Programación Web Básica.

UNQ

Prof. Jorge Nicolás Terradillos
Prof. Fernando Blanco

En la clase de hoy...

-Bucles y contadores

-Objetos en JavaScript

-Propiedades y metodos

-Objeto DOCUMENT

01

Bucles lógicos

Repetición

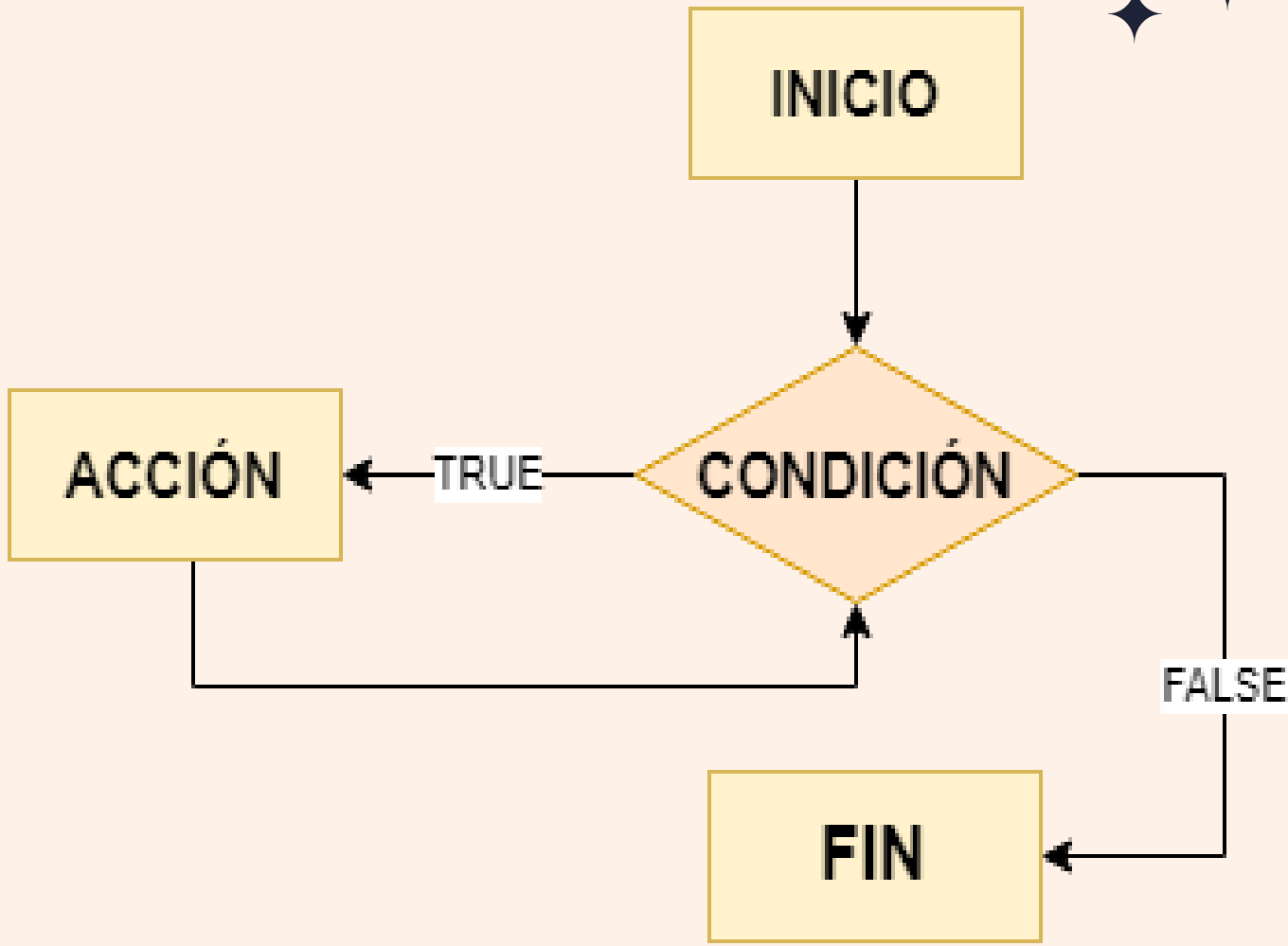
¿Cómo podríamos hacer una función que imprimiera varias veces un mismo valor?

```
function imprimirMiValor(miValor){
  console.log(miValor);
  console.log(miValor);
  console.log(miValor);
...
}
```

