Practica 02-01 Bucles JS

Nombre		Curso	
Apellidos		Fecha	

Los bucles ofrecen una forma rápida y sencilla de hacer algo repetidamente. Este capítulo de la Guía de JavaScript presenta las diferentes declaraciones de iteración disponibles para JavaScript.

Puedes pensar en un bucle como una versión computarizada del juego en la que le dices a alguien que dé *X* pasos en una dirección y luego *Y* pasos en otra. Por ejemplo, la idea "Ve cinco pasos hacia el este" se podría expresar de esta manera como un bucle:

js

```
for (let step = 0; step < 5; step++) {
  // Se ejecuta 5 veces, con valores del paso 0 al 4.
  console.log("Camina un paso hacia el este");
}
```

Hay muchos diferentes tipos de bucles, pero esencialmente, todos hacen lo mismo: repiten una acción varias veces. (¡Ten en cuenta que es posible que ese número sea cero!).

Los diversos mecanismos de bucle ofrecen diferentes formas de determinar los puntos de inicio y terminación del bucle. Hay varias situaciones que son fácilmente atendidas por un tipo de bucle que por otros.

Las declaraciones para bucles proporcionadas en JavaScript son:

- **Declaración for**
- **Declaración do...while**
- **Declaración while**

No Recomendados

- Declaración labeled
- Declaración break
- Declaración continue

Arreglos

- Declaración for...in
- Declaración for...of

Declaración for

Un ciclo for se repite hasta que una condición especificada se evalúe como false. El bucle for de JavaScript es similar al bucle for de Java y C.

Una declaración for tiene el siguiente aspecto:

```
for ([expresiónInicial]; [expresiónCondicional]; [expresiónDeActualización])
  instrucción
```

Cuando se ejecuta un bucle for, ocurre lo siguiente:

1. Se ejecuta la expresión de iniciación expresiónInicial, si existe. Esta expresión normalmente inicia uno o más contadores de bucle, pero la sintaxis permite una expresión de cualquier grado de complejidad. Esta expresión también puede declarar variables.
2. Se evalúa la expresión expresiónCondicional. Si el valor de expresiónCondicional es verdadero, se ejecutan las instrucciones del bucle. Si el valor de condición es falso, el bucle for termina. (Si la expresión condición se omite por completo, se supone que la condición es verdadera).
3. Se ejecuta la instrucción. Para ejecutar varias instrucciones, usa una declaración de bloque ({ ... }) para agrupar esas declaraciones.
4. Si está presente, se ejecuta la expresión de actualización expresiónDeActualización.
5. El control regresa al paso 2.

Ejemplofor

En el siguiente ejemplo, la función contiene una instrucción for que cuenta el número de opciones seleccionadas en una lista de desplazamiento (el elemento <select> de HTML representa un control que proporciona un menú de opciones que permite múltiples selecciones). La instrucción for declara la variable i y la inicia a 0. Comprueba que i es menor que el número de opciones en el elemento <select>, realiza la siguiente instrucción if e incrementa i después de cada pasada por el bucle.

html

```
<form name="selectForm">
  <p>
    <label for="musicTypes"
      >Elija algunos tipos de música, luego haga clic en el botón de
      abajo:</label    >

    <select id="musicTypes" name="musicTypes" multiple="multiple">
      <option selected="selected">R&B</option>
      <option>Jazz</option>
      <option>Blues</option>
      <option>New Age</option>
      <option>Classical</option>
      <option>Opera</option>
    </select>
  </p>
  <p><input id="btn" type="button" value="¿Cuántos están seleccionados?" /></p>
</form>

<script>
  function howMany(selectObject) {
    let numberSelected = 0;
```


```
  for (let i = 0; i < selectObject.options.length; i++) {
    if (selectObject.options[i].selected) {
      numberSelected++;
    }
  }
  return numberSelected;
}

let btn = document.getElementById("btn");
btn.addEventListener("click", function () {
  alert(
    "Número de opciones seleccionadas: " +
      howMany(document.selectForm.musicTypes),
  );
});
</script>
```

Declaración do...while

La instrucción do...while se repite hasta que una condición especificada se evalúe como falsa.

Una declaración do...while tiene el siguiente aspecto:

```
do
  expresión
while (condición);
```

exposición siempre se ejecuta una vez antes de que se verifique la condición. (Para ejecutar varias instrucciones, usa una declaración de bloque ({ ... }) para agrupar esas declaraciones).

Si condición es true, la declaración se ejecuta de nuevo. Al final de cada ejecución, se comprueba la condición. Cuando la condición es false, la ejecución se detiene y el control pasa a la declaración que sigue a do...while.

Ejemplo

En el siguiente ejemplo, el bucle do itera al menos una vez y se repite hasta que i ya no sea menor que 5.

let i = 0; do { i += 1; console.log(i); } while (i < 5);

Declaración while

Una declaración while ejecuta sus instrucciones siempre que una condición especificada se evalúe como true. Una instrucción while tiene el siguiente aspecto:

```
while (condición)
  expresión
```

Si la condición se vuelve false, la instrucción dentro del bucle se deja de ejecutar y el control pasa a la instrucción que sigue al bucle.

La prueba de condición ocurre *antes* de que se ejecute la expresión en el bucle. Si la condición devuelve true, se ejecuta la expresión y la condición se prueba de nuevo. Si la condición devuelve false, la ejecución se detiene y el control se pasa a la instrucción que sigue a while.

Para ejecutar varias instrucciones, usa una declaración de bloque ({ ... }) para agrupar esas declaraciones.

Ejemplo 1

El siguiente ciclo del while se repite siempre que n sea menor que 3:

js

```
let n = 0;
let x = 0;
while (n < 3) {
  n++;
  x += n;
}
```

Con cada iteración, el bucle incrementa n y agrega ese valor a x. Por lo tanto, x y n toman los siguientes valores:

- Después de la primera pasada: n = 1 y x = 1
- Después de la segunda pasada: n = 2 y x = 3
- Después de la tercera pasada: n = 3 y x = 6

Después de completar la tercera pasada, la condición n < 3 ya no es true, por lo que el bucle termina.**Ejemplo 2**

Evita los bucles infinitos. Asegúrate de que la condición en un bucle eventualmente se convierta en false; de lo contrario, el bucle nunca terminará. Las declaraciones en el siguiente bucle while se ejecutan indefinidamente porque la condición nunca se vuelve false:

js

```
// ¡Los bucles infinitos son malos!
while (true) {
  console.log("¡Hola, mundo!");
}
```


Bootstrap
HTML
CSS
Java
Script

Declaración labeled

Una label proporciona una instrucción con un identificador que te permite hacer referencia a ella en otra parte de tu programa. Por ejemplo, puedes usar una etiqueta para identificar un bucle y luego usar las declaraciones break o continue para indicar si un programa debe interrumpir el bucle o continuar su ejecución.La sintaxis de la instrucción etiquetada es similar a la siguiente:label : instrucción

El valor de label puede ser cualquier identificador de JavaScript que no sea una palabra reservada. La declaración que identifica a una etiqueta puede ser cualquier enunciado.

Ejemplo

En este ejemplo, la etiqueta markLoop identifica un bucle while.

markLoop: while (theMark === true) { doSomething(); }

Declaración break

Usa la instrucción break para terminar un bucle, switch o junto con una declaración etiquetada.

- Cuando usas break sin una etiqueta, inmediatamente termina el while, do-while, for o switch y transfiere el control a la siguiente declaración.
- Cuando usas break con una etiqueta, termina la declaración etiquetada especificada.

La sintaxis de la instrucción break se ve así:

```
break;
break [label];
```

1. La primera forma de la sintaxis termina el bucle envolvente más interno o el switch.
2. La segunda forma de la sintaxis termina la instrucción etiquetada específica.

Ejemplo 1

El siguiente ejemplo recorre en iteración los elementos de un arreglo hasta que encuentra el índice de un elemento cuyo valor es theValue:

js

```
for (let i = 0; i < a.length; i++) {
  if (a[i] === theValue) {
    break;
  }
}
```

Ejemplo 2: romper una etiqueta

js

```
let x = 0;
let z = 0;
```


```
labelCancelLoops: while (true) {
  console.log("Bucles externos: " + x);
  x += 1;
  z = 1;
  while (true) {
    console.log("Bucles internos: " + z);
    z += 1;
    if (z === 10 && x === 10) {
      break labelCancelLoops;
    } else if (z === 10) {
      break;
    }
  }
}
```

Declaración continue

La instrucción continue se puede usar para reiniciar un while, do-while, for, o declaración label.

- Cuando utilizas continue sin una etiqueta, finaliza la iteración actual del while, do-while o for y continúa la ejecución del bucle con la siguiente iteración. A diferencia de la instrucción break, continue no termina la ejecución del bucle por completo. En un bucle while, vuelve a la condición. En un bucle for, salta a la expresión-incremento.
- Cuando usas continue con una etiqueta, se aplica a la declaración de bucle identificada con esa etiqueta.

La sintaxis de la instrucción continue se parece a la siguiente:

```
continue [label];
```

Ejemplo 1

El siguiente ejemplo muestra un bucle while con una instrucción continue que se ejecuta cuando el valor de i es 3. Por lo tanto, n toma los valores 1, 3, 7 y 12.

js

```
let i = 0;
let n = 0;
while (i < 5) {
  i++;
  if (i === 3) {
    continue;
  }
  n += i;
  console.log(n);
}
//1,3,7,12

let i = 0;
let n = 0;
while (i < 5) {
  i++;
```

```
  if (i === 3) {
    // continue;
  }
  n += i;
  console.log(n);
}
// 1,3,6,10,15
```

Ejemplo 2

Una declaración etiquetada checkiandj contiene una declaración etiquetada checkj. Si se encuentra continue, el programa termina la iteración actual de checkj y comienza la siguiente iteración. Cada vez que se encuentra continue, checkj reitera hasta que su condición devuelve false. Cuando se devuelve false, el resto de la instrucción checkiandj se completa y checkiandj reitera hasta que su condición devuelve false. Cuando se devuelve false, el programa continúa en la declaración que sigue a checkiandj.

Si continue tuviera una etiqueta de checkiandj, el programa continuaría en la parte superior de la declaración checkiandj.

let i = 0; let j = 10; checkiandj: while (i < 4) { console.log(i); i += 1; checkj: while (j > 4) { console.log(j); j -= 1; if ((j % 2) === 0) { continue checkj; } console.log(j + 'es impar.'); } console.log('i = ' + i); console.log('j = ' + j); }

Declaración for...in

La instrucción for...in itera una variable especificada sobre todas las propiedades enumerables de un objeto. Para cada propiedad distinta, JavaScript ejecuta las instrucciones especificadas. Una declaración for...in tiene el siguiente aspecto:

```
for (variable in objeto)
  instrucción
```

Ejemplo

La siguiente función toma como argumento un objeto y el nombre del objeto. Luego itera sobre todas las propiedades del objeto y devuelve una cadena que enumera los nombres de las propiedades y sus valores.

js

```
function dump_props(obj, obj_name) {
  let result = "";
  for (let i in obj) {
    result += obj_name + "." + i + " = " + obj[i] + "<br>";
  }
  result += "<hr>";
  return result;
}
```

Para un objeto car con propiedades make y model, result sería:

js

```
car.make = Ford;
car.model = Mustang;
```

Arrays

Aunque puede ser tentador usar esto como una forma de iterar sobre los elementos Array, la instrucción for...in devolverá el nombre de sus propiedades definidas por el usuario además de los índices numéricos.

Por lo tanto, es mejor usar un bucle for tradicional con un índice numérico cuando se itera sobre arreglos, porque la instrucción for...in itera sobre las propiedades definidas por el usuario además de los elementos del arreglo, si modificas el objeto Array (tal como agregar propiedades o métodos personalizados).

Declaración for...of

La declaración for...of crea un bucle que se repite sobre objetos iterables (incluidos Array, Map (en-US), Set, objetos arguments y así sucesivamente), invocando un bucle de iteración personalizado con declaraciones que se ejecutarán para el valor de cada distinta propiedad.

```
for (variable of objeto)
  expresión
```

El siguiente ejemplo muestra la diferencia entre un bucle for...of y un bucle for...in. Mientras que for...in itera sobre los nombres de propiedad, for...of itera sobre los valores de propiedad:

js

```
const arr = [3, 5, 7];
arr.foo = "hola";

for (let i in arr) {
  console.log(i); // logs "0", "1", "2", "foo"
}

for (let i of arr) {
  console.log(i); // logs 3, 5, 7
}
```

Ejercicio 1. Escribir un programa que cree muestre todos los elementos de este array con for of
const arr1 = ["Malaga", "Sevilla", "Cordoba", "Huelva", "Cadiz", "Almeria", "Granada"];
en una lista html con color verde.

1. Malaga
2. Sevilla
3. Cordoba
4. Huelva
5. Cadiz
6. Almeria
7. Granada

Ejercicio 2. Modifica el anterior para que el verde sea un degradado (000000 va a 00ff00)

1. **Malaga**
2. **Sevilla**
3. **Cordoba**
4. **Huelva**
5. **Cadiz**
6. **Almeria**
7. **Granada**

Ejercicio 3. Usa for in para ver todas las asignaturas de este objeto y for of para ver el contenido

var Mis_notas = { mates: 9, Lengua: 6, Efisica: 5, Quimica: 9, fisica: 5, ingles: 5 };

Ejercicio 4. Sea estos dos arrays

const arr1 = ["Malaga", "Sevilla", "Cordoba", "Huelva", "Cadiz", "Almeria", "Granada"];

const arr2 = ["bonita", "Guapa", " Bella", "Pintoresca", "Hermosa"];

Usa for of doble para combinarlos todos y mostrarlos en una tabla

Malaga-bonita	Sevilla-bonita	Cordoba-bonita	Huelva-bonita	Cadiz-bonita	Almeria-bonita	Granada-bonita
Malaga-Guapa	Sevilla-Guapa	Cordoba-Guapa	Huelva-Guapa	Cadiz-Guapa	Almeria-Guapa	Granada-Guapa
Malaga- Bella	Sevilla- Bella	Cordoba- Bella	Huelva- Bella	Cadiz- Bella	Almeria- Bella	Granada- Bella
Malaga-Pintoresca	Sevilla-Pintoresca	Cordoba-Pintoresca	Huelva-Pintoresca	Cadiz-Pintoresca	Almeria-Pintoresca	Granada-Pintoresca
Malaga-Cadiz	Sevilla-Cadiz	Cordoba-Cadiz	Huelva-Cadiz	Cadiz-Cadiz	Almeria-Cadiz	Granada-Cadiz

Ejercicio 5. Crea un script que cree un formulario con tantos botones como haya en el siguiente
const arr1 = ["Malaga", "Sevilla", "Cordoba", "Huelva", "Cadiz", "Almeria", "Granada"];
ponle esto dentro de cada boton

<input type="button" onclick='alert("La ciudad mas bonita es Malaga")'>

Ejercicio 6.

Escribe un programa que dibuje la frase "hola que tal" 10 veces cada vez mas pequeña ... puedes usar algo como document.write(<p style="font-size: "+i+"px"> hola que tal</p>);