اما نمی‌توانید یک آرایه جدید نسبت به متغیر ایجاد کنید.



5. تفاوت با ایستایی در زبان‌های دیگر:

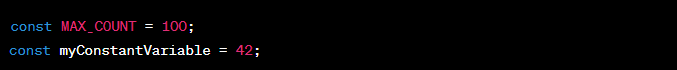
در زبان‌های دیگر مفهوم "ایستا" به معنایی مشخص تر از آن چیزی است که در جاوااسکریپت با const تعبیه شده است. در جاوااسکریپت، const به معنای "قابل تغییر نیست" است، اما این به معنای این نیست که مقدار آن برابر با یک مقدار ثابت و ایستا است.

6. توصیه برای استفاده از const:

استفاده از const به عنوان روش اولویتی برای تعریف متغیرها تا جای ممکن توصیه می‌شود. این کمک می‌کند کد شما خواناتر و قابل‌ترکیب‌تر باشد و در صورتی که مقدار یک متغیر تغییر نخواهد کرد، از این امکان بهره‌مند شوید.

7. نامگذاری متغیرهای ایستا:

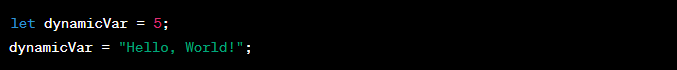
بهتر است متغیرهای ایستا با const نام‌هایی داشته باشند که از حروف بزرگ تشکیل شده و به صورت snake\_case یا camelCase نوشته شوند تا در کد به راحتی قابل تشخیص باشند.



با استفاده از متغیرهای ایستا، می‌توانید کد خود را از اشکالات مرتبط با تغییر ناخواسته مقادیر در طول زمان محافظت کنید و قابلیت خواندن و نگهداری کد خود را افزایش دهید.

متغیر های پویا در جاوااسکریپت چگونه پیاده سازی شده اند؟

در جاوااسکریپت، متغیرهای پویا با استفاده از let یا var پیاده‌سازی می‌شوند. این کلمات کلیدی برای تعریف متغیرهای قابل تغییر در طول اجرای برنامه به کار می‌روند. این متغیرها به عبارتی قابلیت تغییر نوع و مقدار را دارند.

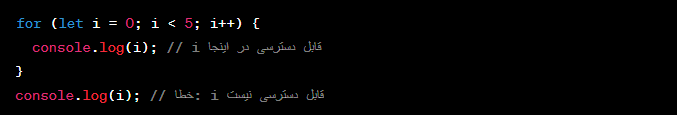


در این مثال، dynamicVar با استفاده از let تعریف شده و ابتدا به یک عدد اختصاص داده شده و سپس مقدار آن به یک رشته تغییر کرده است.

نکات مهم:

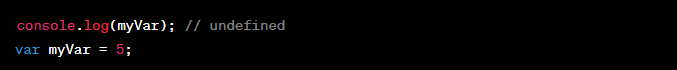
let در ES6:

کلمه کلیدی let در ES6 معرفی شد و به خوبی از var جایگزین شده است. از let به دلیل مزایای بیشتری مانند داشتن scope بلوکی (block-scoped) استفاده می‌شود.

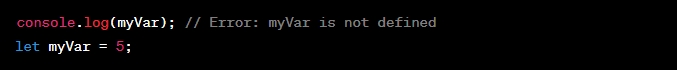


var و Hoisting:

var نیز همچنان در جاوااسکریپت استفاده می‌شود، اما دارای خصوصیت hoisting است که به این معناست که متغیرها با var ممکن است پیش از تعریف آن‌ها مورد دسترسی قرار گیرند.

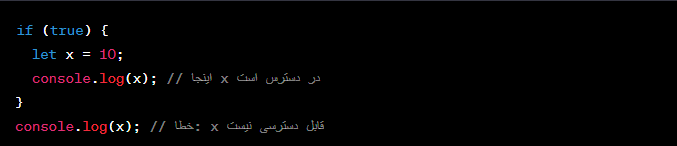


در مقایسه با let که اینگونه کار نمی‌کند:



استفاده در محدوده‌های مختلف:

let اطلاعات را در scope بلوکی (بین {}) نگه می‌دارد و این به معنای آن است که می‌توانید متغیرهایی با همین نام را در بلوک‌های مختلف بسازید.

