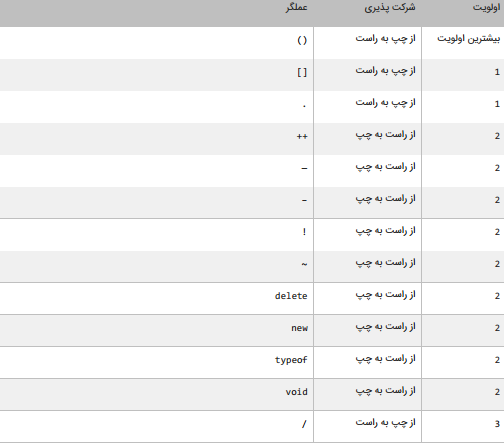
**پارت دوم**

# **4-2 تقدم عملگرها**

تقدم عملگرها مشخص میکند که در محاسباتی که بیش از دو عملوند دارند ابتدا کدام عملگر اثرش را اعمال کند. عملگرها در جاوااسکریپت در محاسبات دارای حق تقدم هستند. به عنوان مثال :

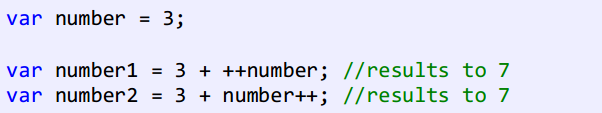


اگر حق تقدم عملگرها را رعایت نکنیم و عبارت بالا را از سمت چپ به راست انجام دهیم نتیجه 9 خواهد شد 3=2+1 سپس 9=3×3 و در آخر 9=1/9 اما کامپایلر با توجه به تقدم عملگرها محاسبات را انجام میدهد. برای مثال عمل ضرب و تقسیم نسبت به جمع و تفریق تقدم دارند. بنابراین در مثال فوق ابتدا عدد 2 ضربدر 3 و سپس نتیجه آنها تقسیم بر 1 میشود که نتیجه 6 به دست میآید. در آخر عدد 6 با 1 جمع میشود و عدد 7 حاصل میشود. در جدول زیر تقدم برخی از عملگرهای جاوااسکریپت آمده است :





ابتدا عملگرهای با بالاترین و سپس عملگرهای با پایینترین حق تقدم در محاسبات تأثیر میگذارند. به این نکته توجه کنید که تقدم عملگرها ++ و – به مکان قرارگیری آنها بستگی دارد ) در سمت چپ یا راست عملوند باشند (. به عنوان مثال :



در هر دو عبارت بالا به دلیل تقدم باالتر عملگر ++ نسبت به عملگر +، ابتدا به مقدار number یک واحد اضافه شده و 4 میشود و سپس مقدار جدید با عدد 3 جمع میشود و در نهایت عدد 7 به دست میآید. برای ایجاد خوانایی در تقدم عملگرها و انجام محاسباتی که در آنها از عملگرهای زیادی استفاده میشود از پرانتز استفاده میکنیم.



در مثال باال ابتدا هر کدام از عباراتی که داخل پرانتز هستند مورد محاسبه قرار میگیرند. به نکتهای در مورد عبارتی که در داخل پرانتز سوم قرار دارد توجه کنید. در این عبارت ابتدا مقدار داخلیترین پرانتز مورد محاسبه قرار میگیرد یعنی مقدار 6 ضربدر 7 شده و سپس از 5 کم میشود. اگر دو یا چند عملگر با حق تقدم یکسان موجود باشد ابتدا باید هر کدام از عملگرها را که در ابتدای عبارت میآیند مورد ارزیابی قرار دهید. به عنوان مثال:



هر دو عملگر \* و / دارای حق تقدم یکسانی هستند. بنابر این شما باید از چپ به راست آنها را در محاسبات تأثیر دهید. یعنی ابتدا 3 را ضربدر 2 میکنید و سپس عدد 8 را بر 4 تقسیم میکنید. در نهایت نتیجه دو عبارت را جمع کرده و در متغیر number قرار میدهید.

وقتی در یک عبارت محاسباتی، چند عملگر وجود داشته باشد، یک مشکل معنایی که به وجود می آید، ترتیب

ارزیابی عملگرهاست. برای مثال، در عبارت z ∗ y + x آیا باید ابتدا عملگرجمع ارزیابی شود و یا عملگر ضرب؟

− بدین منظور، تقدم عملگرها تعریف می شوند. برای مثال، اگر تقدم ضرب بیشتر از جمع تعریف شود، آنگاه

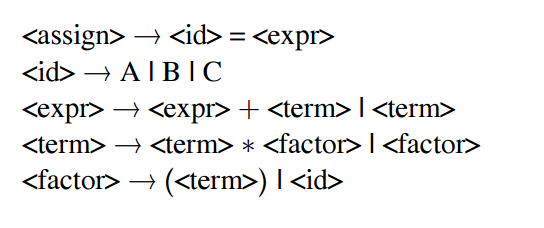
عملگر ضرب باید قبل از جمع ارزیابی شود.

