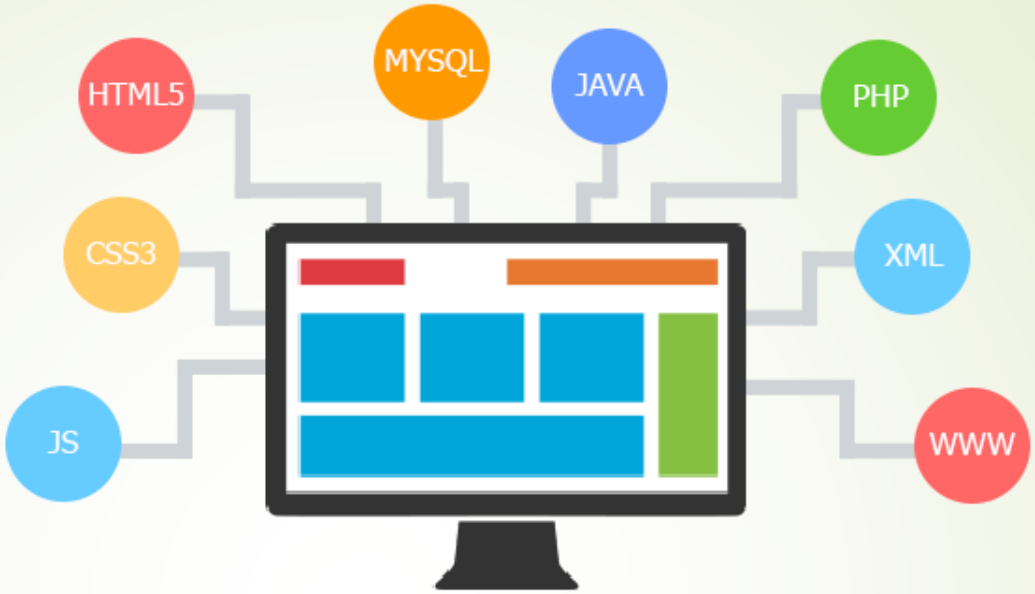


Desarrollo de Aplicaciones Web

DOCENTE: Daniel López Lozano

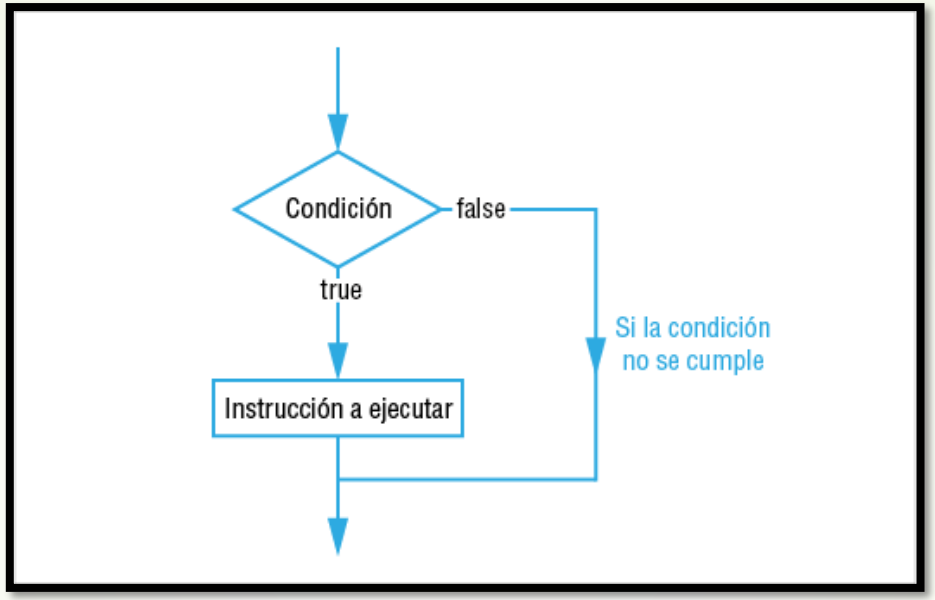
escuelaartegranada

Tema 2.
Fundamentos de JavaScript

Índice de contenidos

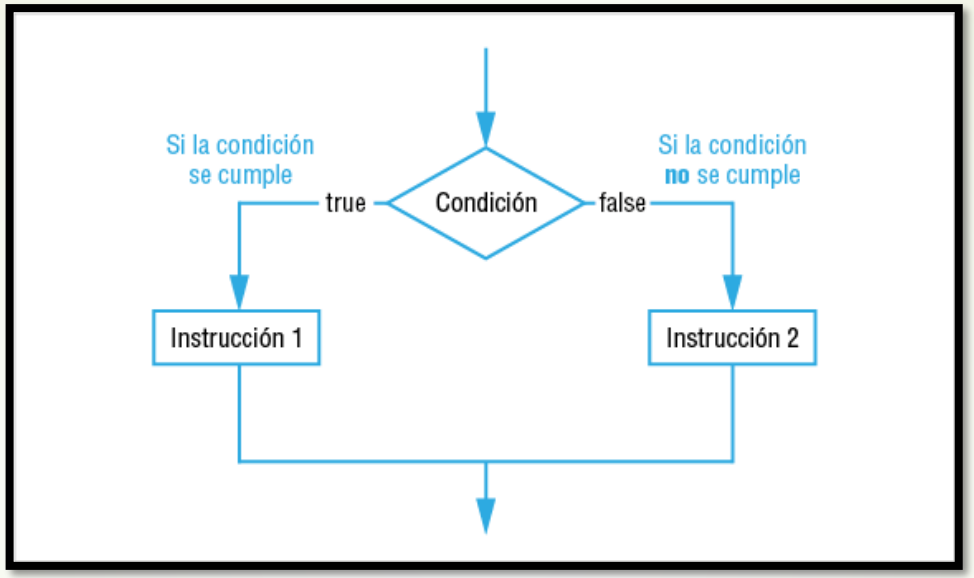
- Las estructuras condicionales realizan unas acciones u otras dependiendo del **estado de las variables**.
- Vamos a ver las estructuras **if-elseif-else y switch-case**
- La principal diferencia es que la estructura **if-else** comprueba si una **condición de cualquier tipo** es true o false.
- Mientras que la estructura **switch** solo permite definir **condiciones de igualdad** con varios valores concretos.