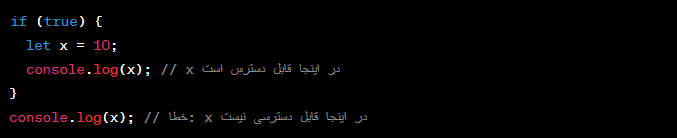


استفاده از let به عنوان روش اولویتی برای تعریف متغیرها تا جای ممکن توصیه می‌شود، زیرا این کلمه کلیدی اصطلاحاً مزایای بیشتری نسبت به var دارد و اشکالات زیادی که در var وجود دارد را برطرف کرده است. استفاده از let باعث ایجاد کدی خواناتر و قابل‌ترکیب‌تر می‌شود.

متغیرهای پویا در جاوااسکریپت با استفاده از let و گاهی هم var تعریف می‌شوند. دو کلمه‌ی کلیدی let و var به شما اجازه می‌دهند تا متغیرها را تعریف کرده و به آن‌ها مقدار اختصاص دهید. این متغیرها می‌توانند در طول اجرای برنامه تغییر کنند، به خصوص اگر با let تعریف شده باشند.

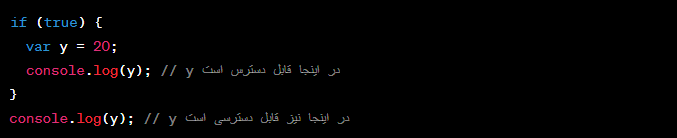
1. استفاده از let:

let به شما این امکان را می‌دهد که متغیرها را در scope بلوکی (block scope) تعریف کنید. این به معنای این است که متغیرها تا انتهای بلوک (بین {}) محدود می‌شوند.



2. استفاده از var:

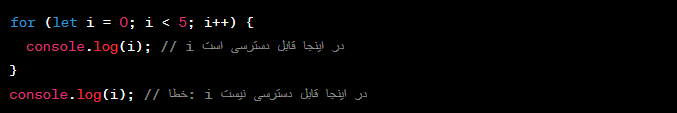
var همچنان در جاوااسکریپت وجود دارد، اما به دلیل خصوصیاتی نظیر hoisting و قابل دسترسی در خارج از scope بلوکی که تعریف شده‌اند، کمتر مورد استفاده قرار می‌گیرد.



به عنوان یک نکته مهم، var ها hoisting دارند که به این معنی است که متغیرها می‌توانند پیش از تعریف آن‌ها مورد استفاده قرار گیرند.

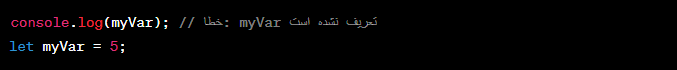
3. مبنای Block Scope در let:

let به شما این امکان را می‌دهد که scope بلوکی (بین {}) را ایجاد کنید. این بهبود از نظر اصول برنامه‌نویسی و ایمنی کد است.



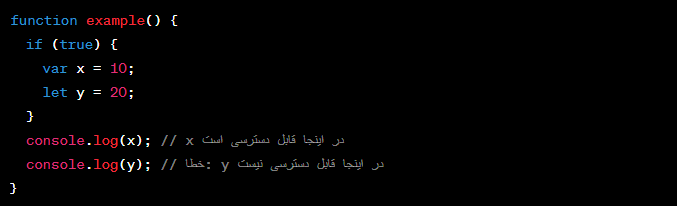
4. Hoisting با let:

let نیز hoisting دارد، اما با این تفاوت که اگر متغیر قبلاً تعریف شده باشد ولی هنوز مقداردهی نشده باشد، به undefined تنظیم می‌شود.



5. تفاوت در محدودیت‌های Block Scope:

let به عنوان یک بهترین عملکرد در محدوده‌های بلوکی استفاده می‌شود، در حالی که var به محدودیت‌های قدیمی‌تری از قبیل محدوده‌های تابعی (function scope) محدود می‌شود.



در کل، به منظور ایجاد متغیرهای پویا که مقدار و نطام تغییرات در طول اجرا دارند، از let برخوردار از Block Scope استفاده کنید. استفاده از let به کدتان انعطاف بیشتری می‌دهد و اشکالات مرتبط با var را کاهش می‌دهد.