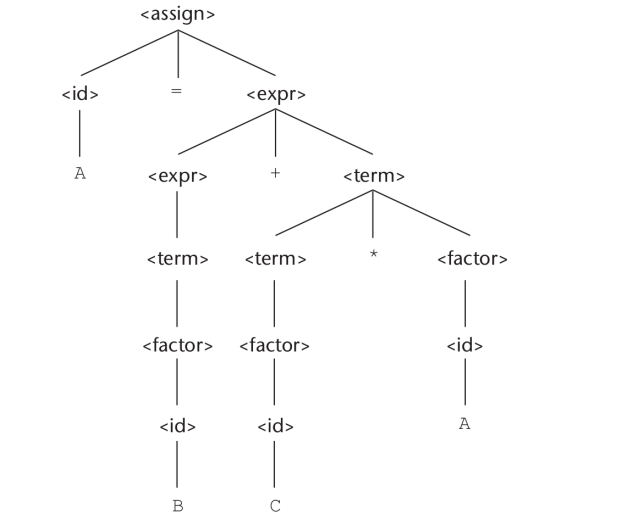
− برای گرامر مبهم قبلی می توانیم یک گرامر غیر مبهم بنویسیم به طوری که درخت تجزیه ابتدا عملگر جمع و

سپس عملگر ضرب را تجزیه کند. بدین ترتیب وقتی از برگ های درخت تجزیه برای ارزیابی یک عبارت آغاز

می کنیم ابتدا عملگر ضرب را اعمال می کنیم.

− گرامر غیر مبهم زیر معادل گرامر مبهم قبلی است

  
بدین ترتیب برای عبارت A ∗ C + B = A تنها یک درخت تجزیه به صورت زیر وجود خواهد داشت.

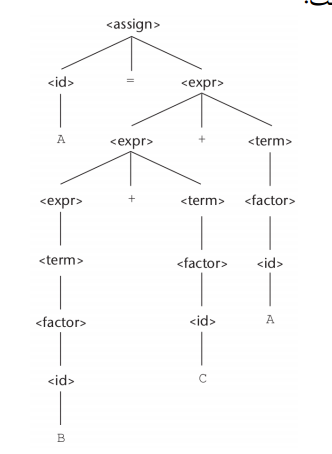


# **2-5 وابستگی عملگرها**

وقتی در یک عبارت دو عملگر وجود داشته باشد که تقدم برابر داشته باشند، به قوانین معنایی نیاز داریم تا مشخص می کند که در شرایطی که تقدم یکسان بدانیم کدام عملگر باید زودتر اجرا شود. وابستگی عملگرها است به کدام عملگر اولویت بالاتری داده می شود.

برای مثال عبارت A + Ⅽ + B = A را در نظر بگیرید. درخت تجزیه برای این عبارت طبق گرامر غیر مبهمی

که قبلا ارائه شده به صورت زیر است.



در این درخت تجزیه عملگر جمع اول زودتر محاسبه می شود. این عملیات صحیح است اگر وابستگی عملگر

جمع از سمت چپ باشد که معمولا هم همینطور است.

در ریاضیات می گوییم عملگر جمع خاصیت شرکت پذیری دارد، بدین معنی که وابستگی از چپ با وابستگی از راست معادل است یعنی: (A+B) + C = A + (B+C)

− در طراحی گرامر یک زبان برنامه نویسی وابستگی عملگرها باید در نظر گرفته شود.

− برای مثال در زبان سی تقدم عملگر \* و ++ یکسان است، اما وابستگی آنها از راست به چپ است، بدین

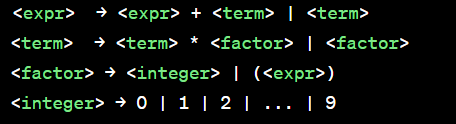
معنا که اگر این دو عملگر در کنار یکدیگر قرار بگیرند، کامپایلر ابتدا عملگر سمت راست را محاسبه می کند.

− عبارت ++p \*ابتدا مقدار اشاره گر را افزایش می دهد و سپس مقدار آن را ارزیابی می کند، زیرا تقدم این دو

عملگر یکسان و وابستگی آنها از راست به چپ است.

# **2-6 گرامر برای پیروی از تقدم**

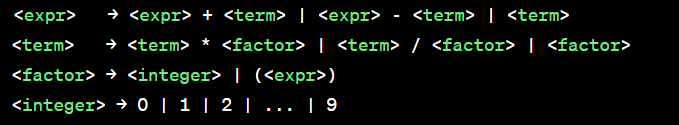
برای نوشتن گرامری که از تقدم عملگرها پیروی کند، می‌توانید از یک گرامر بازگشتی پویا استفاده کنید. در اینجا یک گرامر ساده برای تقدم عملگرها با استفاده از اعداد صحیح و عملگرهای جمع و ضرب به شکل BNF آورده شده است:



این گرامر به ما این امکان را می‌دهد که عباراتی با اعداد صحیح، جمع، و ضرب بسازیم. عملگرهای جمع و ضرب با توجه به این گرامر اولویت دارند، به این معنی که ضرب از جمع اولویت بیشتری دارد.



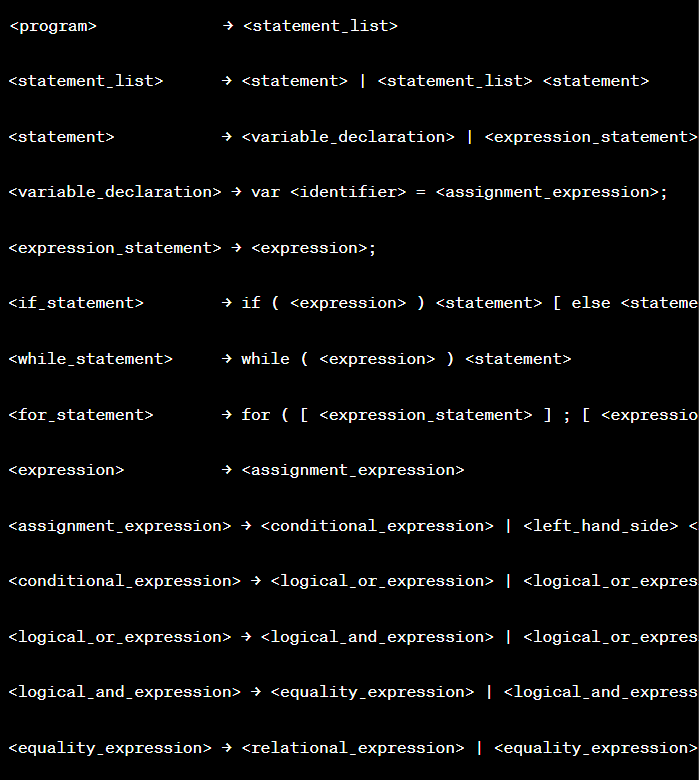
می‌تواند با استفاده از این گرامر نمایش داده شود. این نوع گرامرها معمولاً در توصیف اصول محاسباتی و زبان‌های برنامه‌نویسی به کار می‌روند.

