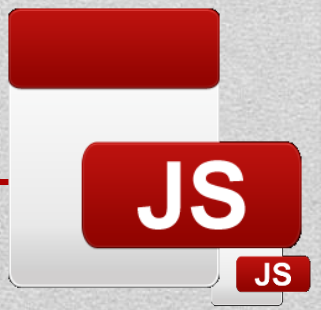
Lenguajes Interpretados en el Cliente

Sentencias repetitivas y matrices

Clase 3



Objetivos

- Mostrar el funcionamiento de las sentencias de control repetitivas, ciclos o lazos.
- Dominar el funcionamientos de cada una de las estructuras de control repetitivas.
- Aplicar las sentencias repetitivas en la solución de problemas prácticos.
- Utilizar sentencias de control de flujo de lazos (*break* y *continue*).
- Aprender a utilizar la sentencia repetitiva para objetos.

Objetivos

- Comprender el funcionamiento de los tipos de datos conocidos como ***arrays*** (arreglos o matrices).
- Sustituir el uso de múltiples variables por el uso de arreglos en los casos en que sea conveniente.
- Adquirir dominio en la forma en que deben utilizarse los ***arrays*** en un script.
- Hacer un buen uso de las sentencias repetitivas ***arrays*** con para mejorar el rendimiento.
- Aplicar el uso de ***arrays*** asociativos en la solución de problemas.
- Utilizar funciones de JavaScript para manipulación de ***arrays***.

Contenido

1. Sentencias repetitivas.
2. Sentencia ***while***.
3. Sentencia ***do-while***.
4. Sentencia ***for***.
5. Sentencia ***for-in***.
6. Sentencias de control de ciclo (***break*** y ***continue***).
7. Sentencia ***with***.
8. Definición de matriz o arreglo
9. Aspectos básicos sobre matrices en JS.

Contenido

1. Declaración de matrices en JavaScript.
2. Asignación de los elementos de una matriz.
3. Acceso a los elementos de una matriz.
4. Añadir o modificar elementos en una matriz.
5. Eliminar elementos en una matriz.
6. Obtener el tamaño de una matriz o arreglo.
7. Matrices asociativas.
8. Matrices multidimensionales.
9. Funciones para trabajar con matrices.

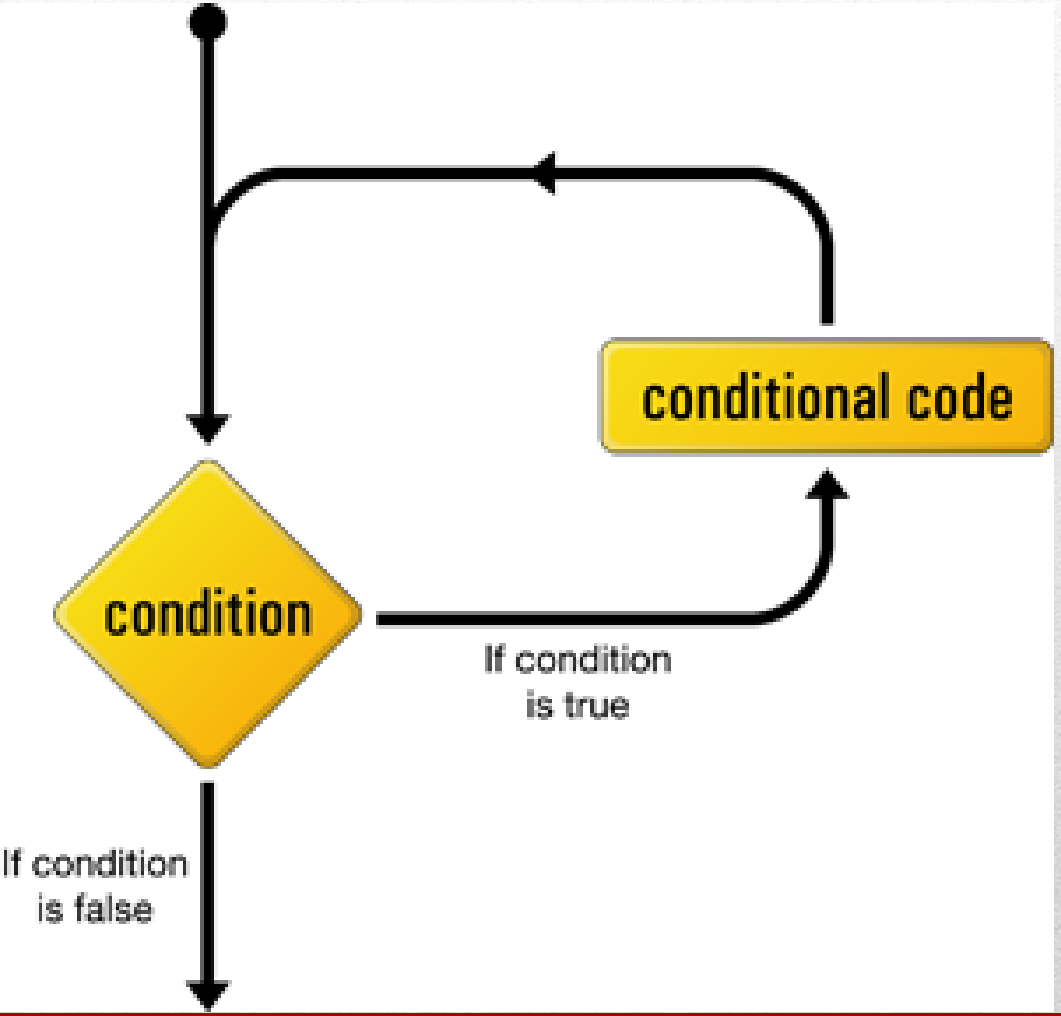
Sentencias repetitivas

- Son estructuras de control que permiten repetir un bloque de instrucciones una o varias veces hasta que una condición de paro se cumpla y lo detenga.
- JavaScript proporciona las estructuras repetitivas tradicionales, como el ***while***, ***do-while*** y ***for***.
- Además, cuenta con la sentencia repetitiva ***for-in*** utilizada con objetos.

Sentencia *while*

- Es el ciclo, lazo o bucle más básico de JavaScript cuyo propósito es ejecutar una instrucción o un bloque de instrucciones una o más veces mientras una expresión condicional sea evaluada como verdadera
- El lazo termina cuando la expresión condicional es evaluada como falsa o cuando se encuentre una instrucción ***break*** dentro del bloque de instrucciones; lo que suceda primero.
- Un lazo o bucle ***while*** puede no ejecutarse ni una vez si la condición es evaluada falsa desde la primera vez.

Sintaxis del *while*



Sintaxis del *while*

```
while (condicion) {
    //bloque de instrucciones
}
```

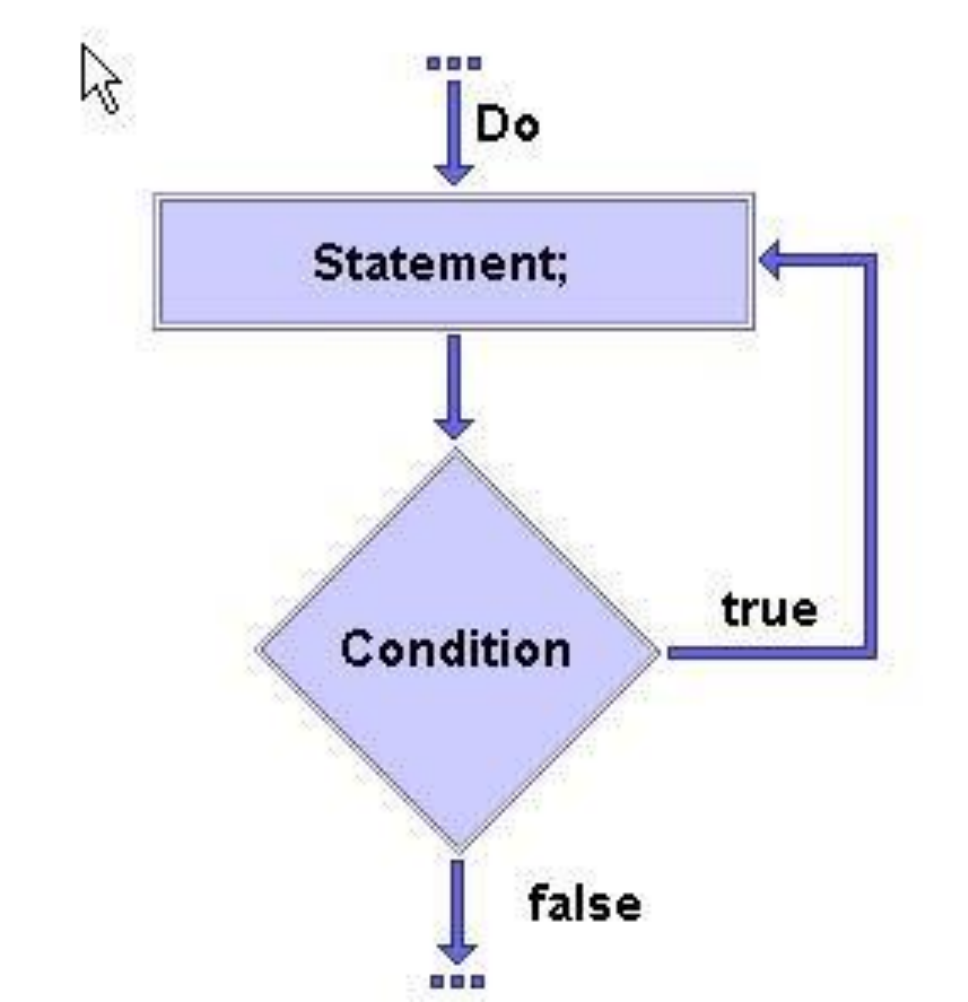
Ejemplo:

```
var i=0,count = 0;
while (i<10){
    count++;
    i+=2;
}
document.write(count);
```

Sentencia *do-while*

- Es muy similar al ciclo ***while***, con la diferencia que en este lazo primero se ejecutan las instrucciones del bloque y por último se evalúa la condición en cada iteración.
- Esto implica que el bucle siempre se ejecutará, al menos una vez, independientemente del resultado de la condición.
- Si al evaluar la condición resulta verdadera, entonces el bucle se sigue ejecutando. El bucle se detiene si la evaluación de la condición resulta falsa.

Sintaxis del do-while



Sintaxis del do-while

```
do {
    //Bloque de instrucciones
}while(condicion);
```

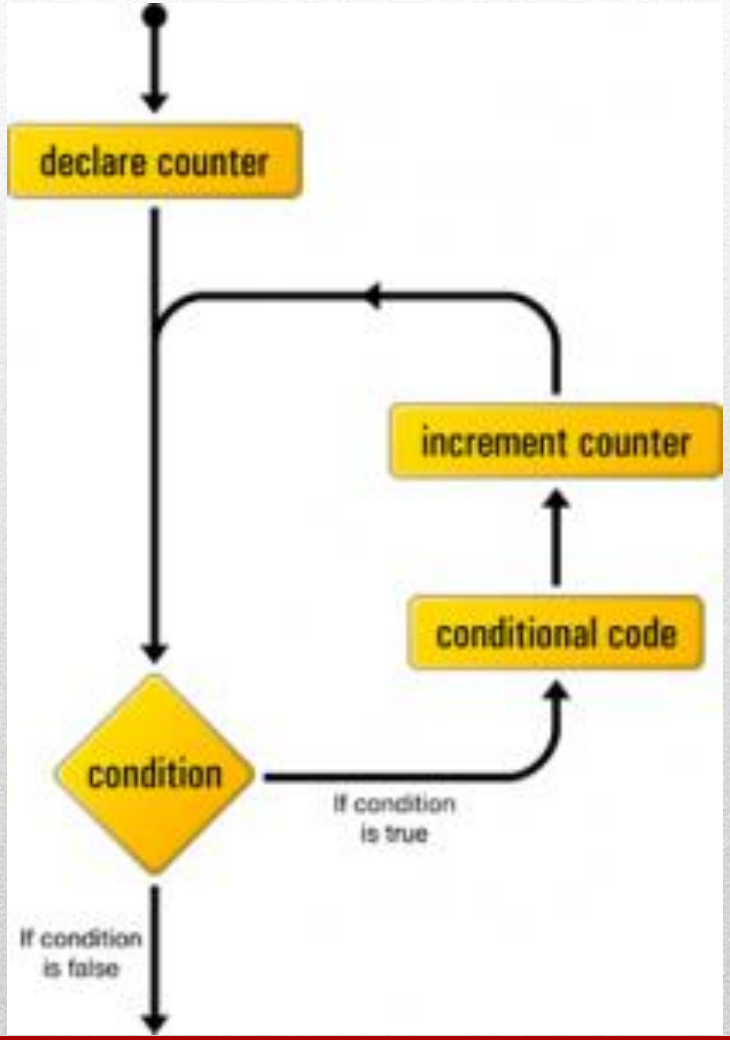
Ejemplo:

```
do {
    r=confirm("¿Desea continuar
  (S/N)?");
}while(r!=false);
```

Sentencia *for*

- Es una sentencia repetitiva que incluye una inicialización, una condición de paro y una instrucción de iteración conocida como incremento/decremento.
- Este tipo de lazo o bucle es conveniente cuando se conoce con exactitud el número de veces que se va a ejecutar el lazo o cuando se conocen los límites inferior y superior del mismo.

Sintaxis del *for*



Sintaxis del *for*

```
for(inicialización;condición;incremento
) {
    //bloque de instrucciones
}
```

Ejemplo:

```
count = 0;
for (i=1; i<10; i+=2){
    count += i;
}
```

Sentencias de control de ciclo

- ***break***

  Permite terminar prematuramente la ejecución del ciclo o lazo, sin tener que esperar hasta que se llegue hasta la última iteración del mismo.

- ***continue***

  Permite saltarse el resto de sentencias que haga falta ejecutar de una iteración específica y seguir de una vez con la siguiente.

Realización de ejemplos

- Ejemplo con sentencias ***for***
- Ejemplo con sentencias ***while***
- Ejemplo con sentencias ***do-while***

Sentencia *for-in*

- Esta instrucción se utiliza para realizar un recorrido por todas las propiedades de un objeto, a manera de bucle o ciclo.
- Esta sentencia termina automáticamente después de acceder a la última propiedad del objeto.
- No debe utilizarse esta instrucción con un tipo de dato que no sea un objeto. Hacerlo produciría un error en tiempo de ejecución

Sintaxis de la sentencia *for-in*

```
for (variable in objeto) {
    //bloque de instrucciones
}
```

Ejemplo:

```
for (property in window){
    document.write("property");
    document.write("<br>");
}
```

Sentencia *with*

- Esta instrucción permite escribir código de forma abreviada cuando se hace referencia a los objetos.
- Se puede evitar tener que digitar varias veces el mismo nombre de objeto en un bloque de instrucciones. En lugar de ello, puede hacerse referencia al nombre del objeto una sola vez en una instrucción *with*.

Sintaxis del *with*

```
with (objeto){
    //Bloque de instrucciones;
}
```

Ejemplo:

```
with (document){
    write("El salario es: " + salario + "<br>");
    write("El descuento es: " + descuento + "<br>");
    write("El salario neto es: " + neto + "<br>");
}
```

Definición de arreglo o matriz

- Al igual que en otros lenguajes un arreglo o matriz es un tipo de dato compuesto que puede contener tipos de datos simples y compuestos.
- Cada valor almacenado en un arreglo es denominado elemento y cada elemento tiene una posición, que puede ser numérica o no. Este indicador de posición del elemento es denominado índice del arreglo o matriz.
- Los elementos de una matriz en JavaScript comienzan desde la posición 0 y no la 1, cuando hablamos de arreglos indexados numéricamente.

Aspectos básicos sobre matrices

- En JavaScript, además de los matrices indexados numéricamente, existen los arreglos asociativos.
- Una matriz asociativa es aquella cuyos elementos son referenciados por índices de tipo cadena, en lugar de ser referenciados por índices numéricos.
- Para acceder a un elemento de una matriz debe utilizarse el nombre de la matriz y, a continuación, y entre corchetes su índice numérico o asociativo.

Declaración de matrices en JavaScript

- Declaración tradicional con corchetes:

```
var impares = [];
```

En esta primera forma se coloca el nombre del identificador y luego, entre corchetes (que pueden quedar vacíos) la lista de valores que almacenará.

- Utilizando el constructor Array():

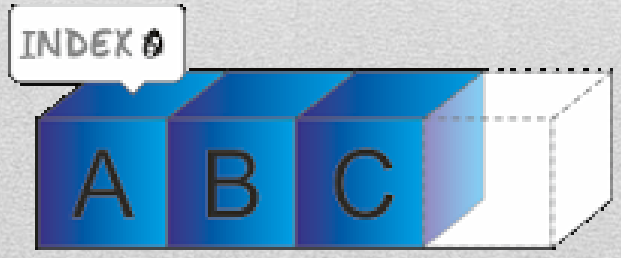
```
var pares = new Array();
```

En esta segunda forma se utiliza el constructor Array. Si lo desea puede incluir los elementos de una vez entre los paréntesis del constructor.

Especificar el tamaño de la matriz

- Es posible especificar el número de elementos que podrá contener una matriz al momento de declararlo si se especifica entre corchetes o entre los paréntesis del constructor Array, un solo valor numérico:

```
var enteros = [25];
var enteros = new Array(25);
```



Declaración de matrices en JavaScript

Ejemplos:

```
//Utilizando corchetes
var bebidas = ["Café", "Té", "Chocolate"];
var primos = [2,3,5,7,11,13,17,19,23,29];

//Utilizando el constructor Array()
var carnes = new Array("Pescado", "Pollo",
  "Res");
var noprimos = new
  Array(1,4,6,8,9,10,12,14,15,16,18,20,21,22,24
  ,25,26,27,28);
```

Asignar elementos a una matriz

- Para introducir valores en un arreglo puede utilizar varias estrategias:
  1. Asignando un dato a un elemento del arreglo de forma directa.
  2. Asignando una lista de valores de una vez en el arreglo.
  3. Pasando los argumentos en el constructor Array de una sola vez.

Asignar elementos a una matriz

```
//Directamente
musica[0] = "rock";
musica[1] = "pop";
musica[2] = "disco";

//Utilizando corchetes
musica = ["rock", "pop", "disco"];

//Utilizando el constructor Array()
musica = new Array("rock", "pop", "disco");
```

Asignar elementos a una matriz

- La asignación a los elementos de un arreglo no se hace necesariamente en el código fuente.
- Por lo general, los valores a los elementos se asignarán cuando se esté ejecutando el script por parte del usuario.
- Para asignar valores a una matriz por parte de los usuarios, resulta conveniente hacer uso de sentencias repetitivas como el ***for***, el ***while*** o el ***do-while***.

Asignar elementos a una matriz

```
//Utilizando lazo for
for(var i=0; i<10; i++){
    notas[i] += parseInt(prompt("Nota: ",""));
}
//Utilizando lazo while
var i=0;
while(i<10){
    notas[i] += parseInt("Nota: ","");
    i++;
}
```

Acceso a los elementos de una matriz

- Para acceder a los elementos almacenados en un arreglo se utiliza el identificador del arreglo y a continuación, entre corchetes el índice del arreglo, numérico o asociativo.

```
mayor = numeros[10];
monedasv = monedas['elsalvador'];
alert(notas['apsI']['ricardoelias']);
```

- También pueden utilizarse los ciclos ***for***, ***while***, ***do-while*** y ***for-in*** para hacer un recorrido por todos o por algunos de los elementos de un arreglo.

Acceso a los elementos de una matriz

- Una precaución con respecto al acceso a los elementos de una matriz es tener el cuidado de no acceder a posiciones de la matriz no establecidas o indefinidas.
- Esto se debe a que JavaScript permite asignar valores a índices de arreglo de forma no consecutiva.
- El valor que será mostrado cuando se acceda a una posición de arreglo no definida será el valor ***undifined***.

Añadir o modificar elementos a una matriz

- Para agregar un nuevo elemento a una matriz no es necesario asignar más memoria para en el arreglo.
- Los nuevos valores se agregan de forma directa, tal y como se crean la primera vez.
- Es por esto que no tiene sentido definir un arreglo como de un tamaño específico, el único propósito de hacerlo sería la claridad del código.

Añadir o modificar elementos a una matriz

- Ejemplo:

```
var planetas = ["Marte", "Saturno", "Júpiter"];
var jovianos = planetas;
jovianos[0] = "Neptuno";
```

- Al acceder a los elementos del arreglo planetas se nos mostrarían los planetas: Neptuno, Saturno y Júpiter. El valor de Marte sería reemplazado por el de neptuno.
- La razón es que se al hacer la asignación de una variable arreglo a otra en la instrucción `jovianos = planetas`, la asignación se realiza por referencia.

Eliminar elementos de una matriz

- Para eliminar un elemento específico de una matriz puede utilizar el operador ***delete***.
- Este operador establece el elemento invocado al valor ***undifined***.
- El tamaño del arreglo no es modificado al utilizar el operador ***delete***.

Eliminar elementos de una matriz

- Ejemplo:

```
var colores = ["rojo", "verde", "azul"];
delete colores[1];
alert("El valor de colores[1] es: " +
colores[1]);
```

El valor del tamaño del arreglo seguirá siendo 3, incluso si se borra el último elemento del arreglo colores.

Obtener el tamaño de una matriz

- Para obtener el tamaño de un arreglo se puede utilizar la propiedad ***length*** para facilitar la tarea.
- La propiedad ***length*** recupera el índice de la siguiente posición disponible (sin ocupar) al final del arreglo.
- Este valor obtiene el índice de esa posición incluso si existen posiciones con índice menor sin ocupar.
- En otras palabras, ***length*** devuelve el índice del primer espacio disponible después del último elemento establecido con un valor.

Obtener el tamaño de una matriz

- Ejemplo:

```
var saludos = new Array();
saludos[5] = "Buenas noches";
alert(saludos.length);
```

- El valor devuelto al aplicar ***length*** al arreglo saludos será 6, aunque únicamente se haya establecido valor para el índice 5 y a pesar de que los índices 0, 1, 2, 3 y 4 estén indefinidos.a

Arreglos o matrices asociativas

- En JavaScript los índices de los arreglos pueden ser literales de cadena y no solamente números como en otros lenguajes.
- Para poder recorrer los elementos de una matriz indexada con cadenas puede utilizarse la sentencia repetitiva for-in, al igual que como se hace con objetos. De hecho en JavaScript una matriz es considerada como un tipo especial de objeto.

Arreglos o matrices asociativas

- Ejemplo:

```
var tags = new Array();
  tags['html']  = "Inicio de un documento HTML";
  tags['head']  = "Cabecera del documento HTML";
  tags['title'] = "Título del documento HTML";
  tags['body']  = "Cuerpo del documento HTML";
 …
  for(etiqueta in tags){
     tabla += "<tr>";
     tabla += "<td>" + etiqueta + "</td>\n<td>" +
     tags[etiqueta] + "</td>\n";
     tabla += "</tr>";
  }
```

Matrices multidimensionales

- JavaScript admite trabajar con matrices multidimensionales; sin embargo, estas no están definidas explícitamente en el núcleo del lenguaje.
- Para poder simular el trabajo con matrices multidimensionales JavaScript utiliza matrices de matrices. Lo que significa que se pueden definir como elementos de una matriz, otra matriz.

Matrices multidimensionales

```
var i,j;
var paresimpares = [
        [0,2,4,6,8,10,12,14,16,18,20],
        [1,3,5,7,9,11,13,15,17,19]
];
for(i=0; i<paresimpares.length; i++){
   document.write("<h4>[");
   for(j=0; j<paresimpares[i].length; j++)
      document.write(" " + paresimpares[i][j] + " ");
   document.write("]</h4>");
}
```

Métodos para el manejo de matrices

- **join()**. Convierte todos los elementos de una matriz en cadenas y luego, los concatena. Se puede especificar en un argumento opcional el caracter de separación entre los elementos. Si no se especifica se asume por defecto la coma como caracter de separación.

```
var num = [1, 2, 3];
var strnum = num.join();   //Salida: s="1,2,3"
var letras = ['a', 'e', 'i', 'o', 'u'];
var strlet = letras.join("-"); //Salida:strlet="a-e-i-
                                        //o-u"
```

Métodos para el manejo de matrices

- **concat()**. Crea y devuelve una matriz que contiene los elementos de la matriz original sobre la que se ha aplicado el método concat(), seguidos por cada uno de los argumentos proporcionados en concat().

```
var impares = [1,3,5];
alert(impares.concat(7,9));//Salida:1,3,5,7,9
```

Métodos para el manejo de matrices

- **sort()**. Ordena los elementos de una matriz de forma lexicográfica. Esto lo hace convirtiendo primero los elementos de la matriz en cadenas y luego aplica el orden lexicográfico.

```
var numeros = [14,52,3,45,36];
alert(numeros.sort());//Salida:14,3,36,45,52
```

Lenguajes Interpretados en el Cliente

FIN

Clase 3