UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
Centro de e-Learning
Secretaría de Cultura y Extensión Universitaria

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Diplomatura en programación web full stack con React JS

UTN.BA | Centro de e-Learning
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
Secretaría de Cultura y Extensión Universitaria

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo 3:

Javascript y maquetado avanzado

Unidad 3:

Javascript (parte 1)

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Presentación

En esta unidad vemos el lenguaje JavaScript que nos ayudará a crear páginas web interactivas.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Objetivos

Que los participantes logren...

- Conocer y aprender el lenguaje Javascript.
- Entender y utilizar las variables.
- Conocer para qué sirven las estructuras de control, bucles y funciones.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Bloques temáticos

1. ¿Qué es Javascript?
2. Características básicas.
3. Variables.
4. Estructuras de control.
5. Bucles.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

1.¿Qué es Javascript?

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

Cómo incluir JavaScript en documentos HTML

La integración de JavaScript y HTML es muy flexible, ya que existen al menos tres formas para incluir código JavaScript en las páginas web.

Incluir JavaScript en el mismo documento HTML

El código JavaScript se encierra entre etiquetas **<script>** y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código JavaScript dentro de la cabecera del documento (la etiqueta **<head>**):

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script type="text/javascript">
        alert("Un mensaje de prueba");
    </script>
</head>
<body>

</body>
</html>
```

Para que la página HTML resultante sea válida, es necesario añadir el atributo type a la etiqueta **<script>.** Los valores que se incluyen en el atributo **type** están estandarizados y para el caso de JavaScript, el valor correcto es **text/javascript.**

Este método se emplea cuando se define un bloque pequeño de código o cuando se quieren incluir instrucciones específicas en un determinado documento HTML que completen las instrucciones y funciones que se incluyen por defecto en todos los documentos del sitio web.

El principal inconveniente es que si se quiere hacer una modificación en el bloque de código, es necesario modificar todas las páginas que incluyen ese mismo bloque de código JavaScript.

Definir JavaScript en un archivo externo

Las instrucciones JavaScript se pueden incluir en un archivo externo de tipo JavaScript que los documentos HTML enlazan mediante la etiqueta **<script>**. Se pueden crear todos los archivos.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

JavaScript que sean necesarios y cada documento HTML puede enlazar tantos archivos JavaScript como necesite.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script type="text/javascript" src="js/codigo.js">

    </script>
</head>
<body>

</body>
</html>
```

Además del atributo type, este método requiere definir el atributo src, que es el que indica la URL correspondiente al archivo JavaScript que se quiere enlazar. Cada etiqueta **<script>** solamente puede enlazar un único archivo, pero en una misma página se pueden incluir tantas etiquetas **<script>** como sean necesarias.

Los archivos de tipo JavaScript son documentos normales de texto con la extensión .js, que se pueden crear con cualquier editor de texto como Notepad, Wordpad, EmEditor, UltraEdit, Vi, etc.

La principal ventaja de enlazar un archivo JavaScript externo es que se simplifica el código HTML de la página, que se puede reutilizar el mismo código JavaScript en todas las páginas del sitio web y que cualquier modificación realizada en el archivo JavaScript se ve reflejada inmediatamente en todas las páginas HTML que lo enlazan.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

2. Características básicas

La sintaxis de un lenguaje de programación se define como el conjunto de reglas que deben seguirse al escribir el código fuente de los programas para considerarse como correctos para ese lenguaje de programación.

La sintaxis de JavaScript es muy similar a la de otros lenguajes de programación como Java y C.

Las normas básicas que definen la sintaxis de JavaScript son las siguientes:

- No se tienen en cuenta los espacios en blanco y las nuevas líneas: como sucede con HTML, el intérprete de JavaScript ignora cualquier espacio en blanco sobrante, por lo que el código se puede ordenar de forma adecuada para entenderlo mejor (tabulando las líneas, añadiendo espacios, creando nuevas líneas, etc.)
- **Se distinguen las mayúsculas y minúsculas**: al igual que sucede con la sintaxis de las etiquetas y elementos HTML. Sin embargo, si en una página HTML se utilizan indistintamente mayúsculas y minúsculas, la página se visualiza correctamente, siendo el único problema la no validación de la página. En cambio, si en JavaScript se intercambian mayúsculas y minúsculas el script no funciona.
- **No se define el tipo** de las variables: al crear una variable, no es necesario indicar el tipo de dato que almacenará. De esta forma, una misma variable puede almacenar diferentes tipos de datos durante la ejecución del script.
- No es necesario terminar cada sentencia con el **carácter de punto y coma (;)**: en la mayoría de lenguajes de programación, es obligatorio terminar cada sentencia con el carácter ;. Aunque JavaScript no obliga a hacerlo, es conveniente seguir la tradición de terminar cada sentencia con el carácter del punto y coma (;).
- Se pueden **incluir comentarios**: los comentarios se utilizan para añadir información en el código fuente del programa. Aunque el contenido de los comentarios no se visualiza por pantalla, si que se envía al navegador del usuario junto con el resto del script, por lo que es necesario extremar las precauciones sobre la información incluida en los comentarios.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