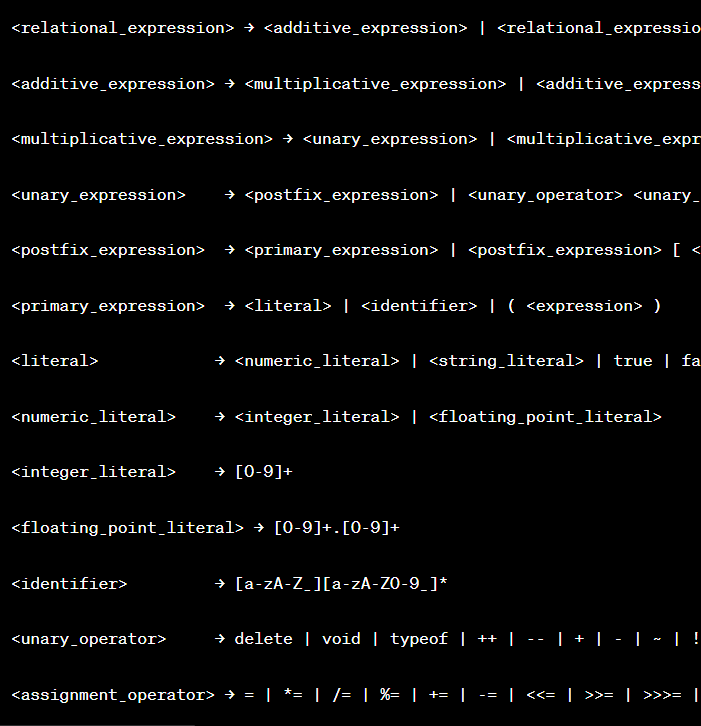


این گرامر به شما اجازه می‌دهد تا عباراتی با عملگرهای جمع، تفریق، ضرب، تقسیم و پرانتزها بسازید. عملگرهای با اولویت بالاتر بهترین اولویت را دارند.



می‌تواند با استفاده از این گرامر نمایش داده شود. این نوع گرامرها بسیار مفیدند در توصیف عبارات ریاضی و زبان‌های برنامه‌نویسی که عملیات محاسباتی مختلف را پشتیبانی می‌کنند.





این گرامر یک ساختار کلی از جاوا اسکریپت را نشان می‌دهد و شامل عبارات، تعریف متغیر، دستورات شرطی، حلقه‌ها، و عملگرهای مختلف است. توجه داشته باشید که این گرامر بسیار ساده‌تر از گرامر واقعی جاوا اسکریپت است.

# **7-2 معناشناسی عملیاتی**

در معناشناسی عملیاتی معنای عبارات یک برنامه با استفاده از تأثیر اجرای آنها بر روی ماشین توصیفمی شود.

تأثیر بر روی ماشین به معنی دنباله ای از تغییرات بر روی حالت ماشین است و حالت ماشین مجموعه ای از مقادیر بر روی حافظۀ آن است.

عملگر جمع (+) در جاوا اسکریپت:

زبان اسمبلی:



عملگر تفریق (-) در جاوا اسکریپت:

زبان اسمبلی:



عملگر ضرب (\*) در جاوا اسکریپت:

زبان اسمبلی:



عملگر تقسیم (/) در جاوا اسکریپت:

زبان اسمبلی:



## **1-7-2 ساختارهای تصمیم**

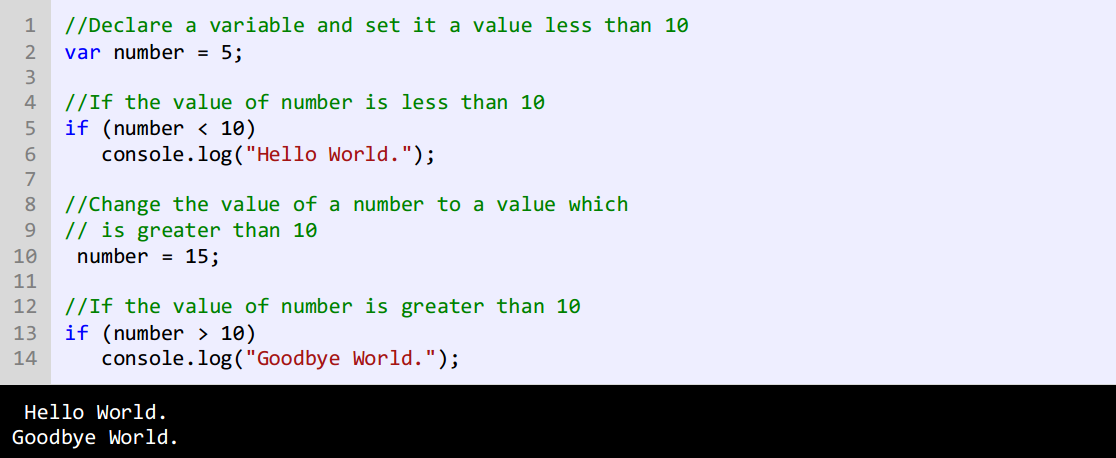
تقریبا همه زبانهای برنامه نویسی به شما اجازه اجرای کد را در شرایط مطمئن می دهند. حال تصور کنید که یک برنامه دارای ساختار تصمیم گیری نباشد و همه کدها را اجرا کند. این حالت شاید فقط برای چاپ یک پیغام در صفحه مناسب باشد ولی فرض کنید که شما بخواهید اگر مقدار یک متغیر با یک عدد برابر باشد سپس یک پیغام چاپ شود آن وقت با مشکل مواجه خواهید شد. JavaScript راه های مختلفی برای رفع این نوع مشکالت ارائه می دهد. در این بخش با مطالب زیر آشنا خواهید شد :