Tenemos que diseñar un **bucle** que se repita tantas veces como lo necesitamos

Bucles

En JavaScript tenemos varias estructuras de bucles, entre ellas el “while”, donde dada una condición booleana (como con el if), se ejecuta un bloque de código mientras esa condición siga siendo verdadera



```
function imprimirMiValor(miValor){
   while (true){
      console.log(miValor);
    }
}
```

Aquí entraremos a un bucle infinito ya que la condición true siempre será true

Acumulador

Para detener el bloque iniciamos un contador de las veces que se repite y cada vez que se repite, este contador aumenta en 1

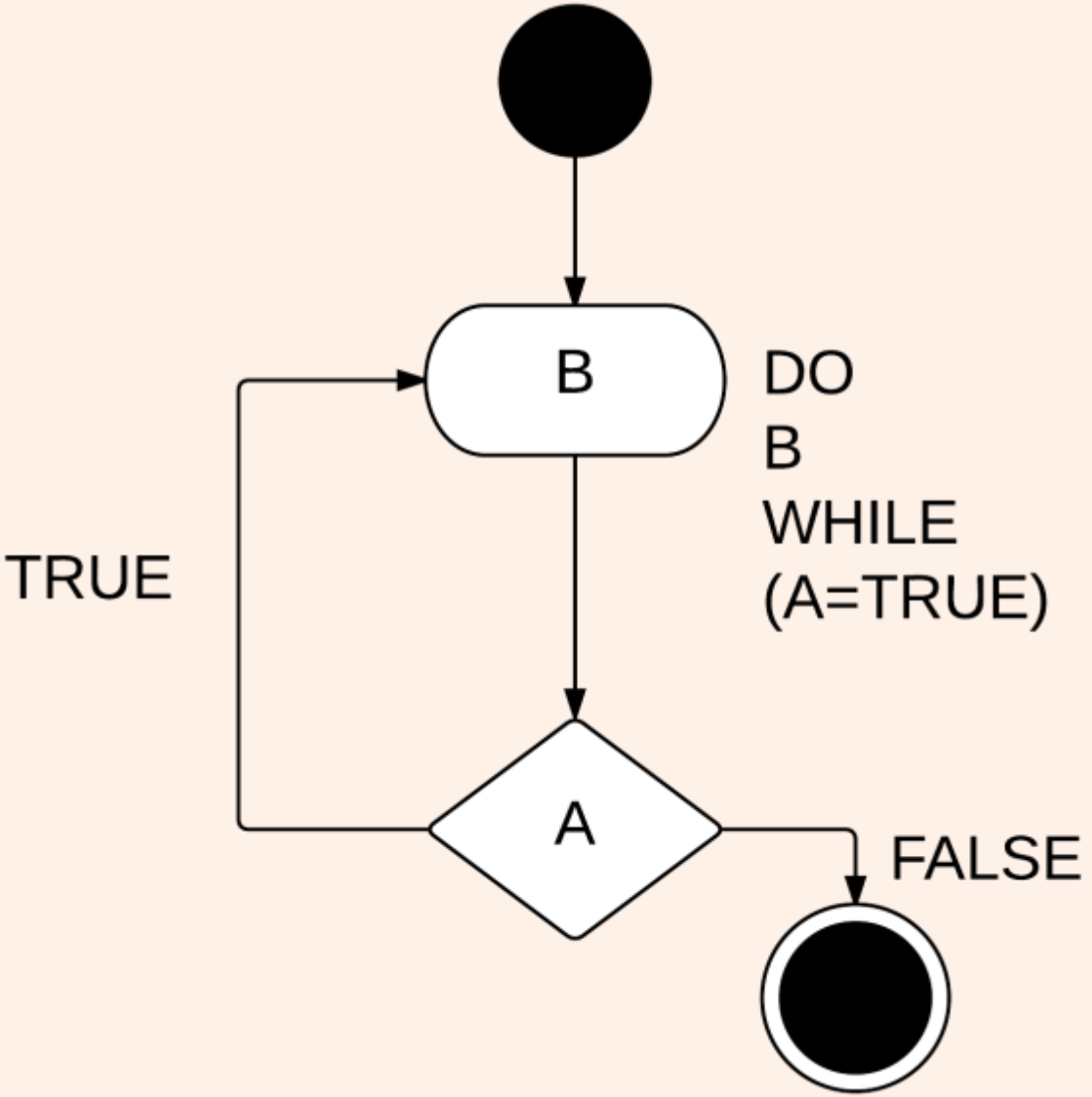
```
function imprimirMiValor(miValor){
  let contador = 0;
  while (contador<5){   //hace que se detenga cuando llega a 5
      console.log(miValor);
      contador++;
    }
}
```

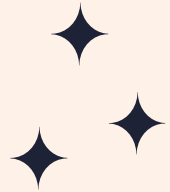
```
function imprimirXVeces(miValor,cantidad){
   let contador = 0;
   while (contador<cantidad){
//podemos pasar la cantidad de repeticiones por parametro
        console.log(miValor);
        contador++;
    }
}
```

Do While

También se puede usar la sintaxis “do {bloque de código} while (condición)”

La diferencia es que así primero ejecuta el código y después revisa si cumple la condición





```
let result = “”;
let i = 0;
do {
  i = i + 1;
  result = result + i;
}
while (i < 5);
console.log(result); // resultado: "12345"
```

Actividad

-Punto 1

Crear una función “imprimirArray” que, tomando un array, imprima todos los elementos que contiene.

-Punto 2

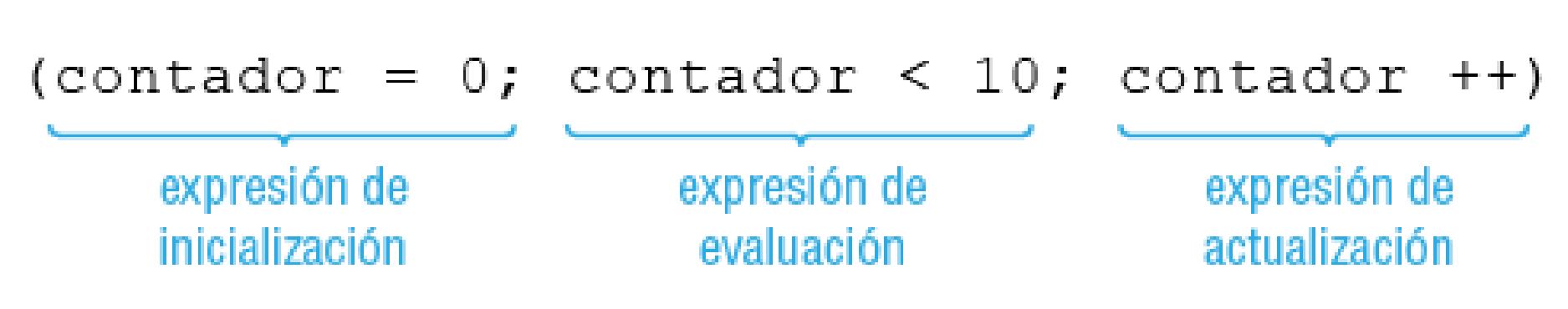
Crear una función “caracteresTotales” que, tomando un array de strings, nos sume que tan largo son los strings que contiene en total

-Punto 3

Crear una función “limpiarDeImpares” que, tomando un array de números, lo retorne sin los números impares (pueden usar una función auxiliar que verifique si un numero es impar)

Contador

Usar acumuladores (como en los casos anteriores) es muy común, pero si sabemos cuantas veces queremos repetir un bloque de código se suele usar otra estructura de bucle: **for**



El contador va en los paréntesis que siguen al for y hay que colocar a partir de cuanto se cuenta, hasta cuanto y de que forma va contabilizando (sumando o restando)

Ejemplo 1:

```
for (i = 0; i < 5; i++) {
//"i" es el contador, aumenta de uno en uno hasta llegar a 5
  console.log("Valor de i:", i);
}
```

Ejemplo 2:

```
misCoches = ["Ferrari","Ford","Seat","Audi","Tesla"];
   for (contador = 0; contador < misCoches.length; contador++) {
      console.log("Coche de la marca: " + misCoches[contador]);
}
```

También podemos usar más de un contador a la vez:

```
for (i = 0, j = 5; i < 5; i++, j--) {
        console.log("Valor de i y j:", i, j);
}
```

Si i++ aumenta en 1 el valor de i en cada iteración, lo que hace j-- es disminuir en 1 el valor de j en cada iteración.

Bucles en arrays

```
var miArray = [“Hola",“Mundo"];
var largoDeArray = miArray.length;
for (var i = 0; i < largoDeArray; i++) {
    console.log(miArray[i]);
    //Imprimir palabra por palabra del array
}
```

02

Objetos en JavaScript

Agrupar valores

Supongamos que tenemos estas variables que representan un auto:

```
var modelo = “Clio”;
var año = 2016;
var kilometros = 50000;
var patente = “AJJ720”;
```

Si quisiéramos tratar todas estas variables como parte de una misma entidad, deberemos agruparla en un **objeto**

Concepto de objetos

Un objeto es una colección de propiedades (valores que estan asociados con ciertos nombres), que representa una **entidad real** de forma **abstracta**

```
let auto = {
 modelo: “Clio”;
 año: 2016;
 kilometros: 50000;
 patente: “AJJ720”;
}
```

Por ejemplo, en vez de tener muchas variables para el auto, lo englobamos en un ente abstracto de “auto”

Propiedades

Podemos acceder a las propiedades de los objetos llamándolas por su nombre, así como también podemos modificarlas:

```
console.log(auto.modelo); // «Clio»

auto.modelo = “Megane”;

console.log(auto.modelo); // «Megane»
```

Métodos

Cada objeto puede tener funciones específicas, conocidas como «métodos», que se definen dentro del objeto. Primero definimos el objeto, luego sus propiedades y métodos

```
let auto{
 modelo: “Clio”;
 año: 2016;
 arrancar: function(){
     console.log(“Brooom”);
   }
}
```

Actividad

-Punto 1

Creemos el objeto «telefono» que tiene por propiedad un número, contactos (un array con strings) y un proveedor (un string con el nombre de la empresa proveedora).

-Punto 2

Crear dos métodos para teléfono: uno que se llame «sonar» y que haga que la consola imprima «Ring ring ring» y otro que sea «llamar» y que tome un número, busque en sus contactos ese dirección dentro del array e imprima en la consola «Llamando a [nombre de la dirección]». Por ejemplo, si tengo de contactos a [“Martín”, “Lucas”, “Karen”] y hago telefono.llamar(1) me devuelve “Llamando a Lucas”.

03

Objetos HTML

Objeto document

Para JavaScript, una página HTML de por sí es un objeto al que podemos acceder con sus propias propiedades y métodos: el objeto «**document**»

```
console.log(document.title)
//retorna el título de nuestra página
```

Podemos acceder directamente a title porque es parte de los metadatos, con otras partes del documento

También podemos ir preguntando por distintos elementos dentro del objeto «document»

```
console.log(document.url)
//muestra la dirección de nuestra página

console.log(document.body)
//muestra el elemento body completo de la página

console.log(document.images)
//muestra un array con todos los elementos <img> de la página
```

Buscar elementos

Si queremos ver un elemento HTML específico con un ID concreto se usa el getElementById(), al cuál se le pasa el id que conocemos como un string

```
<body>
  <h1 id="encabezado">Mi página</h1>
  <p>Blablablablabla</p>
<script>
   document.getElementById("encabezado");
</script>
</body>
```

Aclaración: ahora deberíamos dejar el script al final del body porque así se renderizan primero los elementos y después los puede tomar JS

Modificar elementos

Una vez que buscamos un elemento, podemos revisar y cambiar su contenido con el método «innerHTML»

```
<script>
    let nuevoTitulo = “Mi Nueva página!”;
    document.getElementById(“encabezado”).innerHTML = nuevoTitulo;
</script>
```

Todas estas acciones también se puede utilizar en funciones o métodos de objetos del script

Escribir página

Por otro lado, podemos escribir directamente sobre el documento con el comando document.**write**(«aquí va lo que se quiera escribir»)

```
<script>
    document.write(“<h1>Agregar una orden</h1>”);
</script>
```

Aquí depende donde aparecerá lo nuevo si tenemos el script al principio o al final del body

Crear elementos

Otra de las cosas que se pueden hacer es crear los elementos desde el mismo script con el comando document.**createElement**

```
let nuevoElemento = document.createElement("p");
/*creamos el elemento pasandole un tipo de HTML y lo guardamos en una variable*/

nuevoElemento.innerHTML = "Creamos un nuevo elemento!";
//aquí agregamos texto al contenido
```

Agregar elementos

Con lo anterior creamos el objeto, pero no especificamos dónde aparecen. Para eso usaremos el metodo «appendChild», que tiene la siguiente sintaxis:

```
<script>
 let nuevoElemento = document.createElement("p");
 nuevoElemento.innerHTML = "Creamos un nuevo elemento!";
 document.body.appendChild(nuevoElemento);
</script>
```

La próxima analizaremos en detalle este método

Resumiendo

Bucles	Contadores	Objetos	Document	Elementos
Para repetir acciones podemos usar bucles que respete condiciones lógicas(while y do while)	También podemos usar bucles con contadores (for) con cantidad de repeticiones más fijas	Entidades abstractas que reunen datos (propiedades) y funciones específicas (métodos)	El documento HTML en sí puede ser tratado como un objeto con sus propiedades y métodos	Los elementos HTML son interpretados en JS como propiedades del objeto document

Eso es todo

Se escuchan dudas, quejas y propuestas

Fin de la clase