عملیات تعریف متغیر

تعریف متغیر به شکل var :

;<variable-definition> → var <identifier> = <expression>

تعریف متغیر به شکل let :

;<variable-definition> → let <identifier> = <expression>

تعریف متغیر به شکل const :

;<variable-definition> → const <identifier> = <expression>

گرامری که ارائه داده شد، یک قالب برای تعریف متغیر در زبان جاوااسکریپت است. برای توضیح بیشتر، اجزای این گرامر به شرح زیر هستند:

<variable-definition>: این نشان‌دهنده شروع تعریف متغیر است.

var: این کلیدواژه به معنای تعریف یک متغیر استفاده می‌شود. در جاوااسکریپت، می‌توان از var، let یا const برای تعریف متغیر استفاده کرد. این گرامر از var استفاده کرده است.

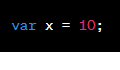
<identifier>: این بخش نمایانگر نام متغیر (شناسه) است. در جاوااسکریپت، نام متغیرها باید با یک حرف شروع شوند و می‌توانند شامل حروف، اعداد، یا خط تیره باشند.

=: علامت مساوی، که به متغیر مقدار اولیه اختصاص می‌دهد.

<expression>: عبارتی که مقدار اولیه متغیر را تعیین می‌کند. این می‌تواند هر نوع عبارت معتبر در جاوااسکریپت باشد.

در نهایت، ; نشان‌دهنده پایان دستور یا تعریف متغیر است.

مثال:



این مثال یک متغیر به نام x تعریف کرده و مقدار اولیه آن را برابر با ۱۰ قرار داده است. این تعریف متغیر با گرامر ارائه شده سازگار است.

عملیات جمع

برای تعریف گرامر عملیات جمع در زبان جاوااسکریپت می‌توانید از گرامر زیر استفاده کنید. در اینجا یک گرامر ساده برای عملیات جمع ارائه شده است:

<addition-expression> → <operand> + <operand>

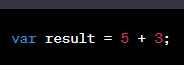
این گرامر یک عبارت جمع را توصیف می‌کند که شامل دو <operand> (اپرند) است که توسط + افزوده شده‌اند.

: <addition-expression> این نشان‌دهنده شروع یک عبارت جمع است.

: <operand> این بخش نمایانگر عملوندها یا اجزای عبارت جمع است.

+ : این علامت جمع نشان‌دهنده عمل جمع است.

از این گرامر می‌توانید برای توصیف عبارت‌های جمعی مانند زیر استفاده کنید:



در این مثال، <addition-expression> با <operand> های 5 و 3 توسط + جمع می‌شود و نتیجه در متغیر result ذخیره می‌شود.

عملیات تفریق

برای تعریف گرامر عملیات تفریق در زبان جاوااسکریپت می‌توانید از گرامر زیر استفاده کنید. در اینجا یک گرامر ساده برای عملیات تفریق ارائه شده است:

< subtraction-expression> → <operand> - <operand>

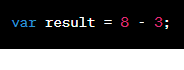
این گرامر یک عبارت تفریق را توصیف می‌کند که شامل دو <operand> (اپرند) است که توسط - تفریق شده‌اند.

: < subtraction-expression> این نشان‌دهنده شروع یک عبارت تفریق است.

: <operand> این بخش نمایانگر عملوندها یا اجزای عبارت تفریق است.

- : این علامت تفریق نشان‌دهنده عمل تفریق است.

از این گرامر می‌توانید برای توصیف عبارت‌های تفریق مانند زیر استفاده کنید:



در این مثال، < subtraction-expression> با <operand> های 8 و 3 توسط - تفریق می‌شود و نتیجه در متغیر result ذخیره می‌شود.

عملیات ضرب

برای تعریف گرامر عملیات ضرب در زبان جاوااسکریپت می‌توانید از گرامر زیر استفاده کنید. در اینجا یک گرامر ساده برای عملیات ضرب ارائه شده است:

< multiplication-expression> → <operand> × <operand>

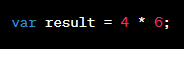
این گرامر یک عبارت ضرب را توصیف می‌کند که شامل دو <operand> (اپرند) است که توسط × ضرب شده‌اند.

: < multiplication-expression> این نشان‌دهنده شروع یک عبارت ضرب است.

: <operand> این بخش نمایانگر عملوندها یا اجزای عبارت ضرب است.

× : این علامت ضرب نشان‌دهنده عمل ضرب است.

از این گرامر می‌توانید برای توصیف عبارت‌های ضرب مانند زیر استفاده کنید:



در این مثال، < multiplication-expression> با <operand> های 4 و 6 توسط × ضرب می‌شود و نتیجه در متغیر result ذخیره می‌شود.