### 1-1-7-2 دستور if

میتوان با استفاده از دستور if و یک شرط خاص که باعث ایجاد یک کد میشود یک منطق به برنامه خود اضافه کنید. دستور if سادهترین دستور شرطی است که برنامه میگوید اگر شرطی برقرار است کد معینی را انجام بده. ساختار دستور if به صورت زیر است :

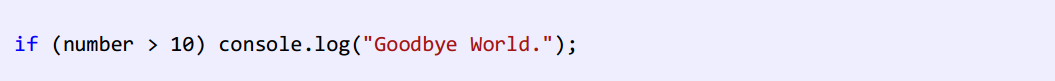


قبل از اجرای دستور if ابتدا شرط بررسی میشود. اگر شرط برقرار باشد یعنی درست باشد سپس کد اجرا میشود. شرط یک عبارت مقایسهای است. میتوان از عملگرهای مقایسهای برای تست درست یا اشتباه بودن شرط استفاده کرد. اجازه بدهید که نگاهی به نحوه استفاده از دستور if در داخل برنامه بیندازیم. برنامه زیر پیغام World Hello را اگر مقدار number کمتر از 10 و World Goodbye را اگر مقدار number از 10 بزرگتر باشد در صفحه نمایش میدهد.

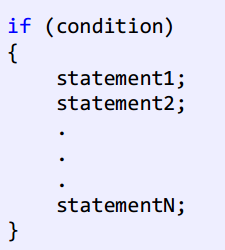


در خط 2 یک متغیر با نام number تعریف و مقدار 5 به آن اختصاص داده شده است. وقتی به اولین دستور if در خط 5 میرسیم برنامه تشخیص میدهد که مقدار number از 10 کمتر است یعنی 5 کوچکتر از 10 است.

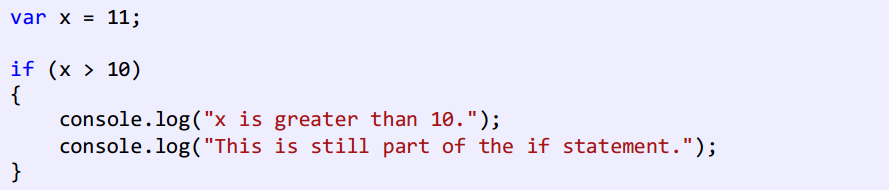
منطقی است که نتیجه مقایسه درست میباشد بنابراین دستور if دستور را اجرا میکند )خط 6 )و پیغام World Hello چاپ میشود. حال مقدار number را به 15 تغییر میدهیم )خط 10 .)وقتی به دومین دستور if در خط 13 میرسیم برنامه مقدار number را با 10مقایسه میکند و چون مقدار number یعنی 15 از 10 بزرگتر است برنامه پیغام World Goodbye را چاپ میکند )خط 14 .)به این نکته توجه کنید که دستور if را میتوان در یک خط نوشت :



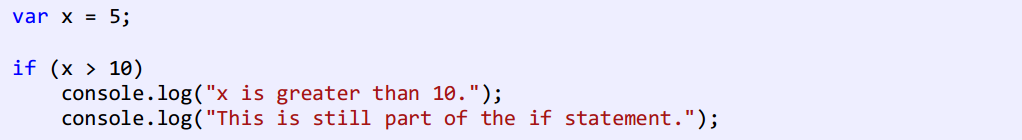
شما میتوانید چندین دستور را در داخل دستور if بنویسید. کافیست که از یک آکوالد برای نشان دادن ابتدا و انتهای دستورات استفاده کنید. همه دستورات داخل بین آکوالد جز بدنه دستور if هستند. نحوه تعریف چند دستور در داخل بدنه if به صورت زیر است :



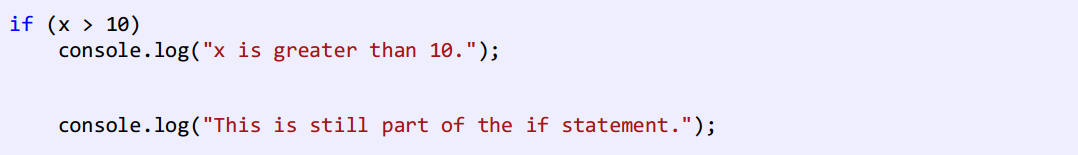
این هم یک مثال ساده :



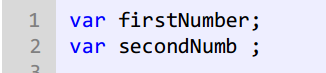
در مثال باال اگر مقدار x از 10 بزرگتر باشد دو پیغام چاپ میشود. حال اگر به عنوان مثال آکوالد را حذف کنیم و مقدار x از 10 بزرگتر نباشد مانند کد زیر :

