O operador >> move todos os bits em seu primeiro operando para a direita por o número de lugares especificados no segundo operando (um número inteiro entre 0 e 31).Bits que são deslocados para a direita são perdidos.O Bits preenchidos à esquerda dependem do bit de sinal do original operando, a fim de preservar o sinal do resultado.Se o primeiro Operando é positivo, o resultado tem zeros colocados nos bits altos;se O primeiro operando é negativo, o resultado possui aqueles colocados na alta bits.Mudar um valor positivo para o lado, um lugar é equivalente a Dividindo por 2 (descartando o restante), mudando para a direita dois lugares é equivalente à divisão inteira até 4, e assim por diante.7 >> 1 Avalia 3, por exemplo, mas observe que e -7 >> 1 avalia como -4.

Mudar à direita com preenchimento zero (>>>)

O operador >>> é como o operador >>, exceto que os bits mudados para a esquerda são sempre zero, independentemente do sinal do primeiro operando.Isso é útil quando você deseja tratar 32 bits assinados valores como se fossem números inteiros não assinados.?1 >> 4 avalia para -1, Mas ?1 >>> 4 avalia para 0x0fffffff, por exemplo.Isso é o único dos operadores JavaScript bit -new que não pode ser usado com valores bigint.Bigint não representa números negativos por Definindo a parte alta da maneira que os números inteiros de 32 bits, e este operador faz sentido apenas para o complemento desses dois em particular representação.

4.9 Expressões relacionais

Esta seção descreve os operadores relacionais da JavaScript.Esses Os operadores testam um relacionamento (como "iguais", "menos que" ou ?Propriedade de?) entre dois valores e retorno verdadeiro ou falso Dependendo se esse relacionamento existe.Expressões relacionais sempre avalie com um valor booleano, e esse valor é frequentemente usado para controlar o fluxo de execução do programa em se, enquanto e para