**عملگرها**

ما عملگرهای مختلفی را از زمان مدرسه به خاطر داریم. مانند جمع + ، تفریق - ، ضرب \* ، تقسیم و دیگر عملگرها.

در این بخش ما بر روی زوایایی از عملگرها متمرکز می‎شویم که در مدرسه نیاموخته‌ایم.

**اصلاحات : unary (یونِری) – binary (باینری) – operand (اُپِرِند)**

پیش از ادامه بیایید مفهوم این اصطلاحات را بفهمیم.

یک operand – همان چیزی است که عملگرهای برای آنها اعمال می‌شوند. برای نمونه در ضرب 5 \* 2 دو operand داریم. برخی آن را آرگمان (argument) نیز می‌خوانند.

یک عملگر زمانی unary است که فقط یک operand داشته باشد. برای نمونه unary منفی کننده – که علامت یک عدد را برعکس می‌کند.

کد

یک عملگر زمانی binary است که دو operand داشته باشد. همان عملگر منفی کننده در شکل binary هم وجود دارد.

کد

در اصل در اینجا ما در مورد دو عملگر مجزا صحبت کردیم. اولی عملگر unary منفی کننده (که علامت عدد را برعکس می‌کرد) و دیگری عملگر تفریق binary (تفریق دو عدد از هم).

**تلفیق رشته‌ها، عملگر + binary**

معمولا از عملگر + برای جمع اعداد استفاده می‌شود، اما زمانیکه این عملگر به رشته‌‌‌ها اعمال شود، آنها را ادغام می‌کند.

کد

در نظر داشته باشید که اگر یکی از operand ها string باشد، دیگری نیز به string تبدیل می‌شود.

برای نمونه :

کد

همانطور که می‌بینید مهم نیست که operand ابتدا آمده باشد یا نه، قاعده ساده است: اگر هر کدام از operand ها string بود، دیگری نیز به string تبدیل می‌شود.

با این حال در نظر داشته باشید که عملیات از چپ به راست انجام می‌شود. اگر دو عدد در پیش از یک string آمده باشند، ابتدا آن دو عدد جمع می‌شوند و سپس به string تبدیل می‌شوند.

کد

تبدیل و الصاق رشته‌ها یکی از ویژگی‌های خاص عملگر + است. دیگر عملگرهای ریاضیاتی فقط با اعداد کار می‌کنند و همیشه operand خود را به number تبدیل می‌کنند.

برای نمونه :

کد

**تبدیل به عدد ، عملگر unary +**

علامت جمع + به دو شکل وجود دارد: به صورت عملگر binary که بالاتر از آن استفاده کردیم و به صورت عملگر unary .

عملگر + که به یک مقدار اعمال می‌شود، هیچ عملی رو اعداد انجام نمی‌دهد. اما اگر operand یک عدد نباشد، عملگر + آن را به عدد تبدیل می‌کند.

به عنوان مثال :

کد

در واقع این همان کاریست که Number(…) انجام می‌دهد ولی به شکلی کوتاه‌تر.

نیاز به تبدیل رشته‌ به عدد اغلب پیش می‌آید. برای نمونه اگر در حال دریافت مقادیری از اینپوت‌های HTML هستیم معمولا به این موضوع نیاز پیدا می‌کنیم.

اگر خواستیم آنها را جمع ببندیم چطور ؟

عملگر + به صورت رشته‌ای آنها را بهم اضافه می‌کند :

کد

اگر خواستیم با آنها مانند عدد برخورد کنیم، آنها را به عدد تبدیل کرده و سپس آنها را جمع می‌کنیم:

کد

از دیدگاه یک ریاضی‌دان، تعدد علامت + ممکن است عجیب به نظر برسد، اما از دیدگاه یک برنامه‌نویس اینطور نیست. unary plusهایی که در ابتدا آمده‌اند رشته‌ها را عدد تبدیل می‌کنند و سپس binary plus اعداد را با هم جمع می‌کند.

چرا unary plus ها پیش از binary plus ها اعمال شدند ؟ به این دلیل که اولویت آنها بالاتر است.

**اولویت عملگرها**

اگر در یک عبارت بیش از یک عملگر وجود داشته باشد، ترتیب اجرای آنها بر اساس تقدم آنها خواهد بود.

از زمان مدرسه همه ما می‌دانیم که در یک عمل ریاضی مانند 1 + 2 \* 2 ابتدا عمل ضرب انجام می‌شود و سپس عمل جمع. این همان اولویت عملگرها است. اینکه عمل ضرب اولویت بالاتری نسبت به جمع دارد.

پرانتزها بر هر اولویتی، اولویت دارند و زمانی‌که از ترتیب قرارگیری عملگرها راضی نیستیم می‌توانیم با پرانتزها این اولویت را تغییر دهیم.

عملگرهای مختلفی در جاوا اسکریپت وجود دارد و هر کدام اولویت مربوط به خود را دارا می‌باشند. در جدول زیر عملگری که عدد بالاتری دارد اولویتش بالاتر است. همینطور اگر دو عملگر عدد یکسانی داشتند اولویت اجرا از چپ به راست (در کد) می‌باشد.

کد

همانطور که می‌بینیم unary + اولویت 16 دارد که از binary + با اولویت 13 بالاتر است. به همین دلیل است که در عبارت “+apples + +oranges” علامت‌های unary plus پیش از علامت جمع اجرا می‌شوند.