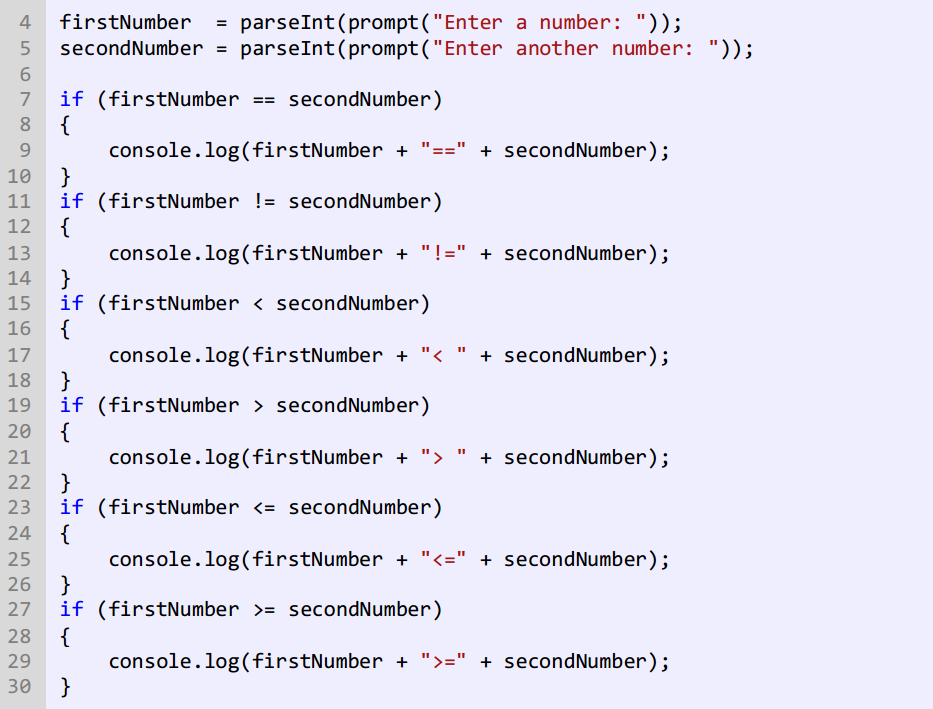


کد باال در صورتی بهتر خوانده میشود که بین دستورات فاصله بگذاریم.



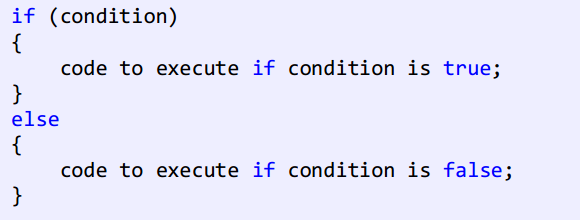
میبیند که دستور دوم یعنی خط آخر در مثال باال، جز دستور if نیست. اینجاست که چون ما فرض را بر این گذاشتهایم که مقدار x از 10کوچکتر است پس خط(?Really. (statement if the of part still is This چاپ میشود. در نتیجه اهمیت وجود آکوالد مشخص میشود. به عنوان تمرین همیشه حتی اگر فقط یک دستور در بدنه if داشتید برای آن یک آکوالد بگذارید. فراموش نکنید که از قلم انداختن یک آکوالد باعث به وجود آمدن خطا شده و یافتن آن را سخت میکند. مثالی دیگر در مورد دستور if:



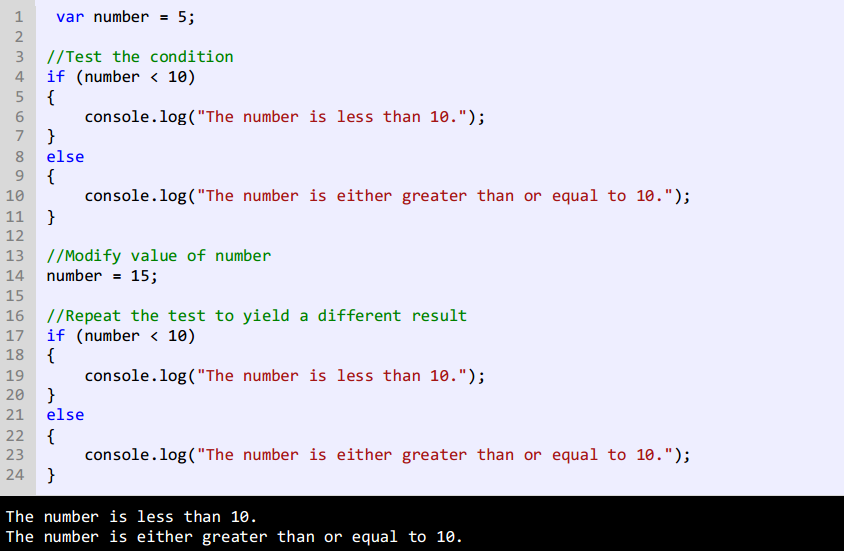


### **2-1-7-2 دستور if…else**

دستور if فقط برای اجرای یک حالت خاص به کار میرود یعنی اگر حالتی برقرار بود کار خاصی انجام شود. اما زمانی که شما بخواهید اگر شرط خاصی برقرار شد یک دستور و اگر برقرار نبود دستور دیگر اجرا شود باید از دستور else if استفاده کنید. ساختار دستور else if در زیر آمده است :



از کلمه کلیدی else نمیتوان به تنهایی استفاده کرد بلکه بدنه else دارید استفاده از آکوالد اختیاری است. کد داخل بلوک else فقط در صورتی اجرا میشود که شرط داخل دستور if نادرست باشد. در زیر نحوه استفاده از دستور else…if آمده است.