Esquemas if-else



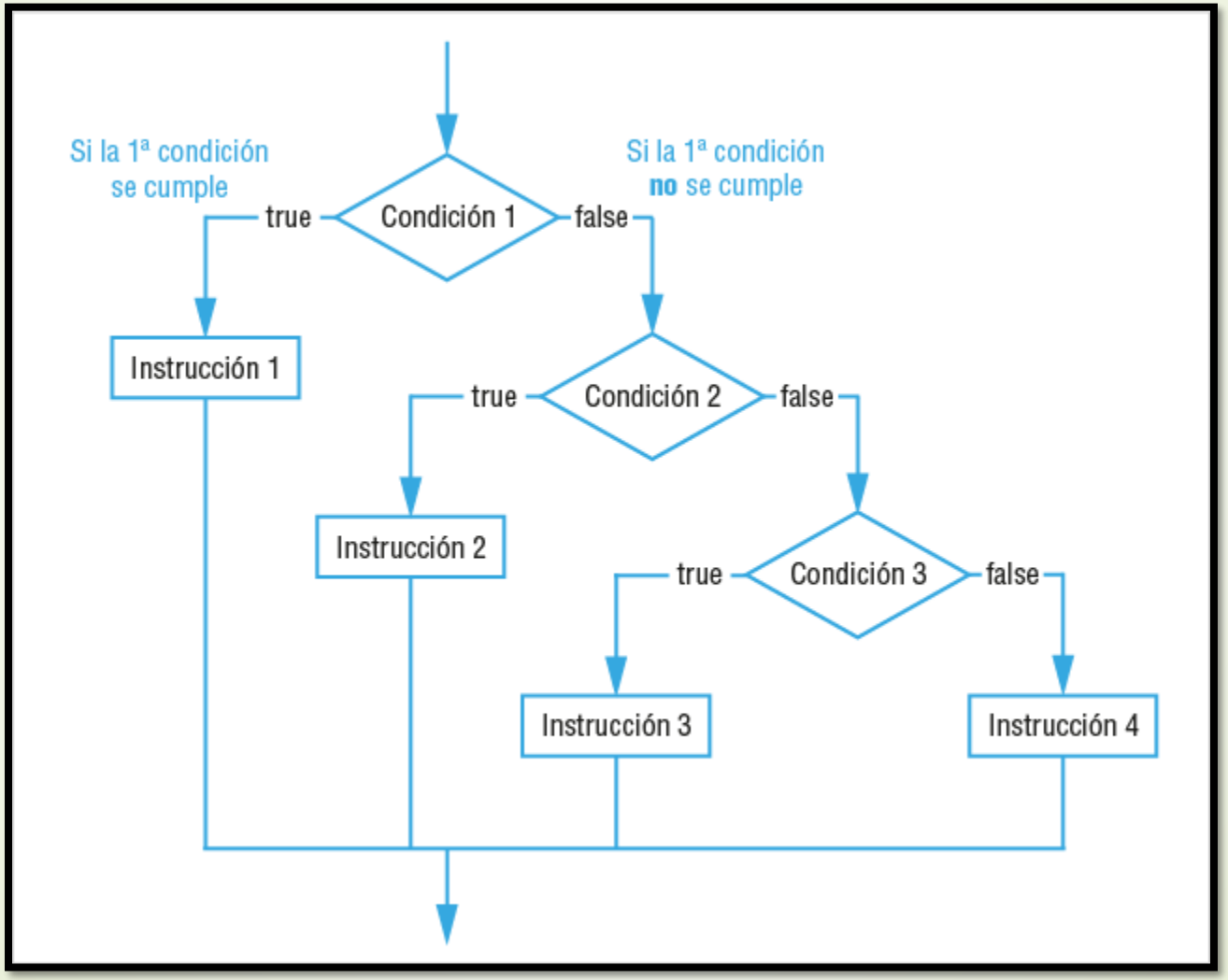

```
let miEdad = prompt("Introduce tu edad: ");
if (miEdad >30)
{
    alert("Ya eres una persona adulta");
}
```

Esquemas if-else




```
let semaforo=prompt("Color del semaforo");
if(semaforo=="verde")
{
   alert("Puede pasar");
}else{
   alert("Detengase");
}
```

Esquemas if-else



- Una forma de hacer una comprobación más completa seria la siguiente.

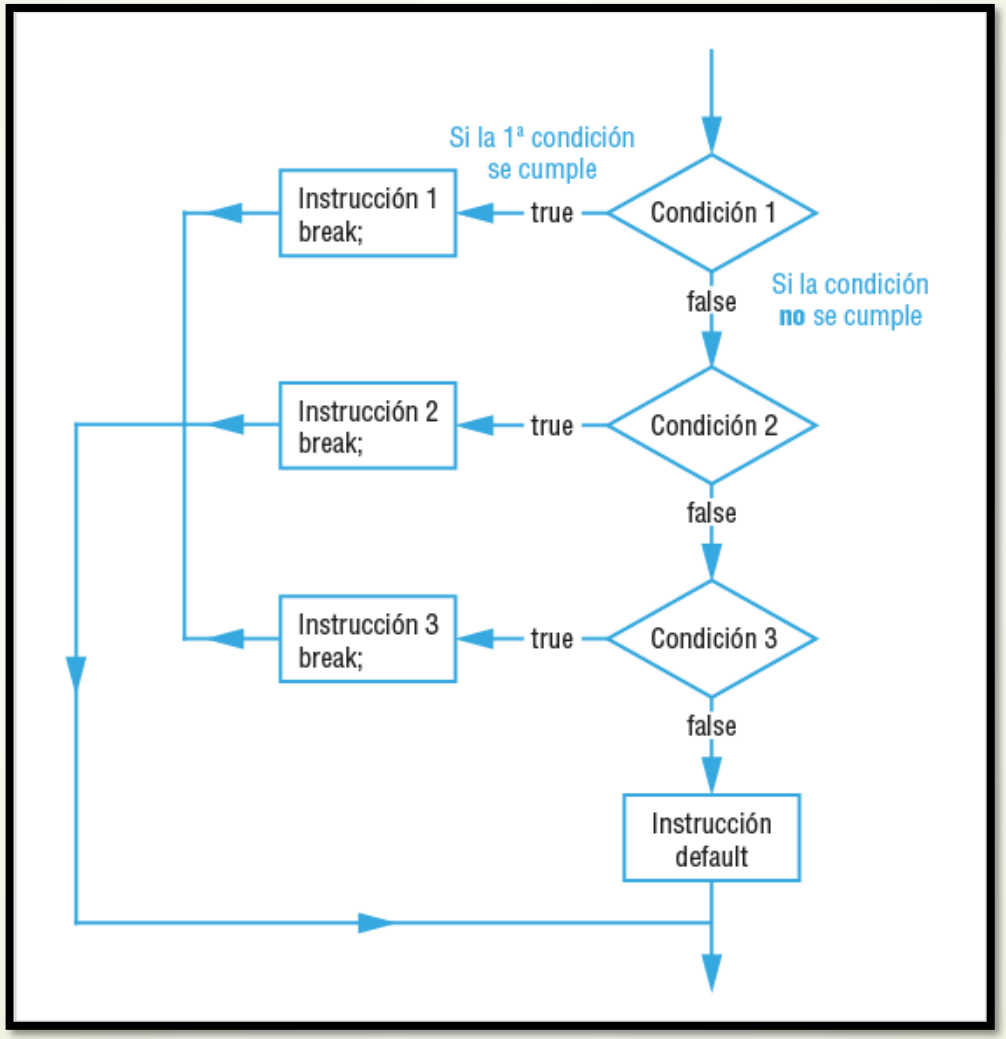
```
if(nota>=5)
{
  if(nota<7)
  {
    document.write("<h2>Aprobado</h2>");
  }else{
    document.write("<h1>Notable</h1>");
  }
}else{
  document.write("<h3>Suspenso</h3>");
}
```

- De manera equivalente se puede usar la forma if-elseif

```
if(nota>=7)
{
    document.write("<h1>Notable</h1>");
}else if(nota>=5){
    document.write("<h2>Aprobado</h2>");
}else{
    document.write("<h3>Suspenso</h3>");
}
```

```
let resolucion = prompt("Introduce la resolucion maxima: ");
if ( resolucion < 400) {
    alert("Resolución móvil");
}else if(resolucion < 800) {
    alert("Resolución tablet");
}else if(resolucion < 1280){
    alert("resolución portatil");
}else{
    alert("resolución monitor");
}
```

Esquema switch



Esquemas switch

```
let dia=prompt("Introduce el dia de la semana");
switch(dia) {
    case 1:
        alert("Lunes");
    break;
    case 2:
        alert("Martes");
    break;
    ...
    default:
        alert("No es un dia de la semana");
}
```

Comparación if y switch

```
let materia=prompt("¿Qué vas a estudiar?");
if(materia=="matematicas")
{
    alert("Va de numeros");
}else if(materia=="lengua"){
    alert("Va de letras");
}else if(materia=="ingles"){
    alert("Va de hablar");
}else if(materia=="ciencias"){
    alert("Va de investigar");
else{
    alert("No sé de que va");
}
```

```
let materia=prompt("¿Qué vas a estudiar?");
switch(materia)
{
    case "matematicas":
      alert("Va de numeros");
      break;
    case "lengua":
      alert("Va de letras");
      break;
    case "ingles":
      alert("Va de hablar");
      break;
    case "ciencias":
      alert("Va de investigar");
      break;
    default:
      alert("No sé de que va");
}
```

- El bucle for permite ejecutar un bloque de código un número fijo y conocido de veces.

- Ese numero fijo de veces se establece mediante un condición de <= ó =>.

```
for (expresión inicial; condición; incremento)
{
    // Instrucciones a ejecutar dentro del bucle.
}
```

Bucle for

```
let nombre;
for (let veces=1;veces<=10;veces++)
{
   nombre=prompt("Introduce un nombre");
   alert("<h4>El nombre numero"+veces+"es "+nombre+"</h4>");
}
```

```
let x ="", i;
for (i=1; i<=6; i++) {
        x = x + "<h" + i + ">Cabecera " + i + "</h" + i + ">";
}
document.write(x);
```

Array con for y sus variantes

```
let lenguajes=["HTML","CSS","JavaScript"];
for(let i=0;i<lenguajes.length;i++)
{
   document.write("<input type='button' value='"+lenguajes[i]+"'>");
}
//funciona en array y objetos
for(let i in lenguajes)
{
   document.write("<input type='button' value='"+lenguajes[i]+"'>");
}
//funciona en arrays y colecciones
for(let elemento of lenguajes)
{
   document.write("<input type='button' value='"+elemento+"'>");
}
```

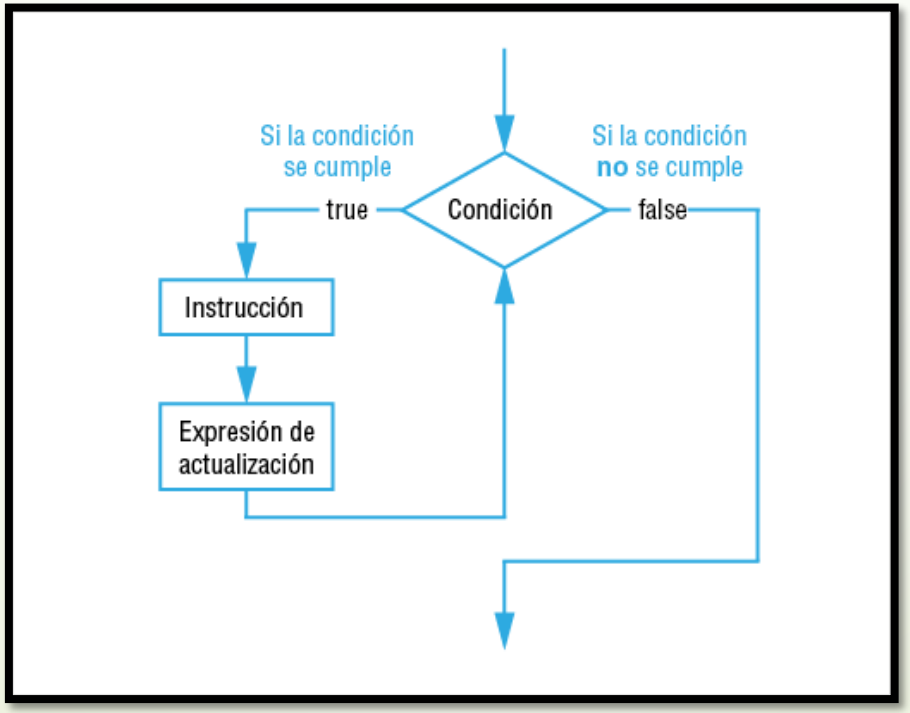
Anidar bucles for para crear tablas

```
var filas=prompt("Cuantas filas quieres en la tabla?");
var columnas=prompt("Cuantas columnas quieres en la tabla?");

document.write("<table>");
for(var i=1;i<=filas;i++)
{
    document.write("<tr>");
    for(var j=1;j<=columnas;j++)
    {
       document.write("<td>Datos</td>");
    }
    document.write("</tr>");
}
document.write("</table>");
```

- Con el bucle for, podemos repetir una tarea un número fijo de veces.

- El bucle while permite crear bucles que se ejecutan cero o más veces de manera indefinida/ilimitada.



Ejemplos while

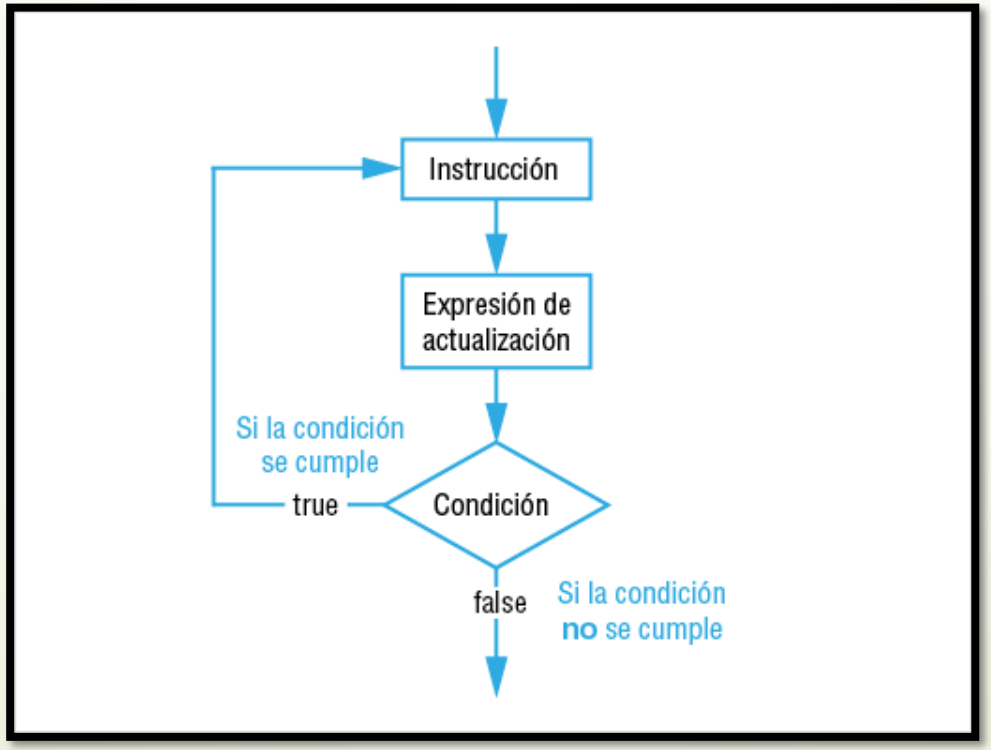
```
let i=0;
while(i<10)
{
  document.write("<br>El numero es " + i);
  i++;
}
```

```
let edad=prompt("Introduce tu edad");
while(edad<0 || isNaN(edad))
{
   edad=prompt("Introduce tu edad");
}
```

```
let secreta="eureka";
let intento="";
while(intento!=secreta)
{
   intento=prompt("¿Que palabra es?");
}
```

- El bucle do-while es una variante del bucle while que se ejecuta siempre al menos una vez.

- Permite escribir de manera más clara en ciertas situaciones



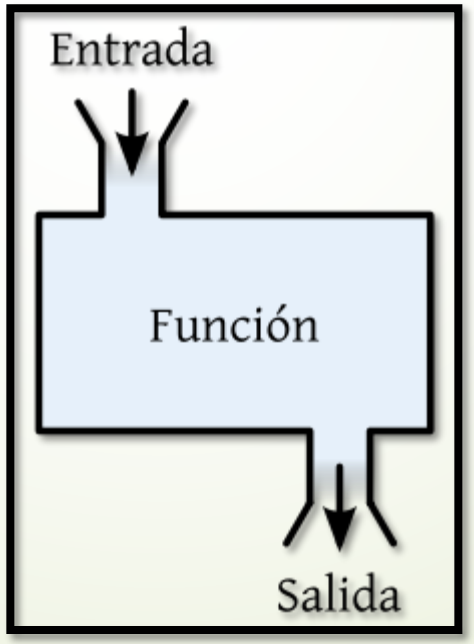
Ejemplos do-while

```
let i=1;
var respuesta;
do
{
   respuesta=confirm("¿Desea salir? Intento"+ i);
   i++;
}while (respuesta!=true);
```

```
let color;
do
{
  color=prompt("indica un color distinto de blanco");
}while(color=="white" || color=="#FFF" || color=="#FFFFFF");
```

- Una función es una herramienta que nos permite definir una serie de instrucciones como si fuera solo una evitando así duplicar código.

- Utilizar funciones permite modularizar el código y hacer que sea más entendible, además de ser necesario para la gestión de eventos.



Funciones en JavaScript

```
function nombreFuncion(parametro1,parametro2,...)
{
    ...
}
```

```
function SumarIVA(cantidad,porcentaje)
{
   let  total;

   total=cantidad+cantidad*porcentaje/100;
   alert(total);
}
SumarIVA(400,18);
```

```
function ElMayor(num1,num2)
{
   if(num1>num2)
   {
      alert(num1+" es el mayor");
   }else{
      alert(num2+" es el mayor");
   }
}
ElMayor(4,7);
```

Diseño modular de funciones

```
function SumarIVA(cantidad,porcentaje)
{
   let  total;

   total=cantidad+cantidad*porcentaje/100;
   return total;
}

let resultado=SumarIVA(400,18);
//Puedo sacarla por pantalla por HTML o lo que quiera
```

Diseño modular de funciones

```
function ElMayor(num1,num2)
{
    let mayor;
    if(num1>num2)
    {
        mayor=num1;
    }else{
        mayor=num2;
    }
    return mayor;
}

let ganador=ElMayor(4,7);
//Puedo sacarla por pantalla por HTML o lo que quiera
```

Funciones y Arrays

```
//Hace la media de un array de numeros
function Media(numeros)
{
     let suma=0;
    for(let i=0;i<=numeros.length;i++)
    {
        suma=suma+numeros[i];
    }
    return suma;
}
//...

let datos=prompt("Introduce numeros separados por espacios");
let numeros=datos.split(" ");
document.write("La suma es: "+Media(numeros));
```

- El ámbito de una variable, es la zona del programa donde existe la variable y se puede operar con ella.

- **Variables locales**: Son variables como el contador de un bucle for o declaradas dentro de una función. **No** se pueden acceder desde **fuera de su bloque**.

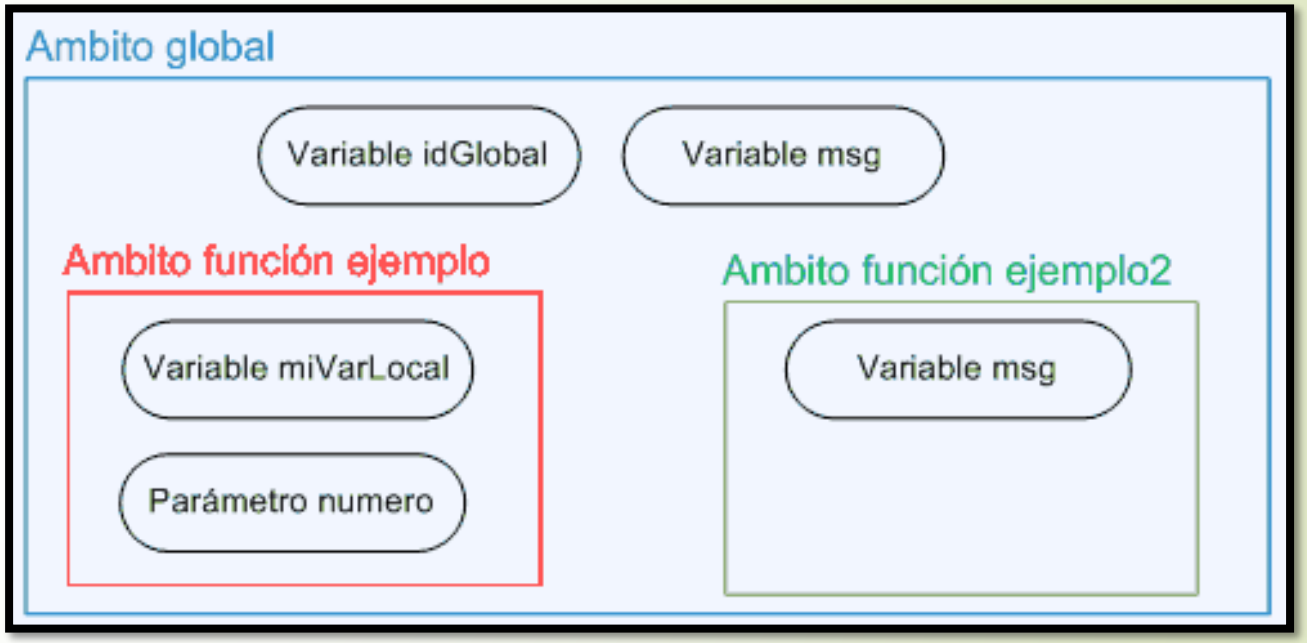
- **Variables globales**: Son variables que no se han definido dentro de ningún bloque y por tanto **están disponibles** en cualquier parte de la **aplicación**.

Ámbito variables

```
let idGlobal=33;
let msg = 'Variable global';

function ejemplo(numero) {
 let miVarLocal = 'Soy una variable local';
 ejemplo2();
}

function ejemplo2(){
  let msg = 'Mensaje: '+idGlobal;
}
```



Uso de variables globales

```
function EmpezarJuego()
{
    let puntuacion=0;
}

function Puntuacion(num)
{
    let puntuacion+=num;
}

EmpezarJuego();
alert("Tu puntuacion es de "+puntuacion+" puntos");
Puntuacion(20);
alert("Tu puntuacion es de "+puntuacion+" puntos");
Puntuacion(-10);
```

Esto no funcionaria

Uso de variables globales

```
let puntuacion;
function EmpezarJuego()
{
    puntuacion=0;
}

function Puntuacion(num)
{
    puntuacion+=num;
}

EmpezarJuego();
alert("Tu puntuacion es de "+puntuacion+" puntos");
Puntuacion(20);
alert("Tu puntuacion es de "+puntuacion+" puntos");
Puntuacion(-10);
alert("Tu puntuacion es de "+puntuacion+" puntos");
```

Esto si funcionaria

- Las **funciones y las variables globales** que se declaren en un fichero js **son visibles** en todo el documento html y otros ficheros js que se incluyan después de él **dentro del documento html**.

- Una practica muy común es incluir utilidades de programación en un fichero js aparte y otro fichero js haga uso de dichos códigos (ficheros principal) sin necesidad de incluirlos en nuestro fichero principal.

```
<script type="text/javascript" src="utilidades.js"></script>
<script type="text/javascript" src="principal.js"></script>
```

Dentro del fichero utilidades.js

```
let puntuacion;
function EmpezarJuego()
{
    puntuacion=0;
}

function Puntuacion(num)
{
    puntuacion+=num;
}
```

Dentro del fichero principal.js

```
EmpezarJuego();
alert("Tu puntuacion es de "+puntuacion+" puntos");
Puntuacion(20);
alert("Tu puntuacion es de "+puntuacion+" puntos");
Puntuacion(-10);
alert("Tu puntuacion es de "+puntuacion+" puntos");
```

- Existen distintas maneras de definir objetos en JavaScript. Una de las mas comunes es el uso de llaves {}.


```
let micoche={
    name:"Fiat",
    model:"500",
    weight: 850,
    color:"white"
};
```

Manejo básico de objetos

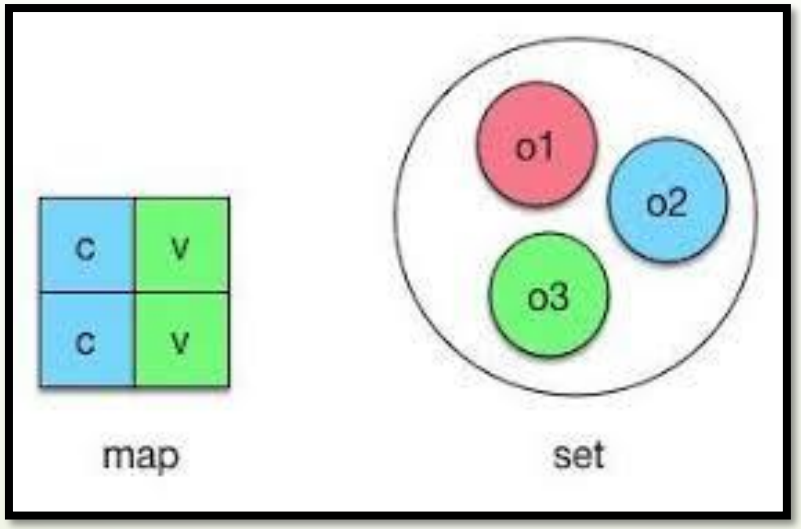
```
//Mostrar informacion del objeto
document.write(micoche.name);
document.write(micoche["color"]);

//Modificar informacion del objeto
micoche.model="500C";
micoche["model"]="500C";

//Recorrer la informacion del objeto
for (dato in micoche)
{
   document.write(dato+" : "+micoche[dato]);
}
```

- Los arrays y los objetos son estructuras muy importante aunque en algunos aspectos insuficientes.

- Desde ECMA Script 6 JavaScript dispone de estructuras de datos más avanzadas como son los Map y Set.



- Los Map (diccionarios) representan la información en una estructura clave-valor. En este aspecto son idénticos a los objetos.

- Los Map además ofrecen una serie de operaciones predefinidas muy útiles.

```
let palabras=new Map();

palabras.set("casa","house");
palabras.set("perro","dog");
palabras.set("rojo","red");
alert(palabras.size());
palabras.delete("casa");
```

```
let busqueda=prompt("¿Que palabra buscas?");
if(palabras.has(busqueda))
{
   alert(palabras.get(busqueda));
}else{
   alert("No tenemos esa palabra");
}

palabras.clear();
```

Recorrido de un Map

```
for (let entrada of palabras) {
  document.write(entrada);
}
// ['casa', 'house']
// ['perro', 'dog']
// ['rojo', 'red']

for (let [clave, valor] of palabras) {
  document.write(valor);
}
//'house'
// 'dog'
// 'red'
```

- Los Set representan una colección de datos donde no puede haber **elementos repetidos** que es la principal diferencia con un Array.

```
let invitados = new Set();

invitados.add("John");
invitados.add("Pete");
invitados.add("Mary");
invitados.add("John");
invitados.add("Mary");
```

```
//Solo aparecen una vez
alert(invitados.size());

for (let persona of invitados) {
  document.write(persona);
}

invitados.delete("Pete");
```

Depuración de programas

- Por muy buenos programadores que seamos, a veces no somos capaces de encontrar la **causa de un error**.

- Nosotros trasladamos una idea que tenemos a un programa que ejecuta un ordenador y aunque vayamos por buen camino algo **puede fallar**. Los detalles a la hora programar son importantes.

- En cierta ocasiones es interesante poder ver como evoluciona **paso a paso** nuestro programa y ver como van cambiando nuestras variables a cada instruccion.

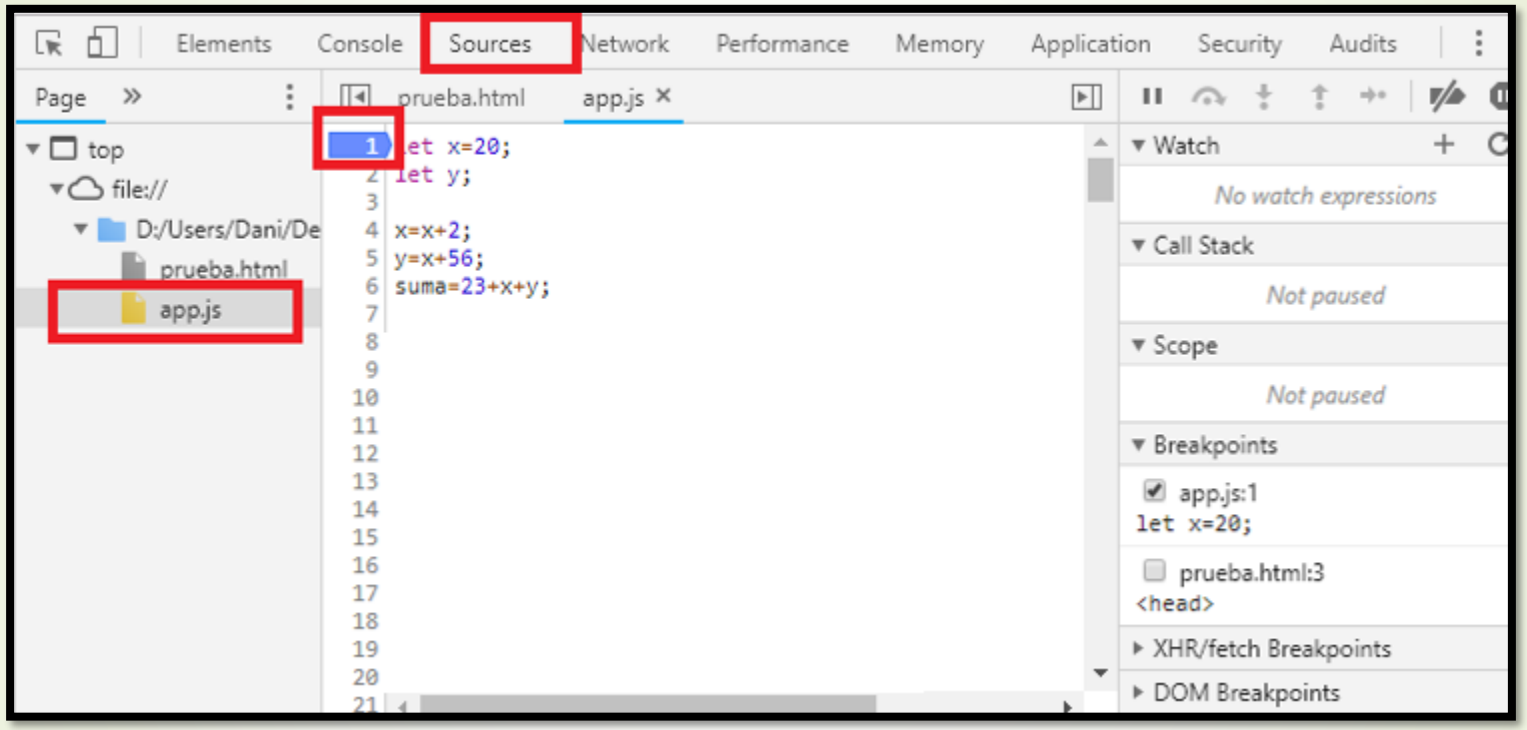
- En un principio imprimimos el valor de nuestras variables de las que **sospechamos** cada cierta distancia en nuestro código.

- Los **depuradores o debugger** surgen para facilitar dicha tarea y ya no es necesario modificar el código.

- Hay que entrar en el **inspector de código** de Google Chrome con F12 o botón derecho inspeccionar.

- En la pestaña **Sources** hay un navegador de archivos y debemos buscar nuestro código JavaScript.

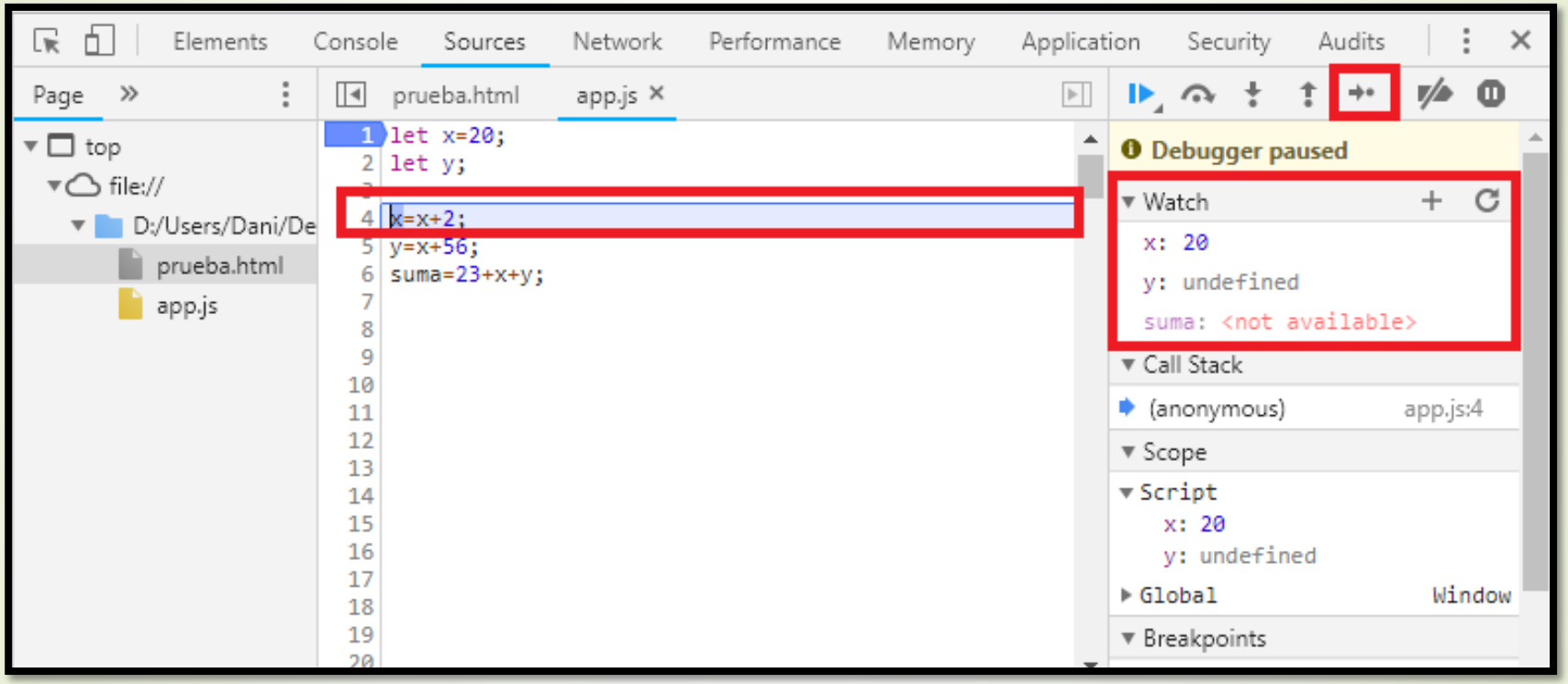
Abrir depurador y código



Clic en la línea de código a partir de la cual queremos empezar

Recargamos la pagina

Seleccionar variables y avanzar



Tenemos que añadir variables en la sección Watch y pulsar el botón avanzar que esta encima de las variables

Bibliografía

- Gauchat, Juan Diego: “El gran libro de HTML5, CSS3 y Javascript”. Editorial Marcombo. 2012
- Vara, J.M. y otros: “Desarrollo Web en Entorno Cliente. CFGS”. Editorial Ra-Ma. 2012
- “Programación en Javascript”. Colección de artículos disponibles en la url. http://www.desarrolloweb.com/manuales/ Última visita: Septiembe 2017.
- W3SCHOOL “Manual de referencia y Tutoriales” http://www.w3schools.com/ Última visitaSeptiembre 2017
- Comesaña, J.L. : Técnico en Desarrollo de Aplicaciones Web. http://www.sitiolibre.com/daw.php Última visita Septiembre 2017
- Pildoras Informaticas. Curso de JavaScript. https://www.pildorasinformaticas.es/course/javascript-desde-0/ Ultima visita Septiembre 2018.