# 2023-09-15

Declaración de variables en JS o ES:

  // hay cuatro formas de declarar variables:

  //las antiguas: 2009 - ES5

    edad1 = 18;

    var edad2 = 18;

  //las modernas: 2015 - ES6

      let edad3= 18;

      const edad4 = 18;

La primera forma esta PROHIBIDISIMO usarla

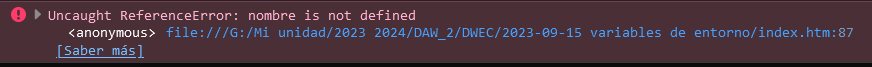
    function pruebaVariables(){

          nombre = 'Juan';

    }

//Aqui saldá un error en la siguiente linea,

  console.log(nombre);



A Adolfo, le veremos casi siempre usando la palabra reservada **const** para declarar variables, y nombrar a variables en **mayúsculas** si son **variables constantes** de toda la vida, ej.: *PI; numero e*, *IVA*; y **minúsculas** para el **resto de variables**.

# 2023-09-18

<script>

    'use strict';

    var nombre    = 'Javier';

    var apellidos = 'Ibarra Muñoz';

    var poblacion = 'Lorca';

    var pais      = 'España';

    //concatenación de cadenas

    var mensaje1 = 'Hola, soy ' + nombre + ' ' + apellidos + '. Vivo en ' + poblacion + '. Nací en ' + pais + '.';

    console.log(mensaje1);

    //interpolacion de cadenas - string literals - f-String

    var mensaje2 = `Hola, soy ${nombre} ${apellidos}. Vivo en ${poblacion}. Nací en ${pais}.`;

    console.log(mensaje2);

    function mostrarMensaje (nombre= "Adrian", apellidos= "García Puriano", poblacion = "Purias", pais = "Marruecos"){

        var mensaje2 = `Hola, soy ${nombre} ${apellidos}. Vivo en ${poblacion}. Nací en ${pais}.`;

        console.log(mensaje2);

    }

    mostrarMensaje();

    mostrarMensaje('Adrián', undefined, 'Purias', 'el hospital')

</script>

# 2023-09-20

## Arrays en JS

### Añadir elementos al Array

const frutas = new Array();

frutas.push('melón');

frutas.push('🥝');

frutas.push('🥑','🍋');

### Formas de recorrer un array:

**//1ra Forma: La de toda la vida, la de primero**

for (let i=0; i< frutas.length; i++){

    console.log(frutas[i]);

}

Ventaja, podemos empezar desde una posición requerida, podemos ir de uno en uno o de varios en varios, o incluso de atrás a delante.

**//2da forma: foreach de tipo in**

for (const i in frutas){ //Se puede usar let, se puede usar const…

    // console.log(`${parseInt(i)+1})`, frutas[i]);

    console.log(`${parseInt(i)+1}) ${frutas[i]}`);

}

**//3ra forma: foreach de tipo of**

for (const fruta of frutas){

    console.log(fruta);

}

Se corresponde con la que usábamos el año pasado en programación

#### Formas raras

**//4ta 1ra forma: métodos de la clase ARRAY**

function mostrarFruta(fruta){

    console.log(fruta);

    //Al no poner return, automaticamente es como si se ejecutase: 'return Undefined;'

}

frutas.forEach(mostrarFruta);

Este tipo de formas de recorrer el array, conforme avancemos en el curso, debería de ser la forma de recorrer los bucles que estaremos usando.

**//4ta 2da forma: metodo de la clase ARRAY, pero con la funcion en el método.**

frutas.forEach(

    function mostrarFruta(fruta){

        console.log(fruta);

    }

);

**//4ta 3ra forma: lambda**

frutas.forEach(

    function (fruta){

        console.log(fruta)

    }

);

A diferencia de la 4.2 forma, en este caso, la función no lleva nombre, es una función anónima, también conocida como **función lambda**: una función que no tiene referencia. Este tipo de funciones solo pueden ser usadas en situaciones donde necesitemos crear una función en un momento indicado y solo ahí. Se le puede llamar también **función flecha**. Explicada a continuación:

**//4ta 4ta forma: función flecha**

//      function (fruta){

//              console.log(fruta)

//      }

//paso 1: cortar el nombre

// function (fruta){

//     console.log(fruta)

// };

//paso 2: poner la flecha

// function (fruta) => {

//     console.log(fruta)

// };

// paso 3: borrar funcion

// (fruta) => {

//     console.log(fruta)

// };

//4to paso: sin hay un solo parámetro quitamos los paréntesis

// fruta => {

//     console.log(fruta)

// };

// 5to paso: si hay una sola línea de código, quitamos las llaves

// fruta =>

//     console.log(fruta)

// ;

// 6to si hay una sola línea la subimos y le quitamos el punto y coma

// fruta => console.log(fruta)

frutas.forEach(fruta => { console.log(fruta)});

**//5ta forma: crear constante con la función flecha**

const mostrarFruta = fruta => console.log(fruta);