مقایسه سرعت تخصیص متغیر های ایستا و پویا را در جاوااسکریپت:

در جاوااسکریپت، سرعت تخصیص متغیرهای ایستا (با `const`) و متغیرهای پویا (با `let` یا `var`) تأثیر مستقیمی بر عملکرد ندارد، زیرا جاوااسکریپت یک زبان اجرایی (interpreted language) است و نه یک زبان کامپایل شده (compiled language). در زبان‌های کامپایل شده، تفاوت در تخصیص متغیرها ممکن است به دلیل فاصله زمانی کمتر بین تولید کد و اجرا مشهود باشد، اما در جاوااسکریپت، این تفاوت به دلیل ماهیت دینامیک و تفسیری زبان کمتر قابل مشاهده است.

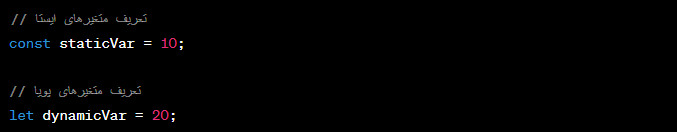
در واقع، در محیط اجرایی جاوااسکریپت، به طور کلی اهمیت زیادی به نحوه تعریف متغیر (ایستا یا پویا) نمی‌دهند و بیشتر به الگوها و عملکرد کد توجه می‌کنند.

متغیرهای ایستا با `const` تعریف شده‌اند و می‌توانند بهترین عملکرد را در برخی موارد ارائه دهند. از طرف دیگر، متغیرهای پویا با `let` یا `var` قابلیت انجام تغییرات دارند که در برخی موارد انعطاف‌پذیری بیشتری ارائه می‌دهد.

توصیه می‌شود که در انتخاب نوع متغیر بر اساس نیازها و اهداف کد نظر داشته باشید تا برنامه‌ی شما خواناتر و قابل‌ترکیب‌تر باشد. اهمیت اصلی باید به خواناتری و نگهداری کد داده شود تا بهینه‌سازی‌های جزئی در این زمینه تأثیر محدودی داشته باشند.

سرعت تخصیص متغیرها در جاوااسکریپت به طور کلی از نحوه اجرای کد و ماهیت زبان تفسیری (interpreted) آن تاثیر می‌پذیرد. در زبان‌های تفسیری مانند جاوااسکریپت، کد به صورت مستقیم توسط موتور اجرایی (مانند موتور V8 در مرورگرها یا موتور Node.js) تفسیر می‌شود و نیازی به یک مرحله کامپایل (compile) قبل از اجرا ندارد.

بنابراین، تفاوت در تخصیص متغیرهای ایستا (const) و پویا (let یا var) در جاوااسکریپت به نظر می‌رسد که تأثیر مستقیمی بر عملکرد کد نداشته باشد. مهمترین نکته این است که متغیرها به صورت دینامیک تایپ (Dynamic Typing) در جاوااسکریپت تعریف می‌شوند که به برنامه‌نویس اجازه می‌دهد تا در طول زمان نوع متغیرها را تغییر دهد.



در این مثال، هر دو staticVar و dynamicVar به صورت پویا تعریف شده‌اند، اما staticVar با const تعریف شده که نشان‌دهنده این است که مقدارش تا زمان اجرای برنامه تغییر نخواهد کرد. این تغییر در تعریف متغیرها به تنهایی تأثیر چندانی بر سرعت اجرا نخواهد گذاشت.

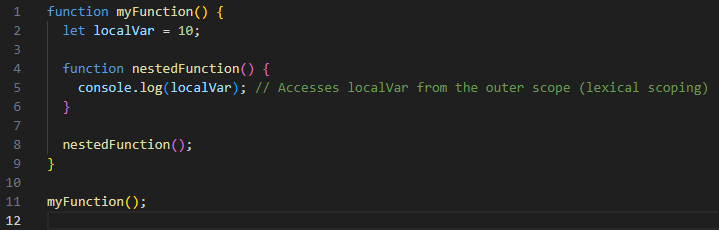
برای بهینه‌سازی کد، می‌توانید تلاش کنید که از متغیرهای ایستا (const) در جاهایی که مقدار ثابت استفاده می‌شود و از متغیرهای پویا (let یا var) در جاهایی که نیاز به تغییر مقدار وجود دارد، بهره ببرید. اما تفاوت در سرعت تخصیص به میزان قابل توجهی نبوده و بیشتر توجه به خواناتری و نگهداری کد باید داده شود.

