



<http://www.robertsallent.com>

[@robertsallent](https://twitter.com/robertsallent)



v.22.12

# JS05: bifurcaciones 1

---

Operadores relacionales.

Bifurcaciones simples, dobles y múltiples.



# Índice



- Flujo de programa.
- Operadores relacionales.
- Selección o bifurcación.
  - Simple (*if*).
  - Doble (*if-else*).
  - Selección anidada.
  - *else if*.
  - Selección múltiple (*switch*).
- Ejercicios.

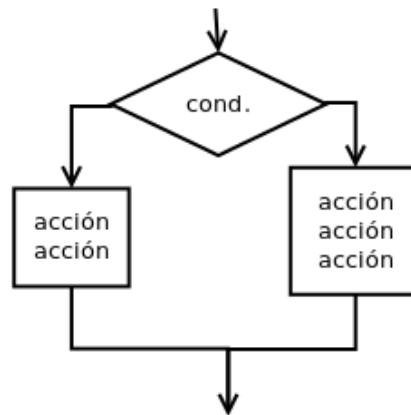
# Flujo del programa



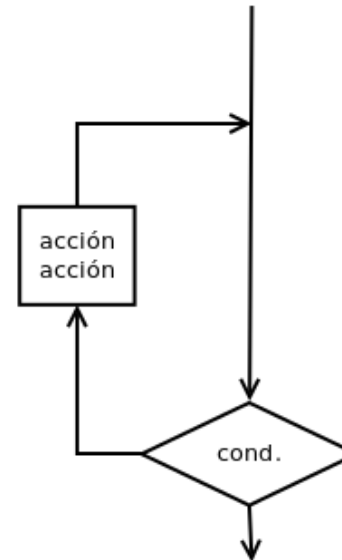
- En la presentación anterior se comentó que existen tres tipos de estructuras básicas para el flujo en nuestros programas:



**Secuencial**



**Selección** (o bifurcación)



**Iteración** (o bucle).

# Selección o bifurcación



- La estructura de **selección** o **bifurcación**, que es la que trataremos en esta presentación, permite ejecutar unas instrucciones u otras en función de una condición.

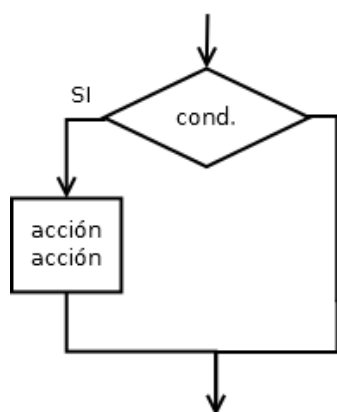


# Selección o bifurcación

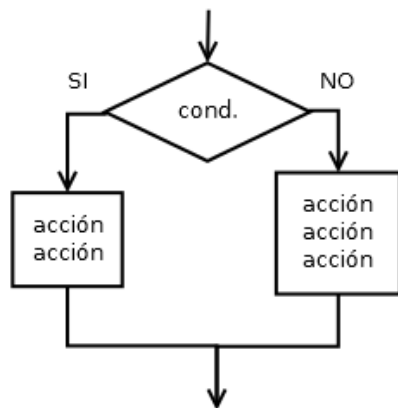


- Existen varios tipos de bifurcaciones:
  - **Bifurcaciones simples:** las instrucciones solamente se ejecutan si se cumple la condición
  - **Bifurcaciones dobles:** en función de si la condición se cumple o no, se ejecutan unas instrucciones u otras.
  - **Bifurcaciones múltiples:** la expresión a evaluar puede retornar múltiples valores y se harán unas operaciones u otras en función del valor que tome dicha expresión.

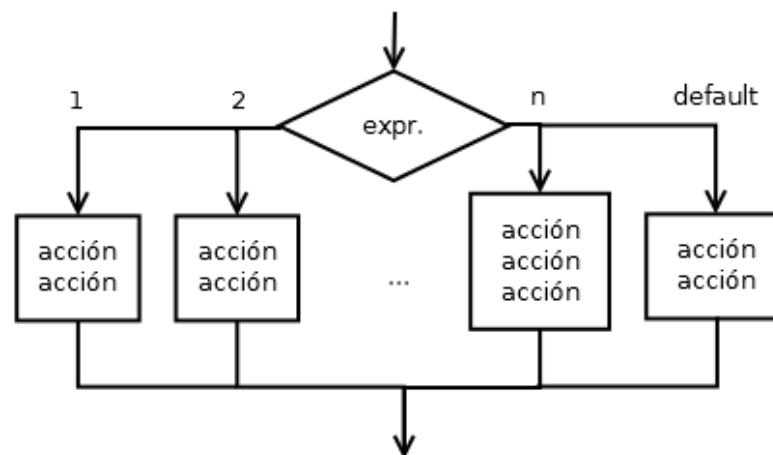
# Ejemplo tipos de bifurcación



Bifurcación simple



Bifurcación doble



Bifurcación múltiple

# Operadores relacionales

---

Operadores para usar en comparaciones



# Operadores relacionales



- Antes de hablar de bifurcaciones, vamos a mencionar los **operadores relacionales**, que necesitaremos para expresar las condiciones.
- Estos operadores permiten conocer la relación que existe entre dos valores (igualdad, mayor que, menor que...) .
  - Devuelven **verdadero** (*true*) o **falso** (*false*), esto es *boolean*.
  - Se usan mucho en la toma de decisiones, en bifurcaciones y bucles.
  - Aunque se pueden aplicar sobre distintos tipos de datos, por ahora los usaremos con números.



# Operadores relacionales



Operador	Significado
<	Menor que
<=	Menor o igual a
>	Mayor que
>=	Mayor o igual a
==	Igual a
!=	Diferente de
===	Idéntico a (valor y tipo)
!==	No idéntico a

# Ejemplo operadores relacionales (1/2 JS)



```
<head>
  <meta charset="utf-8">
  <title>Operadores relacionales</title>
  <script>
    function compara(){
      // tomar los valores de los inputs
      var n1 = parseFloat(inNumero1.value);
      var n2 = parseFloat(inNumero2.value);

      // si n1 es igual que n2
      → if(n1==n2){
        // mostrar que son iguales
        outResultado.innerHTML='IGUALES';

        // en caso contrario
      }else{
        // mostrar que son diferentes
        outResultado.innerHTML='DIFERENTES';
      }
    }
  </script>
</head>
```

# Ejemplo operadores relacionales (2/2 HTML)



```
<body>
  <h1>COMPARACIÓN</h1>

  <span>Numero 1:</span>
  <input type="number" id="inNumero1">
  <br>
  <span>Numero 2:</span>
  <input type="number" id="inNumero2">
  <br>
  <button onclick="compara()">Compara</button>
  <br>
  <span>Resultado:</span>
  <output id="outResultado"></output>
</body>
```

# Ejemplo resultado



## COMPARACIÓN

Numero 1:

Numero 2:

Compara

Resultado: **DIFERENTES**

## COMPARACIÓN

Numero 1:

Numero 2:


Compara

Resultado: **IGUALES**

# Ejercicio



- Modifica el ejemplo anterior usando distintos operadores relacionales en la comparación.



```
if(n1==n2){  
    // mostrar que son iguales  
    resultado.innerHTML='IGUALES';  
}
```

- Cambia también los mensajes mostrados para que todo tenga sentido, por ejemplo si usas el operador `<`, el mensaje mostrado debería ser “el primer número es más pequeño”.

# Bifurcación simple

---

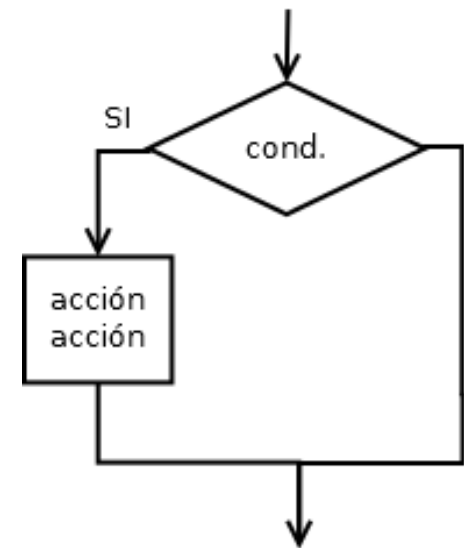
*If*



# Bifurcación simple



- En una **bifurcación simple**, solamente se realizarán las operaciones en caso que se cumpla la condición.
- Si no se cumple la condición, no se hace nada y se sigue con las instrucciones que se encuentren por debajo.
- En *JavaScript*, como en muchos otros lenguajes, usaremos una sentencia de tipo `if`.



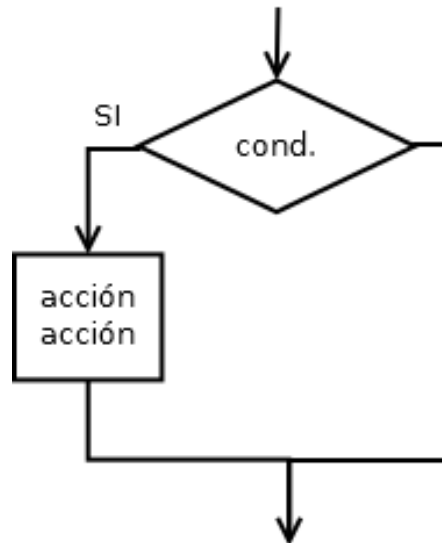
# Ejemplo sintaxis de la bifurcación simple



pseudocódigo

```
Si cond. entonces  
    acción  
    acción  
Fin si
```

diagrama de flujo

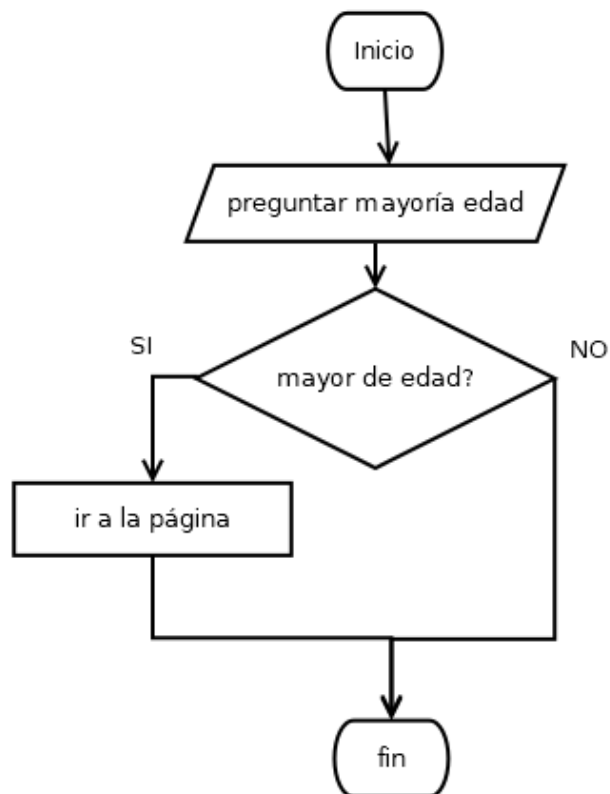


implementación

```
if(cond){  
    acción;  
    acción;  
}
```



# Ejemplo bifurcación simple (1/2)



# Ejemplo bifurcación simple (2/2)



```
<script>
  function entrar(){
    //pregunta si eres mayor de edad
    var mayor = confirm('Eres mayor de edad?');

    //si lo eres
    if(mayor){
      //te lleva a la página
      location.href="https://es.wikipedia.org/wiki/Resident_Evil";
    }
  }
</script>
```

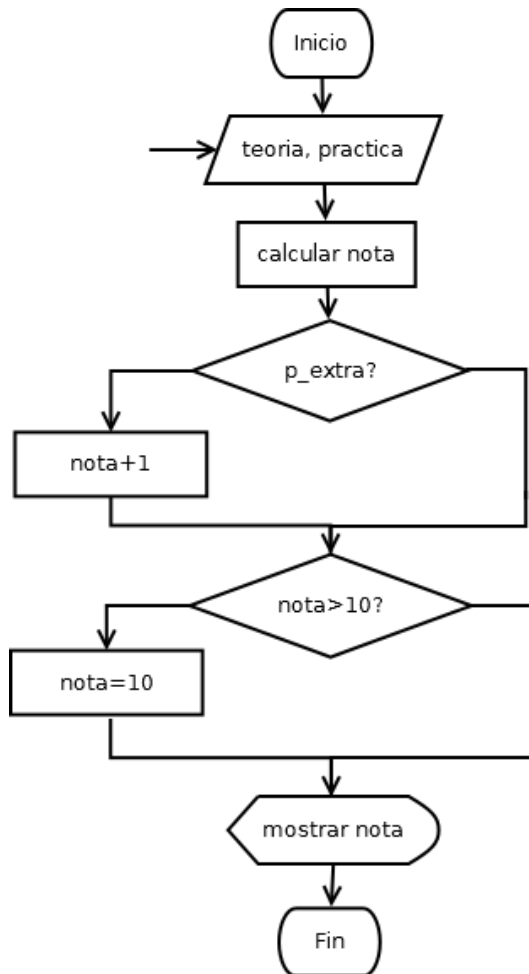
```
<p>Haz clic en el botón para ir a la página.</p>
<button onclick="entrar()">Resident Evil</button>
```

# Ejercicio



- En la siguiente diapositiva encontrarás un ejemplo de función que contiene dos bifurcaciones simples.
  - Se trata de un programa que calcula la nota final del curso a partir de la nota de teoría, la de práctica y el punto extra en un trabajo.
- Haz la interfaz gráfica para que ese programa funcione.
- Observa que necesitarás dos *inputs* de tipo numérico para las notas de teoría y práctica, así como un input de tipo *checkbox* para indicar el punto extra y un *output* para el resultado.

# Ejercicio diagrama y código JS



```
<script>
function calcular(){
    // tomar las notas de los inputs
    var teoria = parseFloat(inTeoria.value);
    var practica = parseFloat(inPractica.value);

    var nota = teoria*0.3+practica*0.7; // calcula

    if(inExtra.checked) // si hay punto extra...
        nota++;

    if(nota>10) // si la nota es > 10 ...
        nota = 10;

    outNota.innerHTML = nota; // pone el resultado
}
</script>
```

# Ejercicio resultado



## Calcular nota

Nota teoría:

Nota práctica:

☐ Punto extra por entregar trabajo

Calcular

Tu nota es: 6.5

## Calcular nota

Nota teoría:

Nota práctica:

☒ Punto extra por entregar trabajo

Calcular

Tu nota es: 7.5

# Bifurcación doble

---

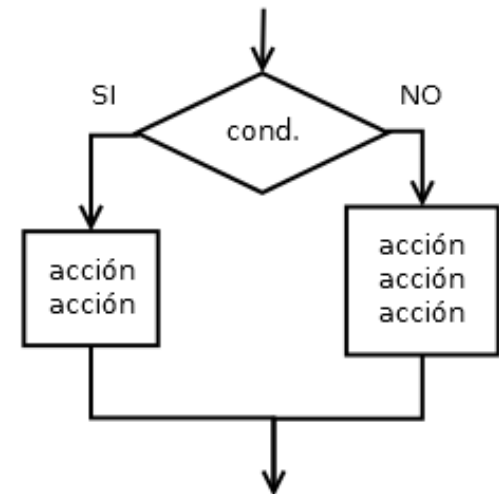
*If ... else*



# Bifurcación doble



- En una **bifurcación doble**, se realizan unas operaciones u otras en función de si se cumple la condición o no.
- La estructura que debemos utilizar, tanto en *JavaScript* como en muchos otros lenguajes, es la de un *if* con *else*.



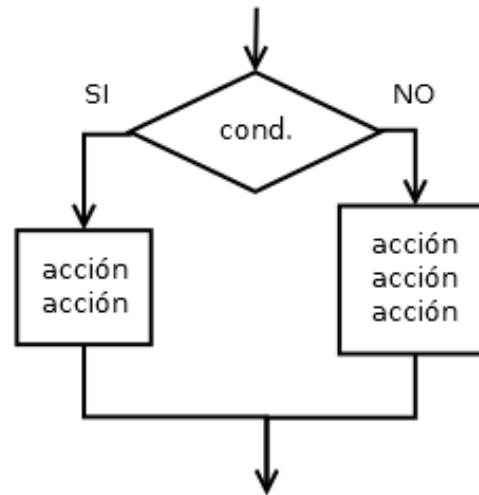
# Ejemplo bifurcación doble



## pseudocódigo

```
Si cond. entonces  
  acción  
  acción  
Si no entonces  
  acción  
  acción  
  acción  
Fin si
```

## diagrama de flujo

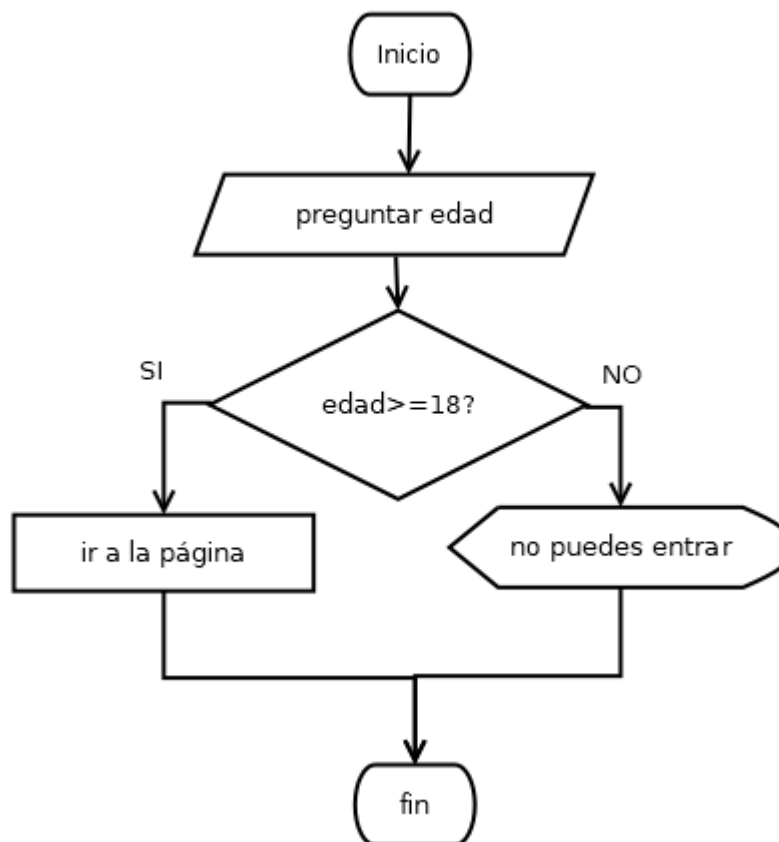


## implementación

```
if(cond){  
  acción;  
  acción;  
}else{  
  acción;  
  acción;  
  acción;  
}
```



# Ejemplo bifurcación doble (1/2)



# Ejemplo bifurcación doble (2/2)



```
<script>
  function entrar(){
    //pregunta tu edad
    var edad = parseInt(prompt('Indica tu edad'));

    //si lo eres
    if(edad>=18){
      //te lleva a la página
      location.href="https://es.wikipedia.org/wiki/Resident_Evil";

      //en caso contrario
    }else{
      alert('No puedes visitar la página!');
    }
  }
</script>
```

```
<p>Haz clic en el botón para ir a la página.</p>
<button onclick="entrar()">Resident Evil</button>
```

# Consideración con las llaves



- Como hemos visto, tanto el bloque `if` como el bloque `else` van acompañados de llaves `{ }`, de esta forma se marca el inicio y final.
- Sin embargo, si solamente existe una instrucción en el interior, se pueden omitir las llaves:

```
if(edad>=18)
    location.href="https://juegayestudia.com";
else
    alert('No puedes visitar la página!');
```

- Por ahora os aconsejaré que pongáis siempre las llaves.

# El operador ternario ?:



- Existe también un operador condicional ternario ?: que, permite reducir el número de líneas que escribimos en bifurcaciones sencillas.

- Su sintaxis es:

```
salida = condición? 'valor si verdadero': 'valor si falso';
```

- Lo que sería equivalente a:

```
if(condicion)
    salida = 'valor si verdadero';
else
    salida = 'valor si falso';
```

# Ejemplo operador condicional ?:



```
<script>
function entrar(){
    // pregunta tu edad
    var edad = parseInt(prompt('Indica tu edad'));

    // operador ternario --> condición? valor si verdadero : valor si falso
    // te llevará a una u otra página dependiendo de si se cumple la condición
    location.href = edad>=18? "https://juegayestudia.com" : 'http://disney.es';
}
</script>

<button onclick="entrar()">JUEGA Y ESTUDIA</button>
```

Condición

Valor si verdadero

Valor si falso

# Bifurcaciones anidadas

---

## Bifurcaciones dentro de bifurcaciones

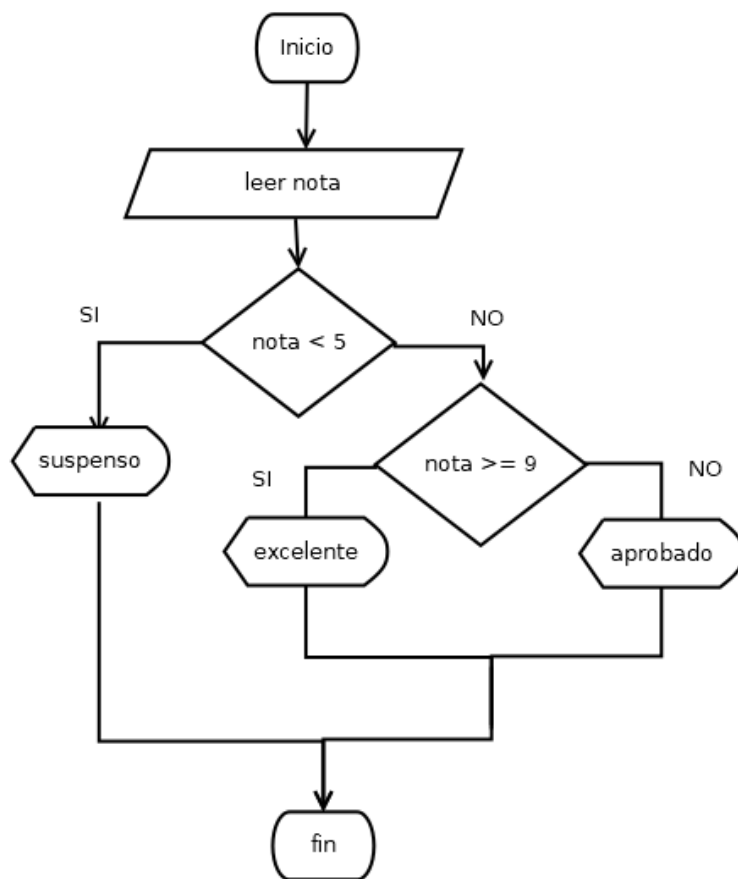


# Bifurcación anidada



- Se pueden anidar bifurcaciones unas dentro de otras, es decir, puede haber un *if* o *if-else* dentro de otro bloque *if* o *else*.
  - Este tipo de estructuras son muy habituales, aunque ahora os puedan resultar algo complicadas.
  - Permiten tener más de dos alternativas diferentes en algún punto del flujo de nuestro programa.
- A continuación se muestra un ejemplo, pero podéis intentar realizarlo a modo de ejercicio, observando el diagrama de flujo y la interfaz gráfica mostrada.

# Ejemplo bifurcación anidada (1/3)



Nota:

Calcular

Tu evaluación final es: **aprobado**



# Ejemplo bifurcación anidada (2/3)



```
<span>Nota:</span>
<input type="number" id="inNota">
<button onclick="calcular()">Calcular</button>
<br>
<span>Tu evaluación final es:</span>
<output id="outCalificacion"></output>
```

# Ejemplo bifurcación anidada (3/3)



```
<script>
  function calcular(){

    var nota = parseFloat(inNota.value); // tomar la nota

    if(nota<5){ // si la nota es inferior a 5...
      outCalificacion.innerHTML='suspense';
    }else{      // en caso contrario

      if(nota>=9){ // si la nota es igual o mayor a 9
        outCalificacion.innerHTML='excelente';
      }else{      // en caso contrario (de 5 a 8.999)
        outCalificacion.innerHTML='aprobado';
      }
    }
  }
</script>
```

# else if



- Podemos usar una combinación `else if` de la siguiente forma, permitiendo expresar una nueva condición en el caso que la anterior fuera falsa:

```
function calcular(){  
    var nota = parseFloat(inNota.value);  
  
    if(nota<5){  
        outCalificacion.innerHTML='suspense';  
    }else if(nota>=9){  
        outCalificacion.innerHTML='excelente';  
    }else{  
        outCalificacion.innerHTML='aprobado';  
    }  
}
```

# *else if*



- Personalmente no suelo fomentar el uso del *else if*, puesto que creo que puede llegar a ser algo confuso para varios niveles de anidamiento.
- La estructura *if else* con otros *if else* anidados se corresponde más fielmente al diagrama de flujo y creo que resulta más simple cuando se está comenzando a programar.
- La elección, sin embargo, es personal así que usad el *else if* si os gusta más.

# Switch

---

## Bifurcación múltiple



# Bifurcación múltiple



- Como hemos visto, anidando bifurcaciones se permiten más de dos caminos alternativos para nuestro flujo de programa.
- Sin embargo, estas estructuras de múltiples caminos hechas con *ifs* anidados a veces pueden resultar complicadas de pensar, de escribir y poco eficientes en cuanto a rendimiento.
- La mayoría de lenguajes de programación cuentan con la estructura de control de flujo **switch** para implementar las **bifurcaciones múltiples**.

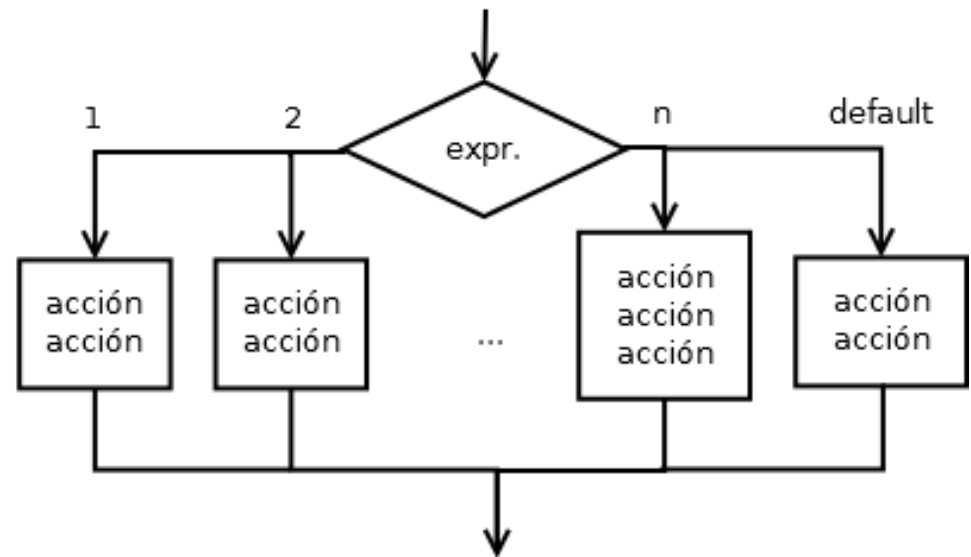
# Ejemplo bifurcación múltiple



pseudocódigo

según expr. hacer  
caso 1: acciones  
caso 2: acciones  
...  
de otro modo: acciones  
fin según

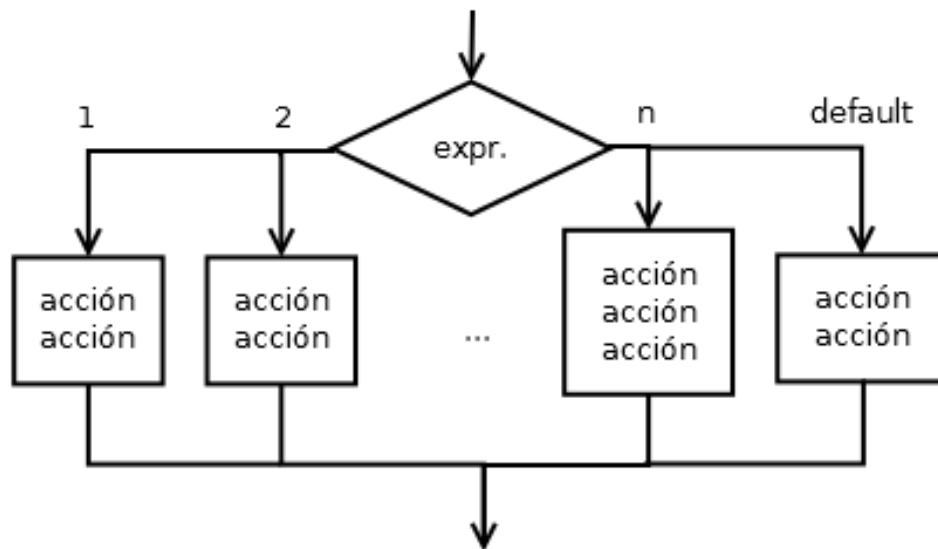
diagrama de flujo



# Ejemplo bifurcación múltiple



diagrama de flujo



implementación

```
switch(expr){  
  case 1: acción;  
          acción;  
          break;  
  
  case 2: acción;  
          acción;  
          break;  
  
  ...  
  
  default: acción;  
           acción;  
  
}
```



# Ejemplo



- En el siguiente ejemplo vamos a crear una pequeña aplicación que calcule el precio de la entrada para un partido en función de la ubicación seleccionada.
- El precio base será de 20 euros, a partir de ahí calculará un sobrecoste de la siguiente forma:
  - Los goles no tienen sobrecoste.
  - El lateral es un 50% más caro.
  - Tribuna baja es el doble de cara y tribuna alta 2.5 veces más.
  - El palco cuesta cinco veces más que la entrada más barata.



# Ejemplo bifurcación múltiple (1/2 HTML)



```
<h1>Calcula el precio</h1>

<span>Selecciona ubicación: </span>
<select id="inUbicacion">
  <option value="1">Gol Norte</option>
  <option value="2">Gol Sur</option>
  <option value="3">Lateral</option>
  <option value="4">Tribuna baja</option>
  <option value="5">Tribuna alta</option>
  <option value="6">Palco</option>
</select>

<button onclick="precio()">Calcular precio</button>
<br>

<span>El precio de tu entrada es: </span>
<output id="outPrecio"></output>
```

# Ejemplo bifurcación múltiple (2/2 JS)



```
<script>
function precio(){
    var precio = 20;

    // recupera la ubicación seleccionada (de 1 a 6)
    var ubicacion = parseInt(inUbicacion.value);

    switch(ubicacion){
        case 1: precio *= 1;
                break;
        case 2: precio *= 1;
                break;
        case 3: precio *= 1.5;
                break;
        case 4: precio *= 2;
                break;
        case 5: precio *= 2.5;
                break;
        default: precio *= 5;
    }

    outPrecio.innerHTML = precio+'€';
}
</script>
```

# Ejemplo resultado



## Calcula el precio

Selecciona ubicación:

El precio de tu entrada es: 20€

## Calcula el precio

Selecciona ubicación:

El precio de tu entrada es: 50€



# El caso *default*



- Si el *switch* dispone del caso ***default***, éste se ejecutará siempre que el valor evaluado en la expresión no coincida con ninguno de los casos anteriores.
  - En el ejemplo anterior se ha usado el *default* en lugar del caso 6.
  - Si llegara un 7, 8, 9... la entrada al partido valdría también 100 euros.
- Si el valor no coincide con ningún caso y no existe caso por defecto, no se hace nada. El programa continuará normalmente con el código que haya después del *switch*.

# La expresión



- En *JavaScript* se permite hacer `switch` con expresiones numéricas o de tipo *String* (cadena de texto).
- En el ejemplo anterior, usamos una expresión entera, así que indicamos cada caso mediante un número entero (sin entrecomillar): `case 1`.
- Si queremos comparar textos en lugar de números, debemos poner comillas (simples o dobles) en los casos, por ejemplo: `case "go1N"`.
- Hay lenguajes que por motivos de eficiencia, son más restrictivos y solamente permiten que las expresiones se puedan evaluar como número entero o carácter (por ejemplo en C).

# Ejemplo expresión de tipo texto (1/2 HTML)



```
<h1>Calcula el precio</h1>

<span>Selecciona ubicación: </span>
<select id="inUbicacion">
  <option value="golN">Gol Norte</option>
  <option value="golS">Gol Sur</option>
  <option value="late">Lateral</option>
  <option value="triB">Tribuna baja</option>
  <option value="triA">Tribuna alta</option>
  <option value="palc">Palco</option>
</select>

<button onclick="precio()">Calcular precio</button>
<br>

<span>El precio de tu entrada es: </span>
<output id="outPrecio"></output>
```



# Ejemplo expresión de tipo texto (2/2 JS)



```
<script>
function precio(){
    var precio = 20;

    // recupera la ubicación seleccionada
    var ubicacion = inUbicacion.value; ←

    switch(ubicacion){
        case "golN": precio *= 1;
                        break;
        case "golS": precio *= 1;
                        break;
        case "late":  precio *= 1.5;
                        break;
        case "triB":  precio *= 2;
                        break;
        case "triA":  precio *= 2.5;
                        break;
        default:      precio *= 5;
    }

    outPrecio.innerHTML = precio+'€';
}
</script>
```



# El *break*



- El **break** **marca el final de un caso**. Es importante no olvidarlo puesto que de hacerlo seguramente incurriremos en un error.
  - Si no ponemos **break**, cuando entremos en el caso, éste no finalizará y seguirá ejecutando las instrucciones del siguiente.
  - En el último caso no es necesario el **break**, puesto que no hay más instrucciones a continuación.
  - Hay situaciones en las que podemos omitir intencionadamente el **break**, por ejemplo para situaciones acumulativas o para indicar rengos dejando casos en blanco. **Esto no es lo habitual**.

# Ejemplo omitiendo intencionadamente el *break*



```
<script>
function precio(){
    var precio = 20;

    // recupera la ubicación seleccionada
    var ubicacion = inUbicacion.value;

    switch(ubicacion){
        case "golN":
        case "golS": break; ←
        case "late": precio *= 1.5;
                     break;
        case "triB": precio *= 2;
                     break;
        case "triA": precio *= 2.5;
                     break;
        default: precio *= 5;
    }
    outPrecio.innerHTML = precio+'€';
}
</script>
```

# Ejercicio omitiendo el *break*



- Implementa la interfaz gráfica para este programa y haz que funcione:

```
function comunidad(){
    var provincia = inProvincia.value.toUpperCase();

    switch(provincia){
        case "BARCELONA":
        case "GIRONA":
        case "LLEIDA":
        case "TARRAGONA": outResultado.innerHTML = "Catalunya";
                           break;
        case "BADAJOZ":
        case "CACERES":   outResultado.innerHTML = "Extremadura";
                           break;
        default : outResultado.innerHTML = "De otro lugar";
    }
}
```

# Ejercicio resultado



## Procedencia

Provincia:

Tu comunidad es: **Catalunya**

## Procedencia

Provincia:

Tu comunidad es: **De otro lugar**

# Ejercicios

---

## Selección o bifurcación



# Ejercicios



- Para la realización de los siguientes ejercicios debes realizar los pasos siguientes:
  - Dibujar un **croquis** de la interfaz gráfica.
  - Indicar el **tipo e identificador** de los elementos con los que interactuará el usuario.
  - **Definir el algoritmo** (pseudocódigo o diagrama de flujo).
  - **Implementar** el programa.
  - Realizar **pruebas**, anotando y corrigiendo los fallos que encuentres.

# Ejemplo diseñando la interfaz gráfica



**Calculadora de IVA**  
Espe's Bussines S.L.

Importe sin IVA  euros ← input #inPrecio

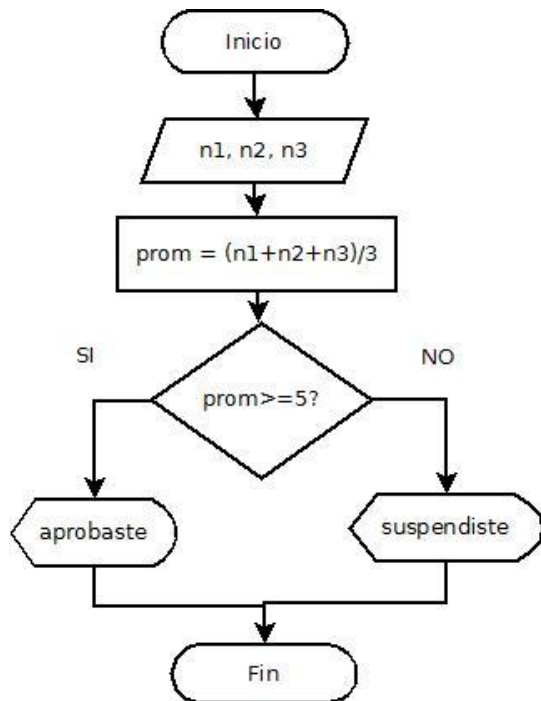
General: 21% ▼ ← select #inIva

Importe final: 121 euros ← output #outPrecio

# Ejercicios



1. Implementa el programa descrito a continuació:



Nota 1:	<input type="text" value="6"/>
Nota 2:	<input type="text" value="5"/>
Nota 3:	<input type="text" value="10"/>
<input type="button" value="Calcular"/>	
APROBADO	

Nota 1:	<input type="text" value="4"/>
Nota 2:	<input type="text" value="5"/>
Nota 3:	<input type="text" value="3"/>
<input type="button" value="Calcular"/>	
SUSPENSO	



# Ejercicios



2. Se quiere un programa que calcule el nuevo salario de los trabajadores de una empresa siguiendo el siguiente criterio:
- A los trabajadores de nivel 1 se les aumenta el sueldo un 30%.
  - A los de nivel 2 se les aumenta un 5%.
  - A los de otros niveles, que no son el 1 o el 2, se les baja un 7%.

**Calculadora**

Salario:

Nivel:

Tu nuevo salario es: **2600**

**Calculadora**

Salario:

Nivel:

Tu nuevo salario es: **1680**

**Calculadora**

Salario:

Nivel:

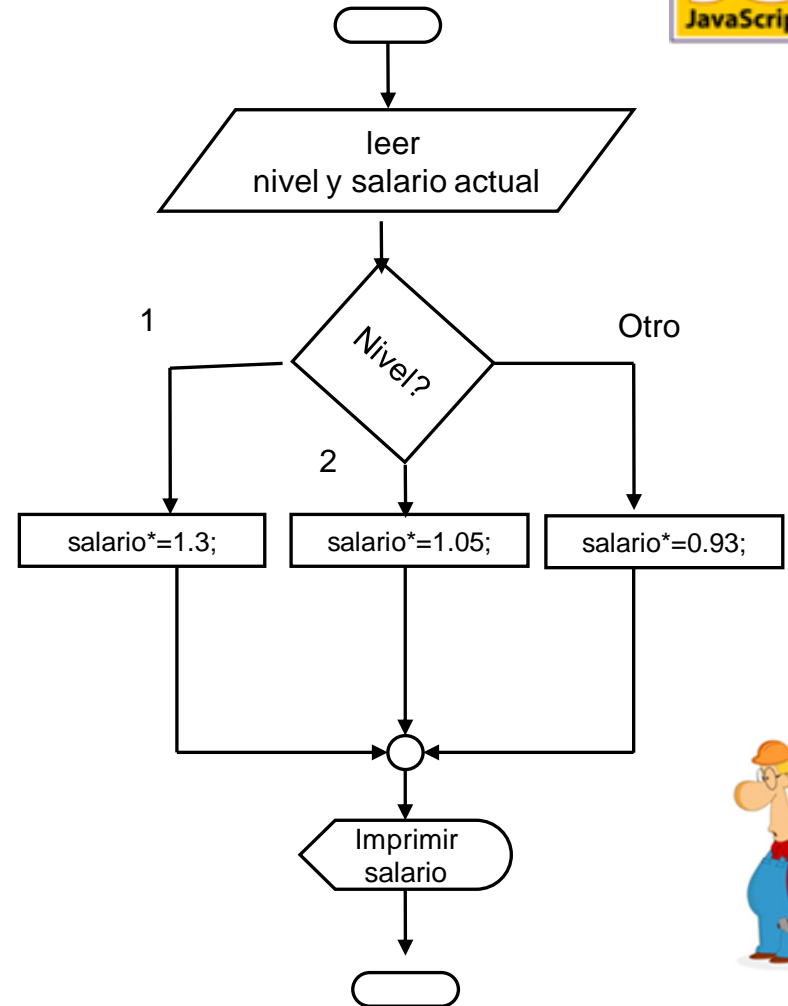
Tu nuevo salario es: **1116**



# Ejercicios



- Puedes utilizar el siguiente diagrama de flujo como punto de partida.
- Si se te ocurre un diseño mejor, lo puedes implementar.



# Ejercicios



3. Modifica la web que hiciste que calculaba el perímetro, área y diámetro de una circunferencia, de manera que ahora muestre un desplegable (select) que permita seleccionar solamente una de las tres operaciones.

Radio:

Operación:

Area

▼

Calcular

El resultado es

314.1592653589793

# Ejercicios



4. Implementa una nueva versión de la web de los cálculos con círculos, en la que se puedan seleccionar las operaciones mediante *checkbox* (una o más). No sea buena idea usar un *switch*.

Radio:

☐ Área  
☒ Perímetro  
☒ Diámetro

El área es  
El perímetro es 62.83185307179  
El diámetro es 20

# Ejercicios



5. Crea un programa que pida tres números y nos imprima por pantalla el mayor de los tres.

Número 1:

Número 2:

Número 3:

El mayor es el **80**

Haz el diagrama de flujo antes de comenzar con la implementación.

# Ejercicios



6. Necesitamos una aplicación para calcular las notas de los test de teoría que haremos en el curso.
- Se tiene que poder indicar **el número de preguntas total** del test, así como el **porcentaje de descuento por pregunta fallada** (0%, 25%, 33%, 50%, 66%, 75% o 100%), las no contestadas no descuentan.
  - Al introducir el **total de errores y aciertos**, se deberá mostrar el **número de preguntas no contestadas**, la **nota sin descontar los fallos** y **nota final** tras descontarlos.
  - También **se indicará el texto**: “aprobado”, “suspense” o “excelente” (9 a 10).

# Ejercicios



## Calculadora de notas

Para calcular el resultado de los test

### Entrada de datos

**Parámetros**

PREGUNTAS:

DESCUENTO POR ERROR:

**Resultados**

ACIERTOS:

ERRORES:

Aciertos: 30

Errores: 5

No Contestadas: 15

Nota sin descontar: 6

Final sobre 10: **5.67**

**APROBADO**

Robert Sallent - L'Estudem

# Ejercicios



7. Realiza una web que sirva para resolver ecuaciones de segundo grado en forma  $ax^2+bx+c=0$ . Se deben contemplar los siguientes casos:
- Si se introduce un valor no numérico, no se puede realizar el cálculo.
  - Hay ecuaciones de segundo grado que pueden no tener solución real.
  - Si el valor de “a” es 0, no se trata de una ecuación de segundo grado.

*Sigue →*



# Ejercicios



- Se debe realizar el diagrama de flujo antes de comenzar con la programación.
- Si no recuerdas cómo se resolvían ecuaciones de segundo grado, documéntate por tus propios medios. Básicamente es aplicar la fórmula, aunque es buena idea usar el discriminante (lo que se encuentra dentro de la raíz) para determinar el número de soluciones.
- Si tienes que realizar raíces cuadradas, puedes usar `Math.sqrt()` ; .

# Ejercicios



8. Vamos a hacer el **juego de adivinar el número**.

El juego consiste en que la máquina piensa un número ENTERO aleatorio (`Math.random()`) entre 1 y 128 y nosotros lo tenemos que adivinar en el menor número de intentos posibles.

Cada vez que no acertemos, **se nos indicará si el número es mayor o menor**.

*Opcional: haz que se recuerde el número de intentos, si se necesitan más de 7 para adivinar el número, el programa te dirá “qué malo eres!”.*

# Ejercicios



**Adivina el numero**

Número:

---

Resultado: Acertaste  
Intentos: 7

An illustration at the bottom of the form showing a group of people in silhouette with their arms raised in celebration, and a cluster of four balloons (blue, orange, red, and yellow) on the right side.