عملیات توان

برای تعریف گرامر عملیات توان در زبان جاوااسکریپت می‌توانید از گرامر زیر استفاده کنید. در اینجا یک گرامر ساده برای عملیات توان ارائه شده است:

< exponentiation-expression> → <operand> \*\* <operand>

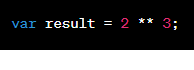
این گرامر یک عبارت توان را توصیف می‌کند که شامل دو <operand> (اپرند) است که توسط \*\* به توان رسیده‌اند.

: <exponentiation-expression> این نشان‌دهنده شروع یک عبارت توان است.

: <operand> این بخش نمایانگر عملوندها یا اجزای عبارت توان است.

\*\*: این علامت توان نشان‌دهنده عمل توان است.

از این گرامر می‌توانید برای توصیف عبارت‌های توان مانند زیر استفاده کنید:



در این مثال، <exponentiation-expression> با <operand> های 2 و 3 توسط \*\* به توان می‌رسند و نتیجه در متغیر result ذخیره می‌شود.

عملیات تقسیم

برای تعریف گرامر عملیات تقسیم در زبان جاوااسکریپت می‌توانید از گرامر زیر استفاده کنید. در اینجا یک گرامر ساده برای عملیات تقسیم ارائه شده است:

< division-expression> → <operand> / <operand>

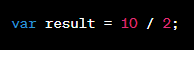
این گرامر یک عبارت تقسیم را توصیف می‌کند که شامل دو <operand> (اپرند) است که توسط / تقسیم شده‌اند.

: < division-expression> این نشان‌دهنده شروع یک عبارت تقسیم است.

: <operand> این بخش نمایانگر عملوندها یا اجزای عبارت تقسیم است.

/: این علامت تقسیم نشان‌دهنده عمل تقسیم است.

از این گرامر می‌توانید برای توصیف عبارت‌های تقسیم مانند زیر استفاده کنید:



در این مثال، < division-expression> با <operand> های 2 و 3 توسط / تقسیم شده‌اند و نتیجه در متغیر result ذخیره می‌شود.

عملیات باقی‌مانده تقسیم

برای تعریف گرامر عملیات باقی‌مانده تقسیم در زبان جاوااسکریپت می‌توانید از گرامر زیر استفاده کنید. در اینجا یک گرامر ساده برای عملیات باقی‌مانده تقسیم ارائه شده است:

< remainder-expression> → <operand> % <operand>

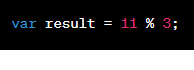
این گرامر یک عبارت باقی‌مانده تقسیم را توصیف می‌کند که شامل دو <operand> (اپرند) است که توسط % باقی‌مانده تقسیم یکی بر دیگری بدست می‌آید.

: < remainder-expression> این نشان‌دهنده شروع یک عبارت باقی‌مانده تقسیم است.

: <operand> این بخش نمایانگر عملوندها یا اجزای عبارت باقی‌مانده تقسیم است.

%: این علامت تقسیم نشان‌دهنده عمل باقی‌مانده تقسیم است.

از این گرامر می‌توانید برای توصیف عبارت‌های باقی‌مانده تقسیم مانند زیر استفاده کنید:



در این مثال، < remainder-expression> با <operand> های 11 و 3 توسط % باقی‌مانده تقسیم‌شان بدست می‌آید و نتیجه در متغیر result ذخیره می‌شود.

عملیات افزایش

برای تعریف گرامر عملیات افزایش در زبان جاوااسکریپت می‌توانید از گرامر زیر استفاده کنید. در اینجا یک گرامر ساده برای عملیات افزایش ارائه شده است:

++<increment-expression> → <operand>

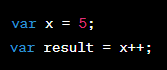
این گرامر می‌گوید که یک عبارت افزایش از یک <operand> تشکیل شده است که به آن یک واحد افزوده شده است.

: < increment-expression> این نشان‌دهنده شروع یک عبارت افزایش است.

: <operand> این بخش نمایانگر عملوندها یا اجزای عبارت افزایش است.

++: این علامت افزایش نشان‌دهنده عمل افزایش است.

از این گرامر می‌توانید برای توصیف عبارت‌های افزایش مانند زیر استفاده کنید:



در این مثال مقدار x یک واحد افزایش یافته و در result ذخیره می‌شود و x مقدار قبلی خود را نگه می‌دارد.

عملیات کاهش

برای تعریف گرامر عملیات کاهش (decrement) در زبان جاوااسکریپت، می‌توانید از یک گرامر به صورت زیر استفاده کنید:

--<decrement-expression> → <operand>

در این گرامر:

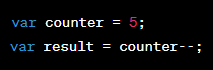
<decrement-expression>: نشان‌دهنده شروع یک عبارت کاهش است.

