clase n°10
JavaScript

<animate a/> programar
revøluciøn* digital_

utilizando window.alert ()

- El método de ***alert*** muestra un cuadro de alerta visual en la pantalla. El parámetro del método de alerta se muestra al usuario en texto sin formato:

```
window.alert(message);
```

- Debido a que la ***window*** es el objeto global, puede llamar también usar la siguiente forma abreviada:

```
alert(message);
```

Probamos lo siguiente:

```
> alert(" Cual es tu nombre:?");
< undefined
```

- Usar los ***logs***

  A la hora de desarrollar podemos enviar avisos a la consola directamente desde nuestra aplicación. Para ello usaremos los siguientes comandos:

  - `console.log("texto"): para un mensaje normal`
  - `console.warn("texto"): para un mensaje de aviso`
  - `console.error("texto"): para un mensaje de error`

```
> console.warn("Boca");
▶ Boca
< undefined
> console.error("La Bombonera");
▶ La Bombonera
< undefined
> console.log("Lo Mejor");
  Lo Mejor
< undefined
```

utilizando window.prompt ()

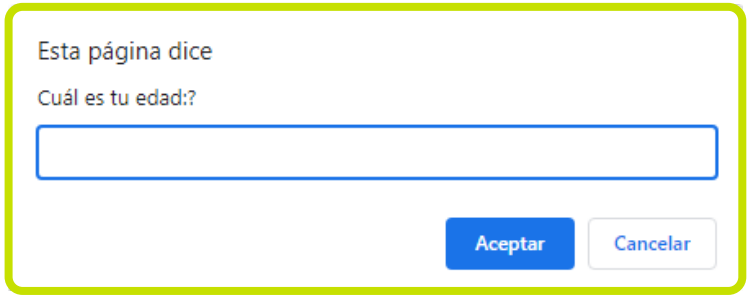
- Una forma fácil de obtener una entrada de un usuario es mediante el método ***prompt() .***

Sintaxis:

```
prompt(text, [default]);
```

- ***texto :*** el texto que se muestra en el cuadro de solicitud.
- ***predeterminado :*** un valor predeterminado para el campo de entrada (opcional).

```
> var age = prompt("Cuál es tu edad:?");
  console.log(age);
```



- Ingresamos la edad y aceptamos

- Agreguemos un valor predeterminado y veamos que pasa.

- Luego probemos nuevamente un ***console.log***

Si el usuario hace clic en el botón Aceptar , se devuelve el valor de entrada. De lo contrario, el método devuelve null.
El valor de retorno del prompt siempre es una cadena, a menos que el usuario haga clic en Cancelar, en cuyo caso devuelve un null.
Safari es una excepción porque cuando el usuario hace clic en Cancelar, la función devuelve una cadena vacía.
Desde allí, puede convertir el valor de retorno a otro tipo, como un entero .

utilizando window.confirm ()

- El método window.confirm() muestra un diálogo modal con un mensaje opcional y dos botones, Aceptar y Cancelar.

Ahora, tomemos el siguiente ejemplo:

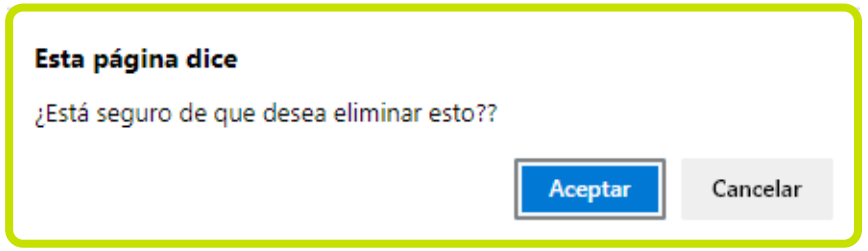
```
result = window.confirm(message);
```

- Aquí, el mensaje es la cadena opcional que se mostrará en el cuadro de diálogo y el resultado es un valor booleano que indica si se seleccionó Aceptar o Cancelar (verdadero significa OK).

window.confirm() se usa normalmente para solicitar la confirmación del usuario antes de realizar una operación peligrosa como eliminar algo en un Panel de control:

- Si lo necesita para su uso posterior, simplemente puede almacenar el resultado de la interacción del usuario en una variable:

var deleteConfirm = window.confirm("¿Está seguro de que desea eliminar esto??");



Los cuadros de diálogo son ventanas modales: impiden que el usuario acceda al resto de la interfaz del programa hasta que se cierre el cuadro de diálogo. Por este motivo, no debe abusar de ninguna función que cree un cuadro de diálogos.

utilizando var

- Sirve para declarar una variable.

  Una variable es una especie de “contenedor” al que le ponemos un nombre y que nos sirve para guardar algún valor.
  Este valor puede ser cualquier tipo de dato que soporte el lenguaje de programación (números, strings, booleans, arrays, objetos).
  Dependiendo del lenguaje de programación la forma de declarar una variable varía, en el caso de JavaScript tenemos
  que primero usar una de las tres palabras reservadas seguido del nombre de la variable y luego utilizamos el signo “igual”
  para asignarle un valor.

```
// Declarando varios tipos de variable
var name = "Pedro" // String
var students = 40 // Number
var countries = ["Venezuela", "Colombia", "Perú"] // Array
var grades = { Carlos: "B", Paula: "A" } // Object
var success = true // Boolean
var nothing = null // null
```

  Si quiero ver que valor está contenido en una variable puedo utilizar conole.log para que se muestre en consola.

```
console.log(students);
// resultado 40
```

- Nombré todas las variables en inglés porque quise. Realmente puedes nombrarlas en español si quieres (evitando las “ñ”
  y las tildes para ahorrarte dolores de cabeza). El idioma que elijas no es importante. Lo realmente importante es seguir
  algunas reglas al momento de nombrarlas:

- Las variables pueden contener letras (a-z), números (0-9), el símbolo de dólar o peso ($) y underscores (_)
- Si bien el nombre puede contener números, la misma no puede empezar con ningún número.
- No pueden haber espacios en blancos.
- No pueden ser ninguna de las palabras reservadas de JavaScript (reserved keywords).
- Los nombres de variable son case sensitive, es decir que se diferencian las mayúsculas de las minúsculas (“esto” no es igual a “Esto” que no es igual a “ESTO”).

¿Qué es el Scope?

Es el alcance que tiene la variable.
Se refiere dónde está disponible esa variable para JavaScript.

Existen dos tipos de scope: ***local y global.***
Las variables globales son las declaradas fuera de los bloques de código, en cambio las locales son las que son declaradas dentro de bloques de código. Ejemplos:

```
// Declarando variable global
var vaso = "lleno"

function beber() {
    // Declarando una variable local
    var vaso = "vacío"
    console.log(vaso)
}

console.log(vaso)
beber()
console.log(vaso)

//  Resultado:
//  lleno
//  vacío
//  lleno
```

```
"Buenas Tardes" // String
'Buenas Noches' // String

1000 // number
-2.3 // number
// boolean
true
false
//array
['Diego','Armando','Sergio']
[1,2,3]
[true, false,true, false]

//object
{
  'username': "Diego",
  "puntos": 30,
  "Partidojugado": 2,
  "profesional": false
}
{
  "username": "Sergio",
  "puntos": 60,
  "Partidojugado": 3,
  "profesional": true
}
```

Dentro de js probamos el console.log

Lo haremos con las distintas variables.

- console.log("cadena");
- console.log(1000);
- console.log(true);
- console.log([1,2,3]);
- console.log([1,2,3,4]);
- console.log({"Usuario":"Diego","Puntos":10});
- var nombre = "Diego Armando";
  console.log(nombre);

También podría haber declarado con ***var*** como ***let*** o ***const***

let te permite declarar variables limitando su alcance (scope) al bloque, declaración, o expresión donde se está usando. a diferencia de la palabra clave var la cual define una variable global o local en una función sin importar el ámbito del bloque

Probemos:

```
const PI = 3.1415;
PI=100;
console.log(PI);
```

```html
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Test page</title>
  </head>
  <body>
    Inline script (option 1):
    <script>
      // YOUR CODE HERE
    </script>
    External script (option 2):
    <script src="your-code-file.js"></script>
  </body>
</html>
```

utilizando la API DOM

DOM significa ***D*** ocumento ***O*** bject ***M*** odel. Es una representación orientada a objetos de documentos estructurados como XML y HTML .

consola on-line.

JS Bin - Collaborative JavaScript Debugging

http://jsbin.com/fuzijox/edit?html,js,console,output

HTML - 1script.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
<title>JS Bin</title>
</head>
<body>

<p id="paragraph"></p>
<p> Probando </p>
<script type="application/javascript" src="1script.js">

</script>
</body>
</html>
```

La siguiente etiqueta HTML:

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

Aquí le damos el nombre paragraph al id

JavaScript - 1script.js

```
document.getElementById("paragraph").textContent = "Hello, World";
```