آیا حوزه تعریف در زبان جاوااسکریپت ایستا است یا پویا؟ با ذکر مثال توضیح دهید.

در جاوا اسکریپت، دامنه تعاریف متغیر در درجه اول استاتیک است، زیرا بر اساس جایی که متغیرها در کد اعلام می شوند و نه در شرایط زمان اجرا تعیین می شوند. با این حال، ماهیت پویای جاوا اسکریپت همچنین رفتارهای خاصی را معرفی می کند که می تواند به صورت پویا در مورد این کلمه کلیدی و توابع تو در تو در نظر گرفته شود.

محدوده استاتیک

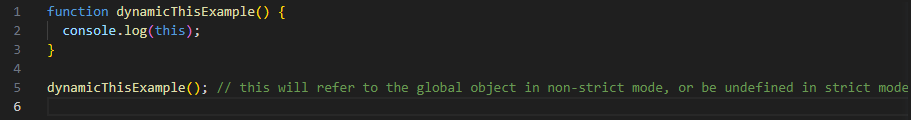
در محدوده استاتیک (همچنین به عنوان محدوده واژگانی شناخته می شود)، دامنه یک متغیر با مکان ان در کد منبع و زمینه واژگانی که در ان اعلام شده است تعیین می شود. موتور جاوا اسکریپت در زمان کامپایل می داند که باید به کدام محدوده متغیر برای حل یک متغیر نگاه کند. به عنوان مثال:



در این مورد، NestedFunction به دلیل محدوده استاتیک به localVar دسترسی دارد. محیط واژگانی NestedFunction شامل دامنه myFunction است که به ان اجازه دسترسی به localVar را می دهد.

محدوده داینامیک

در حالی که محدوده متغیر عمدتا استاتیک است، استفاده از کلمه کلیدی this در جاوا اسکریپت نوعی رفتار دامنه پویا را معرفی می کند. مقدار this به صورت پویا بر اساس چگونگی فراخوانی یک تابع تعیین می شود، نه جایی که تعریف می شود. این رفتار به انعطاف پذیری جاوا اسکریپت کمک می کند. مثال زیر را در نظر بگیرید:



مقدار this در dynamicThisExample بستگی به این دارد که چگونه تابع فراخوانی می شود، و ان را به یک شکل از محدوده پویا در زمینه اجرای تابع تبدیل می کند.

به طور خلاصه، دامنه تعاریف متغیر در جاوا اسکریپت در درجه اول استاتیک است. با این حال، رفتار پویای کلمه کلیدی this، شکلی از محدوده پویا را در توابع معرفی می کند. درک تعامل بین محدوده واژگانی، کلمه کلیدی this برای نوشتن کد جاوا اسکریپت قابل پیش بینی و قابل نگهداری بسیار مهم است.

پیاده‌سازی حوضه تعریف پویا در جاوا اسکریپت :

اضافه کردن حوزه تعریف پویا (Dynamic Scope) به جاوااسکریپت یک تغییر اساسی در طراحی زبان است و نیاز به تغییرات گسترده‌ای دارد. در حال حاضر، جاوااسکریپت از حوزه تعریف لکسیکال (Lexical Scope) استفاده می‌کند که به عنوان Scope یا Scope Chain نیز شناخته می‌شود. در حوزه تعریف لکسیکال، Scope ها بر اساس مکان تعریف توابع در کد قرار دارند و از این رو، دسترسی به متغیرها بر اساس نحوه تو در تویی از یکدیگر تعیین می‌شود.

برای ایجاد حوزه تعریف پویا، نیاز به تغییرات در نحوه عملکرد Scope ها و مکانیسم تعیین متغیرها داریم. این تغییرات به شدت تأثیرگذار بوده و احتمالاً با ساختار فعلی جاوااسکریپت تداخل دارد. با این حال، می‌توانیم یک سناریو ساده را برای نمونه توضیح دهیم:

سناریو ساده:

فرض کنید که می‌خواهیم یک ویژگی تعریف پویا به جاوااسکریپت اضافه کنیم. برای این کار، یک نحوه اجرا و تعیین Scope دیگر بر اساس اجرا‌ها ایجاد می‌کنیم.

