Material Imprimible

Curso de JavaScript

Módulo 2

Sintaxis de JavaScript

La sintaxis de JavaScript es el conjunto de reglas, cómo se construyen los programas de JavaScript:

```
var x, y, z; // Cómo declarar variables
x = 5; y = 6; // Cómo asignar valores
z = x + y; // Cómo calcular valores
```

Valores de JavaScript

La sintaxis de JavaScript define dos tipos de valores: valores fijos y valores variables.

Los valores fijos se llaman **literals**. Los valores variables se llaman **variables**.

Literales JavaScript

Las reglas más importantes para escribir valores fijos son:

Los números se escriben con o sin decimales:

```
<p id="demo"> </p>
<script>
document.getElementById("demo").innerHTML = 10.50;
</script>
```

Las cadenas son texto, escrito entre comillas dobles o simples:

```
<p id="demo"> </p>
<p id="demo2"> </p>
```

```
<script>
document.getElementById("demo").innerHTML = 'John Doe' ;
</script>
<script>
document.getElementById("demo2").innerHTML = "John Doe" ;
</script>
```

Variables de JavaScript

En un lenguaje de programación, las **variables** se utilizan para **almacenar** valores de datos.

JavaScript usa la palabra clave var para **declarar** variables.

Se usa **un signo igual** para **asignar valores** a las variables.

En este ejemplo, x se define como una variable. Entonces, a x se le asigna (dado) el valor 6:

```
<! DOCTYPE html>
<html>
<body>

<h2> Variables de JavaScript </h2>

<p> En este ejemplo, x se define como una variable.
Entonces, a x se le asigna el valor de 6: </p>

<p id = "demo"> </p>

<script>
var x;
x = 6;
document.getElementById ("demo"). innerHTML = x;
</script>

</body>
```

```
</html>
```

Operadores JavaScript

JavaScript usa **operadores aritméticos** (+ - * /) para **calcular** valores:

```
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = (5 + 6) * 10;
</script>
```

JavaScript usa un **operador de asignación** (=) para **asignar** valores a las variables:

```
<p id="demo"></p>
<script>
var x, y;
x = 5;
y = 6;
document.getElementById("demo").innerHTML = x + y;
</script>
```

Expresiones JavaScript

Una expresión es una combinación de valores, variables y operadores, que se calcula en un valor.

El cálculo se llama evaluación.

Por ejemplo, 5 * 10 evalúa a 50:

```
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 5 * 10;
</script>
```

Las expresiones también pueden contener valores variables:

```
<p id="demo"></p>
<script>
var x;
x = 5;
document.getElementById("demo").innerHTML = x * 10;
```

```
</script>
```

Los valores pueden ser de varios tipos, como números y cadenas.

Por ejemplo, "John" + "" + "Doe", se evalúa como "John Doe":

```
<! DOCTYPE html>
<html>
<cuerpo>
<h2> Expresiones de JavaScript </h2>
<p> Las expresiones se calculan en valores. </p>
<p id = "demo"> </p>
<script>
document.getElementById ("demo"). innerHTML = "John" + "" + "Doe";
</script>
</body>
</html>
```

Palabras clave de JavaScript

Las **palabras clave de** JavaScript se utilizan para identificar acciones a realizar.

La palabra clave var le dice al navegador que cree variables:

```
<! DOCTYPE html>
<html>
<body>

<h2> La palabra clave var crea variables </h2>

<p id = "demo"> </p>

<script>
var x, y;
x = 5 + 6;
```

```
y = x * 10;
document.getElementById ("demo"). innerHTML = y;
</script>

</body>
</html>
```

Comentarios de JavaScript

No todas las declaraciones de JavaScript se "ejecutan".

Código después de barras dobles `//` o entre `/*` y `*/` se trata como un **comentario**.

Los comentarios se ignoran y no se ejecutarán:

```
<p id = "demo"> </p>
<script>
var x;
x = 5;
// x = 6; No sera ejecutado
document.getElementById ("demo"). innerHTML = x;
</script>
```

Identificadores de JavaScript

Los identificadores son nombres.