در خط 1 یک متغیر به نام number تعریف کردهایم و در خط 4 تست میکنیم که آیا مقدار متغیر number از 10 کمتر است یا نه و چون کمتر است در نتیجه کد داخل بلوک if اجرا میشود )خط 6 )و اگر مقدار number را تغییر دهیم و به مقداری بزرگتر از 10 تغییر دهیم)خط 14 ،)شرط نادرست میشود )خط 17 )و کد داخل بلوک else اجرا میشود )خط 23 .)مانند بلوک if نباید به آخر کلمه کلیدی else سیمیکولن اضافه شود.

### 3-1-7-2 عملگر شرطی

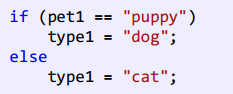
عملگر شرطی (:?) در جاوااسکریپت مانند دستور شرطی else…if عمل میکند. در زیر نحوه استفاده از این عملگر آمده است:



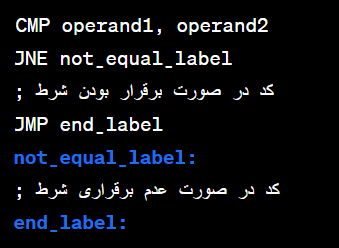
عملگر شرطی تنها عملگر سه تایی جاوااسکریپت است که نیاز به سه عملوند دارد، شرط، یک مقدار زمانی که شرط درست باشد و یک مقدار زمانی که شرط نادرست باشد. اجازه بدهید که نحوه استفاده این عملگر را در داخل برنامه مورد بررسی قرار دهیم.



برنامه باال نحوه استفاده از این عملگر شرطی را نشان میدهد. خط 6 به صورت زیر ترجمه میشود: اگر مقدار pet1 برابر با puppy سپس مقدار dog را در type1 قرار بده در غیر این صورت مقدار cat را type1 قرار بده. خط 7 به صورت زیر ترجمه میشود: اگر مقدار pet2 برابر با kitten سپس مقدار cat را در type2 قرار بده در غیر این صورت مقدار .dog حال برنامه باال را با استفاده از دستور else if مینویسیم:



زبان اسمبلی:



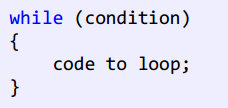
## **2-7-2 تکرار**

ساختارهای تکرار به شما اجازه میدهند که یک یا چند دستور کد را تا زمانی که یک شرط برقرار است تکرار کنید. بدون ساختارهای تکرار شما مجبورید همان تعداد کدها را بنویسید که بسیار خسته کننده است. راه بهتر برای نوشتن کدهای باال استفاده از حلقهها است. حلقهها در جاوااسکریپت عبارتند از :

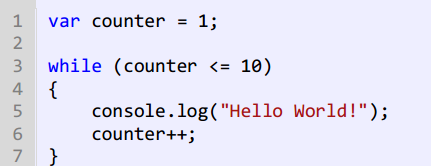
* While
* Do while
* For

### **1-2-7-2 حلقه while**

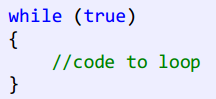
ابتداییترین ساختار تکرار در جاوااسکریپت حلقه While است. ابتدا یک شرط را مورد بررسی قرار میدهد و تا زمانیکه شرط برقرار باشد کدهای درون بلوک اجرا میشوند. ساختار حلقه While به صورت زیر است :



به یک متغیر شمارنده در داخل بدنه حلقه نیاز داریم. این شمارنده برای آزمایش شرط مورد استفاده قرار میگیرد و ادامه یا توقف حلقه به نوعی به آن وابسته است. این شمارنده را در داخل بدنه باید کاهش یا افزایش دهیم. در برنامه زیر نحوه استفاده از حلقه While آمده است :

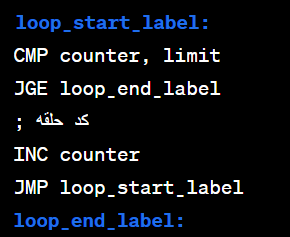


اگر مقدار شمارنده یک بماند و آن را افزایش ندهیم و یا مقدار شرط هرگز false نشود یک حلقه بینهایت به وجود میآید. به این نکته توجه کنید که در شرط باال به جای عالمت > از => استفاده شده است. اگر از عالمت > استفاده میکردیم کد ما 9 بار تکرار میشد چون مقدار اولیه 1 است و هنگامی که شرط به 10 برسد false میشود چون 10 10 < نیست. اگر میخواهید یک حلقه بی نهایت ایجاد کنید که هیچگاه متوقف نشود باید یک شرط ایجاد کنید که همواره درست (true (باشد.



این تکنیک در برخی موارد کارایی دارد و آن زمانی است که شما بخواهید با استفاده از دستورات break و return که در آینده توضیح خواهیم داد از حلقه خارج شوید.

