**JAVASCRIPT – ARROW FUNCTION**

[**https://academind.com/tutorials/this-keyword-function-references**](https://academind.com/tutorials/this-keyword-function-references)

[**https://www.youtube.com/watch?v=Pv9flm-80vM**](https://www.youtube.com/watch?v=Pv9flm-80vM)

* Vamos a probar casos de funciones

Index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Funcion Flecha</title>

  </head>

  <body>

    <button>Agregar Nombre</button>

    <ul id="names"></ul>

    <script src="app.js"></script>

  </body>

</html>

App.js

class NameField {

  constructor(name) {

    const field = document.createElement("li");

    field.textContent = name;

    const nameListHosk = document.querySelector("#names");

    nameListHosk.appendChild(field);

  }

}

class NameGenerator {

  constructor() {

    const btn = document.querySelector("button");

    btn.addEventListener("click", addName());

  }

  addName() {

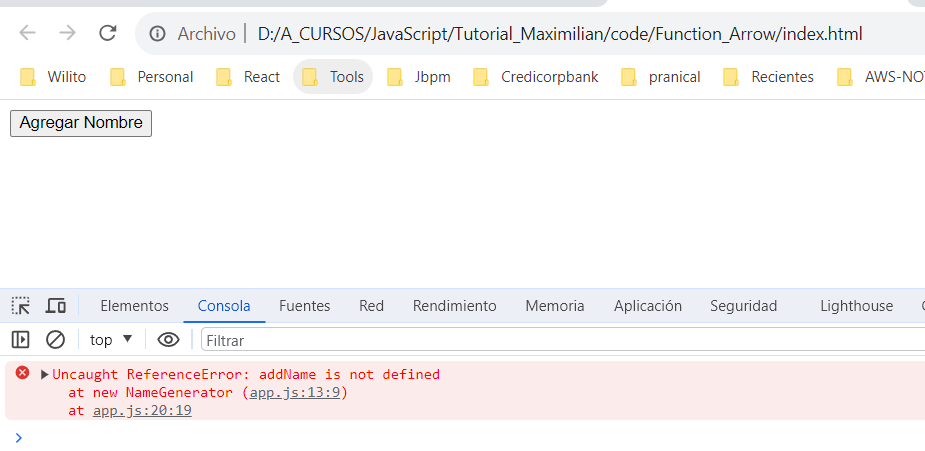
    const name = new NameField("WLOPERA");

  }

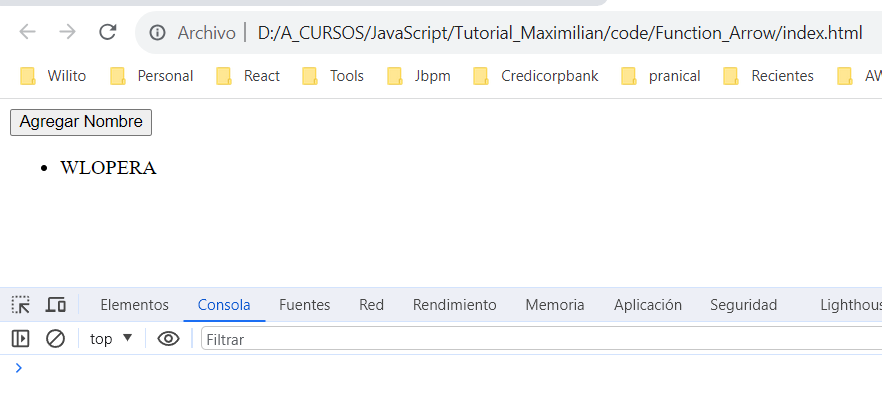
}

const generator = new NameGenerator();

* Llamo al index.html en un navegador



* Si le agrego ***this*** a la llamada a la función ***addName*** funciona llamándose automáticamente (opción click al ***boton***). Se llama automáticamente por los paréntesis de la función ***this.addName()***



* Lo que realmente quiero es que se registre el evento (listener) del boton click y que se ejecute solo cuando presione el boton y el oyente realice la acción click
* Si remueve los paréntesis de la función si funciona solo cuando le de click al boton. Esto es porque paso una referencia de mi función (una dirección de memoria) y el JS le dice al navegador (llama a la función) cuando se da click sobre el boton

class NameGenerator {

  constructor() {

    const btn = document.querySelector("button");

    btn.addEventListener("click", this.addName);

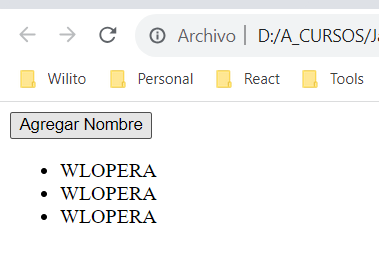
  }

  addName() {

    const name = new NameField("WLOPERA");

  }

}



* Vamos a crear una lista de nombre y un valor actual que permita mostrar los valores, uno a uno, al presionar el click del boton. Uso del this en objetos y clases.

class NameGenerator {

  constructor() {

    const btn = document.querySelector("button");

this.names = ["Messi", "Neymar", "Ronaldo", "Suarez"];

    this.currentName = 0;

    btn.addEventListener("click", this.addName);

  }

  addName() {

    const name = new NameField(this.names[this.currentName]);

    this.currentName++;

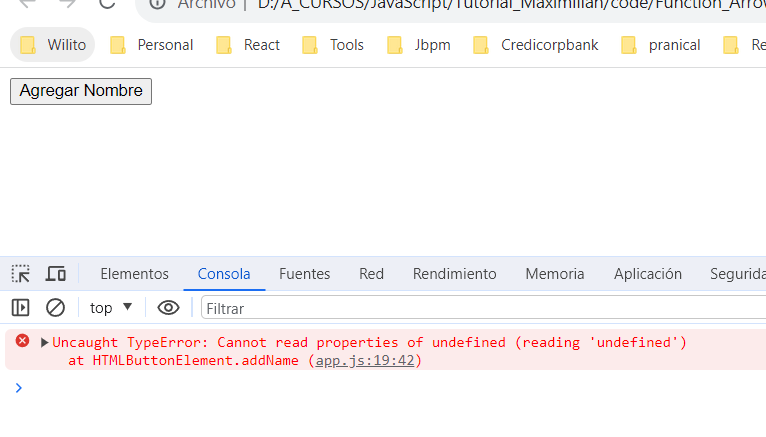
    if (this.currentName >= this.names.length) {

      this.currentName = 0;

    }

  }

}



* Cambio el código para ver quien llama a la clase NameGenerator y quien llama a la función addName y el que llama en ese momento es el this.

class NameGenerator {

  constructor() {

    const btn = document.querySelector("button");

    this.names = ["Messi", "Neymar", "Ronaldo", "Suarez"];

    this.currentName = 0;

    btn.addEventListener("click", this.addName);

    console.log("Quien me llama a la clase NameGenerator: ", this)

  }

  addName() {

    console.log("Quien llama a la funcion addName: ", this)

    const name = new NameField(this.names[this.currentName]);

    this.currentName++;

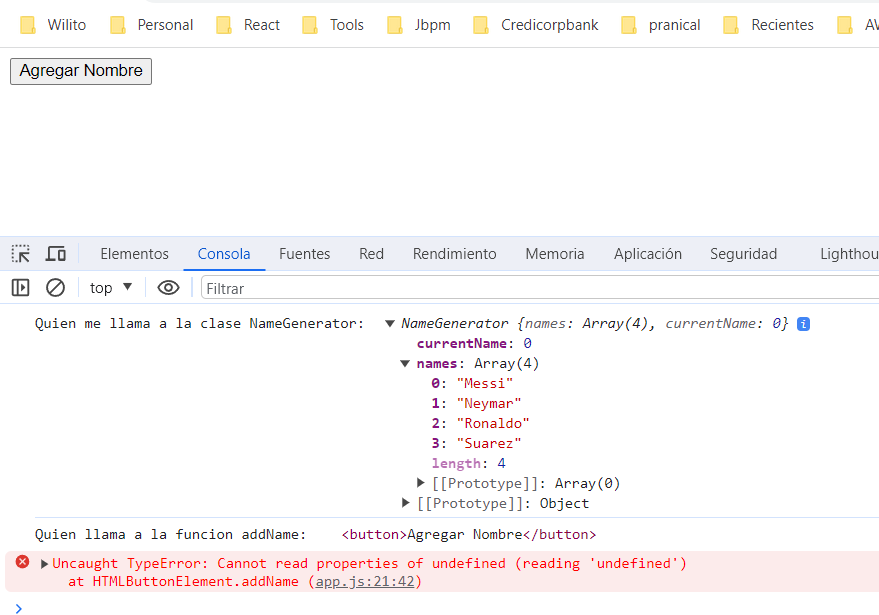
    if (this.currentName >= this.names.length) {

      this.currentName = 0;

    }

  }

}



Son dos lugares diferentes (this diferentes). El this es quien esta ejecutando el código en ese momento.

* Si desde la clase NameGenerator llamo a la funcion addName allí si funcionaria directo el click

class NameGenerator {

  constructor() {

      const btn = document.querySelector("button");

      this.names = ["Messi", "Neymar", "Ronaldo", "Suarez"];

      this.currentName = 0;

      this.addName()

      btn.addEventListener("click", this.addName);

      console.log("Quien me llama a la clase NameGenerator: ", this)