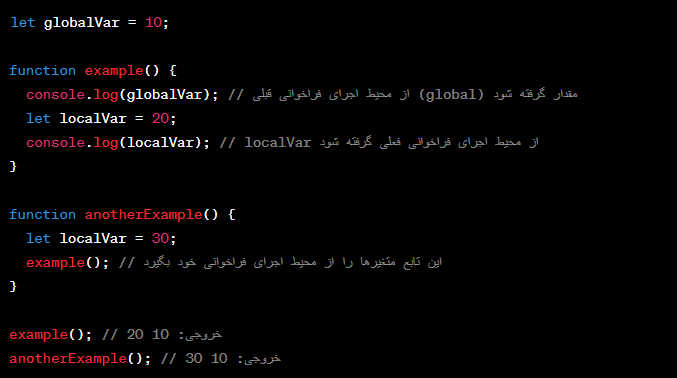
ایده اصلی:

متغیرهای تعریف شده در یک تابع به صورت پویا بر اساس محیط اجرای فراخوانی‌ها تغییر کنند.

تابع‌ها اطلاعات Scope خود را در هر فراخوانی ذخیره کنند.

اگر یک متغیر در محیط اجرا وجود نداشته باشد، به محیط اجرای قبلی نگاه کند.

پیاده‌سازی ساده:



این کد ساده نمایانگر یک ایده مبتنی بر حوزه تعریف پویا است. در هر فراخوانی تابع، متغیرها از محیط اجرای آن تابع گرفته شده و اگر یک متغیر در محیط اجرای فعلی وجود نداشته باشد، به محیط اجرای قبلی نگاه می‌کنند.

جاوااسکریپت در حال حاضر از حوزه تعریف لکسیکال استفاده می‌کند که به این معناست که Scope ها بر اساس مکان تعریف توابع در کد قرار دارند. این به این معناست که دسترسی به متغیرها بر اساس نحوه تو در تویی از یکدیگر تعیین می‌شود. حتی اگر یک تابع در داخل یک تابع دیگر تعریف شده باشد، Scope آن تابع داخلی محدود به Scope تابع بیرونی خود خواهد بود.

برای اجرای این تغییر و افزودن حوزه تعریف پویا، نیاز به تغییرات گسترده‌ای در موتورهای اجرای جاوااسکریپت (مثل V8 که توسط Google برای مرورگر Chrome ایجاد شده است) و استانداردهای جاوااسکریپت داریم. این تغییرات بسیار پیچیده و تأثیرگذار هستند و نیاز به تعیین استانداردهای جدید، تغییرات در موتورهای جاوااسکریپت، و حتی اصلاح یا تغییر در کد‌های موجود دارند.

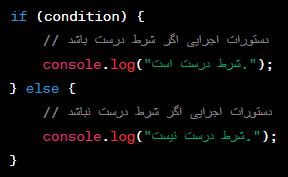
یک پیاده‌سازی واقعی این تغییرات بسیار پیچیده و بسیار وسیع خواهد بود و احتمالاً نیاز به مشارکت جامع از سوی توسعه‌دهندگان و تصمیم‌گیران جاوااسکریپت دارد. در زمینه توسعه زبان‌های برنامه‌نویسی، تغییرات اساسی مانند این نیاز به بررسی دقیق، آزمایش، و تأثیرات کلان روی اکوسیستم جاوااسکریپت دارند.

در حال حاضر، استفاده از حوزه تعریف لکسیکال در جاوااسکریپت به‌عنوان یک استاندارد تقریباً ثابت شده است و تغییر آن به یک سیاق بزرگتر نیاز دارد که از ابتدا در طراحی زبان لحاظ شود.

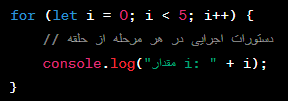
بلوک ها در جاوااسکریپت چگونه تعریف شده اند؟

در جاوااسکریپت، بلوک‌ها (blocks) به صورت گروهی از ادامه دادن دستورات می‌باشند که می‌توانند به عنوان یک واحد اجرایی در نظر گرفته شوند. بلوک‌ها در اغلب موارد با دستورات شرطی (مانند `if` یا `else`) و حلقه‌ها (مانند `for` یا `while`) مرتبط هستند. یک بلوک با استفاده از زوج آکولاد `{}` تعریف می‌شود.