زبان اسمبلی:

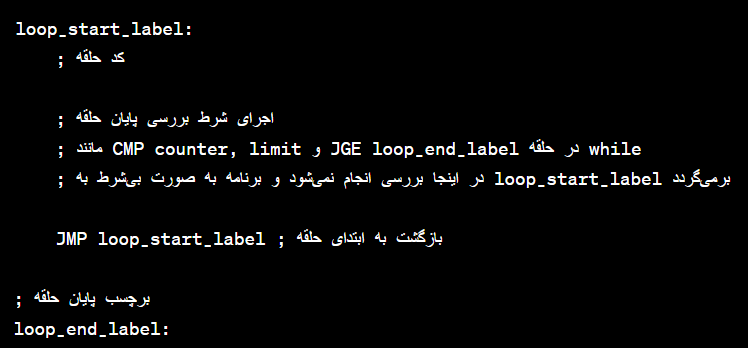


### **2-2-7-2 حلقه do while**

حلقه while do یکی دیگر از ساختارهای تکرار است. این حلقه بسیار شبیه حلقه while است با این تفاوت که در این حلقه ابتدا کد اجرا می شود و سپس شرط مورد بررسی قرار میگیرد. ساختار حلقه while do به صورت زیر است :

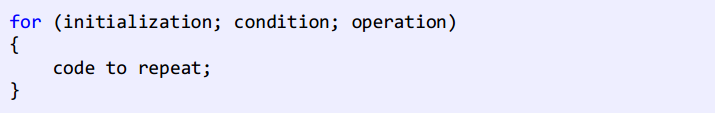


زبان اسمبلی:



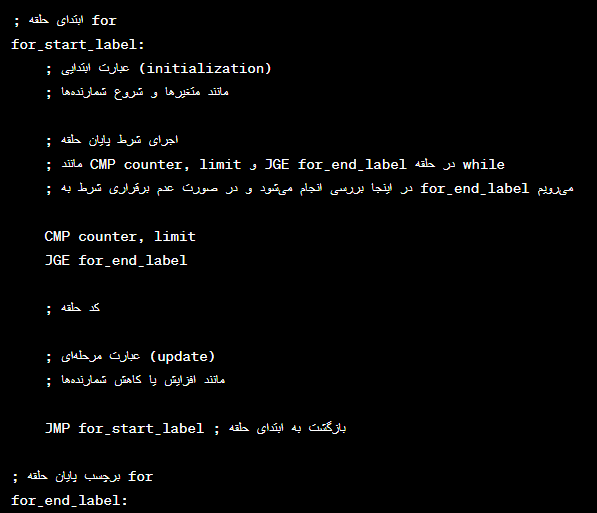
### **3-2-7-2 حلقه for**

یکی دیگر از ساختارهای تکرار حلقه for است. این حلقه عملی شبیه به حلقه while انجام میدهد و فقط دارای چند خصوصیت اضافی است. ساختار حلقه for به صورت زیر است :



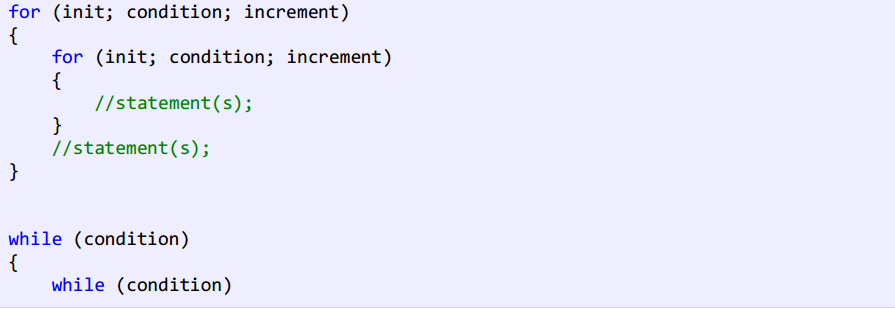
* مقدار دهی اولیه (initialization (اولین مقداری است که به شمارنده حلقه میدهیم. شمارنده فقط در داخل حلقه for قابل دسترسی است.
* شرط (condition (در اینجا مقدار شمارنده را با یک مقدار دیگر مقایسه میکند و تعیین میکند که حلقه ادامه یابد یا نه.
* عملگر (operation (که مقدار اولیه متغیر را کاهش یا افزایش میدهد.

زبان اسمبلی:



### **4-2-7-2 حلقه های تودرتو (Nested Loops)**

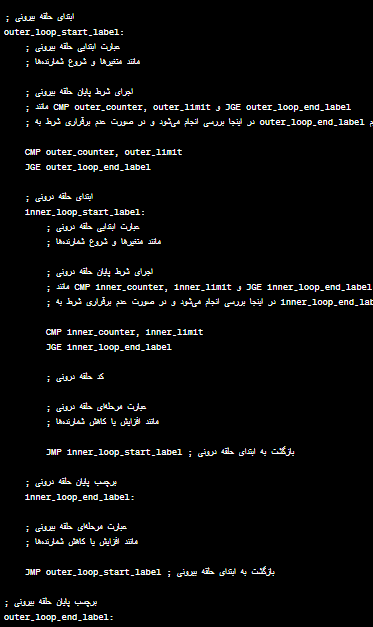
جاوااسکریپت به شما اجازه میدهد که از حلقهها به صورت تو در تو استفاده کنید. اگر یک حلقه در داخل حلقه دیگر قرار بگیرد، به آن حلقه تو در تو گفته میشود. در این نوع حلقهها، به ازای اجرای یک بار حلقه بیرونی، حلقه داخلی به طور کامل اجرا میشود. در زیر نحوه ایجاد حلقه تو در تو آمده است :