En JavaScript, los identificadores se usan para nombrar variables (y palabras clave, funciones y etiquetas).

Las reglas para los nombres legales son muy similares en la mayoría de los lenguajes de programación.

En JavaScript, el primer caracter debe ser una letra, un guión bajo (_) o un signo de pesos ($).

Los caracteres posteriores pueden ser letras, dígitos, guiones bajos o signos de pesos.

Los números no están permitidos como primer carácter.

De esta manera, JavaScript puede distinguir fácilmente los identificadores de los números

JavaScript distingue mayúsculas de minúsculas

Todos los identificadores de JavaScript distinguen entre **mayúsculas y minúsculas**.

Las variables lastNamey lastname, son dos variables diferentes:

```
<! DOCTYPE html>
<html>
<body>
<h2> JavaScript distingue mayúsculas y minúsculas </h2>
<p> Intente cambiar apellido por apellido. </p>
<p id = "demo"> </p>
<script>
var lastname, lastName;
lastName = "Doe";
lastname = "Peterson";
document.getElementById ("demo"). innerHTML = lastName;
</script>
</body>
</html>
```

JavaScript no interpreta **VAR** o **Var** como la palabra clave **var** .

JavaScript y Camel Case

Históricamente, los programadores han utilizado diferentes formas de unir varias palabras en un nombre de variable:

Guiones:

First-name, last-name, master-card, inter-city.

Los guiones no están permitidos en JavaScript. Están reservados para sustracciones.

Guion bajo:

first_name, last_name, master_card, inter_city.

Upper Camel Case (Estuche Pascal):

FirstName, LastName, MasterCard, InterCity.

Lower Camel Case:

Los programadores de JavaScript tienden a usar mayúsculas y minúsculas que comienzan con una letra minúscula:

firstName, lastName, masterCard, interCity.

Conjunto de caracteres de JavaScript

JavaScript usa el conjunto de caracteres **Unicode**.

Unicode cubre (casi) todos los caracteres, signos de puntuación y símbolos del mundo.

Comentarios de JavaScript

Los comentarios de JavaScript se pueden usar para explicar el código de JavaScript y hacerlo más legible.

Los comentarios de JavaScript también se pueden usar para evitar la ejecución, al probar código alternativo.

Comentarios de línea única

Los comentarios de una sola línea comienzan con //.

//JavaScript ignorará cualquier texto entre y al final de la línea (no se ejecutará).

Este ejemplo usa un comentario de una sola línea antes de cada línea de código:

```
<!DOCTYPE html>
<html>
<body>
<h1 id = "myH"> </h1>
<p id = "myP"> </p>
<script>
// Cambiar título:
document.getElementById ("myH"). innerHTML = "Comentarios de JavaScript";
// Cambiar párrafo:
document.getElementById ("myP"). innerHTML = "Mi primer párrafo";
</script>
</body>
</html>
```

Este ejemplo utiliza un comentario de una sola línea al final de cada línea para explicar el código:

```
<p id = "demo"> </p>

<script>
var x = 5; // Declara x, dale el valor de 5
var y = x + 2; // Declara y, dale el valor de x + 2
// Escribe y en la demostración:
document.getElementById ("demo"). innerHTML = y;
</script>
```

Comentarios multilínea

Los comentarios de varias líneas comienzan /*y terminan con */.

Cualquier texto entre /*y */será ignorado por JavaScript.

Este ejemplo utiliza un comentario de varias líneas (un bloque de comentarios) para explicar el código:

```
<h1 id = "myH"> </h1>
<p id = "myP"> </p>

<script>
/ *
El código a continuación cambiará
el encabezado con id = "myH"
y el párrafo con id = "myP"
* /
document.getElementById ("myH"). innerHTML = "Comentarios de JavaScript";
document.getElementById ("myP"). innerHTML = "Mi primer párrafo";
</script>
```

Es más común usar comentarios de una sola línea.

Los comentarios de bloque a menudo se usan para la documentación formal.

Uso de comentarios para evitar la ejecución

El uso de comentarios para evitar la ejecución de código es adecuado para la prueba de código.

Agregar //delante de una línea de código cambia las líneas de código de una línea ejecutable a un comentario.

Este ejemplo usa // para evitar la ejecución de una de las líneas de código:

```
//document.getElementById("myH").innerHTML = "Mi primera página";
document.getElementById ("myP"). innerHTML = "Mi primer párrafo";
```

Este ejemplo utiliza un bloque de comentarios para evitar la ejecución de varias líneas:

```
/ *
document.getElementById ("myH"). innerHTML = "Mi primera página";
document.getElementById ("myP"). innerHTML = "Mi primer párrafo";
* /
```

Muy parecido al álgebra

En este ejemplo, price1, price2, y total, son variables:

```
var precio1 = 5;
var precio2 = 6;
var total = precio1 + precio2;
```

En programación, al igual que en álgebra, utilizamos variables (como price1) para mantener los valores.

En programación, al igual que en álgebra, usamos variables en expresiones (total = precio1 + precio2).

A partir del ejemplo anterior, puede calcular el total como 11.

Las variables de JavaScript son contenedores para almacenar valores de datos.

Identificadores de JavaScript

Todas las **variables de** JavaScript deben **identificarse** con **nombres únicos**.

Estos nombres únicos se denominan **identificadores**.

Los identificadores pueden ser nombres cortos (como x e y) o nombres más descriptivos (edad, suma, volumen_total).

Las reglas generales para construir nombres para variables (identificadores únicos) son:

- Los nombres pueden contener letras, dígitos, guiones bajos y signos de pesos.
- Los nombres deben comenzar con una letra
- Los nombres también pueden comenzar con $ y _
- Los nombres distinguen entre mayúsculas y minúsculas (y e Y son variables diferentes)
- Las palabras reservadas (como las palabras clave de JavaScript) no se pueden usar como nombres

El operador de asignación

En JavaScript, el signo igual (=) es un operador de "asignación", no un operador "igual a".

Esto es diferente del álgebra. Lo siguiente no tiene sentido en álgebra:

x = x + 5

En JavaScript, sin embargo, tiene mucho sentido: asigna el valor de x + 5 a x.

(Calcula el valor de x + 5 y pone el resultado en x. El valor de x se incrementa en 5.)

El operador "igual a" se escribe como ==en JavaScript.

Tipos de datos de JavaScript

Las variables de JavaScript pueden contener números como 100 y valores de texto como "John Doe".

En programación, los valores de texto se denominan cadenas de texto.

JavaScript puede manejar muchos tipos de datos, pero por ahora, solo piense en números y cadenas de caracteres.

Las cadenas se escriben entre comillas dobles o simples. Los números se escriben sin comillas.

Si pone un número entre comillas, se tratará como una cadena de texto.

```
<! DOCTYPE html>
<html>
<body>

<h2> Variables de JavaScript </h2>
```

```
<p> Las cadenas se escriben entre comillas. </p>
<p> Los números se escriben sin comillas. </p>

<p id = "demo"> </p>

<script>
var pi = 3.14;
var persona = "John Doe";
var respuesta = '¡Sí, lo soy!';

document.getElementById ("demo"). innerHTML =
pi + "<br>" + persona + "<br>" + respuesta;
</script>

</body>
</html>
```

Declarando (Creando) Variables JavaScript

Crear una variable en JavaScript se llama "declarar" una variable.

Usted declara una variable de JavaScript con la varpalabra clave:

```
var carName;
```

Después de la declaración, la variable no tiene valor (técnicamente tiene el valor de undefined).

Para asignar un valor a la variable, use el signo igual:

```
carName = "Volvo";
```

También puede asignar un valor a la variable cuando la declara:

```
var carName = "Volvo";
```

En el siguiente ejemplo, creamos una variable llamada carName y le asignamos el valor "Volvo".

Luego "sacamos" el valor dentro de un párrafo HTML con id = "demo":

```
<p id="demo"></p>
<script>
var carName = "Volvo";
document.getElementById("demo").innerHTML = carName;
</script>
```

Es una buena práctica de programación declarar todas las variables al comienzo de un script.

Una declaración, muchas variables

Puede declarar muchas variables en una declaración.

Comience la declaración con var y separe las variables por **coma**:

```
var person = "John Doe", carName = "Volvo", price = 200;
```

Una declaración puede abarcar varias líneas:

```
<p id="demo"></p>

<script>
var person = "John Doe",
carName = "Volvo",
price = 200;
document.getElementById("demo").innerHTML = carName;
</script>
```

Valor = indefinido

En los programas de computadora, las variables a menudo se declaran sin un valor. El valor puede ser algo que debe calcularse o algo que se proporcionará más adelante, como la entrada del usuario.

Una variable declarada sin un valor tendrá el valor undefined.

La variable carName tendrá el valor undefineddespués de la ejecución de esta declaración:

```
var carName;
```

Re-declarando variables de JavaScript

Si vuelve a declarar una variable de JavaScript, no perderá su valor.

La variable carNameseguirá teniendo el valor "Volvo" después de la ejecución de estas declaraciones:

```
var carName = "Volvo";
var carName;
```

Aritmética JavaScript

Al igual que con el álgebra, puede hacer aritmética con variables de JavaScript, utilizando operadores como =y +:

```
<!DOCTYPE html>
<html>
<body>
<h2> Variables de JavaScript </h2>
<p> El resultado de sumar 5 + 2 + 3: </p>
<p id = "demo"> </p>
<script>
var x = 5 + 2 + 3;
document.getElementById ("demo"). innerHTML = x;
</script>
</body>
</html>
```

También puede agregar cadenas, pero las cadenas se concatenarán:

```
var x = "John" + " " + "Doe";
```

Prueba también esto:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Variables</h2>
```

```
<p> El resultado de agregar "5" + 2 + 3:</p>

<p id="demo"></p>

<script>
x = "5" + 2 + 3;
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

Si pone un número entre comillas, el resto de los números se tratarán como cadenas y se concatenarán.

Ahora que pasa con esto:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Variables</h2>
<p> El resultado de agregar 2 + 3 + "5":</p>
<p id="demo"></p>
<script>
var x = 2 + 3 + "5"
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

JavaScript Signo pesos $

Recuerde que los identificadores de JavaScript (nombres) deben comenzar con:

- Una letra (AZ o az)
- Un signo de pesos ($)
- O un guión bajo (_)

Como JavaScript trata un signo de dólar como una letra, los identificadores que contienen $ son nombres de variables válidos:

```
<p id="demo"></p>
<script>
var $ = 2;
var $myMoney = 5;
document.getElementById("demo").innerHTML = $ + $myMoney;
</script>
```

El uso del signo de pesos no es muy común en JavaScript, pero los programadores profesionales a menudo lo usan como un alias para la función principal en una biblioteca de JavaScript.
En la biblioteca JavaScript jQuery, por ejemplo, la función principal $se usa para seleccionar elementos HTML. En jQuery $("p");significa "seleccionar todos los elementos p".
jQuery es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web

Subrayado de JavaScript (_)

Como JavaScript trata el guión bajo como una letra, los identificadores que contienen _ son nombres de variables válidos:

```
var _lastName = "Johnson";
var _x = 2;
var _100 = 5;
```

El uso del guion bajo no es muy común en JavaScript, pero una convención entre los programadores profesionales es usarlo como un alias para las variables "privadas (ocultas)".

Fuentes:

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Grammar_and_types

https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Math

http://progra.usm.cl/apunte/materia/expresiones.html