مثال از یک بلوک در یک دستور شرطی (`if`):



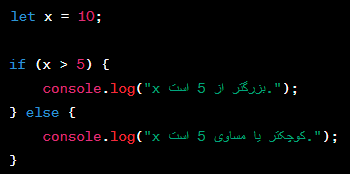
یک مثال دیگر از یک بلوک در یک حلقه (`for`):



در اینجا، بلوک داخلی هر دستور `if` یا `for` توسط آکولاد‌ها `{}` تعیین می‌شود. دستورات داخل بلوک باید توسط آکولاد‌ها محصور شوند، حتی اگر تنها یک دستور باشند. استفاده از آکولادها از ابهام جلوگیری می‌کند و کد را برای خوانایی بهتر افزایش می‌دهد.

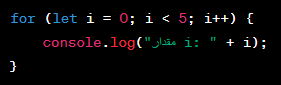
در جاوااسکریپت، بلوک‌ها با استفاده از آکولادها `{}` تعریف می‌شوند و دستورات داخل آکولادها به عنوان یک واحد اجرایی در نظر گرفته می‌شوند. زمانی که یک دستور شرطی یا یک حلقه تعریف می‌شود، دستورات متعلق به آن دستور داخل آکولادها قرار می‌گیرند.

مثال ۱: دستور شرطی (if-else)



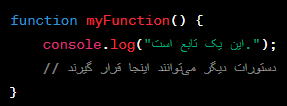
در این مثال، دستورات `console.log` داخل بلوک `if` و `else` قرار دارند.

مثال ۲: حلقه (for)



در این مثال، دستور `console.log` داخل بلوک حلقه `for` قرار دارد. بلوک اجرایی حلقه با `{}` شروع و پایان می‌یابد.

مثال ۳: تعریف تابع



در تعریف یک تابع نیز، دستورات تابع داخل یک بلوک آکولاد قرار دارند.

استفاده از بلوک‌ها باعث بهبود خوانایی کد، حفظ ساختار و قابلیت اطمینان در اجرای بخش‌های مختلف کد می‌شود. آکولادها به تعداد یکتا دستورات مربوطه را مشخص می‌کنند و از اشکالات جلوگیری می‌کنند.

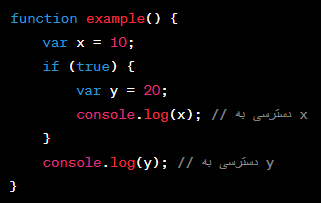
ایا کلمات کلیدی ویژه ای برای اعمال تغییر در حوزۀ تعریف متغیرها در جاوااسریپت وجود دارند؟

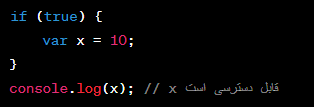
در جاوااسکریپت، متغیرها می‌توانند با استفاده از سه کلمه کلیدی `var`، `let` و `const` تعریف شوند. این کلمات کلیدی تفاوت‌هایی در تعریف متغیرها دارند:

1. var:

`var` یک متغیر را تعریف می‌کند و از آن به عنوان متغیر تابعی استفاده می‌کند (function-scoped).

ممکن است به هر نقطه‌ای در داخل یک تابع دسترسی داشته باشد، حتی اگر داخل بلوک شرطی یا حلقه باشد.

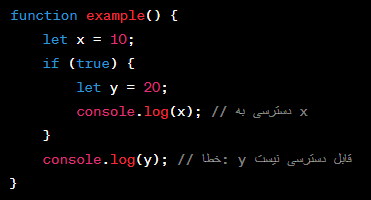


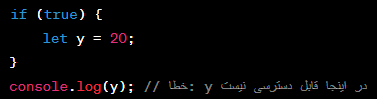


2. let:

`let` متغیرها را به عنوان متغیرهای بلوکی تعریف می‌کند (block-scoped).

این به این معناست که دسترسی به `let` در محدوده تعریف شده (بلوک، شرط، حلقه) محدود است.

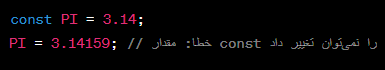




3. const:

`const` نیز متغیرها را به عنوان متغیرهای بلوکی تعریف می‌کند.

اما با این تفاوت که یکبار مقدار دهی شده‌اند، نمی‌توان آن‌ها را مجدداً تغییر داد.



به کمک این کلمات کلیدی، توسعه‌دهندگان در جاوااسکریپت می‌توانند متغیرهای خود را با توجه به نیازهای مختلف و محدودیت‌ها تعریف کنند.