<operand>: عملوندها یا اجزای عبارت کاهش.

--: عملگر کاهش.

این گرامر می‌گوید که یک عبارت کاهش از یک <operand> تشکیل شده است که یک واحد از آن کم می‌شود.

برای مثال:



در اینجا:

<decrement-expression> توسط counter-- توصیف می‌شود.

<operand> یک بار تکرار شده و مقدار counter را نشان می‌دهد.

-- نشان‌دهنده عملگر کاهش است.

بعد از اجرای این کد، مقدار counter از 5 به 4 کاهش پیدا می‌کند، و مقدار result هم مقدار قبلی counter (یعنی 5) را نگه می‌دارد. این عمل به معنای کاهش counter پس از استفاده از مقدار فعلی آن است.

عملیات مساوی

برای تعریف گرامر عملیات مساوی (equal to) در زبان جاوااسکریپت، می‌توانید از یک گرامر به صورت زیر استفاده کنید:

<equal-to-expression> → <operand> == <operand>

در این گرامر:

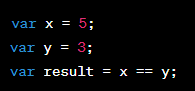
<equal-to-expression>: نشان‌دهنده شروع یک عبارت مساوی است.

<operand>: عملوندها یا اجزای عبارت مساوی.

==: علامت مساوی.

این گرامر می‌گوید که یک عبارت مساوی از دو <operand> که توسط == تساوی آن‌ها بررسی می‌شود، تشکیل شده است.

برای مثال:



در اینجا:

<equal-to-expression> توسط x == y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

== نشان‌دهنده علامت مساوی است.

در این مثال تساوی دو operand بررسی شده و نتیجه(true,false) در result ذخیره می‌شود.

عملیات مقدار مساوی و نوع مساوی

برای تعریف گرامر عملیات مقدار مساوی و نوع مساوی (equal value and equal type) در زبان جاوااسکریپت، می‌توانید از یک گرامر به صورت زیر استفاده کنید:

<equal-value-and-type-expression> → <operand> === <operand>

در این گرامر:

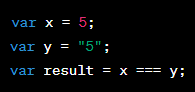
<equal-value-and-type-expression>: نشان‌دهنده شروع یک عبارت مقدار مساوی و نوع مساوی است.

<operand>: عملوندها یا اجزای عبارت مقدار مساوی و نوع مساوی.

===: علامت مساوی و نوع مساوی.

این گرامر می‌گوید که یک عبارت مقدار مساوی و نوع مساوی از دو <operand> تشکیل شده که توسط === مساوی بودن مقدار و نوع آن‌ها بررسی می‌شود.

برای مثال:



در اینجا:

<equal-value-and-type-expression> توسط x === y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

=== نشان‌دهنده علامت مساوی و نوع مساوی است.

در این مثال برابر بودن مقدار و نوع x و y بررسی می‌شود و نتیجه در result ذخیره می‌شود.

عملیات نامساوی بودن مقدار و نوع

برای تعریف گرامر عملیات نامساوی مقدار و نامساوی نوع (not equal value and not equal type) در زبان جاوااسکریپت، می‌توانید از یک گرامر به صورت زیر استفاده کنید:

<not-equal-value-and-type-expression> → <operand> !== <operand>

در این گرامر:

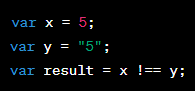
<not-equal-value-and-type-expression>: نشان‌دهنده شروع یک عبارت نامساوی مقدار و نامساوی نوع است.

<operand>: عملوندها یا اجزای عبارت نامساوی مقدار و نامساوی نوع.

!==: علامت نامساوی مقدار و نامساوی نوع.

این گرامر می‌گوید که یک عبارت نامساوی مقدار و نامساوی نوع از دو <operand> که توسط !== نامساوی بودن operand ها بررسی می‌شود، ، تشکیل شده است.

برای مثال:



در اینجا:

<not-equal-value-and-type-expression> توسط x !== y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

!== نشان‌دهنده علامت نامساوی مقدار و نامساوی نوع است.

در این مثال عدم برابر بودن مقدار و نوع x و y بررسی می‌شود و نتیجه در result ذخیره می‌شود.

عملیات نامساوی مقدار یا نامساوی نوع

برای تعریف گرامر عملیات نامساوی مقدار یا نامساوی نوع (not equal value or not equal type) در زبان جاوااسکریپت، می‌توانید از یک گرامر به صورت زیر استفاده کنید:

<not-equal-value-or-type-expression> → <operand> != <operand>

در این گرامر:

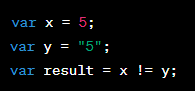
<not-equal-value-or-type-expression>: نشان‌دهنده شروع یک عبارت نامساوی مقدار یا نامساوی نوع است.

