**Condições**

As condições são usadas para controlar a sequência de um programa ou função. Todos funcionam de acordo com o mesmo padrão: uma condição é definida e, dependendo de como isso pode ser avaliado (verdadeiro ou falso), algo é realizado ou não.

**If**

If é a forma mais simples de uma condição. Consiste em duas partes: uma condição e uma instrução ou bloco de instruções. A condição é escrita após a palavra if entre dois parênteses. A instrução ou bloco de instrução segue a condição. Se a afirmação for verdadeira ( true), a afirmação é executada. E se for false, não é executada. Literalmente, tal consulta seria: If [condição] então faça [instrução] . Exemplo:

a = 2;

if (a == 1)

alert (‘a é 1’);

if (a <10 && a> 1) {

         a ;

         alert (‘a é’ + a);

}

**If-else**

Uma condição if-else pode ser vista como uma condição dupla, na qual a segunda condição pode ser validada também. Isso é realizado se a condição if for avaliada com false( ou seja, falso).

A parte alternativa é separada da parte do if pela palavra else. Uma representação literal de tal if-else poderia ser a seguinte: If [condição] então faça [instrução if], caso contrário, faça [instrução else]. Exemplo:

a = 1;

if (a == 2) a ++ else a–;

if (a == 1) {

         a ;

} else {

         a–;

};

**Condição inline**

A condição if-else inline é uma forma simplificada de condição. Consiste em apenas três partes: a condição e duas instruções ou blocos de instruções. Deve haver um ponto de interrogação ( ?) após a condição e dois pontos ( :) entre as duas instruções ou blocos . A função é como uma if-else: se a condição for atendida, a primeira instrução é executada - se não for atendida, a segunda (ou seja, a instrução após os dois pontos) é executada. Exemplo:

a = 1;

a == 2? a: a–;

**Switch**

Em vez de várias condições if ou if-else, pode ser usada a condição switch . Ela contém várias opções e, portanto, pode ser acionada de forma mais conveniente que o if-else.

O switch começa com a palavra switch seguida por um objeto, uma variável ou uma condição entre parênteses. Em seguida, colocamos a cláusula case. Cada case representa uma forma ou valor do objeto que pode assumir. Uma case sempre começa com a palavra case- seguida pelo valor. Um valor pode ser quase qualquer coisa - texto, número, objeto ou tipo. O case termina com dois pontos(:). O case é seguido pelas instruções que devem ser realizadas quando a expressão switch é completamente idêntica com a expressão da case. Uma sequência de instruções termina com a palavra break. Além disso, instruções alternativas podem ser definidas. Eles devem vir após o último break e começar com a palavra default seguida por dois pontos(:). Isso é seguido pelas instruções e não precisa ser um  break para encerrar.

O switch funciona da seguinte maneira: O switch é executado de cima para baixo e é pesquisado entre os cases um valor semelhante. Se um valor for encontrado, as tarefas subsequentes são realizadas até o próximo break e a consulta encerrada. Se nenhum valor semelhante for encontrado, as instruções alternativas(default) - se disponíveis - são executadas. O switch é encerrado e o programa continua. Exemplo:

a = 1;

switch (a)

{

 case 1 :

    alert(‘Cheguei case 1’);

    break;

 case 2: case 3: case 4: case 5 :

    alert(‘Cheguei case 2,3,4,5’);

    break;

 case 6 :

        alert(‘Cheguei case 6’);

        break;

 default:

        alert(‘Cheguei no valor padrão’);

}

**Loop**

Os loops são usados para repetir certas instruções sem ter que informá-las com tanta frequência. Existem vários tipos de loops, bem como alguns acréscimos que só podem ser usados dentro desses loops.

**for**

Um loop for geralmente consiste em um cabeçalho e um corpo. O início do for é introduzido pela palavra for. Isso é seguido pelas três partes do cabeçalho entre parênteses: um valor inicial, uma condição e um contador. Todos os três são separados um do outro por um ponto e vírgula ( ;). O valor inicial é definido por uma variável à qual um valor (inicial) é atribuído. Normalmente, a variável i(i = inteiro = número inteiro) é usada. Uma expressão possível para isso seria, por exemplo i=23;, o valor inicial do for com 23. A condição é feita de forma normal, pois já aprendemos quando tratamos de condições. Se possível, esta condição deve usar a variável do valor inicial, mas também pode ser definida de forma diferente. Uma condição possível seria, por exemplo i<50. A terceira parte é o contador. O contador consiste em uma instrução que deve ser executada após o final do corpo do laço. Normalmente, as variáveis usadas anteriormente também são usadas aqui e aumentadas, por exemplo. Uma expressão de exemplo para isso seria i.

Como funciona o loop for? Primeiro, a variável do valor inicial é criada e o valor especificado é passado para ela. Em seguida, é verificado se a condição foi atendida. Se o resultado for  true, o corpo do loop é executado completamente. Em seguida, a instrução do contador é executada. Até que a condição não seja mais satisfeita, o corpo do loop e a instrução do contador são executados. O loop pode ser descrito com a seguinte frase: Defina [valor inicial] e execute [corpo do loop] e, em seguida, [contador], desde que a [condição] seja satisfeita.

Exemplo:

a = 0;

for(i=1 ; i<10 ; i++) { a += i; }

b = true; c = 1;

for(a=false ; b!=false ; c+=2)

{

         if(c < 10 && a == true){

                  b = false

         }

         if(a == false && c < 10){

                  a = true

         }

}

**for-in**

Um loop for-in é semelhante a uma estrutura loop for, mas tem diferenças. Também consiste em um cabeçalho e um corpo, mas o cabeçalho é construído de forma diferente de um for.

Ele também começa novamente com a palavra for. Isso é seguido por uma variável, depois a palavra in e em seguida um objeto. Tudo isso entre parênteses .

A maneira como um for-in funciona é a seguinte: As propriedades do objeto são usadas como o número de iterações. Veja um exemplo para ver como funciona:

var fruta = {sabor: ‘azeda’, cor: ‘vermelha’, nome:’acerola’}

for (let propriedade in frutas){

    //exibe as propriedades do objeto fruta

    console.log(propriedade + ": " + fruta[propriedade]);

}

**do-while**

O do-while\_consiste em duas partes: uma instrução ou um bloco de instruções e uma condição. A instrução ou o bloco de instruções são introduzidos pela palavra do e estão localizados na frente da condição. A condição é introduzida pela palavra while dentro de dois parênteses após o bloco de instrução.

A forma como funciona é a seguinte: a instrução ou bloco de instruções é sempre executado primeiro. Em seguida, a condição é testada. Se for true, a sequência começa do início até que a condição seja false. A forma como funciona, pode ser entendida na frase:Faça [instrução / s] e se [condição verdadeira] volte ao início, caso contrário, termine o loop. Exemplo:

do{

 a++;

 b = b + (a\*10);

}

while(a < 10)

**while**

O while é semelhante ao do-while. Ele começa com a palavra while seguida pela condição entre parênteses. Então, apenas a declaração ou o bloco de declarações segue. O modo como funciona é exatamente o oposto do do-while: a primeira coisa a fazer é testar a condição. Se isso for true, a instrução ou o bloco de instruções é executado. O processo é então repetido até que a condição seja false. Simplificando, você poderia descrever este loop da seguinte maneira: Contanto que [condição] faça [instrução], caso contrário, saia do loop. Exemplo:

a = 1; b = 1;

while (a == 1) {

         if (b <10) {

                  a = 2

         } else {

                  b

         }

 }

**Rótulos**

O while é semelhante ao do-while. Ele começa com a palavra while seguida pela condição

Rótulos, também conhecidos como marcas, são locais autodefinidos dentro de um loop, eles são usados em conexão com break e continue para controlar a sequência de um loop. Um rótulo é definido por um nome auto-selecionado, ainda não atribuído, e dois pontos subsequentes ( :). Os rótulos de break devem ser controlados a partir de uma instrução e podem conter qualquer tipo de instrução após os dois pontos (função, loop, consulta, atribuição …). Os rótulos de continue devem ser controlados a partir de uma instrução e só podem conter loops como instruções após os dois pontos. Exemplo:

while (a == b)

{

 meurotulo:

 / \* … instruções … \* /

}

**break**

O break sai do loop atual sem executá-lo até o fim. Além disso, o nome de um rótulo pode ser anotado após a instrução break. Este rótulo designa então o loop que deve ser encerrado. Apenas rótulos que pertencem ao loop atual podem ser usados. Exemplo:

a = 0;

rotulo1: while(a < 10)

{

 b = 0;

 while(b < 10)

 {

  if(b>5){break rotulo1;}

  b;

 }

 a++;

}

alert(a);

**continue**

O continue funciona de forma semelhante ao break. A diferença é que ele apenas termina a iteração atual do loop, não o loop inteiro. Todas as instruções que seguem após esta instrução não serão executadas. Além disso, o nome de um rótulo também pode ser informado aqui, após a instrução.

Se este rótulo levar a um while ou  do-while, ele verifica sua condição e executa o loop novamente (se a condição for true).

Se o rótulo levar a um for, seu contador será incrementado e o loop continuará (se a condição for true).Exemplo:

c = ‘’;

rotulo1:

for(a=0 ; a<=9 ; a++)

{

 for(b=0 ; b<=9 ; b++)

 {

  if(a == 3){continue rotulo1;}

  c += a+’‘+b+’  |  ‘;

 }

 c += ‘<br>’

}

document.writeln(’<span>‘+c+’</span>');