



Curso de Front-End

Clase 08



Agenda de la clase



Agenda



- Repaso.
- Booleanos.
- Sentencias, Expresiones y Operadores.
- `Math`.
- Funciones.
- *Arrays*.
- Ejercicios.



Repaso



¿Qué es JavaScript?

- Es el lenguaje de programación de la web.
- Nació en 1995 (desarrollado por Netscape).  Netscape®
- Interactuamos con JavaScript todos los días
En Gmail, Facebook, Google Search, etc. Es difícil encontrar un sitio que no use JS.
- JavaScript  ≠ Java 
- Documentación:
 - MDN: <https://developer.mozilla.org/es/docs/Web/JavaScript>
 - W3Schools: <http://www.w3schools.com/js/default.asp>



Tipos de datos

Hay varios tipos de datos en JS. Hablamos sobre `String` y `Number`:

- `String`
Conjunto de caracteres encerrados por comillas (simples o dobles).
- `Number`
Pueden ser números enteros o decimales.

`typeof` es una funcionalidad que brinda JS para consultar el tipo de “algo”.

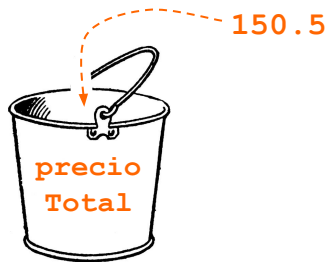


Variables

Las variables son **contenedores que permiten guardar información** (valores) y poder usarla más tarde.



Variables



Ejemplos:

```
var precioA = 100;
```

```
var precioB = 50.5;
```

```
var precioTotal = precioA + precioB;
```

```
var mensaje1 = "Hola" + " " + "Mundo";
```

```
var mensaje2 = "Tengo " + 30 + " años"; // ¿Esto se puede hacer?
```




Tips (1/4)

- Usar mucho la **consola**.

Si no recuerdan qué resultado se obtiene al hacer $15 \% 4$, pruébenlo en la consola. Si algo no les anda, busquen si no aparece algún mensaje de error en la consola.

- Consultar la documentación:
 - **MDN** <https://developer.mozilla.org/es/docs/Web/JavaScript>
 - W3Schools: <http://www.w3schools.com/js/default.asp>
- Usar Google y **Stack Overflow**.
- Preguntar, preguntar y preguntar.

Tips (2/4)

¡Practiquen mucho!



¿Cómo se aprende
a tocar la guitarra?

👉 Sugerimos que practiquen aprox. **6 horas semanales**.

⚠️ Es la única forma de realmente aprovechar el curso.

😓 Además, a medida que transcurre el curso, la diferencia entre los alumnos que practican y los que no, empieza a ser cada vez más grande.



Tips (3/4)

¡Sean pacientes!

Recuerden que aprender a programar suele ser **difícil al principio**.

Es muy importante que cuando se enfrenten a un ejercicio, intenten resolverlo durante un buen rato. A veces van a estar **horas** sin poder escribir una línea de código. Van a pasar horas leyendo respuestas en Stack Overflow, leyendo documentaciones o mirando videos sobre programación en YouTube, tratando de encontrar una solución. No se desanimen, **así es la vida del programador**; nos pasa a todos. Y si después de 8 horas seguidas siguen sin poder resolver el problema, tengan certeza de que todo ese esfuerzo no fue en vano; todo lo contrario.

Se aprende muchísimo intentando, probando y corrigiendo errores. **Probar** y **equivocarse** es fundamental, es parte del aprendizaje.



Tips (4/4)

¡No falten a clase!

Es difícil re-engancharse si faltan, aunque se hagan repasos.

👉 Los **certificados de asistencia** se dan a los alumnos que lo soliciten (son la minoría) y que además hayan asistido a un **90%** de las clases.