<operand>: عملوندها یا اجزای عبارت نامساوی مقدار یا نامساوی نوع.

!=: علامت نامساوی مقدار یا نامساوی نوع.

این گرامر می‌گوید که یک عبارت نامساوی مقدار یا نامساوی نوع از دو <operand> که توسط !== نامساوی بودن مقدار یا نامساوی نوع operand ها بررسی می‌شود، تشکیل شده است.

برای مثال:



در اینجا:

<not-equal-value-or-type-expression> توسط x != y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

!= نشان‌دهنده علامت نامساوی مقدار یا نامساوی نوع است.

در این مثال عدم برابر بودن مقدار یا نوع x و y بررسی می‌شود و نتیجه در result ذخیره می‌شود.

عملیات بزرگ‌تر

برای تعریف گرامر عملیات بزرگتر از (greater than) در زبان جاوااسکریپت، می‌توانید از یک گرامر به صورت زیر استفاده کنید:

<greater-than-expression> → <operand> > <operand>

در این گرامر:

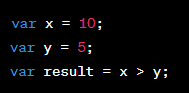
<greater-than-expression>: نشان‌دهنده شروع یک عبارت بزرگتر از است.

<operand>: عملوندها یا اجزای عبارت بزرگتر از.

>: علامت بزرگتر از.

این گرامر می‌گوید که یک عبارت بزرگتر از از دو <operand> که توسط > بزرگتر بودن operand ها بررسی می‌شود، تشکیل شده است.

برای مثال:



در اینجا:

<greater-than-expression> توسط x > y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

> نشان‌دهنده علامت بزرگتر از است.

در این مثال بزگ‌تر بودن x و y بررسی می‌شود و نتیجه در result ذخیره می‌شود.

عملیات کوچک‌تر

برای تعریف گرامر عملیات کمتر از (less than) در زبان جاوااسکریپت، می‌توانید از یک گرامر به صورت زیر استفاده کنید:

<less-than-expression> → <operand> < <operand>

در این گرامر:

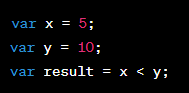
<less-than-expression>: نشان‌دهنده شروع یک عبارت کمتر از است.

<operand>: عملوندها یا اجزای عبارت کمتر از.

<: علامت کمتر از.

این گرامر می‌گوید که یک عبارت کمتر از از دو <operand> که توسط < کوچک تر بودن یکی از دیگری بررسی می‌شود، تشکیل شده است.

برای مثال:



در اینجا:

<less-than-expression> توسط x < y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

< نشان‌دهنده علامت کمتر از است.

در این مثال کوچک‌تر بودن x و y بررسی می‌شود و نتیجه در result ذخیره می‌شود.

عملیات بزرگ‌تر مساوی

برای تعریف گرامر عملیات بزرگ‌تر مساوی (greater than or equal to) در زبان جاوااسکریپت، می‌توانید از یک گرامر به صورت زیر استفاده کنید:

<greater-than-or-equal-to-expression> → <operand> >= <operand>

در این گرامر:

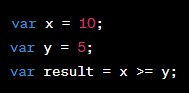
<greater-than-or-equal-to-expression>: نشان‌دهنده شروع یک عبارت بزرگتر مساوی است.

<operand>: عملوندها یا اجزای عبارت بزرگتر مساوی.

>=: علامت بزرگتر مساوی.

این گرامر می‌گوید که یک عبارت بزرگتر مساوی از دو <operand> که توسط >= بزرگ‌تر یکی از دیگری بررسی می‌شود، تشکیل شده است.

برای مثال:



در اینجا:

<greater-than-or-equal-to-expression> توسط x >= y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

>= نشان‌دهنده علامت بزرگتر مساوی است.

در این مثال بزرگ‌تر مساوی بودن x و y بررسی می‌شود و نتیجه در result ذخیره می‌شود.

عملیات کم‌تر مساوی

برای تعریف گرامر عملیات کم‌تر مساوی (less than or equal to) در زبان جاوااسکریپت، می‌توانید از یک گرامر مشابه به صورت زیر استفاده کنید:

<less-than-or-equal-to-expression> → <operand> <= <operand>

در این گرامر:

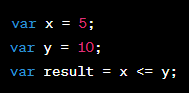
<less-than-or-equal-to-expression>: نشان‌دهنده شروع یک عبارت کمتر مساوی است.

<operand>: عملوندها یا اجزای عبارت کمتر مساوی.

<=: علامت کمتر مساوی.

این گرامر می‌گوید که یک عبارت کمتر مساوی از دو <operand> که توسط <= کمتر مساوی بودن یکی از دیگری بررسی می‌شود، تشکیل شده است.

برای مثال:



در اینجا:

<less-than-or-equal-to-expression> توسط x <= y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

<= نشان‌دهنده علامت کمتر مساوی است.

در این مثال کم‌تر مساوی بودن x و y بررسی می‌شود و نتیجه در result ذخیره می‌شود.

عملیات عامل سه‌گانه

برای تعریف گرامر عملیات عامل سه‌گانه (ternary operator) در زبان جاوااسکریپت، می‌توانید از یک گرامر به صورت زیر استفاده کنید:

<ternary-operator> → if (<logic-expr>) <stⅿt> else <stⅿt>

در این گرامر:

<ternary-operator>: نشان‌دهنده شروع یک عملگر سه‌گانه است.

<logic-expr>: عبارت منطقی مورد بررسی.

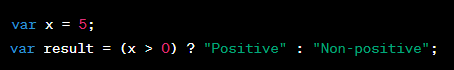
<stⅿt>: دستورات (بلوک کد) که در صورت برقرار بودن یا نبودن عبارت منطقی اجرا می‌شوند.

if: کلمه کلیدی if.

else: کلمه کلیدی else.

این گرامر می‌گوید که یک عملگر سه‌گانه از یک شرط (<logic-expr>) و دو بلوک کد (<stⅿt>) که با if و else مشخص شده‌اند، تشکیل شده است.

برای مثال:



در اینجا:

<ternary-operator> توسط (x > 0) ? "Positive" : "Non-positive" توصیف می‌شود.

<logic-expr> بخش (x > 0) را نشان می‌دهد.

<stⅿt> دو بار تکرار شده و به ترتیب بلوک‌های کد "Positive" و "Non-positive" را نشان می‌دهد.

? نشان‌دهنده عملگر سه‌گانه است که شرط را بررسی می‌کند.

: نشان‌دهنده جداکننده بین دو بلوک کد است.

در این مثال مثبت بودن x بررسی می‌شود و مقدار "Positive" یا "Non-positive" در result ذخیره می‌شود.

عملیات و منطقی

برای تعریف گرامر عملیات و منطقی (logical AND) در زبان جاوااسکریپت، می‌توانید از یک گرامر به صورت زیر استفاده کنید:

<logical-and-expression> → <logic-expr> && <logic-expr>

در این گرامر:

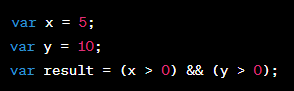
<logical-and-expression>: نشان‌دهنده شروع یک عبارت و منطقی است.

<logic-expr>: عبارت‌های منطقی که با یکدیگر به صورت و منطقی مرتبط هستند.

&&: علامت و منطقی.

این گرامر می‌گوید که یک عبارت و منطقی از دو <logic-expr> که توسط && و منطقی گرفته شده‌اند، تشکیل شده است.

برای مثال:



در اینجا:

<logical-and-expression> توسط (x > 0) && (y > 0) توصیف می‌شود.

<logic-expr> دو بار تکرار شده و به ترتیب عبارت‌های (x > 0) و (y > 0) را نشان می‌دهد.

&& نشان‌دهنده علامت و منطقی است.

در این مثال x و y، and منطقی شده و حاصل در result ذخیره میشود.

عملیات یا منطقی

برای تعریف گرامر عملیات یا منطقی (logical OR) در زبان جاوااسکریپت، می‌توانید از یک گرامر مشابه به صورت زیر استفاده کنید:

<logical-or-expression> → <logic-expr> || <logic-expr>

در این گرامر:

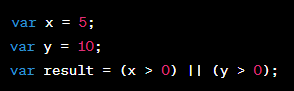
<logical-or-expression>: نشان‌دهنده شروع یک عبارت یا منطقی است.

<logic-expr>: عبارت‌های منطقی که با یکدیگر به صورت یا منطقی مرتبط هستند.

||: علامت یا منطقی.

این گرامر می‌گوید که یک عبارت یا منطقی از دو <logic-expr> که توسط || یا منطقی گرفته شده‌اند، تشکیل شده است.

برای مثال:



در اینجا:

<logical-or-expression> توسط (x > 0) || (y > 0) توصیف می‌شود.

<logic-expr> دو بار تکرار شده و به ترتیب عبارت‌های (x > 0) و (y > 0) را نشان می‌دهد.

|| نشان‌دهنده علامت یا منطقی است.

در این مثال x و y، or منطقی شده و حاصل در result ذخیره میشود.

عملیات OR بیتی

در زبان جاوااسکریپت، عملیات OR بیتی با استفاده از عملگر | انجام می‌شود. برای تعریف گرامر این عملیات، می‌توانید از یک قالب به صورت زیر استفاده کنید:

<bitwise-or-expression> → <operand> | <operand>

در این گرامر:

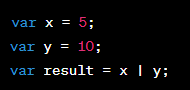
<bitwise-or-expression>: نشان‌دهنده شروع یک عبارت OR بیتی است.

<operand>: عملوندها یا اجزای عبارت OR بیتی.

|: علامت OR بیتی.

این گرامر می‌گوید که یک عبارت OR بیتی از دو <operand> که توسط | OR بیتی گرفته شده‌اند، تشکیل شده است.

برای مثال:



در اینجا:

<bitwise-or-expression> توسط x | y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

| نشان‌دهنده علامت OR بیتی است.

در این مثال x و y، or بیتی شده و حاصل در result ذخیره میشود.

عملیات AND بیتی

در زبان جاوااسکریپت، عملیات AND بیتی با استفاده از عملگر & انجام می‌شود. برای تعریف گرامر این عملیات، می‌توانید از یک قالب مشابه به صورت زیر استفاده کنید:

<bitwise-and-expression> → <operand> & <operand>

در این گرامر:

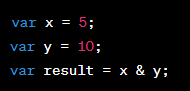
<bitwise-and-expression>: نشان‌دهنده شروع یک عبارت AND بیتی است.

<operand>: عملوندها یا اجزای عبارت AND بیتی.

&: علامت AND بیتی.

این گرامر می‌گوید که یک عبارت AND بیتی از دو <operand> که توسط & AND بیتی گرفته شده‌اند، تشکیل شده است.

برای مثال:



در اینجا:

<bitwise-and-expression> توسط x & y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

& نشان‌دهنده علامت AND بیتی است.

در این مثال x و y، and بیتی شده و حاصل در result ذخیره میشود.

عملیات NOT بیتی

در زبان جاوااسکریپت، عملیات NOT بیتی با استفاده از عملگر ~ انجام می‌شود. برای تعریف گرامر این عملیات، می‌توانید از یک قالب مشابه به صورت زیر استفاده کنید:

<bitwise-not-expression> → ~ <operand>

در این گرامر:

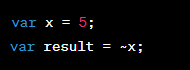
<bitwise-not-expression>: نشان‌دهنده شروع یک عبارت NOT بیتی است.

<operand>: عملوند یا اجزای عبارت NOT بیتی.

~: علامت NOT بیتی.

این گرامر می‌گوید که یک عبارت NOT بیتی از یک <operand> که توسط ~ NOT بیتی گرفته شده‌اند، تشکیل شده است.

برای مثال:



در اینجا:

<bitwise-not-expression> توسط ~x توصیف می‌شود.

<operand> یک بار تکرار شده و مقدار x را نشان می‌دهد.

~ نشان‌دهنده علامت NOT بیتی است.

در این مثال x، not بیتی شده و حاصل در result ذخیره میشود.

عملیات XOR بیتی

در زبان جاوااسکریپت، عملیات XOR بیتی با استفاده از عملگر ^ انجام می‌شود. برای تعریف گرامر این عملیات، می‌توانید از یک قالب به صورت زیر استفاده کنید:

<bitwise-xor-expression> → <operand> ^ <operand>

در این گرامر:

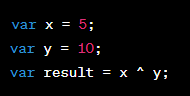
<bitwise-xor-expression>: نشان‌دهنده شروع یک عبارت XOR بیتی است.

<operand>: عملوندها یا اجزای عبارت XOR بیتی.

^ : علامت XOR بیتی.

این گرامر می‌گوید که یک عبارت XOR بیتی از دو <operand> که توسط ^، XOR بیتی گرفته شده‌اند، تشکیل شده است.

برای مثال:



در اینجا:

<bitwise-xor-expression> توسط x ^ y توصیف می‌شود.

<operand> دو بار تکرار شده و به ترتیب مقادیر x و y را نشان می‌دهد.

^ نشان‌دهنده علامت XOR بیتی است.

در این مثال x و y، xor بیتی شده و حاصل در result ذخیره میشود.

عملیات Left Shift بیتی

در زبان جاوااسکریپت، عملیات Left Shift بیتی با استفاده از عملگر << انجام می‌شود. برای تعریف گرامر این عملیات، می‌توانید از یک قالب به صورت زیر استفاده کنید:

<bitwise-left-shift-expression> → <operand> << <operand>

در این گرامر:

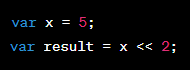
<bitwise-left-shift-expression>: نشان‌دهنده شروع یک عبارت Left Shift بیتی است.

<operand>: عملوندها یا اجزای عبارت Left Shift بیتی.

<<: علامت Left Shift بیتی.

این گرامر می‌گوید که یک عبارت Left Shift بیتی از یک <operand> به تعداد بیت‌ها که توسط << گرفته شده‌اند، تشکیل شده است.

برای مثال:



در اینجا:

<bitwise-left-shift-expression> توسط x << 2 توصیف می‌شود.

<operand> یک بار تکرار شده و مقدار x را نشان می‌دهد.

<< نشان‌دهنده علامت Left Shift بیتی است.

در این مثال، x به چپ شیف داده شده و نتیجه در result ذخیره می‌شود.

عملیات Right Shift بیتی

در زبان جاوااسکریپت، عملیات Right Shift بیتی با استفاده از عملگر >> انجام می‌شود. برای تعریف گرامر این عملیات، می‌توانید از یک قالب به صورت زیر استفاده کنید:

<bitwise-right-shift-expression> → <operand> >> <operand>

در این گرامر:

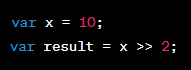
<bitwise-right-shift-expression>: نشان‌دهنده شروع یک عبارت Right Shift بیتی است.

<operand>: عملوندها یا اجزای عبارت Right Shift بیتی.

>>: علامت Right Shift بیتی.

این گرامر می‌گوید که یک عبارت Right Shift بیتی از یک <operand> به تعداد بیت‌ها که توسط >> گرفته شده‌اند، تشکیل شده است.

برای مثال:



در اینجا:

<bitwise-right-shift-expression> توسط x >> 2 توصیف می‌شود.

<operand> یک بار تکرار شده و مقدار x را نشان می‌دهد.

>> نشان‌دهنده علامت Right Shift بیتی است.

در این مثال، x به راست شیف داده شده و نتیجه در result ذخیره می‌شود.

عملیات Unsigned Right Shift بیتی

در زبان جاوااسکریپت، عملیات Unsigned Right Shift بیتی با استفاده از عملگر >>> انجام می‌شود. برای تعریف گرامر این عملیات، می‌توانید از یک قالب به صورت زیر استفاده کنید:

<bitwise-unsigned-right-shift-expression> → <operand> >>> <operand>

در این گرامر:

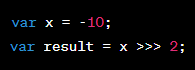
<bitwise-unsigned-right-shift-expression>: نشان‌دهنده شروع یک عبارتUnsigned Right Shift بیتی است.

<operand>: عملوندها یا اجزای عبارت Unsigned Right Shift بیتی.

>>>: علامت Unsigned Right Shift بیتی.

این گرامر می‌گوید که یک عبارت Unsigned Right Shift بیتی از یک <operand> به تعداد بیت‌ها که توسط >>> گرفته شده‌اند، تشکیل شده است.

برای مثال:



در اینجا:

<bitwise-unsigned-right-shift-expression> توسط x >>> 2 توصیف می‌شود.

<operand> یک بار تکرار شده و مقدار x را نشان می‌دهد.

>>> نشان‌دهنده علامت Unsigned Right Shift بیتی است.

در این مثال، x به راست به شکل بی‌علامت شیف داده شده و نتیجه در result ذخیره می‌شود.

حلقه for

برای تعریف گرامر عبارت for در زبان جاوااسکریپت، می‌توانیم از یک ساختار مشابه به صورت زیر استفاده کنیم:

<for-statement> → for (<initialization>; <condition>; <update>) <statement>

<for-statement> → for (<initialization>; <condition>; <update>) <statement> else <statement>

در این گرامر:

<for-statement>: نشان‌دهنده شروع یک عبارت for است.

<initialization>: مرحله شروعی for که معمولاً تعیین متغیرها یا مقداردهی اولیه انجام می‌دهد.

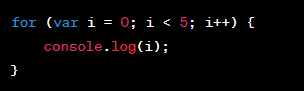
<condition>: شرط تا زمان ادامه حلقه.

<update>: عملیات به‌روزرسانی که در هر ایتریشن اجرا می‌شود.

<statement>: بلوک کد یا عبارتی که در هر ایتریشن اجرا می‌شود.

else: اختیاری و نشان‌دهنده بلوک کدی که اگر شرط for اجرا نشود، اجرا می‌شود.

برای مثال:



در اینجا:

- <for-statement> توسط for (var i = 0; i < 5; i++) { console.log(i); } توصیف می‌شود.

- <initialization> نشان‌دهنده var i = 0 است.

- <condition> نشان‌دهنده i < 5 است.

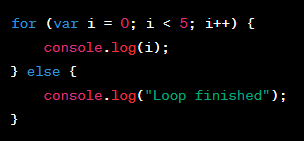
- <update> نشان‌دهنده i++ است.

