

Primeros pasos

Variables

Tipos de datos

Operadores


Condicionales




Bucles

Alertas y ventanas

Funciones


Arrays

 index.html

index.html >  html >  head >  meta

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10      |
11  </body>
12  </html>
```

JS Interno

```
index.html >  html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8      <script type="text/javascript">
9          | alert("Hola bienvenidos");
10         </script>
11 </head>
12 <body>
13     | <h1>Mi entrenamiento JS</h1>
14 </body>
15 </html>
```

ecibidos (658) - jjpardo2002@g x

Recibidos (4.039) - videojuegos@ x



ex.html



Google Maps



YouTube



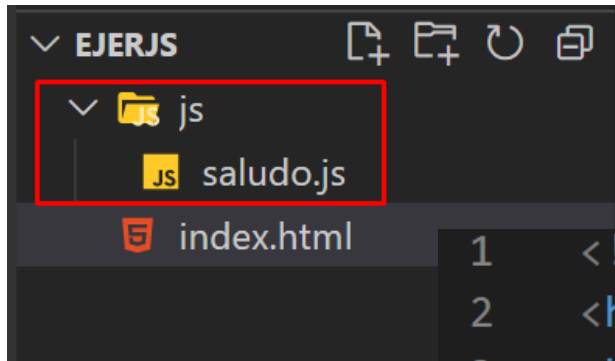
Wikipedia

127.0.0.1:5500 dice

Hola bienvenidos

Aceptar

JS Externo



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8     <!-- <script type="text/javascript">
9         alert("Hola bienvenidos");
10    </script> -->
11    <script type="text/javascript" src="js/saludo.js"></script>
12 </head>
13 <body>
14     <h1>Mi entrenamiento JS</h1>
15 </body>
16 </html>
```

JS Externo – Escribiendo en el Documento HTML

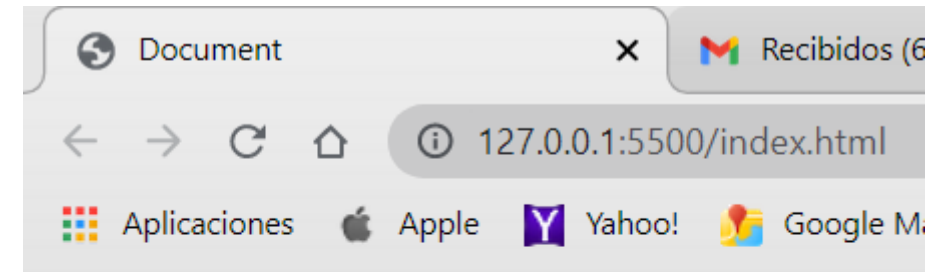
js > JS saludo.js

```
1 alert("Hola bienvenidos");
2 document.write("Escribiendo desde documento externo JS");
```



index.html > html > head

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8   <!-- <script type="text/javascript">
9     alert("Hola bienvenidos");
10  </script> -->
11   <script type="text/javascript" src="js/saludo.js"></script>
12 </head>
13 <body>
14   <h1>Mi entrenamiento JS</h1>
15 </body>
16 </html>
```

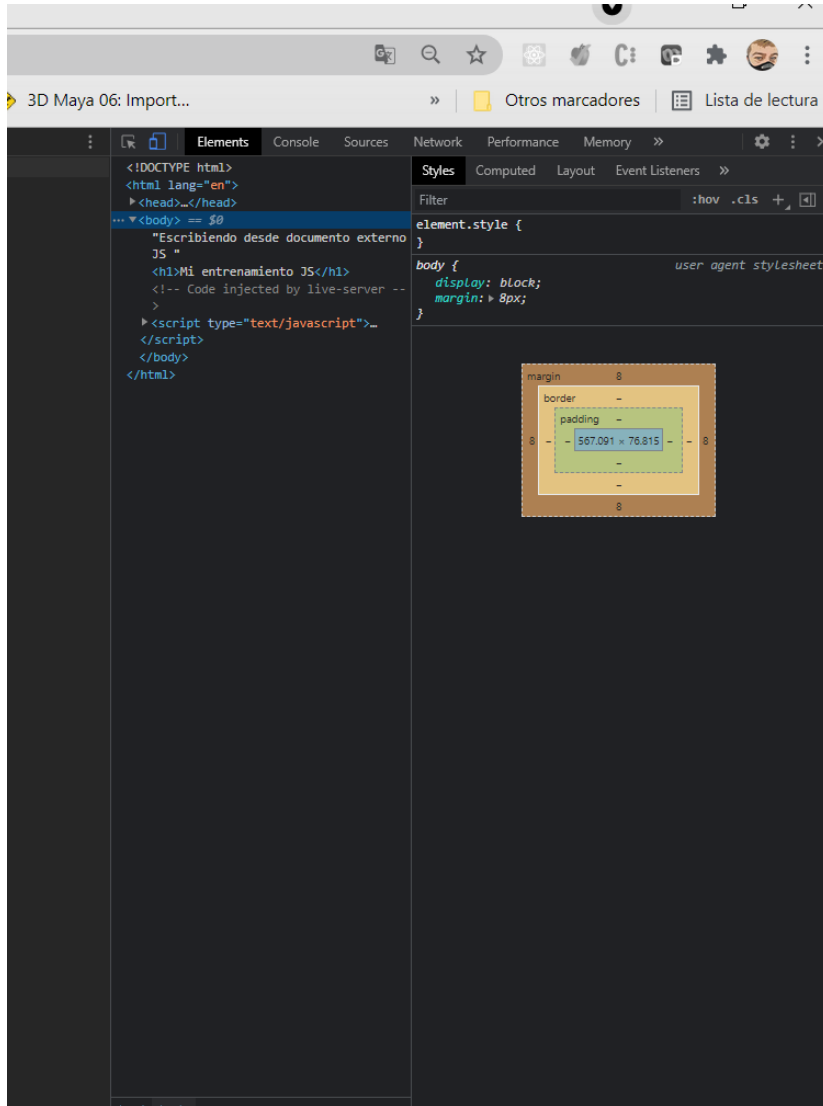


Escribiendo desde documento externo JS

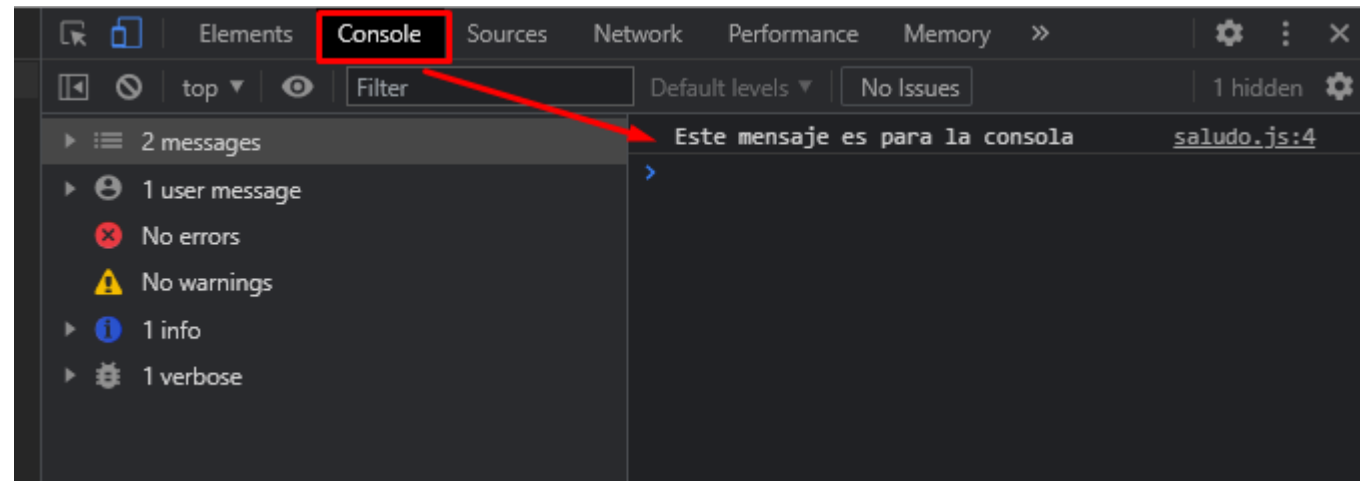
Mi entrenamiento JS

JS Externo – Escribiendo en la consola

F12 (Activa ventana desarrollo Navegador)



```
js > JS saludo.js
1 alert("Hola bienvenidos");
2 document.write("Escribiendo desde documento externo JS");
3 //Mensajes en consola
4 console.log("Este mensaje es para la consola");
```



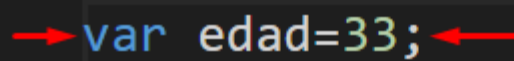
JS Variables

JS Comentarios

```
1  //Esto es un comentario en linea
2  /*
3  | Esto es un Comentario en bloque
4  */
```

JS Variables

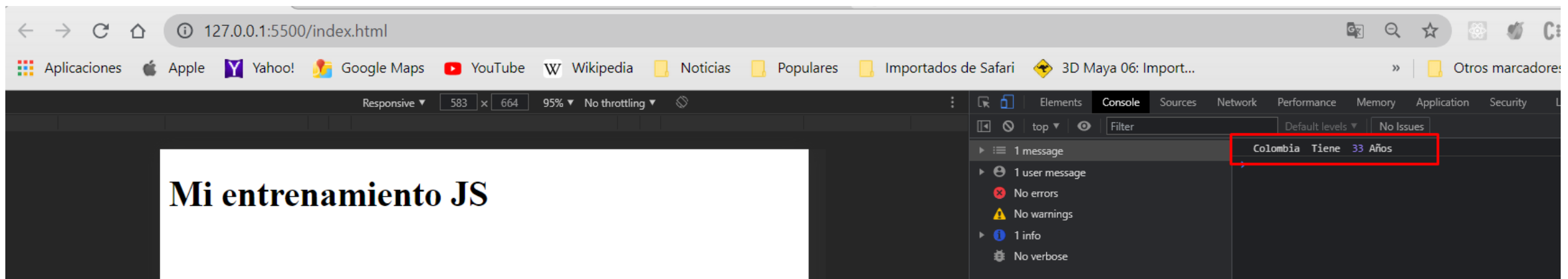
```
1  //Declaracion
2  var pais="Colombia";
3  → var edad=33; ←
```



The diagram illustrates the syntax of variable declarations and assignments in JavaScript. In the third line of code, 'var edad=33;', a red arrow points from the left to the keyword 'var', and another red arrow points from the right to the variable name 'edad'. A third red arrow points upwards to the variable name 'edad'.

Getting Started index.html JS saludo.js JS variables.js X

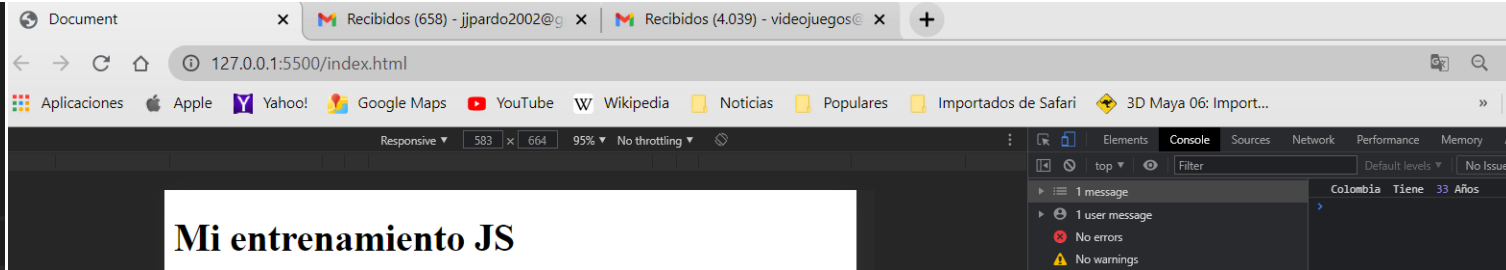
```
js > JS variables.js > ...  
1 //Declaracion  
2 var pais="Colombia";  
3 var edad=33;  
4 console.log(pais," Tiene ", edad, "Años");
```



JS Modo Estricto

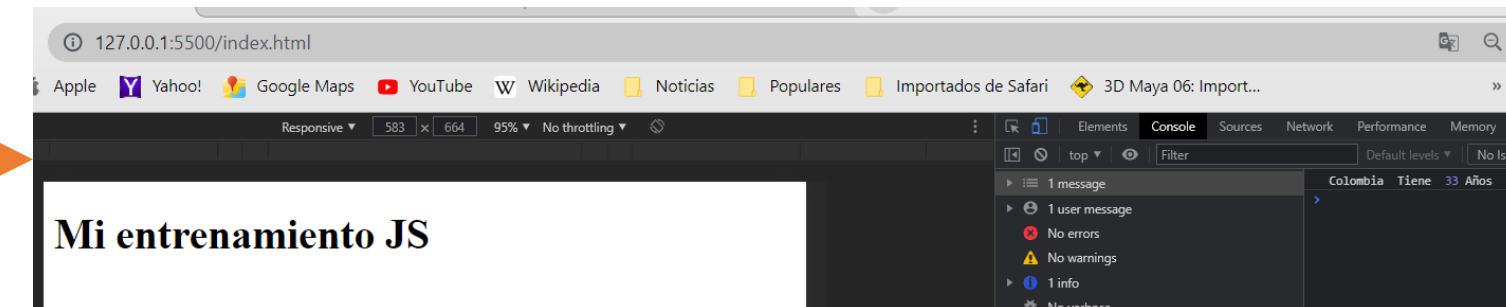
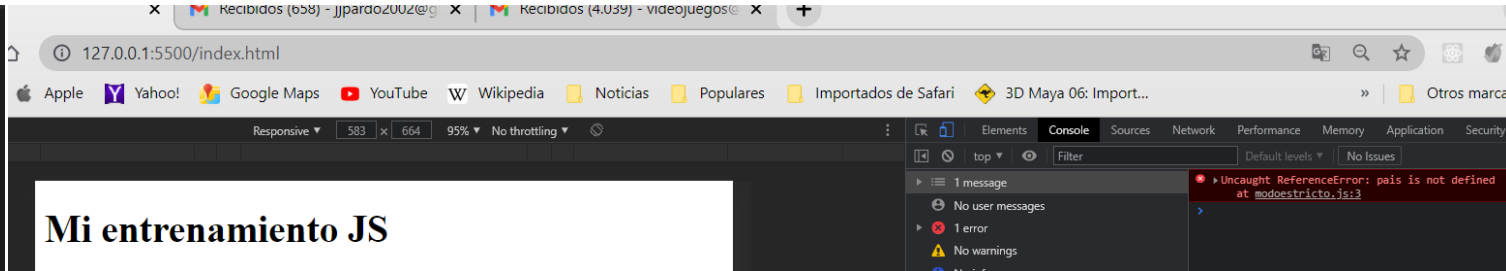
JS Sin Modo Estricto

```
// 'use strict'
// Declaracion
pais="Colombia";
var edad=33;
console.log(pais," Tiene ", edad, "Años");
```



JS Con Modo Estricto

```
js > JS modoestricto.js > ...
1 'use strict'
2 //Declaracion
3 var pais;
4 pais="Colombia";
5 var edad=33;
6 console.log(pais," Tiene ", edad, "Años");
```



JS Variables

Var

```
'use strict'

var edad=20;
console.log(edad);
if(true){
    var edad=12;
    console.log(edad);
}
console.log(edad);
```

```
20
12
12
>
```

let

```
let programa="Angular";
console.log(programa);
if(true){
    let programa="Pascal";
    console.log(programa);
}
console.log(programa);
```

```
Angular
Pascal
Angular
>
```

JS Constantes

```
'use strict'  
const salario=20000;  
let horasw=12;  
console.log("Total a pagar",(salario*horasw));
```

Default levels ▼ | No Issues

Total a pagar 240000

>

Operadores de asignación

Nombre	Operador abreviado	Significado
<u>Asignación</u>	<code>x = y</code>	<code>x = y</code>
<u>Asignación de adición</u>	<code>x += y</code>	<code>x = x + y</code>
<u>Asignación de resta</u>	<code>x -= y</code>	<code>x = x - y</code>
<u>Asignación de multiplicación</u>	<code>x *= y</code>	<code>x = x * y</code>
<u>Asignación de división</u>	<code>x /= y</code>	<code>x = x / y</code>
<u>Asignación de residuo</u>	<code>x %= y</code>	<code>x = x % y</code>
<u>Asignación de exponenciación</u>	<code>x **= y</code>	<code>x = x ** y</code>

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions_and_Operators

Operadores de comparación

<u>Operador</u>	<u>Descripción</u>	<u>Ejemplos que devuelven true</u>
<u>Igual</u> (==)	<u>Devuelve true si los operandos son iguales.</u>	<u>3 == var1</u> <u>"3" == var1</u> <u>3 == '3'</u>
<u>No es igual</u> (!=)	<u>Devuelve true si los operandos <i>no</i> son iguales.</u>	<u>var1 != 4</u> <u>var2 != "3"</u>
<u>Estrictamente igual</u> (===)	<u>Devuelve true si los operandos son iguales y del mismo tipo. Consulta también <u>Object.is</u> y <u>similitud en JS</u>.</u>	<u>3 === var1</u>
<u>Desigualdad estricta</u> (!==)	<u>Devuelve true si los operandos son del mismo tipo pero no iguales, o son de diferente tipo.</u>	<u>var1 !== "3"</u> <u>3 !== '3'</u>
<u>Mayor que</u> (>)	<u>Devuelve true si el operando izquierdo es mayor que el operando derecho.</u>	<u>var2 > var1</u> <u>"12" > 2</u>

<u>Mayor o igual que</u> (>=)	<u>Devuelve true si el operando izquierdo es mayor o igual que el operando derecho.</u>	<u>var2 >= var1</u> <u>var1 >= 3</u>
<u>Menor que</u> (<)	<u>Devuelve true si el operando izquierdo es menor que el operando derecho.</u>	<u>var1 < var2</u> <u>"2" < 12</u>
<u>Menor o igual</u> (<=)	<u>Devuelve true si el operando izquierdo es menor o igual que el operando derecho.</u>	<u>var1 <= var2</u> <u>var2 <= 5</u>

Operadores aritméticos

Operador	Descripción	Ejemplo
<u>Residuo</u> (%)	Operador binario. Devuelve el resto entero de dividir los dos operandos.	12 % 5 devuelve 2.
<u>Incremento</u> (++)	Operador unario. Agrega uno a su operando. Si se usa como operador prefijo (++x), devuelve el valor de su operando después de agregar uno; si se usa como operador sufijo (x++), devuelve el valor de su operando antes de agregar uno.	Si x es 3, ++x establece x en 4 y devuelve 4, mientras que x++ devuelve 3 y, solo entonces, establece x en 4.
<u>Decremento</u> (- -)	Operador unario. Resta uno de su operando. El valor de retorno es análogo al del operador de incremento.	Si x es 3, entonces --x establece x en 2 y devuelve 2, mientras que x-- devuelve 3 y, solo entonces, establece x en 2.
<u>Negación unaria</u> (-)	Operador unario. Devuelve la negación de su operando.	Si x es 3, entonces -x devuelve -3.
<u>Positivo unario</u> (+)	Operador unario. Intenta convertir el operando en un número, si aún no lo es.	+"3" devuelve 3. +true devuelve 1.
<u>Operador de exponenciación</u> (**)	Calcula la base a la potencia de exponente, es decir, $\text{base}^{\text{exponente}}$	2 ** 3 returns 8. 10 ** -1 returns 0.1.

Operadores lógicos

Operadores lógicos

Operador	Uso	Descripción
<u>AND Lógico</u> (<code>&&</code>)	<code>expr1 && expr2</code>	Devuelve <code>expr1</code> si se puede convertir a <code>false</code> ; de lo contrario, devuelve <code>expr2</code> . Por lo tanto, cuando se usa con valores booleanos, <code>&&</code> devuelve <code>true</code> si ambos operandos son <code>true</code> ; de lo contrario, devuelve <code>false</code> .
<u>OR lógico</u> (<code> </code>)	<code>expr1 expr2</code>	Devuelve <code>expr1</code> si se puede convertir a <code>true</code> ; de lo contrario, devuelve <code>expr2</code> . Por lo tanto, cuando se usa con valores booleanos, <code> </code> devuelve <code>true</code> si alguno de los operandos es <code>true</code> ; si ambos son falsos, devuelve <code>false</code> .
<u>NOT lógico</u> (<code>!</code>)	<code>!expr</code>	Devuelve <code>false</code> si su único operando se puede convertir a <code>true</code> ; de lo contrario, devuelve <code>true</code> .

El siguiente código muestra ejemplos del operador `&&` (AND lógico).

```
var a1 = true && true;    // t && t devuelve true
var a2 = true && false;   // t && f devuelve false
var a3 = false && true;    // f && t devuelve false
var a4 = false && (3 == 4); // f && f devuelve false
var a5 = 'Cat' && 'Dog';   // t && t devuelve Dog
var a6 = false && 'Cat';   // f && t devuelve false
var a7 = 'Cat' && false;   // t && f devuelve false
```



El siguiente código muestra ejemplos del operador `||` (OR lógico).

```
var o1 = true || true;    // t || t devuelve true
var o2 = false || true;   // f || t devuelve true
var o3 = true || false;   // t || f devuelve true
var o4 = false || (3 == 4); // f || f devuelve false
var o5 = 'Cat' || 'Dog';   // t || t devuelve Cat
var o6 = false || 'Cat';   // f || t devuelve Cat
var o7 = 'Cat' || false;   // t || f devuelve Cat
```



El siguiente código muestra ejemplos de el operador ! (NOT lógico).

```
var n1 = !true; // !t devuelve false  
var n2 = !false; // !f devuelve true  
var n3 = !'Cat'; // !t devuelve false
```



Operador condicional (ternario).

```
condition ? val1 : val2
```

```
var status = (age >= 18) ? 'adult' : 'minor';
```

Precedencia de los operadores

Tipo de operador	Operadores individuales
miembro	<code>.</code> <code>[]</code>
llamar / crear instancia	<code>()</code> <code>new</code>
negación / incremento	<code>!</code> <code>~</code> <code>-</code> <code>++</code> <code>--</code> <code>typeof</code> <code>void</code> <code>delete</code>
multiplicar / dividir	<code>*</code> <code>/</code> <code>%</code>
adición / sustracción	<code>+</code> <code>-</code>
desplazamiento bit a bit	<code><<</code> <code>>></code> <code>>>></code>
relacional	<code><</code> <code><=</code> <code>></code> <code>>=</code> <code>in</code> <code>instanceof</code>
igualdad	<code>==</code> <code>!=</code> <code>===</code> <code>!==</code>
AND bit a bit	<code>&</code>

AND lógico	&&
OR lógico	
condicional	?:
asignación	= += -= *= /= %= <<= >>= >>>= &= ^= = &&= = ??=
coma	,

Estructuras de Control

```
'use strict'
var edad=18;
if(edad>18){
    console.log('Puede ejercer el dercho al voto popular');
}
```

```
'use strict'
var edad=18;
if(edad>18){
    console.log('Puede ejercer el dercho al voto popular');
}else{
    console.log('No puede ejercer el derecho al voto popular');
}
```

```
// Operadores relacionales
Mayor: >
Menor: <
Mayor o igual: >=
Menor o igual: <=
Igual: ==
Distinto: !=
*/
```

Operadores Lógicos y Comparación

```
//Uso operadores Logicos
var genero="F";
if(edad>18 && genero!='F'){
    console.log('Puede practicar deportes arriesgados');
}else if(edad>18 || genero=='F'){
    console.log('Debes considerar tu decision de practicar deportes arriesgados');
}else{
    console.log('Eres joven para practicar deportes peligrosos');
}
```

Switch

```
var categoria=1;
switch(categoria){
  case 1:
    console.log('No debes pagar para por tu cita medica')
    break;
  case 2:
    console.log('Debes pagar $1000 por tu cita medica')
    break;
  case 3:
    console.log('Debes pagar $2000 por tu cita medica')
    break;
  case 4:
    console.log('Debes pagar $3000 por tu cita medica')
    break;
  case 5:
    console.log('Debes pagar $5000 por tu cita medica')
    break;
  default:
    console.log('Debes pagar $10000 por tu cita medica')
    break;
}
```