Profundizando en JavaScript parte II

VÍCTOR CUSTODIO 2015

booleanos

booleanos

- Muy a menudo, en la programación, se necesita un tipo de datos que sólo puede tener uno de dos valores, como
  - SI NO
  - ENCENDIDO APAGADO
  - TRUE / FALSE
  - Podemos evaluar un boolean con la función Boolean():

```
Boolean(10 > 9)
```

Sería lo mismo que `(10 > 9)` o `10 > 9`

BOOLEANS – todo lo que no sea real es false

```
var x = 0;
Boolean(x);        // 0 devuelve false
```

```
var x = "";
Boolean(x);        // string vacio devuelve false
```

```
var x;
Boolean(x);        // indefinido devuelve false
```

BOOLEANS – todo lo que no sea real es false

```
var x = null;
Boolean(x);      //null devuelve false
```

```
var x = 10 / "H";
Boolean(x);        // NaN devuelve false
```

```
var x;
Boolean(x);        // indefinido devuelve false
```

Operadores

- Ya vimos los operadores comparativos, ==, ===, !=, <,<= etc..
- Veamos ahora los lógicos:

&&	and	(x < 10 && y > 1) es true
||	or	(x == 5 || y == 5) es false
!	not	!(x == y) es true

Operador asignación condicional

- Sintaxis :
  - *Variable = (condición)? valor1:valor2;*
- *Ejemplo*
  - `var puedeVotar= (edad < 18) ? “demasiado joven":“suficiente mayor";`

  - `Realiza los siguientes ejercicios:booealans 1,2,3,4,5,6`

Bloques Condicionales

Bloques condicionales

- Muy a menudo, cuando se escribe código, se desea realizar diferentes acciones para diferentes decisiones.
- Puedes utilizar sentencias condicionales en el código para hacer esto.
- En JavaScript tenemos las siguientes instrucciones condicionales:
  - **If** para especificar un bloque de código que se ejecutará si la condición especificada es verdadera
  - **else** para especificar un bloque de código que se ejecutará, si la misma condición es falsa
  - **else if** para añadir una nueva condición a evaluar si la primera condición es falsa
  - **switch** para especificar muchos bloques de código para ser ejecutados según una condición con mas de 2 posibilidades

If

- Declaración para especificar un bloque de código JavaScript que se ejecutará si la condición es verdadera.
- Sintaxis

if (*condicion*) {
Cloque de código a ser ejecutado si la condicion es verdadera

}

Ejemplo:

```
if (hora < 15) {
     saludo = “buenos días”;
}
```

else

- para especificar un bloque de código que se ejecutará si la condición es falsa
- Sintaxis:

```
if (condicion) {
      Bloque de código a ser ejecutado si la condicion es verdadera
} else {
      Bloque de código a ser ejecutado si la condicion es falsa
}
```

else

- Ejemplo:
- ```
  if (hora< 15) {
  saludo = “Buenos dias";
  } else {
      saludo = “Buenas tardes";
  }
  ```
  ```

Else if

- declaración para especificar una nueva condición si la primera condición es falsa.
- Sintaxis:

```
if (condicion) {
      Bloque de código a ser ejecutado si la condicion es verdadera

} else if(condición dos){
      Bloque de código a ser ejecutado si la condicion1 es falsa y la condición2 es verdadera
}else{

Bloque de código a ser ejecutado si la condicion1 y la condición2 son falsas

}
```

Else if

- Ejemplo:
- ```
  if (hora< 15) {
  saludo = “Buenos dias";
  } else  if (hora< 20) {
      saludo = “Buenas tardes";
  } else{
  saludo = “Buenas noches";
  }
  ```

- Haz los ejercicios: condicionales1,2,3,4,5
  ```

La sentencia switch

- Para seleccionar uno de los muchos bloques de código a ejecutar.
- Sintaxis:
- ```
  switch(expresion) {
      case n:
          bloque de código
          break;
    case n:
      bloque de código
          break;
    default:
      bloque de código por defecto
  }
  ```
  ```

- Así es como funciona:
  - La expresión switch se evalúa una vez.
  - El valor de la expresión se compara con los valores de cada case.
  - Si hay una coincidencia, se ejecuta el bloque asociado de código.
  - Si no hay coincidencias se ejecuta el bloque default

Switch

- Ejemplo
-
```
switch (new Date().getDay()) {
    case 0:
        day = "Sunday";
        break;
    case 1:
        day = "Monday";
        break;
    case 2:
        day = "Tuesday";
        break;
    case 3:
        day = "Wednesday";
        break;
    case 4:
        day = "Thursday";
        break;
    case 5:
        day = "Friday";
        break;
    case 6:
        day = "Saturday";
        break;
}
```

Switch - Break

- Cuando el código JavaScript intérprete **break** se rompe la ejecución del bloque y se sale fuera del switch.
- ```
  switch (new Date().getDay()) {
      case 6:
          text = “hoy es sabado";
          break;
      case 0:
          text = “hoy es domingo";
          break;
      default:
          text = “Deseando que llegue el fin de semana";
  }
  ```
  ```

Switch - Break

- Cuando el código JavaScript intérprete **break** se rompe la ejecución del bloque y se sale fuera del switch.
- ```
  switch (new Date().getDay()) {
      case 6:
          text = “hoy es sabado";
          break;
      case 0:
          text = “hoy es domingo";
          break;
      default:
          text = “Deseando que llegue el fin de semana";
  }
  ```
  ```

Sin break podemos compartir código entre casos

```
switch (new Date().getDay()) {
    case 1:
    case 2:
    case 3:
    default:
        text = “esperando el fin de semana";
        break;
    case 4:
    case 5:
       text = “pronto será fin de semana";
        break;
    case 0:
    case 6:
       text = “es fin de semana";
}
```

- Haz los ejercicios Switch1,2,3,4

Bucles

Bucles

- Los bucles pueden ejecutar un bloque de código un determinado numero de veces.
- Son útiles, si desea ejecutar el mismo código una y otra vez, cada vez con un valor diferente.
- A menudo este es el caso cuando se trabaja con arrays:

Bucle ejemplo

```
text += cars[0] + "<br>";
text += cars[1] + "<br>";
text += cars[2] + "<br>";
text += cars[3] + "<br>";
text += cars[4] + "<br>";
text += cars[5] + "<br>";
```

```
for (i = 0; i < cars.length; i++) {
    text += cars[i] + "<br>";
}
```

Distintos tipos de bucles

- **For** - bucles a través de un bloque de código un número de veces
- **For/in**- recorre las propiedades de un objeto
- **While**- bucles a través de un bloque de código mientras una condición especificada es verdadera
- **do / while** - también bucles a través de un bloque de código mientras una condición especificada es verdadera

Bucle for

- El **bucle for** es la herramienta mas utilizada para crear un bucle.
- El **bucle for** tiene la siguiente sintaxis:

```
for (declaración1 ; declaración2; declaración 3) {
    bloque de código a ser ejecutado
}
```

- **Declaración 1** se ejecuta antes de que comience el bucle (el bloque de código).
- **Declaración 2** define la condición para ejecutar el bucle (el bloque de código).
- **Declaración 3** se ejecuta cada vez que después del bucle (el bloque de código) se ha ejecutado.

Bucle For

```
for (i = 0; i < 5; i++) {
    text += “El numero es" + i + "<br>";
}
```

- Declaración 1 establece una variable antes de que comience el bucle (var i = 0).
- Declaración 2 define la condición para que el bucle siga funcionando (i debe ser inferior a 5).
- Declaración 3 aumenta un valor (i ++) cada vez que el bloque de código en el bucle se ha ejecutado.

Bucle For- Declaración 1

- Normalmente se usará la declaración 1 para iniciar la variable utilizada en el bucle (i = 0).
- Esto no siempre es el caso. La declaración 1 es opcional.
- También podemos inicializar muchos valores en la declaración 1 (separados por comas):
- Ejemplo
  - ```
    for (i = 0, len = cars.length, text = ""; i < len; i++) {
        text += cars[i] + "<br>";
    }
    ```
- Ejemplo:

```
var i = 2;
var len = cars.length;
var text = "";
for (; i < len; i++) {
    text += cars[i] + "<br>";
}
```
```

Declaración 2

- A menudo la declaración 2 se utiliza para evaluar la condición de la variable inicial.
- Si los la condición del la declaración 2 devuelve verdadera, el bucle se iniciará de nuevo, si devuelve falsa, el bucle terminará.
- La declaración 2 también es opcional. (deberemos poner un **break** dentro del bucle si queremos que acabe alguna vez)

Declaración 3

- También es opcional
- Normalmente en la declaración 3 aumentamos/disminuimos el valor de la variable inicial
- (i--) ,(i = i+15) , x++
- ```
  var i = 0;
  var len = cars.length;
  for (; i < len; ) {
      text += cars[i] + "<br>";
      i++;
  }
  ```
  ```

For / in

- En JavaScript for / in recorre las propiedades de un objeto

```
var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
    text += person[x];
}
```

Haz los ejercicios for1,2,3,4,5,6

Bucle While

- El bucle while itera a través de un bloque de código mientras una condición especificada sea cierto.
- Sintaxis:

```
while (condicion) {
    código a ser ejecutado
}
```

While

- En el siguiente ejemplo, el código en el bucle se ejecuta, una y otra vez, siempre y cuando una variable (i) es menor que 10:
- ```
  while (i < 10) {
      text += "The number is " + i;
      i++;
  }
  ```

OJO si te olvidas de aumentar la variable i, el bucle será infinito y el navegador posiblemente bloquee su ejecución.
  ```

Do/While

- El do / while es una variante del bucle while. Este bucle se ejecutará el bloque de código una vez, antes de comprobar si la condición es verdadera, entonces se repetirá el bucle mientras la condición es verdadera.
- Sintaxis:
- ```
  do {
      bloque de código a ser ejecutado
  }
  while (condicion);
  ```
  ```

Do/While

- Ejemplo: El bucle siempre se ejecuta al menos una vez, incluso si la condición es falsa(i=10), ya que el bloque de código se ejecuta antes de que la condición se compruebe:

```
do {
    text += "The number is " + i;
    i++;
}
while (i < 10);
```

- Realiza los ejercicios while1,2,3,4,5

For vs While

- Si habéis estado atentos descubrireis que un bucle while es lo mismo que un bucle for, con la declaración 1 y 3 omitidas.

```
var cars =
["BMW", "Volvo", "Saab", "Ford"];
var i = 0;
var text = "";

while (cars[i]) {
    text += cars[i] + "<br>";
    i++;
}
```

```
var cars =
["BMW", "Volvo", "Saab", "Ford"];
var i = 0;
var text = "";

for (;cars[i];) {
    text += cars[i] + "<br>";
    i++;
}
```

Break

- Ya hemos visto que la sentencia break se utiliza para "saltar" una sentencia switch ().
- La sentencia break también se puede utilizar para saltar un bucle.
- La **sentencia break** rompe el bucle y continúa ejecutando el código después del bucle (si hay):
- Ejemplo:

```
for (i = 0; i < 10; i++) {
    if (i === 3) { break; }
    text += "The number is " + i + "<br>";
}
```

Continue

- La sentencia continue también rompe una iteración del bucle, pero después el bucle se sigue ejecutando
- Ejemplo:

```
for (i = 0; i < 10; i++) {
    if (i === 3) { continue; }
    text += "The number is " + i + "<br>";
}
```

Realiza los ejercicios break1,continue1,2 y 3