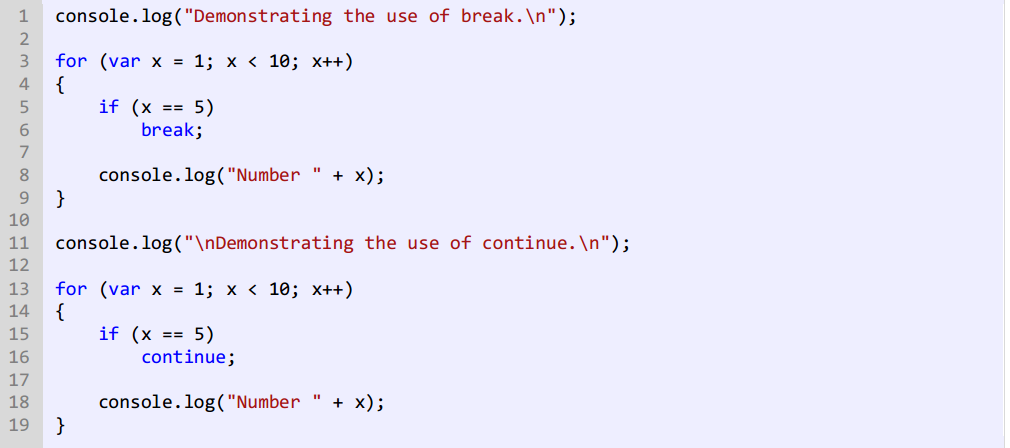
نکتهای که در مورد حلقههای تو در تو وجود دارد این است که، میتوان از یک نوع حلقه در داخل نوع دیگر استفاده کرد. مثال میتوان از حلقه for در داخل حلقه while استفاده نمود.

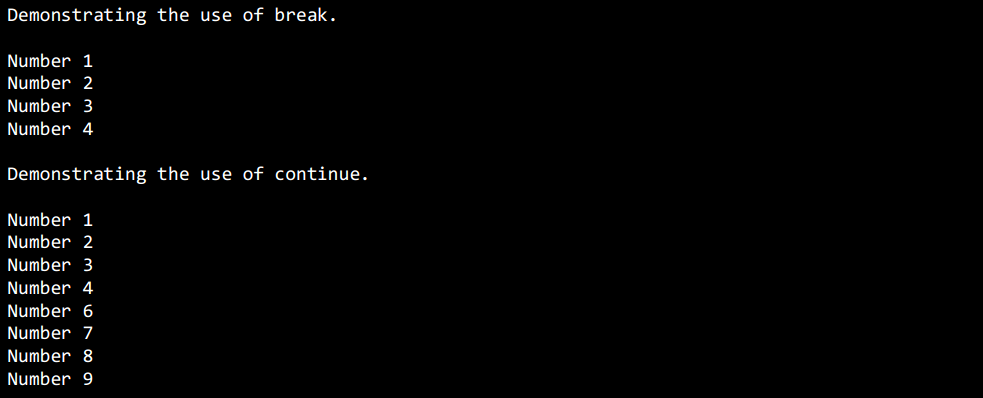
زبان اسمبلی:



### **5-2-7-2 خار ج شدن از حلقه با استفاده از break و continue**

گاهی اوقات با وجود درست بودن شرط میخواهیم حلقه متوقف شود. سؤال اینجاست که چطور این کار را انجام دهید؟ با استفاده از کلمه کلیدی break حلقه را متوقف کرده و با استفاده از کلمه کلیدی continue میتوان بخشی از حلقه را رد کرد و به مرحله بعد رفت. برنامه زیر نحوه استفاده از continue و break را نشان میدهد:





در این برنامه از حلقه for برای نشان دادن کاربرد دو کلمه کلیدی فوق استفاده شده است اگر به جای for از حلقههای while و استفاده میشد نتیجه یکسانی به دست میآمد. همانطور که در شرط برنامه )خط 5 )آمده است، وقتی که مقدار x به عدد 5 برسد، سپس دستور break اجرا )خط 6 )و حلقه بالفاصله متوقف میشود، حتی اگر شرط 10 < x برقرار باشد. از طرف دیگر در خط 15 حلقه for فقط برای یک تکرار خاص متوقف شده و سپس ادامه مییابد. )وقتی مقدار x برابر 5 شود حلقه از 5 رد شده و مقدار 5 را چاپ نمیکند و بقیه مقادیر چاپ میشوند.

تعریف متغیر در جاوا اسکریپت: در بخش سوم به طور مفصل در مورد متغییر صحبت شده است.

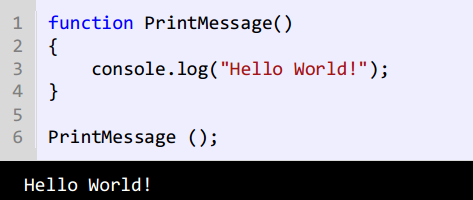
زبان اسمبلی:



## **3-7-2 تابع**

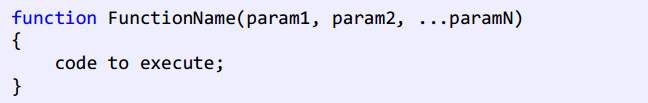
تابع به شما اجازه می دهند که یک رفتار یا وظیفه را تعریف کنید و مجموعه ای از کدها است که در هر جای برنامه می توان از آنها استفاده کرد. توابع در جاوااسکریپت و اکثر زبانهای برنامه نویسی بر دو نوعند :

* توابع از پیش تعریف شده
* توابعی که توسط کاربر تعریف می شوند.



### **1-3-7-2 پارامترها و آرگومان ها**

پارامترها، دادههای خامی هستند که تابع آنها را پردازش میکند و سپس اطالعاتی را که به دنبال آن هستید در اختیار شما قرار میدهد. فرض کنید پارامترها مانند اطالعاتی هستند که شما به یک کارمند میدهید که بر طبق آنها کارش را به پایان برساند. یک تابع میتواند هر تعداد پارامتر داشته باشد. هر پارامتر میتواند از انواع مختلف داده باشد. در زیر یک تابع با N پارامتر نشان داده شده است :



زبان اسمبلی:



