Técnico en

<DESARROLLO DE SOFTWARE>

Introducción a la Programación de Computadoras

Introducción a la Programación de Computadoras

Unidad II: Conceptos básicos para iniciar a programar

1. Conceptos básicos para iniciar a programar

Ahora que ya se tiene un entorno de trabajo establecido, ya se tiene el sistema operativo que se va a utilizar y ya se está familiarizado con al menos un editor de textos con el que va a empezar a programar, entonces ya se puede dar inicio a lo básico de la programación. Para ello vamos a iniciar con los distintos tipos de valores, tipos de datos, variables y operadores, condiciones. Para revisarlo y demostrarlo se estará utilizando JavaScript.

Variables

Son datos que se almacenan en memoria y que pueden cambiar su valor durante la ejecución del programa.

Para identificar una variable, es necesario seguir los estándares utilizados según el lenguaje de programación que se utilice. Pero en general se deben tomar en cuenta los siguientes aspectos:

- Las variables no pueden ser de longitudes extremadamente cortas, ya que posiblemente no identifiquen al valor que se quiere guardar en la variable. Ejemplo: i, t, j, etc.
- Las variables no pueden ser de longitudes extremadamente largas.
- El primer carácter debe ser una letra, no un número y no deben tener signos de puntuación.
- No se permite el uso de espacios en los nombres de las variables.
- Se espera que el nombre de la variable identifique a su valor. Ejemplo: *nombre*, *apellido*, *telefono*, *direccion*, etc. Si vamos a guardar el nombre de una persona, no vamos a nombrar a la variable como “apellido” o “v1”.
- Para separar palabras compuestas en los nombres de variables, se pueden utilizar varias opciones, entre las cuales se encuentran: CamelCase, ej: PrimerApellido o separar las palabras con el signo “_”, ej: primer_apellido.

El fin de estos aspectos es que se tengan variables que representen a su valor y que al momento de volver a leer el código por el mismo programador o por alguien más, éste se entienda.

Antes de utilizar una variable, es necesario declararla. Por ejemplo en JavaScript las variables se declaran con la palabra reservada **var.**

```
var nombre;
var total;
```

También se pueden declarar las variables con la misma palabra reservada, en una sola línea.

```
var nombre, total;
```

Para almacenar un valor en una variable, se asocia un identificador con un valor. Ejemplo en JavaScript:

```
total = 5;
```

También se puede combinar la declaración con el almacenamiento de valores.

```
var asunto = "Inicio de la Unidad 3";
var email = "isabel22@galileo.edu"
```

Mientras no se almacene un valor en una variable declarada, su valor será *undefined* (indefinido).

Constantes

Son datos que se almacenan en memoria, pero no cambian su valor durante la ejecución del programa.

En JavaScript funciona de manera similar que las variables pero se diferencia a la hora de declarar las constantes. Las constantes se declaran con la palabra reservada **const** y de una vez se debe almacenar su valor**.** Ejemplo:

```
const pi = 3.1416;
```

Palabras reservadas

En todos los lenguajes de programación hay ciertas palabras clave que no pueden utilizarse como identificadores en los programas.

Por ejemplo en JavaScript no se puede utilizar ninguna de estas palabras claves como identificadores: break, case, catch, continue, debugger, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with.

Existen otras palabras reservadas para el futuro que también hay que evitar: abstract, boolean, byte, char, class, const, debugger, double, enum, export, extends, final, float, goto, implements, import, int, interface, long, native, package, private, protected, public, short, static, super, synchronized, throws, transient, volatile.

2. Tipos de datos y operadores

Un tipo de dato es una clasificación que define los posibles valores y las operaciones que se pueden realizar. Los tipos de datos simples son los siguientes:

Tipos de datos numéricos

Un tipo de datos numérico representa un subconjunto finito de números. Algunos lenguajes de programación hacen distinción entre números enteros y números decimales. Para cada lenguaje de programación existe un rango de los números que la computadora puede representar, algunos soportan números negativos otros no.

Sintaxis en JavaScript

JavaScript no hace distinción entre enteros y decimales. Puede representar todos los números enteros entre −9007199254740992 (−2^53) y 9007199254740992 (2^53). Cuando un número aparece directamente en JavaScript se le denomina literal numérico.

Ejemplo:

```
1;

2.20;

340232;
```

Para escribir un número negativo se le antepone al número un guion.

Ejemplo:

```
-2;

-423;
```

Los números decimales también pueden utilizar notación científica. Se debe escribir el número seguido de la letra e y el exponente. Ejemplo:

```
3.45e12; // 3.45 x 10^12

2.38E-5; // 2.38 x 10^-5
```

Ejemplo de almacenamiento de valor tipo numérico en variable.

```
var precio = 4.34;
```

Operaciones numéricas

Se pueden realizar las típicas operaciones aritméticas: suma, resta, multiplicación y división. Dependiendo del lenguaje de programación se pueden realizar otro tipo de operaciones aritméticas avanzadas que veremos más adelante. Las operaciones aritméticas también se pueden realizar con variables que almacenan datos del tipo numérico.

Operaciones aritméticas

Operación aritmética	Operador	Ejemplo
Suma	**+**	**3 + 4**
Resta	**-**	**10 - 5**
Multiplicación	*****	**9 * 5**
División	**/**	**27 / 3**
Módulo	**%**	**5 % 2**

Operaciones comparativas

Las operaciones comparativas son operadores en su mayoría binarios nos permiten comparar datos, devolviendo verdadero o falso según sea el caso. Las operaciones comparativas también se pueden realizar con variables que almacenan datos.

Operaciones comparativas en JavaScript

Operación comparativa	Operador	Ejemplo

Igualdad	==	20 == 20 // Verdadero
		23 == 50 // Falso

No igualdad	!=	4 != 3 // Verdadero 86 != 86 // Falso
Mayor que	>	6 > 3 // Verdadero 3 > 6 // Falso
Mayor o igual que	>=	5 >= 5 // Verdadero 2 >= 8 // Falso
Menor que	<	3 < 9 // Verdadero 9 < 3 // Falso
Menor o igual que	<=	40 <= 56 // Verdadero 56 <= 40 // Falso

Tipos de datos lógicos

Un tipo de dato lógico o booleano representa verdadero o falso, encendido o apagado, sí o no. Hay sólo dos posibles valores de este tipo de dato.

En JavaScript utiliza las palabras reservadas **true** y **false** para verdadero y falso. También se puede utilizar 1 y 0 para verdadero y falso. Undefined también se considera falso.

```
var verdadero = true;
var falso = 0;
```

Los valores booleanos generalmente provienen del resultado de las operaciones comparativas, los cuales veremos mas adelante.

```
var verdadero = 5 >= 2;
var falso = 3 < 3;
```

Operaciones lógicas (and, or, not)

Estos operadores comúnmente son utilizados con los operadores comparativos para cumplir múltiples condiciones.

- AND

  Funciona de la misma manera que la conjunción lógica. El resultado es verdadero si ambas expresiones son verdaderas.

- OR

  Funciona de la misma manera que la disyunción lógica. El resultado es verdadero si alguna expresión es verdadera.

- NOT

  Funciona de la misma manera que la negación. El resultado invierte el valor lógico.

Operaciones lógicas en JavaScript

Operación lógica	Operador	Ejemplo
and	&&	true && true // Verdadero true && false // Falso false && true // Falso false && false // Falso
or	||	true || true // Verdadero true || false // Verdadero false || true // Verdadero false || false // Falso
not	!	!true // Falso !false // Verdadero

Tipos de datos carácter

El tipo de dato carácter se utiliza para representar un símbolo de un código predeterminado. Normalmente se utiliza el código ASCII el cual está formado por 128 caracteres en los que se incluyen letras, números, signos de puntuación, espacio y otros.

Cadena de caracteres

Una cadena de caracteres (String) es una secuencia ordenada de caracteres. En algunos lenguajes de programación es diferente la declaración de un carácter a la de una cadena de caracteres.

En **JavaScript** un literal de cadenas de caracteres o un carácter se declara entre comillas dobles o comillas simples.

```
var asunto = "Confirmación de pedido";
var nombre = 'Andrea Morales;
```

3. Estructuras de Control Condicionales

Para la toma de decisiones dentro de algoritmos y en general en cualquier programa, es necesario utilizar instrucciones condicionales. En la mayoría de lenguajes de programación se tienen las siguientes instrucciones:

If

Es una instrucción utilizada para ejecutar una porción de código si y sólo si la condición especificada es verdadera. Ejemplo:

Si tenemos todos los departamentos de Guatemala listados y cada uno tiene asignado un código de la siguiente forma:

Código	Departamento	Cabecera
1	Alta Verapaz	Cobán
2	Baja Verapaz	Salamá
3	Chimaltenango	Chimaltenango
4	Chiquimula	Chiquimula
5	El Progreso	Guastatoya
6	Escuintla	Escuintla
7	Guatemala	Ciudad de Guatemala
8	Huehuetenango	Huehuetenango
9	Izabal	Puerto Barrios
10	Jalapa	Jalapa
11	Jutiapa	Jutiapa
12	Petén	Flores
13	Quetzaltenango	Quetzaltenango
14	Quiché	Santa Cruz del Quiché
15	Retalhuleu	Retalhuleu
16	Sacatepéquez	La Antigua Guatemala
17	San Marcos	San Marcos
18	Santa Rosa	Cuilapa
19	Sololá	Sololá
20	Suchitepéquez	Mazatenango
21	Totonicapán	Totonicapán
22	Zacapa	Zacapa

Si queremos verificar que la persona que estamos atendiendo es del departamento de Guatemala, entonces nuestra variable *departamento* tendría el valor 1 de la siguiente forma:

```
var departamento = 1;
```

y nuestra condición quedaría de la siguiente forma:

```
if (departamento == 1 ) {
   alert("Pertenece al departamento Guatemala");
}
```

Si pertenece a cualquier otro departamento, entonces no hará nada.

If ... else

Es una instrucción utilizada para ejecutar una porción de código si la condición especificada es verdadera, pero también cuenta con otra instrucción para el caso donde la condición es falsa. Continuando con el ejemplo anterior:

```
var departamento = 3;
```

```
if (departamento == 1 ) {
   alert("Pertenece al departamento Guatemala");
} else {
   alert("Pertenece a otro departamento distinto a Guatemala");
}
```

El ejemplo anterior debería mostrar el mensaje "Pertenece a otro departamento distinto a Guatemala"; en cambio si le colocan al valor de la variable departamento 1, entonces debería mostrar el mensaje " Pertenece al departamento Guatemala".

If ... else if ... else

La instrucción "else if" es utilizada para ejecutar una porción de código en caso de que la primera condición sea falsa y se quiera validar una o más condiciones. Ejemplo:

```
var departamento = 5;
```

```
if (departamento == 1 ) {
   alert("Pertenece al departamento Guatemala");
} else if (departamento == 3 ) {
   alert("Pertenece al departamento Chimaltenango");
} else {
   alert("Pertenece a otro departamento distinto a Guatemala o Chimaltenango");
}
```

El ejemplo anterior debería mostrar el mensaje " Pertenece a otro departamento distinto a Guatemala o Chimaltenango "; y dependiendo del valor que tenga la variable "departamento", ese mensaje va a mostrar.

Switch case

Esta instrucción es utilizada cuando se tienen varios bloques de código y se quiere escoger entre uno de ellos dependiendo del valor que tenga una variable en específico.

Ejemplo:

var departamento = 5;

```
switch(departamento) {
        case 1:
                alert("Alta Verapaz");
                break;
        case 2:
                alert("Baja Verapaz");
                break;
        case 3:
                alert("Chimaltenango");
                break;
        case 4:
                alert("Chiquimula");
                break;
        case 5:
```

```
            alert("El Progreso");
            break;
        case 6:
            alert("Escuintla");
            break;
        case 7:
            alert("Guatemala");
            break;
        case 8:
            alert("Huehuetenango");
            break;
        case 9:
            alert("Izabal");
            break;
        case 10:
            alert("Jalapa");
            break;
        case 11:
            alert("Jutiapa");
            break;
        case 12:
            alert("Petén");
            break;
        case 13:
            alert("Quetzaltenango");
            break;
        case 14:
```

```
            alert("Quiché");
            break;
        case 15:
            alert("Retalhuleu");
            break;
        case 16:
            alert("Sacatepéquez");
            break;
        case 17:
            alert("San Marcos");
            break;
        case 18:
            alert("Santa Rosa");
            break;
        case 19:
            alert("Sololá");
            break;
        case 20:
            alert("Suchitepéquez");
            break;
        case 21:
            alert("Totonicapán");
            break;
        case 22:
            alert("Zacapa");
            break;
        default:
```

```
            alert("No es un departamento válido");
            break;
}
```

La instrucción "break" que se muestra en el ejemplo anterior se utiliza para detener la ejecución de un bloque de código y continuar con la siguiente estructura, bucle, ciclo o código del programa.

Si en el ejemplo anterior no hubiéramos colocado la instrucción break al finalizar el bloque de código, se hubiera ejecutado cada uno de los bloques de código de las distintas opciones que se tenían disponibles en el switch.

Referencias

- http://oreilly.com/javascript/excerpts/learning-javascript/javascript-datatypes-variables.html
- http://www.w3schools.com/js/
- http://librosweb.es/libro/javascript/
- John C. Mitchell. Concepts In Programming Languages, Cambridge University Press, 2003.

Material de Apoyo

- Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship. ISBN-13: 978-0132350884. ISBN-10: 9780132350884

Descargo de responsabilidad

La información contenida en este documento descargable en formato PDF o PPT es un reflejo del material virtual presentado en la versión online del curso. Por lo tanto, su contenido, gráficos, links de consulta, acotaciones y comentarios son responsabilidad exclusiva de su(s) respectivo(s) autor(es) por lo que su contenido no compromete al área de e-Learning del Departamento GES o al programa académico al que pertenece.

El área de e-Learning no asume ninguna responsabilidad por la actualidad, exactitud, obligaciones de derechos de autor, integridad o calidad de los contenidos proporcionados y se aclara que la utilización de este descargable se encuentra limitada de manera expresa para los propósitos educacionales del curso.