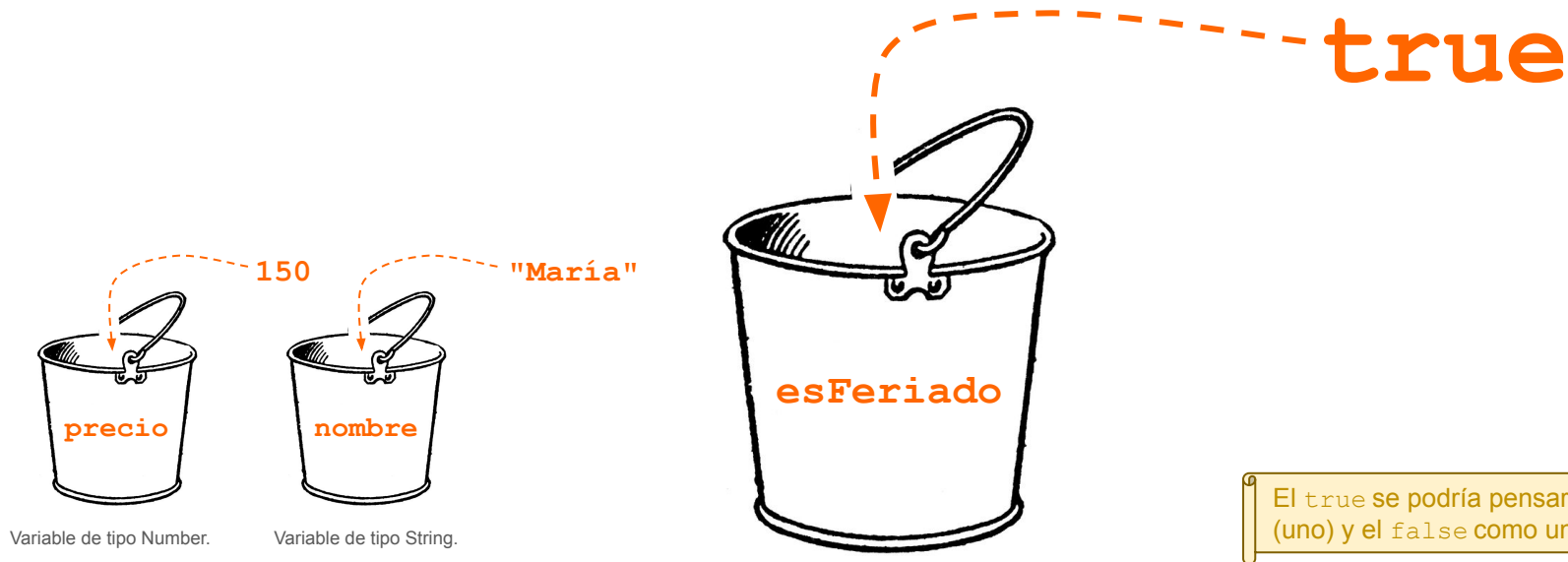
Booleanos



Booleanos (1/2)

Boolean es otro tipo de dato (como lo es String y Number).

Un dato booleano sólo puede tomar 2 valores: **true** o **false**.



El **true** se podría pensar como un 1 (uno) y el **false** como un 0 (cero).



Booleanos (2/2)

Ejemplos:

```
var mariaEsChilena = true;
```

```
var hoyEsFeriado = false;
```

```
!mariaEsChilena; // ¿Qué hace esto?
```



Sentencias, Expresiones y Operadores



Sentencias, Expresiones y Operadores

- **sentencia**: es una instrucción que se le da a la computadora.

Ejemplo: `var precio = 100;`

- **expresión**: cualquier “cosa” que produzca un valor.

Ejemplo: `5 + 9;`

- **operador**: Un operador es básicamente un símbolo matemático que puede actuar sobre uno o dos valores (o variables) y producir un resultado.

Ejemplo: `+, -, *, /, <, >, ==, >=, <=, ===, !`



Quizzes



Quizzes (1/2)

¿Qué retornan las siguientes expresiones?

```
5 * ""; // Nota: "" es un string vacío.
```

```
5 + "";
```

```
5 / "";
```

```
5 + true;
```

```
5 + "3";
```

```
5 - "3";
```

```
"5" - "4";
```

⚠ **Nota:** No es importante saberse todo esto de memoria. De hecho, estas son expresiones con las que probablemente nunca se tengan que enfrentar. Lo interesante es ver ciertas particularidades que tiene JavaScript vs. otros lenguajes.



Quizzes (2/2)

¿Funciona esto? ¿Tiene sentido?

```
var min, max;  
var promedio = (min + max) / 2;
```

¿Funciona esto?

```
var edad = 26;  
console.log("Tengo " + edad + " años.");
```



Algunos buenos hábitos



Buenos hábitos

- Usar espacios y saltos de línea adecuadamente.
- Recordar usar los “;” al final de cada sentencia.
- Usar nombres **descriptivos** para las variables, preferentemente en **inglés**.

MALA PRÁCTICA

```
var x=10,y=5
console.log(x+y)
var ciudad='Montevideo'
window.alert("Hola "+ciudad+".")
```

BUENA PRÁCTICA

```
var ageChild1 = 10;
var ageChild2 = 5;
console.log(ageChild1 + ageChild2);
var city = "Montevideo";
window.alert("Hola " + city + ".");
```



Math



Math

Lo veremos más adelante.

Math es un *objeto* de JavaScript que permite realizar ciertas operaciones matemáticas como ser:

- `Math.min` – Calcular el mínimo entre dos o más números.
- `Math.max` – Calcular el máximo entre dos o más números.
- `Math.random` – Generar un número aleatorio entre 0 y 1.
- `Math.round` – Hacer un redondeo de un número.

Ver [documentación](#).

Recuerden utilizar la `M` de `Math` en mayúscula, ya que JavaScript es *case sensitive*.

```
Math.min(2, 4);      // Retorna 2.
Math.max(781, 49);   // Retorna 781.
Math.random();       // Retorna un número aleatorio entre 0 y 1.
Math.round(34.8);    // Retorna 35.
```




Funciones



Ya hemos visto algunas funciones...

```
alert("¡Hola alumnos de Front-End!");
```

```
console.log(5 + 10);
```

```
Math.random(); // Retorna un número cualquiera entre 0 y 1.
```

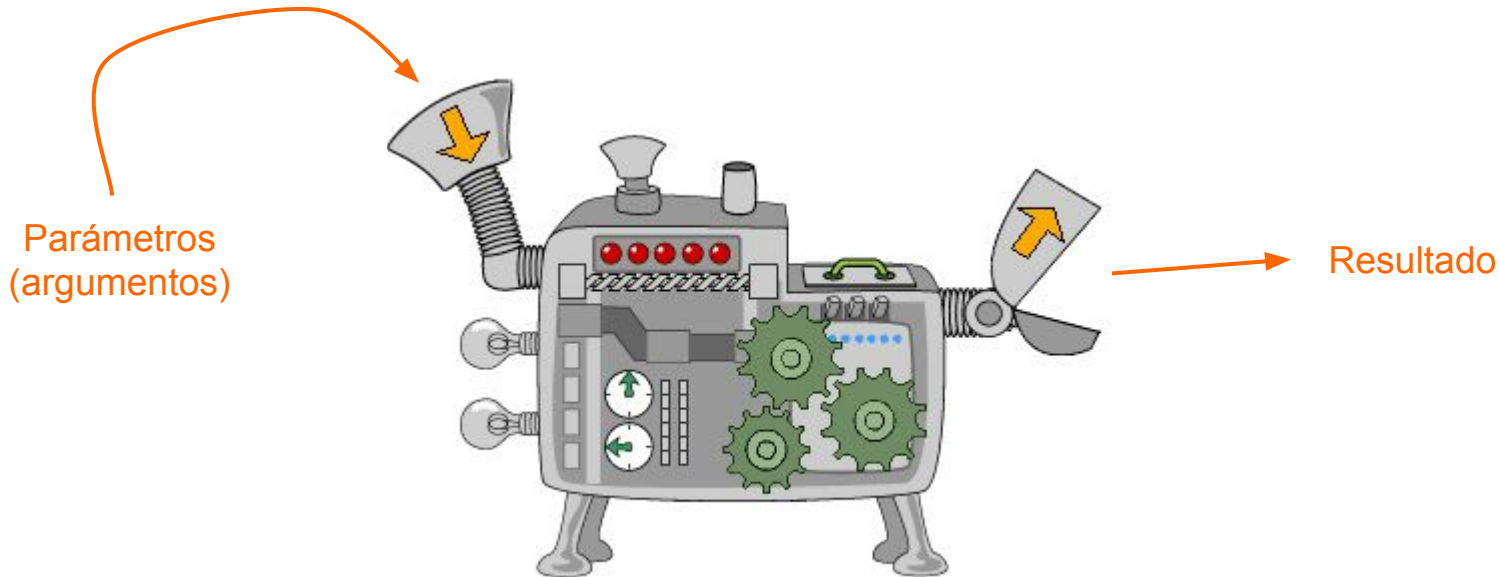
```
Math.max(3, 55); // Retorna 55.
```

```
Math.min(3, 55); // Retorna 3.
```

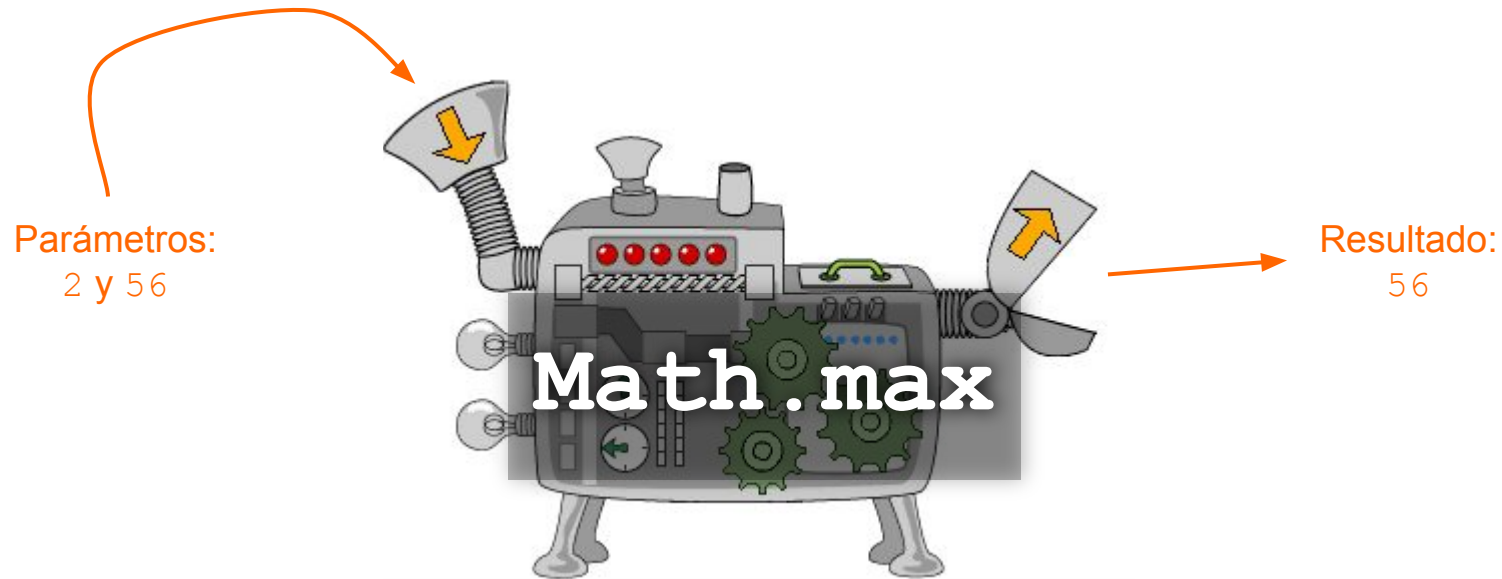
¿Qué tienen en común
las funciones?

¿Qué es una función?

Pueden pensar en una función como en una máquina que recibe materia prima (**parámetros** o **argumentos**), los procesa y luego retorna un producto final (**resultado**).

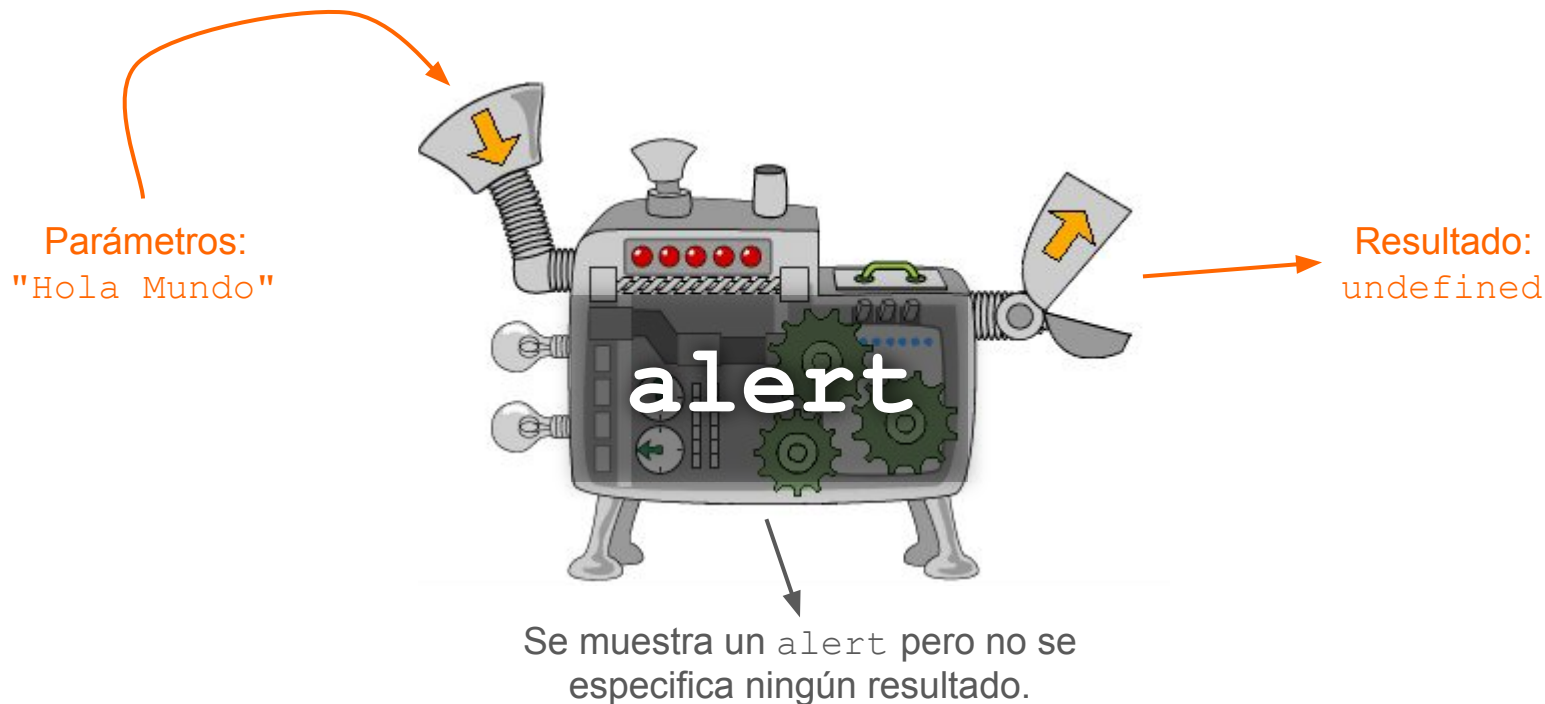


¿Qué es una función? – Ejemplo: `max`

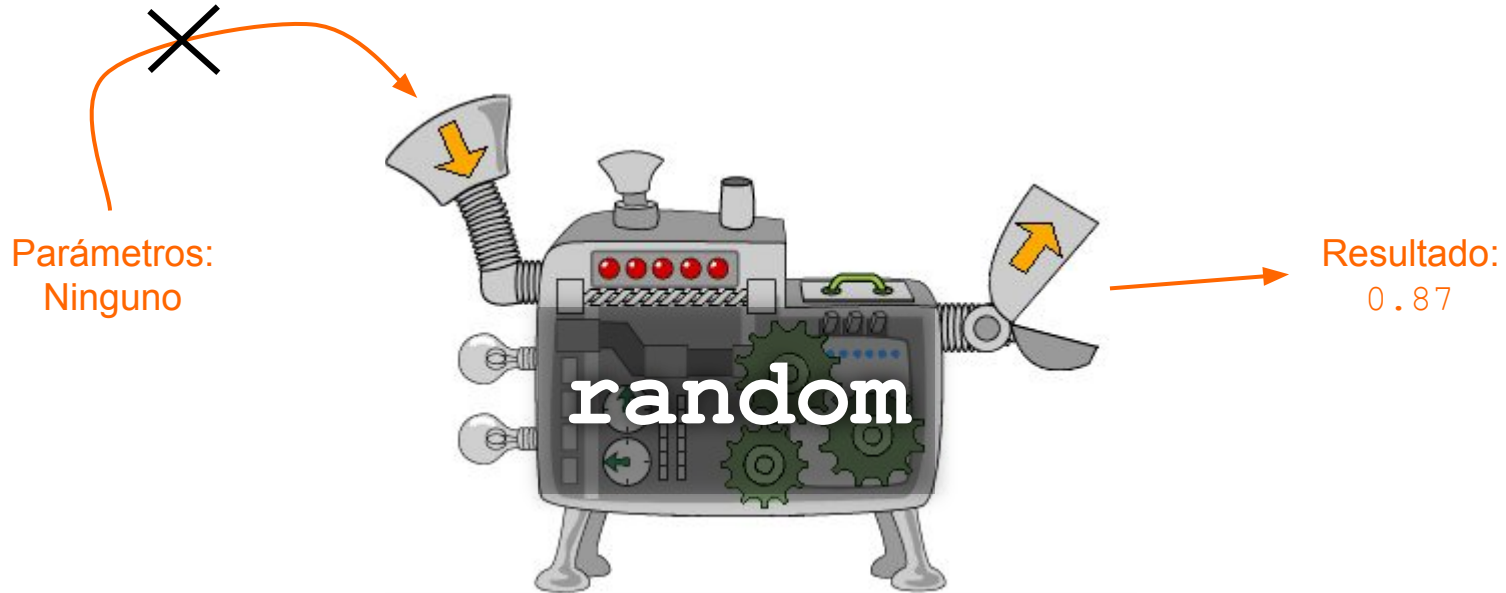




¿Qué es una función? – Ejemplo: `alert`



¿Qué es una función? – Ejemplo: `random`





¿Qué es una función? (en JavaScript)

- Es un **bloque de código (re-utilizable) que cumple determinada tarea.**

Al igual que un programa, una función está compuesta por una secuencia de sentencias llamadas “cuerpo” de la función.

- Re-utilizable: La función se puede usar y re-usar sin límites.
- Cuando se utiliza una función, se dice que se la **llama** (*call a function*).
También se puede decir que se “invoca” una función.

- Toda función retorna (devuelve) un **resultado**.

Si no se especifica el resultado, se retorna (casi siempre): `undefined`. Por eso en la consola muchas veces aparece escrito el texto “undefined”, aparentemente sin sentido.

- Opcionalmente, una función puede **recibir parámetros (argumentos)** cuando se la llama.

Nota técnica: una función en JS es un *objeto*. Lo veremos más adelante.

¿Cómo se **declara** una función?

👉 Aprovechar **VSC** al máximo. Usar el auto-complete al crear una `function`.



```
function nombreDeLaFuncion(arg1, arg2, etc) {  
  /**  
   * Aquí se coloca el "cuerpo" de la función.  
   * Eventualmente, la función puede retornar un  
   * resultado. En ese caso, se utiliza la palabra  
   * `return` para definirlo.  
   */  
}
```

Esta forma de definir una función se llama "Function [Declaration](#)".



¿Cómo se **declara** una función? – Ejemplo

```
function calcularSueldoLiquido(sueldoBruto) {  
    /**  
     * Nota: la siguiente es una simplificación del  
     * cálculo del sueldo líquido de un empleado.  
     */  
    var descuento = 0.80;  
    return sueldoBruto * descuento;  
}
```

Esta forma de definir una función se llama "Function [Declaration](#)".



¿Cómo se **llama** a una función? – Ejemplo

```
calcularSueldoLiquido(30000); // Retorna 24000.
```

```
calcularSueldoLiquido(65000); // Retorna 52000.
```

```
calcularSueldoLiquido(100000); // Retorna 80000.
```



Llamar a una función antes de ser definida

```
alert("El sueldo líquido es: " + calcularSueldoLiquido(30000));  
  
// ...más código...  
  
function calcularSueldoLiquido(sueldoBruto) {  
    var descuento = 0.80;  
    return sueldoBruto * descuento;  
}
```

Esto se puede hacer siempre y cuando la función haya sido definida usando "Function [Declaration](#)".



Otra forma de declarar una función

```
var calcularSueltoLiquido = function(sueltoBruto) {  
    var descuento = 0.80;  
    return sueldoBruto * descuento;  
}  
  
/**  
 * Al hacerlo de esta forma, siempre se debe llamar a la  
 * función LUEGO de haber sido declarada.  
 */  
alert(calcularSueltoLiquido(30000));
```

Esta forma de definir una función se llama “Function [Expression](#)”.

⚠ Por el momento, esta forma de definir funciones **no** será de mucha utilidad. La usaremos en próximas clases.



Evitar que una función afecte variables externas

```
var temperaturaEnCelsius;
```

La función modifica una variable externa. Es una mala práctica y hay que tratar de evitarlo.

```
function toCelsius(fahrenheit) {
```

```
    temperaturaEnCelsius = (5/9) * (fahrenheit-32);
```

```
}
```



Elección de nombres para los parámetros

Al elegir nombres para los parámetros de una función:

- Usar nombres que tengan algún significado.
- No usar: `arg1`, `arg2`, `arg3`, `x`, `y`, `z`...

Mala práctica:

```
function toCelsius(arg) {  
    return (5/9) * (arg-32);  
}
```

Buena práctica:

```
function toCelsius(fahrenheit) {  
    return (5/9) * (fahrenheit-32);  
}
```



Ejercicios de Funciones



Ejercicios

1. Definir una función llamada `duplicar` que reciba como parámetro un número cualquiera y retorne su doble.
2. Definir una función llamada `mostrarNombre` que reciba como parámetro dos strings cualesquiera y despliegue un `alert` conteniendo el texto:
“Tu nombre es María y tu apellido es Pérez”
(suponiendo que los argumentos ingresados fueron “María” y “Pérez”).
3. Definir una función llamada `numeroAleatorio` que retorne un número cualquiera entre 1 y 100. Se puede usar la función `Math.random` (ya definida en JavaScript).



Arrays



Problema

Supongamos que tenemos un programa donde necesitamos **almacenar un conjunto** de marcas de autos.

Con lo que hemos visto hasta ahora, podríamos hacer lo siguiente:

```
var marca1 = "BMW";  
var marca2 = "Peugeot";  
var marca3 = "Chevrolet";  
var marca4 = "Subaru";
```

Esto no está mal, pero es poco eficiente.

Por ejemplo: ¿Cómo harían para averiguar cuántas marcas de auto hay almacenadas en el programa?

¿Cómo harían para averiguar si la marca “Fiat” está en el programa?



Solución

La solución es crear una estructura llamada **Array**:

```
var marcas = ["BMW", "Peugeot", "Chevrolet", "Subaru"];
```

A partir de ahora será más fácil responder preguntas como:

- ¿Cuántas marcas hay almacenadas en el conjunto?
- ¿La marca “Fiat” está en el conjunto?



¿Qué es un *Array*? (en JavaScript)

- Es una **estructura de datos** que permite almacenar un **conjunto** de valores en formato lista. Se lo puede pensar como una “lista de elementos”.
- En lugar de manipular los valores por separado, se manipula el conjunto.
- El largo de los *arrays* es variable y pueden contener elementos repetidos.
- Los valores contenidos en un *array* no tienen por qué ser del mismo tipo. Pueden ser *strings*, *numbers*, *booleans*, funciones, otros *arrays*, etc (todo mezclado). De todas maneras, lo más usual es que los elementos de un *array* sean del mismo tipo.
- En español se les llama “Arreglos” (o incluso “Vectores”).

Nota técnica: un *array* en JS es un *objeto*. Lo veremos más adelante. Ver [más información](#).



¿Cómo se crea un *Array*?

Método 1:

Se utilizan corchetes `[]` y se pasan los valores separados por comas `", "`.

```
var marcas = ["BMW", "Peugeot", "Chevrolet", "Subaru", "Nissan"];
```

Método 2:

Se utilizan las palabras `new` y `Array`, y se pasan los valores separados por comas `", "`.

```
var marcas = new Array("BMW", "Peugeot", "Chevrolet", "Subaru", "Nissan");
```

Ambos métodos son equivalentes, pero el más usado y simple es el primero.



Índices en un *Array*

A cada elemento dentro del *array* le corresponde un índice.

```
var marcas = ["BMW", "Peugeot", "Chevrolet", "Subaru", "Nissan"];
```



Nota: tener en cuenta que el índice del *array* comienza en 0, no en 1.

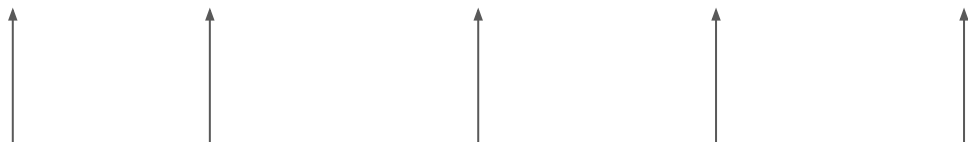
Esto varía según cada lenguaje. Los que comienzan en 0 suelen conocerse como *zero-index array*.



Acceder a un elemento de un *Array*

```
var marcas = ["BMW", "Peugeot", "Chevrolet", "Subaru", "Nissan"];
```

Índice: 0 1 2 3 4



```
var marcaPreferida = marcas[2];
```

¿Cuál es el valor de `marcaPreferida`?



Agregar un elemento a un *Array*

```
var marcas = ["BMW", "Peugeot", "Chevrolet", "Subaru", "Nissan"];
```

Índice: 0 1 2 3 4



```
marcas[5] = "Volvo";
```

¿Qué hubiese pasado si se hacía: `marcas[1] = "Volvo";`?



Largo de un *Array*

```
var marcas = ["BMW", "Peugeot", "Chevrolet", "Subaru", "Nissan"];
```

Índice: 0 1 2 3 4



```
console.log(marcas.length);
```

¿Cuál es el valor de la propiedad `length`?



Métodos de un *Array*

```
var marcas = ["BMW", "Peugeot", "Chevrolet", "Subaru", "Nissan"];
```

Todo *array* en JavaScript viene con una serie de métodos (**funciones**) muy útiles. Ejemplos:

```
marcas.push("Fiat"); // Agrega "Fiat" al final del array (sin especificar el índice).  
marcas.pop(); // Elimina el último elemento del array.  
marcas.shift(); // Elimina el primer elemento del array.  
marcas.unshift("VW"); // Agrega "VW" al inicio del array (mueve los otros elementos).  
marcas.splice(1,2); // Elimina 2 elementos del array, desde la posición 1.  
marcas.toString(); // Retorna un string con los elementos del array separados por comas.  
marcas.join(" - "); // Similar a toString, pero permite especificar separador.
```

Documentación: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#Methods_2



Ejercicio de *Arrays*



Ejercicio de *Arrays*

1. Abrir la consola de Google Chrome.
2. Crear un *array* vacío llamado `marcas`.
3. Insertar las marcas “VW”, “Audi”, “Volvo” y “Fiat” usando el método `push`.
4. Quitar “VW” de la lista.
5. Quitar “Volvo” de la lista.
6. Insertar la marca “Kia” en la primera posición.

Al terminar, verificar que el *array* resultante sea: `["Kia", "Audi", "Fiat"]`.