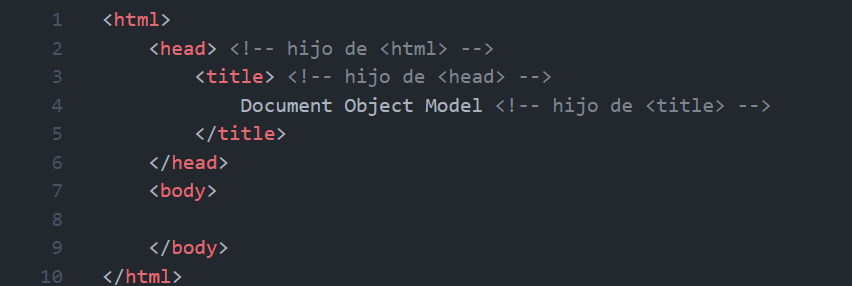
frutas.forEach(mostrarFruta);

# 2023-09-25 DOM Básico y Lista

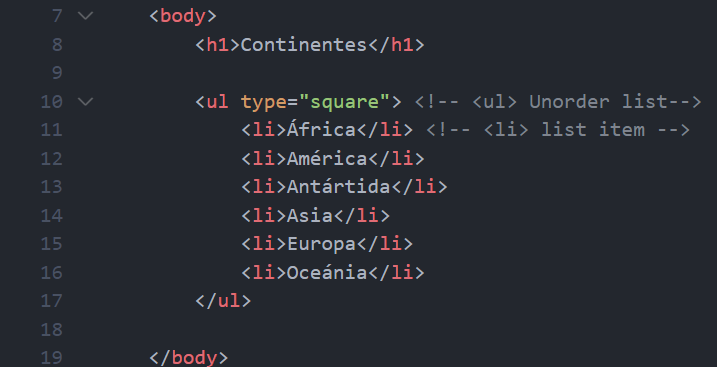
### Document Object Model

La raíz del árbol de HTML es la etiqueta <html>. Sus hijos son <head> y <body>.

<head> su hijo, <title> tiene como hijos las cadenas de texto:



### Listas:



# 2023-10-04 Eventos

Observable 🡪 Cualquier etiqueta que haya en la pantalla.

Hay entidades interesadas en el *observable*, y se llaman *Observers*.

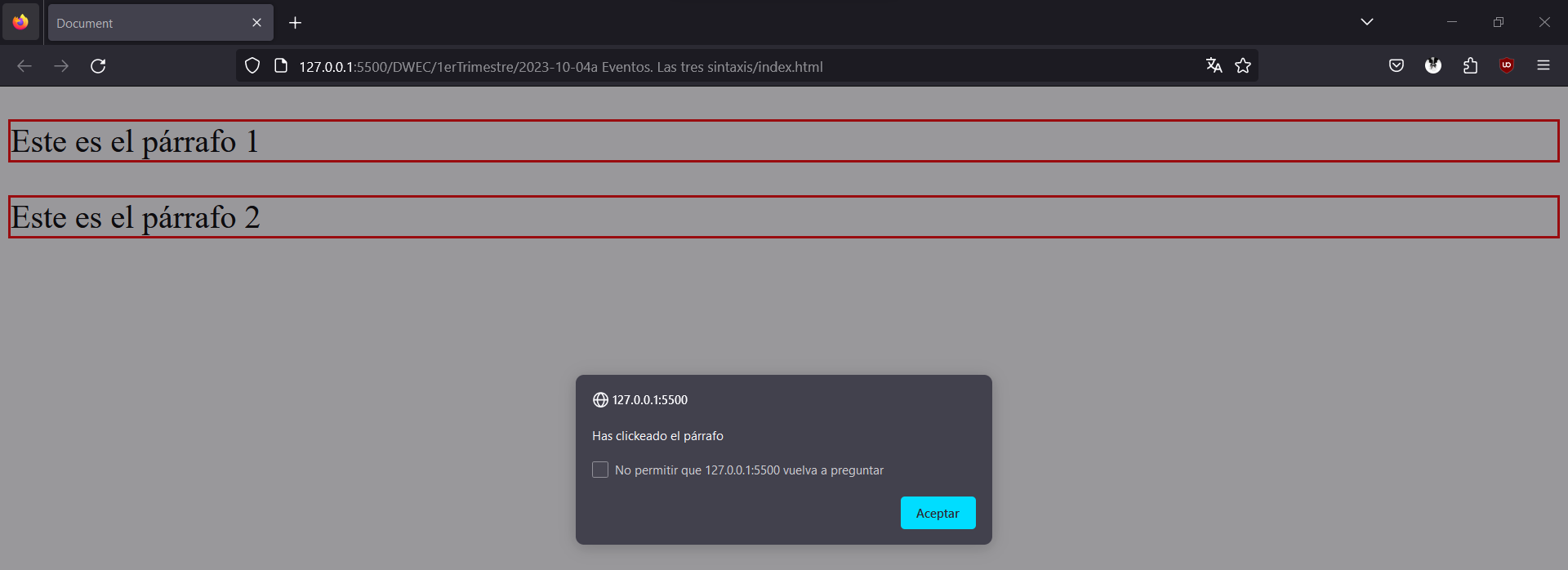
Observers

Los *Observer* reaccionan a un hecho de interés que ocurre en el *Observable*, cuando ocurren esos hechos de interés o Eventos, el *Observable* informa a los *Observers*. Esto ocurre especialmente en las interfaces que tienen que ocurrir muchas cosas.

Hay tres sintaxis:



Al hacer *click* en el primer párrafo en el primer párrafo nos sale un cuadro de dialogo:



Todo lo que aparezca en el entrecomillado de la etiqueta onclick tiene que ser código JavaScript. Priorizar siempre la llamada de funciones, mas que a escribir código puesto que ensucia el código.