- <statement> نشان‌دهنده console.log(i); است.

اگر بخواهیم این گرامر را با بلوک else همراه کنیم:

<for-statement> → for (<initialization>; <condition>; <update>) <statement> else <statement>

مثال:



در اینجا، else نشان‌دهنده بلوک کدی است که اگر شرط for اجرا نشود (مثلا اگر i < 5 از ابتدا صحیح نباشد)، آن بلوک اجرا می‌شود. در مثال فوق، این بلوک else به ازای شرط نقض اجرا می‌شود.

حلقه while

برای تعریف گرامر عبارت while loop در زبان جاوااسکریپت، می‌توانیم از یک ساختار به صورت زیر استفاده کنیم:

<while-loop> → while (<condition>) <statement>

<while-loop> → while (<condition>) <statement> else <statement>

در این گرامر:

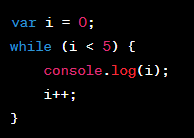
<while-loop>: نشان‌دهنده شروع یک حلقه while است.

<condition>: شرط تا زمانی که حلقه ادامه دارد.

<statement>: بلوک کد یا عبارتی که در هر ایتریشن اجرا می‌شود.

else: اختیاری و نشان‌دهنده بلوک کدی که اجرا می‌شود اگر شرط while اجرا نشود.

برای مثال:



در اینجا:

<while-loop> توسط while (i < 5) { console.log(i); i++; } توصیف می‌شود.

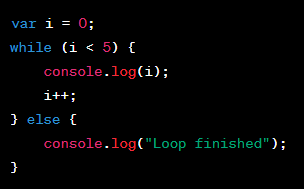
<condition> نشان‌دهنده i < 5 است.

<statement> نشان‌دهنده console.log(i); i++; است.

اگر بخواهیم این گرامر را با بلوک else همراه کنیم:

<while-loop> → while (<condition>) <statement> else <statement>

مثال:



در اینجا، else نشان‌دهنده بلوک کدی است که اجرا می‌شود اگر شرط while اجرا نشود (مثلا اگر i < 5 از ابتدا صحیح نباشد). در مثال فوق، این بلوک else به ازای شرط نقض اجرا می‌شود.

عبارات شرطی if

گرامر زیر یک نمونه از گرامر مرتبط با عبارات شرطی if در زبان جاوااسکریپت است:

<if-stmt> → if (<logical-expr>) <stmt>

<if-stmt> → if (<logical-expr>) <stmt> else <stmt>

در این گرامر:

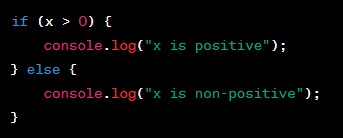
<if-stmt>: نشان‌دهنده شروع یک عبارت شرطی if است.

<logical-expr>: عبارت منطقی که بررسی می‌شود.

<stmt>: بلوک کد یا عبارتی که در صورت برقرار بودن شرط اجرا می‌شود.

else: اختیاری و نشان‌دهنده بلوک کدی که اجرا می‌شود اگر شرط if برقرار نباشد.

برای مثال:



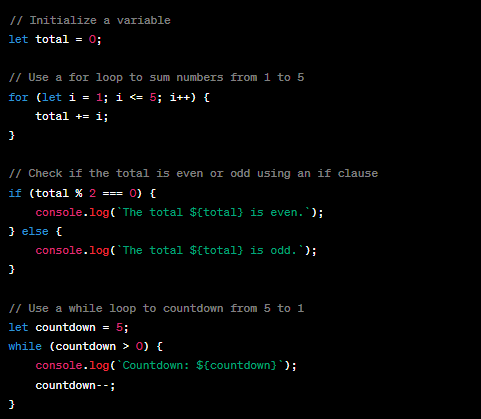
در این مثال:

<if-stmt> توسط if (x > 0) { console.log("x is positive"); } else { console.log("x is non-positive"); } توصیف می‌شود.

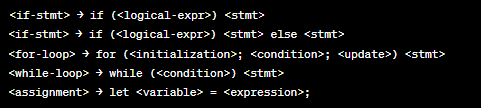
<logical-expr> نشان‌دهنده x > 0 است.

<stmt> نشان‌دهنده console.log("x is positive"); و console.log("x is non-positive"); است.

حال برای جمع بندی، یک گرامر برای کد زیر می‌نویسیم :



برای تبدیل کد JavaScript بالا به گرامر با توجه به قالب ارائه شده، می‌توانیم از گرامر زیر استفاده کنیم:



حالا کد JavaScript بالا را به گرامر تبدیل می‌کنیم:

