JavaScript

Comparador Distinto de !=

El comparador distinto de, se comporta de forma inversa que el comparador ==.

1. Retorna verdadero cuando ambos valores son diferentes.
2. Retorna falso cuando ambos valores son iguales.

```
<script>
    var a = 1, b = 2;

    if (a != b) {
        console.log("Son distintos");
    } else {
        console.log("Son iguales");
    }

    if (a + 1 != b) {
        console.log("Son distintos");
    } else {
        console.log("Son iguales");
    }
</script>
```

Otro Uso

También se puede ocupar el símbolo “**!**” para invertir el valor de verdad de una comparación. Observar que todo se encuentra encerrado entre paréntesis.

```
<script>
    if (!(3 > 1)) {
        console.log("Verdadero");
    } else {
        console.log("Falso");
    }

    if (!(10 > 5 && 0 >= -1)) {
        console.log("Verdadero");
    } else {
        console.log("Falso");
    }
</script>
```

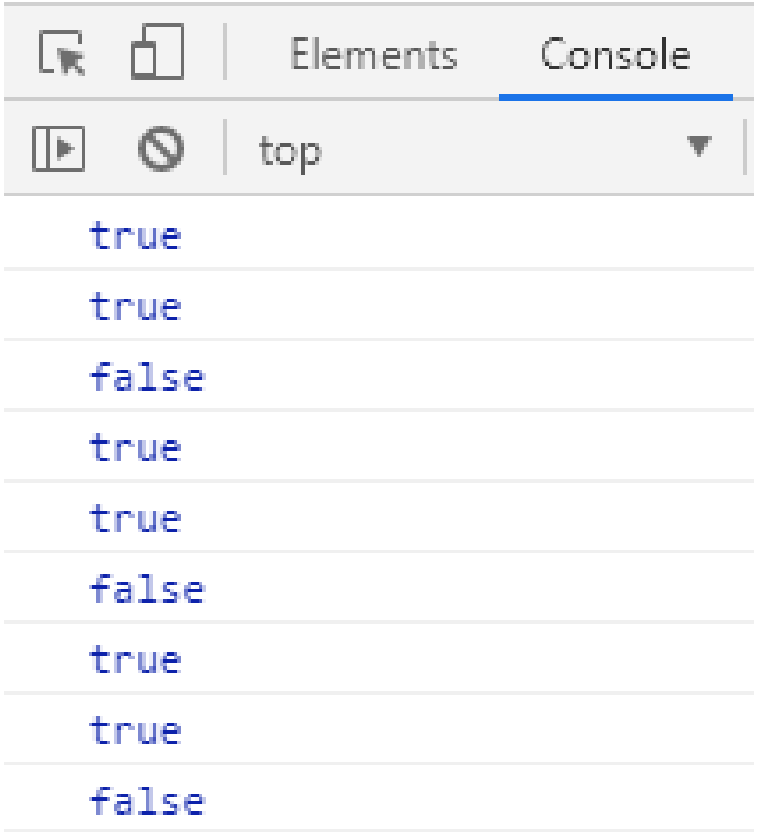
Ejercicios

Al realizar una comparación esta entrega un valor de verdad (true/false). Dependiendo de la respuesta es lo que se imprimirá (true/false).

Desarrolle los ejercicios en su cuaderno y posteriormente verifique el resultado en el Navegador.

```
<script>
    console.log(5 != 3 && 7 != 10);

    console.log(!(5 == 3 && 7 == 10));

    console.log(!(5 > 3 && 5 < 20 && 0 != 1));

    console.log(!!(10 >= 0 && 0 <= 10));

    console.log(5 > 0 && !(4 < 3));

    console.log((!(0 != 0 && 1 != 1) && 5 > 5) || !(2 == 2));

    console.log(10 != 0 || !(4 < 3));

    console.log(!(!(0 != 0) && !(10 >= 3)));

    console.log(!((-1 <= 0 || !(-1 > 2)) && 5 == 5));
</script>
```

Soluciones


Elements
Console
top
true
true
false
true
true
false
true
true
false

Valor Booleano

Un valor boolean está representado por dos estados posible Verdadero o Falso , 0 o 1, Encendido o Apagado.

En JavaScript puede ser almacenado con las palabras true o false.

```
<script>
    var estadoOn = true;
    var estadoFalse = false;

    if (estadoOn == true) {
        console.log("Es verdad!");
    } else {
        console.log("Es falso!");
    }

    if (estadoFalse == true) {
        console.log("Es verdad!");
    } else {
        console.log("Es falso!");
    }
</script>
```

Redundancia

Un boolean ya contiene un valor de verdad requerido por los if.

Por lo que no es necesario compararlo con algo.

```
<script>
    var estadoOn = true;
    var estadoFalse = false;

    if (estadoOn) {
        console.log("Es verdad!");
    } else {
        console.log("Es falso!");
    }

    if (estadoFalse) {
        console.log("Es verdad!");
    } else {
        console.log("Es falso!");
    }
</script>
```

Boolean

Como boolean almacena un valor de verdad, es posible crear una sentencia que genere un valor booleano y posteriormente indicarlo en un if/else.

```
<script>
    var estado_1 = 5 >= 10 && !(-1 != -1);
    var estado_2 = 1 == 1;

    if (estado_1 && estado_2) {
        console.log("Bloque 1");
    } else {
        console.log("Bloque 2");
    }
</script>
```

Observación

Como un boolean almacena un valor de verdadero o falso, es también posible negarlo.

Solamente es necesario agregar un signo “!” antes de usar la variable.

```
<script>
    var estado = true;

    if (!estado) {
        console.log("Bloque 1");
    } else {
        console.log("Bloque 2");
    }
</script>
```

Observación: Al decir !estado está invirtiendo el valor de verdad que está entregando, no está cambiando su valor interno.

Ejercicio

Indique el resultado del siguiente código

```
<script>
    var a = 1, b = 1;
    var estado;

    estado = (a == b && true) || (b != a);

    if (estado) {
        a++;
    }
    else {
        b--;
    }

    console.log(a + " " + b + " " + estado);
</script>
```

Respuesta

2 1 true

Ejercicio

Indique el resultado del siguiente código

```
<script>
    var estado_1 = true, estado_2 = true;
    var a = 1, b = 2, c = 3;

    estado_1 = a >= 2 && b < c;
    estado_1 = !estado_1;

    estado_2 = (true && false) || !true;

    console.log(estado_1 + " " + estado_2);
</script>
```

Respuesta

true false

Ejercicio

Indique el resultado del siguiente código

```
<script>
    var a = 1, b = 1, c = 1;
    var estado;

    if (a >= b) {
        estado = false;
        c++;
    }
    else {
        estado = true;
        a++;
    }

    if (estado) {
        estado = !estado;
    } else {
        b = c;
        estado = a == 2;
    }

    console.log(a + " " + b + " " + c + " " + estado);
</script>
```

Solución

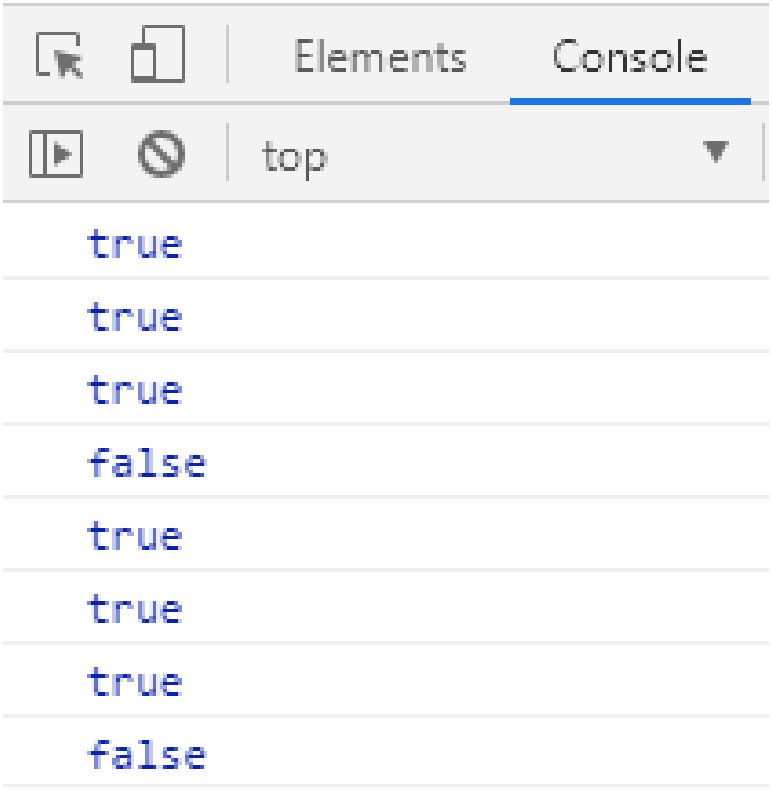
1 2 2 false

Ejercicio

Indique qué imprime

```
<script>
    console.log(1 == 1);
    console.log('123' == 123);
    console.log("898" == '898');
    console.log("898" == '898          ');
    console.log("898" == 898);
    console.log(0 == false);
    console.log(1 == true);
    console.log(2 == true);
</script>
```

Solución


Elements
Console
top
true
true
true
false
true
true
true
false

Comparación - JavaScript

En más de una ocasión JavaScript dará dolores de cabeza intentado comprender su comportamiento, esto se debe porque es bastante permisivo con los tipos de comparaciones versus otros lenguajes de programación.

JavaScript para ejecutar una comparación revisará si los datos son del mismo tipo, si no lo son, intentará transformar el valor para compatibilizar (En el caso que sea necesario), en el caso que no lo consiga el resultado será false por defecto.

Comparación Estricta

Este tipo de comparación evalúa 2 apartados.

1. Tipo de Dato
2. Valor de Dato

Si cumple ambas características será verdadero, si solamente cumple una, será falso.

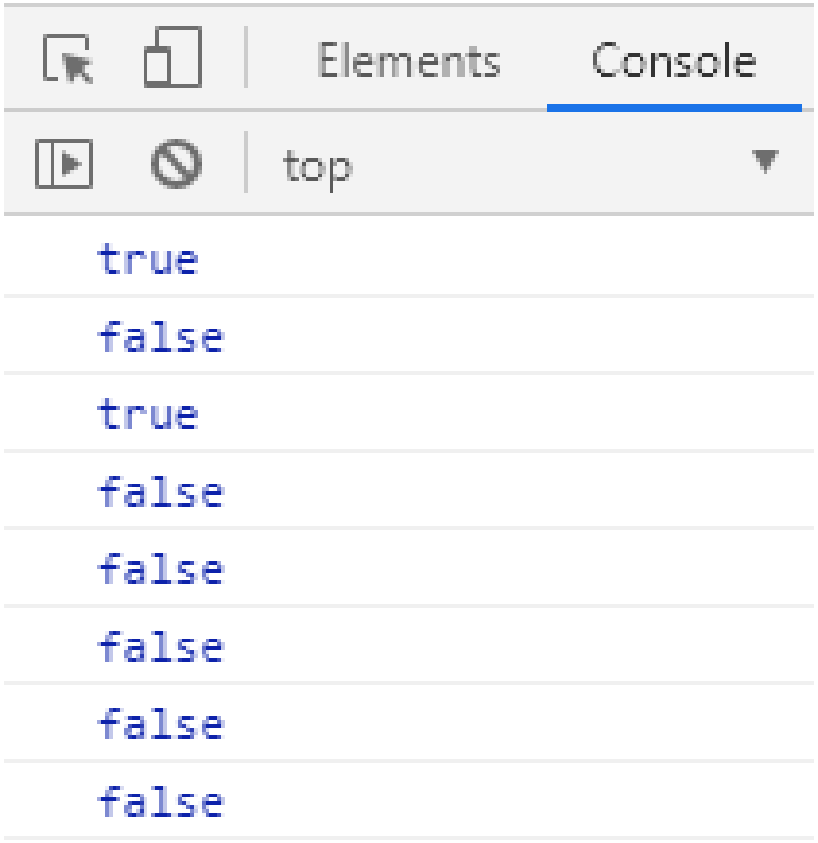
Para indicar que la comparación sea estricta se debe usar 3 es igual **===**.

Ejercicio

Indique qué imprime

```
<script>
    console.log(1 === 1);
    console.log('123' === 123);
    console.log("898" === '898');
    console.log("898" === '898        ');
    console.log("898" === 898);
    console.log(0 === false);
    console.log(1 === true);
    console.log(2 === true);
</script>
```

Solución


Elements
Console
top
true
false
true
false
false
false
false
false

Distinto de - Estricto

También existe la versión estricta para “!=“ la cual es

!==

Observación: Es igual y Distinto de, son las únicas versiones que cuentan con una tipo de comparación estricto.

Comparación

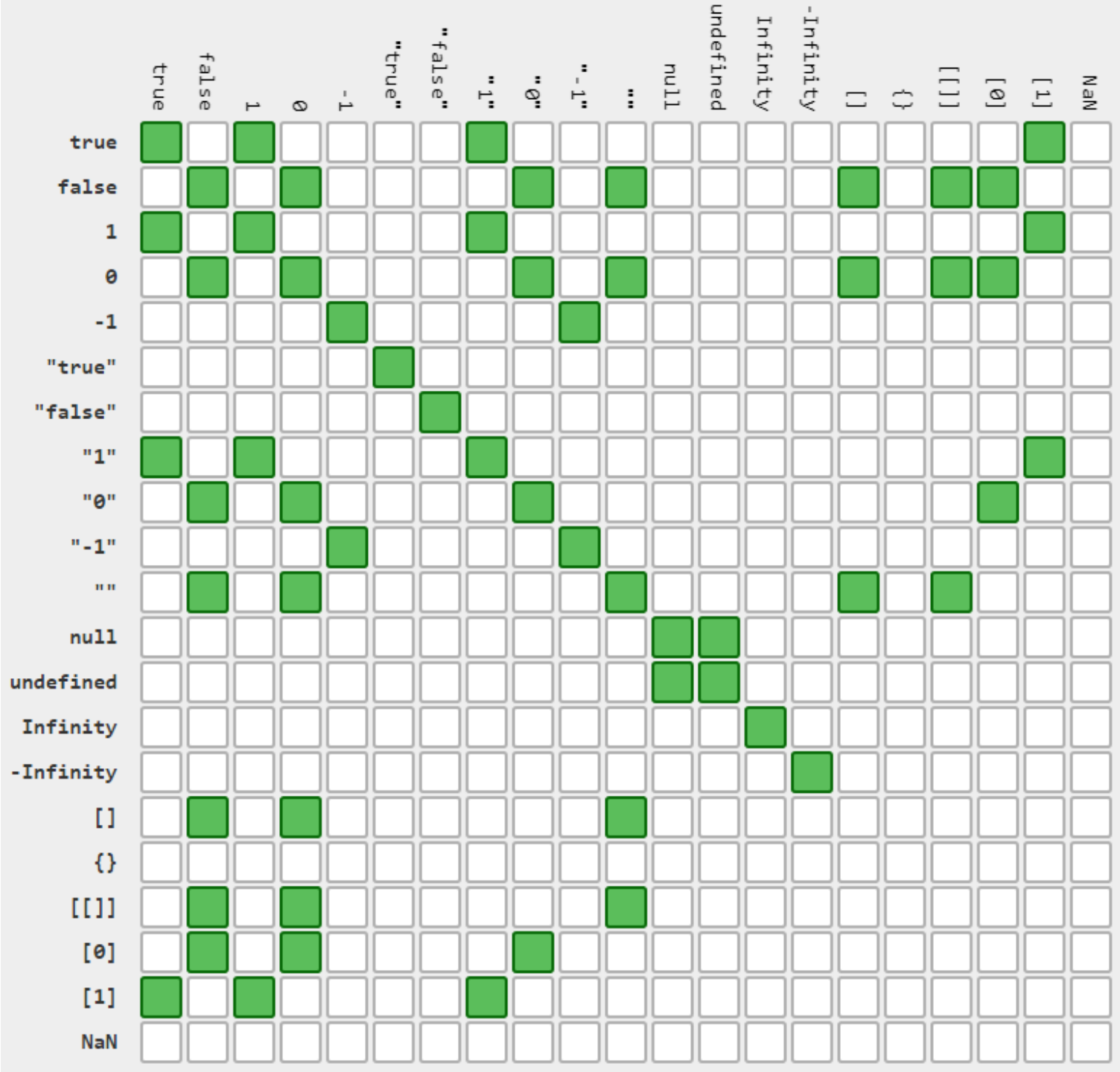


Imagen obtenida de: https://dorey.github.io/JavaScript-Equality-Table/

Comparación estricta

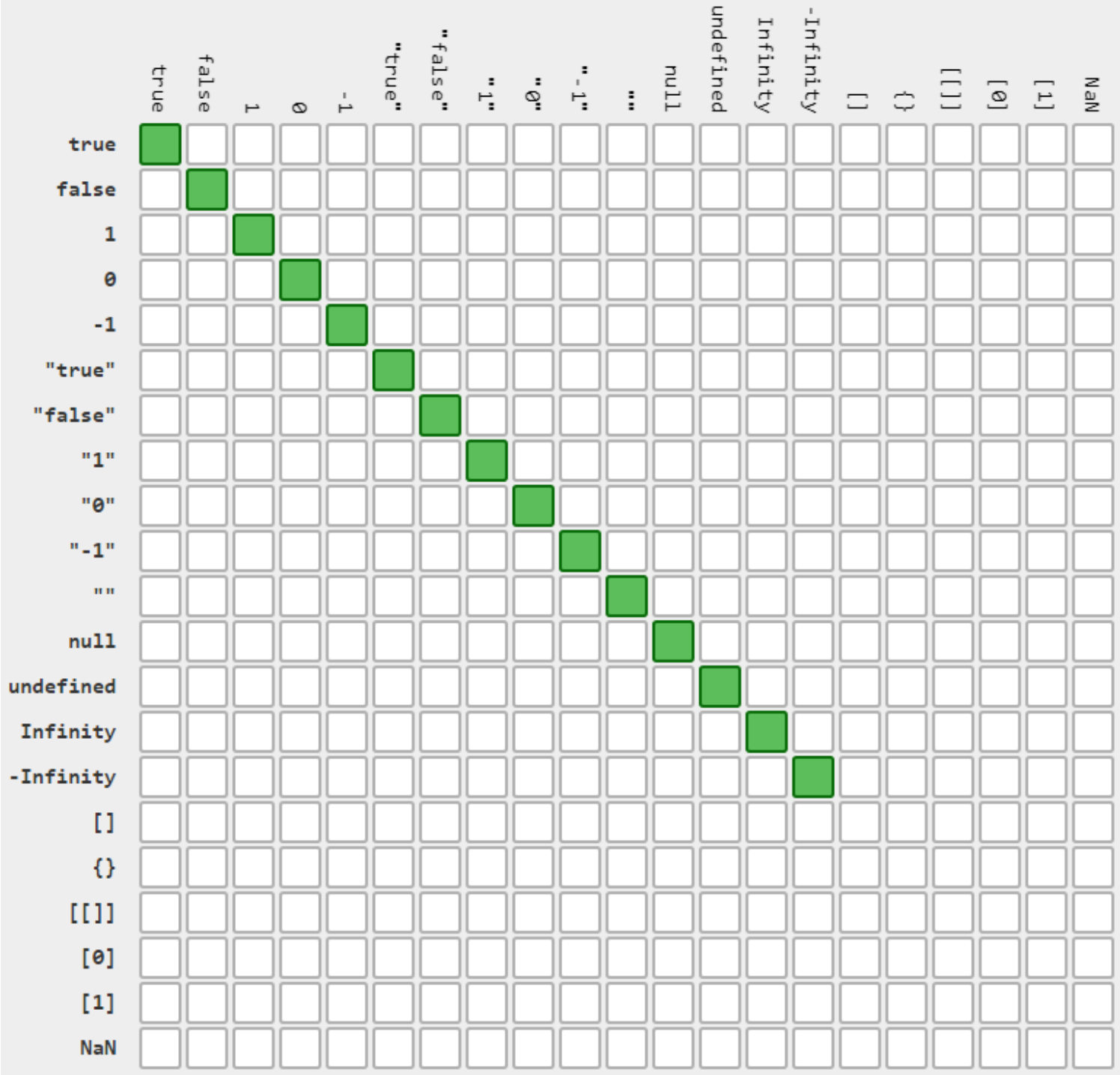


Imagen obtenida de: https://dorey.github.io/JavaScript-Equality-Table/

Moraleja

Siempre utilice la versión estricta para “igual a” o “distinto de”, a menos que tenga una muy buena razón para no hacerlo.

Valor Constante

Una de las últimas características de JavaScript es permitir crear constantes.

Como su nombre lo indica, solamente permitirá la primera asignación de valor en la variable, no permitiendo su modificación posteriormente. (Esto aplica para toda la ejecución)

Para crear una constante se ocupa la palabra clave **const**

```
<script>
    const miValor = "Hola a todos!";
</script>
```

Observación

const hace exactamente lo mismo que var (Crear una variable), con la diferencia que le agrega una restricción de no modificación.

Si se declara una variable const, se le debe asignar su valor por obligación en la misma línea.

Ejecutar los siguientes códigos por separado para observar su comportamiento.

```
<script>
    const miValor;
</script>
```

```
<script>
    const miValor ="Hola";
    miValor = miValor + " Mundo!";
    console.log(miValor);
</script>
```

Ejemplos para usar const

1. Cuando creas una librería Matemática, esta trabaja con valores que son siempre los mismos y que jamás deben ser modificados, como Pi.

2. Cuando creas un videojuego, por temas de rendimiento querrás limitar la cantidad de fotogramas y no desearás cambiar su valor accidentalmente en el desarrollo, o que otro programador lo modifique.

3. Cuando hay variables que se deben compartir a otros apartados del código y se necesita garantizar su no modificación. (Pueden haber procesos que dependen de la integridad de esos valores)

Ejercicio

Diseñe un programa que declare una variable numero e inicie en 0.

Incremente su valor en una unidad (Repita el proceso de incremento 10 veces)

Solución

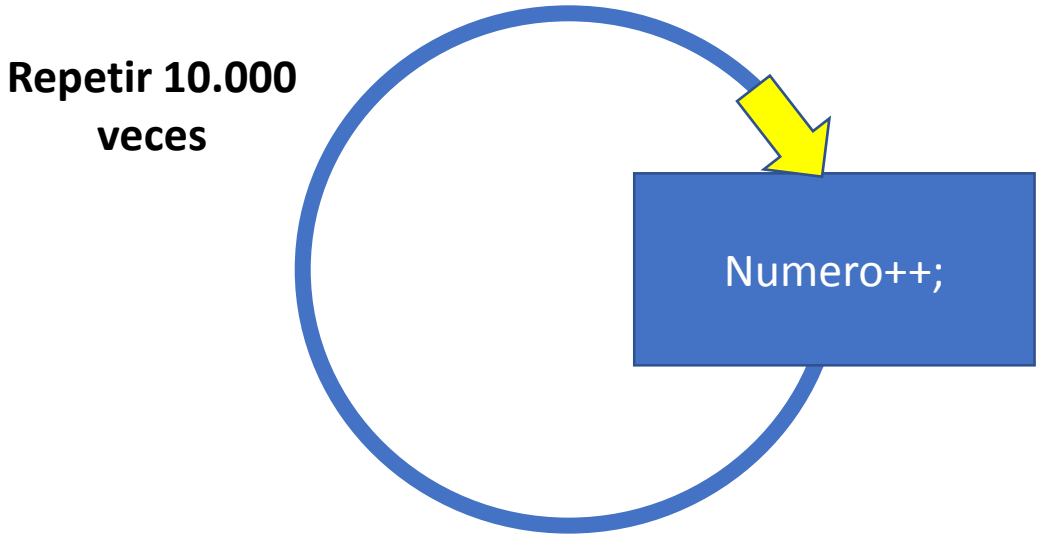
¿Y si después se necesitara repetir 10.000 veces más el incremento?

```
<script>
    var numero = 0;

    numero++; //1
    numero++; //2
    numero++; //3
    numero++; //4
    numero++; //5
    numero++; //6
    numero++; //7
    numero++; //8
    numero++; //9
    numero++; //10
</script>
```

Ciclos/Bucles

Gracias a un ciclo, se puede indicar cuantas veces se debe repetir un conjunto de instrucciones.

El ciclo terminará cuando su condición sea falsa.



Ciclo – while (Mientras)

Mientras su condición sea verdadera continuará indefinidamente ejecutándose.

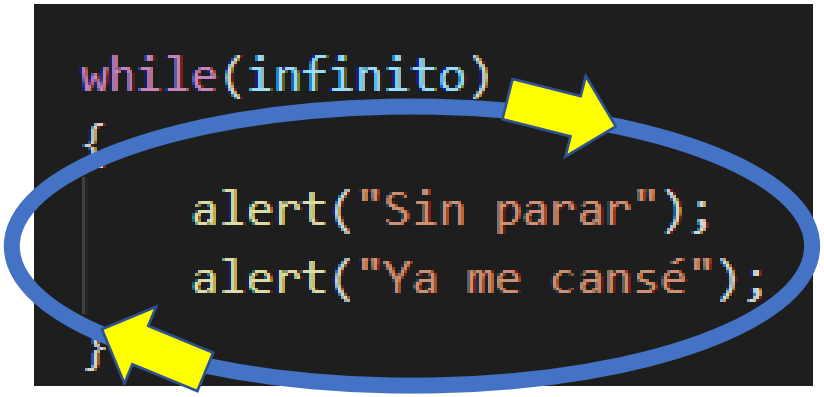
Observe el comportamiento del siguiente código.

```
<script>
    var infinito = true;

    while(infinito)
    {
        alert("Sin parar");
        alert("Ya me cansé");
    }
</script>
```

Recordar

1. Cuando se ejecuta la sentencia while, se le pregunta su condición si es verdadera, en el caso que sea verdadera se ejecuta todo el bloque que se encuentra entre llaves, en caso contrario, el ciclo no se ejecuta.

2. Por lo tanto, si la condición es verdadera se ejecuta todo el bloque hasta llegar a la última línea y se vuelve a subir hasta la sentencia while para repetir **punto 1.**



Ciclo - Determinado

Regularmente se le indica al ciclo cuantas repeticiones deberá hacer, para lograr esta funcionalidad se requerirá de un contador.

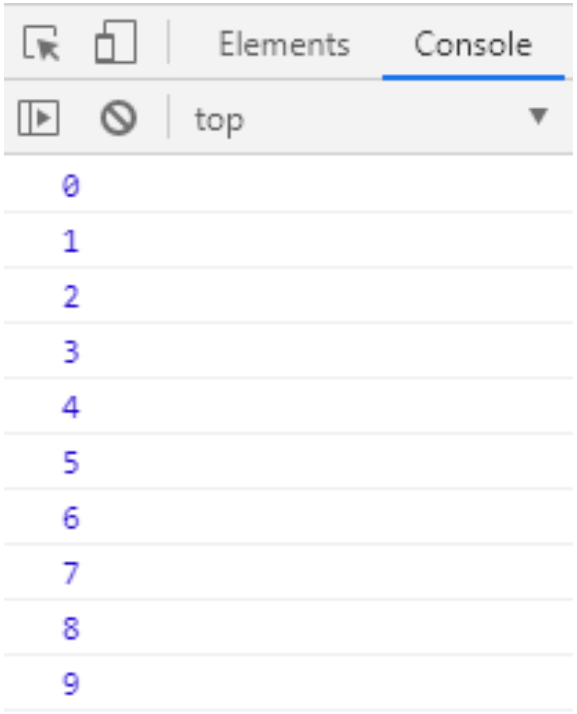
Escribir el siguiente código.

¿Con qué valor quedará i al finalizar el ciclo?

```
<script>
    var i = 0;

    while(i < 3)
    {
        console.log("Repetir! " + i);
        i++;
    }
</script>
```

Ejercicio

Escribir un código que muestre por consola un listado del 0 al 9.



Solución

```
<script>
    var i = 0;

    while(i < 10)
    {
        console.log(i);
        i++;
    }
</script>
```

Ejercicio

Escribir un código que muestre por consola un listado del 1 al 10.

Solución

```
<script>
    var i = 1;

    while(i <= 10)
    {
        console.log(i);
        i++;
    }
</script>
```

Ejercicio

Escribir un programa que permita al usuario indicar hasta qué número desea listar de forma automática.

La lista inicia desde cero hasta el número que se indicó.

Ejemplo:

Ingrese un número: 5

Lista generada:

0

1

2

3

4

5

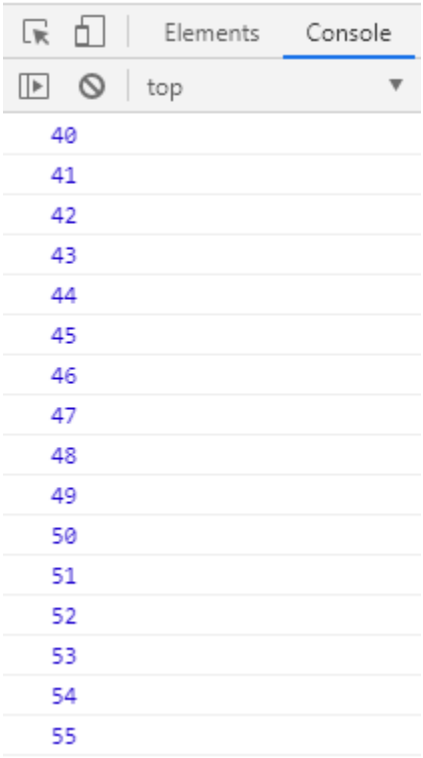
Ejercicio

Escribir un programa que imprima la suma de todos los números que van desde 0 hasta N.

Pida N por prompt.

Ejercicio

Escribir un código que muestre por consola un listado del 40 al 55.



Solución

```
<script>
    var i = 40;

    while(i <= 55)
    {
        console.log(i);
        i++;
    }
</script>
```