3. Variables

Las variables en los lenguajes de programación siguen una lógica similar a las variables utilizadas en otros ámbitos como las matemáticas. Una variable es un elemento que se emplea **para almacenar y hacer referencia a otro valor.** Gracias a las variables es posible crear "programas genéricos", es decir, programas que funcionan siempre igual independientemente de los valores concretos utilizados.

De la misma forma que si en matemáticas no existieran las variables no se podrían definir las ecuaciones y fórmulas, en programación no se podrían hacer programas realmente útiles sin las variables.

Si no existieran variables, un programa que suma dos números podría escribirse como:

resultado = 3 + 1

El programa anterior es tan poco útil que sólo sirve para el caso en el que el primer número de la suma sea el 3 y el segundo número sea el 1. En cualquier otro caso, el programa obtiene un resultado incorrecto.

Sin embargo, el programa se puede rehacer de la siguiente manera utilizando variables para almacenar y referirse a cada número:

```
<script type="text/javascript">
        numero_1 = 3;
        numero_2 = 1;
        resultado = numero_1 + numero_2;
</script>
```

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Los elementos numero_1 y numero_2 son variables que almacenan los valores que utiliza el programa. El resultado se calcula siempre en función del valor almacenado por las variables, por lo que este programa funciona correctamente para cualquier par de números indicado. Si se modifica el valor de las variables numero_1 y numero_2, el programa sigue funcionando correctamente.

Tipos de variables

Aunque todas las variables de JavaScript se crean de la misma forma (mediante la palabra reservada var), la forma en la que se les asigna un valor depende del tipo de valor que se quiere almacenar (números, textos, etc.)

Numéricas

Se utilizan para **almacenar valores numéricos** enteros (llamados integer en inglés) o decimales (llamados float en inglés). En este caso, el valor se asigna indicando directamente el número entero o decimal.

Los números decimales utilizan el carácter . (punto) en vez de , (coma) para separar la parte entera y la parte decimal:

```
const iva = 21; // variable tipo entero
const total = 15000.65 // variable tipo decimal
```

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Cadenas de texto

Se utilizan para **almacenar caracteres, palabras y/o frases de texto**. Para asignar el valor a la variable, se encierra el valor entre comillas dobles o simples, para delimitar su comienzo y su final:

```
const mensaje = "Bienvenidos al curso"
const nombreProducto = 'Compu X'
const letraSeleccionada ='F'
```

Arrays

En ocasiones, a los arrays se les llama vectores, matrices e incluso arreglos. No obstante, el término array es el más utilizado y es una palabra comúnmente aceptada en el entorno de la programación.

Un array es una **colección de variables,** que pueden ser todas del mismo tipo o cada una de un tipo diferente. Su utilidad se comprende mejor con un ejemplo sencillo: si una aplicación necesita manejar los días de la semana, se podrían crear siete variables de tipo texto:

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

```
const dia1 = 'Lunes'
const dia2 = 'Martes'
const dia3 = 'Miércoles'
const dia4 = 'Jueves'
const dia5 = 'Viernes'
const dia6 = 'Sábado'
const dia7 = 'Domingo'
```

Aunque el código anterior no es incorrecto, sí que es poco eficiente y complica en exceso la programación. Si en vez de los días de la semana se tuviera que guardar el nombre de los meses del año, el nombre de todos los países del mundo o las mediciones diarias de temperatura de los últimos 100 años, se tendrían que crear decenas o cientos de variables.

En este tipo de casos, se pueden agrupar todas las variables relacionadas en una colección de variables o array. El ejemplo anterior se puede rehacer de la siguiente forma:

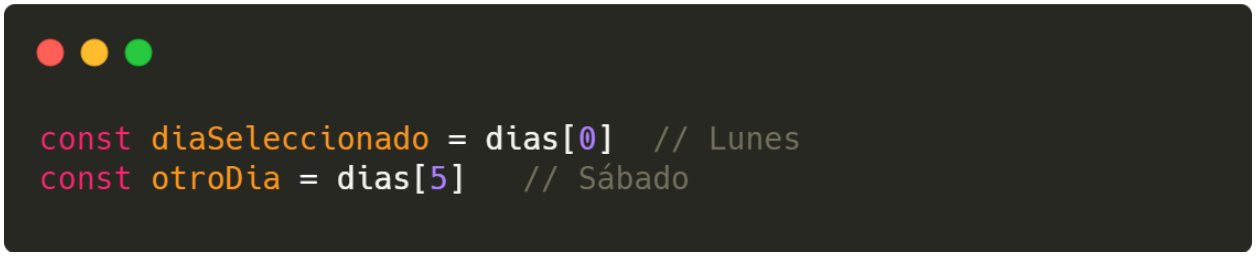
```
const dias = ['Lunes' , 'Martes' , 'Miércoles' , 'Jueves' , 'Viernes',
'Sábado' , 'Domingo']
```

Ahora, una única variable llamada días almacena todos los valores relacionados entre sí, en este caso los días de la semana. Para definir un array, se utilizan los **caracteres [y]** para delimitar su comienzo y su final y se utiliza el carácter , (coma) para separar sus elementos:

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

```
var nombre_array = [valor1, valor2, ..., valorN];
```

Una vez definido un array, es muy sencillo acceder a cada uno de sus elementos. Cada elemento se accede indicando su posición dentro del array. La única complicación, que es responsable de muchos errores cuando se empieza a programar, es que las posiciones de los elementos empiezan a contarse en el 0 y no en el 1:



```
const diaSeleccionado = dias[0]  // Lunes
const otroDia = dias[5]   // Sábado
```

En el ejemplo anterior, la primera instrucción quiere obtener el primer elemento del array. Para ello, se indica el nombre del array y entre corchetes la posición del elemento dentro del array. Como se ha comentado, las posiciones se empiezan a contar en el 0, por lo que el primer elemento ocupa la posición 0 y se accede a él mediante dias[0].

El valor dias[5] hace referencia al elemento que ocupa la sexta posición dentro del array dias. Como las posiciones empiezan a contarse en 0, la posición 5 hace referencia al sexto elemento, en este caso, el valor Sábado.

Diferencia entre var, let y const

let y **const** son dos formas de declarar variables en JavaScript introducidas en ES6 que reducen el ámbito de la variable a bloques (con var el ámbito era la función actual) y no admiten hoisting. Además, las variables declaradas con **const** no pueden ser reasignadas (aunque no significa que su valor sea inmutable, como veremos a continuación).

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Un bloque en JavaScript se puede entender como «lo que queda entre dos corchetes», ya sean definiciones de funciones o bloques **if, while, for** y loops similares. Si una variable es declarada con **let** en el ámbito global o en el de una función, la variable pertenecerá al ámbito global o al ámbito de la función respectivamente, de forma similar a como ocurría con **var.**

Ejemplos de var

```
var i = "global";
function foo() {
    i = "local";
    console.log(i); // local
}
foo();
console.log(i); // local
```

```
var i = "global";
function foo() {
    var i = "local"; // Otra variable local solo para esta función
    console.log(i); // local
}
foo();
console.log(i); // global
```

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Ejemplos de let

```
let i = 0;
function foo() {
    i = 1;
    let j = 2;
    if(true) {
        console.log(i); // 1
        console.log(j); // 2
    }
}
foo();
```

En este caso i es global (definida fuera de un bloque) y j es una variable local.

```
function foo() {
    let i = 0;
    if(true) {
        let i = 1; // Sería otra variable i solo para el bloque if
        console.log(i); // 1
    }
    console.log(i); // 0
}
foo();
```

Si declaramos una variable con let dentro un bloque que a su vez está dentro de una función, la variable pertenece solo a ese bloque.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Ejemplos de const

```
const i = 0;
i = 1; // TypeError: Assignment to constant variable
```

Error al reasignar el valor de una constante.

```
const user = { name: 'Juan' };
user.name = 'Manolo';
console.log(user.name); // Manolo
```

El objeto asignado a la constante sufre un cambio, pero la variable apunta siempre al mismo objeto, por eso no da error.

```
const user = 'Juan';
user = 'Manolo'; // TypeError: Assignment to constant variable
```

Error al reasignar el valor de una constante.

Resumen

var declara una variable de scope global o local para la función sin importar el ámbito de bloque.

let declara una variable de scope global, local para la función o de bloque. **Es reasignable.**

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

const declara una variable de scope global, local para la función o de bloque. **No es reasignable, pero es mutable.**

En general, **let** sería todo lo que se necesita dentro de un bloque, función o ámbito global. **const** sería para variables que no van sufrir una reasignación.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

4. Estructuras de control

Los programas que se pueden realizar utilizando solamente variables y operadores son una simple sucesión lineal de instrucciones básicas.

Sin embargo, no se pueden realizar programas que muestren un mensaje si el valor de una variable es igual a un valor determinado y no muestren el mensaje en el resto de casos. Tampoco se puede repetir de forma eficiente una misma instrucción, como por ejemplo sumar un determinado valor a todos los elementos de un array.

Para realizar este tipo de programas son necesarias las estructuras de control de flujo, que son instrucciones del tipo **"si se cumple esta condición, hazlo; si no se cumple, haz esto otro".** También existen instrucciones del tipo "repite esto mientras se cumpla esta condición".

Si se utilizan estructuras de control de flujo, los programas dejan de ser una sucesión lineal de instrucciones para convertirse en programas inteligentes que pueden tomar decisiones en función del valor de las variables.

Estructura if

La estructura más utilizada en JavaScript y en la mayoría de lenguajes de programación es la estructura if. Se emplea para tomar decisiones en función de una condición. Su definición formal es:

```
if(condición) {
   //bloque a ejecutar ...
}
```

Si la condición se cumple (es decir, si su valor es true) se ejecutan todas las instrucciones que se encuentran dentro de {...}. Si la condición no se cumple (es decir, si su valor es false) no se ejecuta ninguna instrucción contenida en {...} y el programa continúa ejecutando el resto de instrucciones del script.

Ejemplo:

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

```
const mostrarMensaje = true

if(mostrarMensaje) {
  alert('Hola Curso!')
}
```

Estructura if...else

En ocasiones, las decisiones que se deben realizar no son del tipo "si se cumple la condición, hazlo; si no se cumple, no hagas nada". Normalmente las condiciones suelen ser del tipo "si se cumple esta condición, hazlo; si no se cumple, haz esto otro".

Para este segundo tipo de decisiones, existe una variante de la estructura if llamada if...else. Su definición formal es la siguiente:

```
if(condición) {
      ... }
else {
      ... }
```

Si la condición **se cumple** (es decir, si su valor es true) se ejecutan todas las instrucciones que se encuentran dentro del if(). Si la condición **no se cumple** (es decir, si su valor es false) se ejecutan todas las instrucciones contenidas en else { }.

Ejemplo:

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

```
const edad = 18

if (edad >=18){
  alert ('Eres mayor de edad')
} else {
  alert ('Todavía eres menor de edad')
}
```

Si el valor de la variable edad es mayor o igual que el valor numérico 18, la condición del if() se cumple y por tanto, se ejecutan sus instrucciones y se muestra el mensaje "Eres mayor de edad". Sin embargo, cuando el valor de la variable edad no es igual o mayor que 18, la condición del if() no se cumple, por lo que automáticamente se ejecutan todas las instrucciones del **bloque else { }**. En este caso, se mostraría el mensaje "Todavía eres menor de edad".

Estructura switch

La estructura if...else se puede utilizar para realizar **comprobaciones múltiples** y tomar decisiones complejas. Sin embargo, si todas las condiciones dependen **siempre de la misma variable**, el código JavaScript resultante es demasiado redundante:

```
if(numero == 5) {
 ... }
 else if(numero == 8) {
 ... }
 else if(numero == 20) {
 ... }
 else { ...
}
```

En estos casos, la estructura switch es la más eficiente, ya que está especialmente diseñada para manejar de forma sencilla múltiples condiciones sobre la misma

variable. Su definición formal puede parecer compleja, aunque su uso es muy sencillo:

```
switch (valor) {
    case 1: document.write('uno');
      break;
    case 2: document.write('dos');
      break;
    case 3: document.write('tres');
      break;
    case 4: document.write('cuatro');
      break;
    case 5: document.write('cinco');
      break;
    default:document.write('debe ingresar un valor comprendido entre
            1 y 5.');
  }
```

La estructura switch se define mediante la palabra reservada switch seguida, entre paréntesis, del nombre de la variable que se va a utilizar en las comparaciones. Como es habitual, las instrucciones que forman parte del switch se encierran entre comillas { y }

Dentro del switch se definen todas las comparaciones que se quieren realizar sobre el valor de la variable. Cada comparación se indica mediante la palabra reservada case seguida del valor con el que se realiza la comparación. Si el valor de la variable utilizada por switch coincide con el valor indicado por case, **se ejecutan las instrucciones definidas dentro de ese case.**

Normalmente, después de las instrucciones de cada case se incluye la sentencia **break para terminar** la ejecución del switch, aunque no es obligatorio. Las comparaciones se realizan por orden, desde el primer case hasta el último, por lo que es muy importante el orden en el que se definen los case.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

¿Qué sucede si ningún valor de la variable del switch coincide con los valores definidos en los case? En este caso, se utiliza el valor default para indicar las instrucciones que se ejecutan en el caso en el que ningún case se cumpla para la variable indicada.

Aunque default es opcional, las estructuras switch suelen incluirlo para definir al menos un valor por defecto para alguna variable o para mostrar algún mensaje por pantalla.

Operador ternario

El operador ternario es una forma **simplificada** de los **bloques if... else** para asignar valor a una variable o decidir la ejecución de un bloque de código. Su sintaxis es

```
condicion ? <expresion_verdadero> : <expresion_falso>
```

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

```
const esMayor = edad >= 21 ? true : false;
const cuota = esMayor ? 500 : 350;

// es lo mismo que

if (edad >= 21) {
   esMayor = true;
} else {
   esMayor = false;
}

if (esMayor) {
    cuota = 500;
} else {
    cuota = 350;
}
```

En este caso, si el valor de la variable **edad** fuera mayor o igual a 21, el valor de la variable **esMayor** sería verdadero y en consecuencia el valor de la variable **cuota** sería de 500. Como podemos ver el operador ternario nos ahorra bastantes líneas de código en el uso de condicionales simples.

Cabe aclarar que también es posible utilizar otro operador ternario dentro de la expresión de verdadero o la de falso.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

5. Bucles

Estructura for

Las estructuras if y if...else no son muy eficientes cuando se desea ejecutar de forma **repetitiva una instrucción**. Por ejemplo, si se quiere mostrar un mensaje cinco veces, se podría pensar en utilizar el siguiente if:

```
const veces = 0

if(veces< 4){
  alert ('Mensaje')
  veces++
}
```

Se comprueba si la variable veces es menor que 4. Si se cumple, se entra dentro del if(), se muestra el mensaje y se incrementa el valor de la variable veces. Así se debería seguir ejecutando hasta mostrar el mensaje las cinco veces deseadas.

Sin embargo, el funcionamiento real del script anterior es muy diferente al deseado, ya que solamente se muestra una vez el mensaje por pantalla. La razón es que la ejecución de la estructura if() no se repite y la comprobación de la condición sólo se realiza una vez, independientemente de que dentro del if() se modifique el valor de la variable utilizada en la condición.

La estructura for permite realizar este tipo de repeticiones (también llamadas bucles) de una forma muy sencilla. No obstante, su definición formal no es tan sencilla como la de if():

```
for(inicialización; condición; actualización) {
    ... }
```

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

La idea del funcionamiento de un bucle for es la siguiente: "mientras la condición indicada se siga cumpliendo, repite la ejecución de las instrucciones definidas dentro del for. Además, después de cada repetición, actualiza el valor de las variables que se utilizan en la condición".

- La "**inicialización**" es la zona en la que se establecen los valores iniciales de las variables que controlan la repetición.
- La "**condición**" es el único elemento que decide si continua o se detiene la repetición.
- La "**actualización**" es el nuevo valor que se asigna después de cada repetición a las variables que controlan la repetición.

```
const mensaje = 'Hola, estoy dentro de un bucle'

for(let i=0; i<5; i++){
  alert(mensaje)
}
```

Estructura while

La estructura while permite crear bucles que se ejecutan ninguna o más veces, dependiendo de la **condición indicada**. Su definición formal es:

```
while(condición) {
  ...
 }
```

El funcionamiento del bucle while se resume en: "**mientras se cumpla la condición** indicada, repite indefinidamente las instrucciones incluidas dentro del bucle".

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Si la condición no se cumple ni siquiera la primera vez, el bucle no se ejecuta. Si la condición se cumple, se ejecutan las instrucciones una vez y se vuelve a comprobar la condición. Si se sigue cumpliendo la condición, se vuelve a ejecutar el bucle y así se continúa hasta que la condición no se cumpla.

Evidentemente, las variables que controlan la condición deben modificarse dentro del propio bucle, ya que de otra forma, la condición se cumpliría siempre y el bucle while se repetirá indefinidamente.

El siguiente ejemplo utiliza el bucle while para sumar todos los números menores o iguales que otro número:

```
let x = 1;

while(x <= 100){
  document.write(`${x} <br>`);
  x++; // o  x += 1; o x = x + 1;
}
```

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Bibliografía utilizada y sugerida

Artículos de revista en formato electrónico:

CYBMETA. Disponible desde la URL:
https://cybmeta.com/var-let-y-const-en-javascript

Libros y otros manuscritos:

Eguíluz Pérez, Javier. Introducción a JavaScript.España: www.librosweb.es 2008.