  }

  addName() {

    console.log("Quien llama a la funcion addName: ", this)

    const name = new NameField(this.names[this.currentName]);

    this.currentName++;

    if (this.currentName >= this.names.length) {

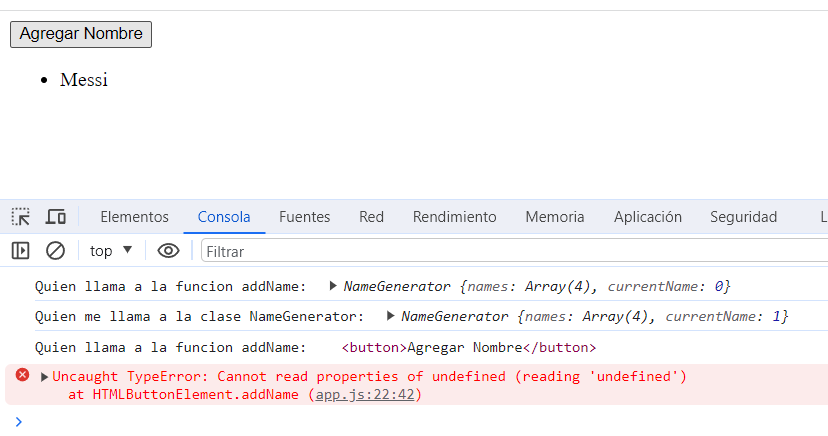
      this.currentName = 0;

    }

  }

}

Al darle click igual falla debió a que la referencia se la está pasando el boón



* Se arregla utilizando función bind o arrow functions
* La funcion.bind(this), el primer argumento de la función bind pasa la referencia al objeto this que queremos en este caso la class NameGenerator que contiene los objetos names y currentName. Pasa el this del contructor no el this del botón

class NameGenerator {

  constructor() {

      const btn = document.querySelector("button");

      this.names = ["Messi", "Neymar", "Ronaldo", "Suarez"];

      this.currentName = 0;

      this.addName()

      btn.addEventListener("click", this.addName.bind(this));

      console.log("Quien me llama a la clase NameGenerator: ", this)

  }

  addName() {

    console.log("Quien llama a la funcion addName: ", this)

    const name = new NameField(this.names[this.currentName]);

    this.currentName++;

    if (this.currentName >= this.names.length) {

      this.currentName = 0;

    }

  }

}



* Una función anónima tampoco funciona aquí

class NameGenerator {

  constructor() {

    const btn = document.querySelector("button");

    this.names = ["Messi", "Neymar", "Ronaldo", "Suarez"];

    this.currentName = 0;

    this.addName();

    btn.addEventListener("click", function () {

        this.addName();

    });

    console.log("Quien me llama a la clase NameGenerator: ", this);

  }

  addName() {

    console.log("Quien llama a la funcion addName: ", this);

    const name = new NameField(this.names[this.currentName]);

    this.currentName++;

    if (this.currentName >= this.names.length) {

      this.currentName = 0;

    }

  }

}



* Otra forma, que funciona, es pasar una función flecha (es6). La Arrow Fucnction resuelve el problema del this. Mantiene el contexto this (la referencia)

class NameGenerator {

  constructor() {

    const btn = document.querySelector("button");

    this.names = ["Messi", "Neymar", "Ronaldo", "Suarez"];

    this.currentName = 0;

    this.addName();

    btn.addEventListener("click", () => {

      this.addName();

    });

    console.log("Quien me llama a la clase NameGenerator: ", this);

  }

  addName() {

    console.log("Quien llama a la funcion addName: ", this);

    const name = new NameField(this.names[this.currentName]);

    this.currentName++;

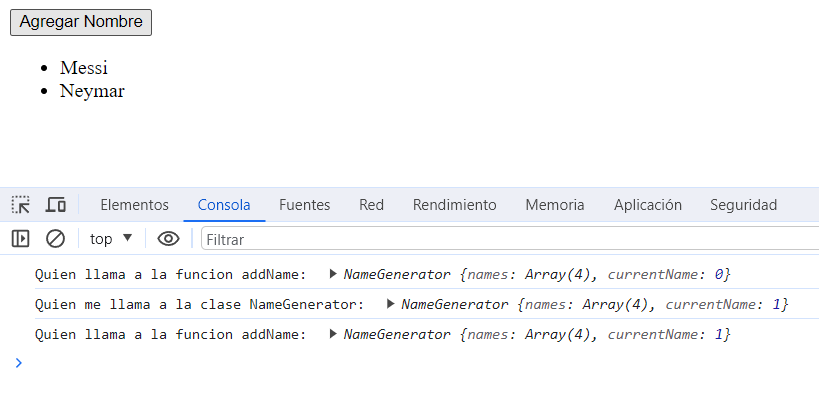
    if (this.currentName >= this.names.length) {

      this.currentName = 0;

    }

  }

}



**Objetos**

* Los objetos pueden manejar funciones anónimas y normales pero el uso de funciones flecha debemos tener cuidado con el uso de this. Las funciones arrow en objetos no tiene sus propios enlaces a ***this*** o ***super***

const person = {

  name: "wlopera",

  age: 50,

  anonymousFunction: function () {

    console.log("Anonima-this:", this);

    console.log("Anonima-this.name:", this.name);

  },

  toString() {

    console.log("Funcion-this:", this);

    console.log("Funcion-this.name:", this.name);

  },

  arrowFunction: (objet) => {

    console.log("Arrow-this:", this, objet);

    console.log("Arrow-this.name:", this.name);

  },

};

console.log("Objeto Persona:", person);

console.log("Nombre Persona:", person.name);

console.log("Edad Persona:", person.age);

console.log();

person.anonymousFunction();

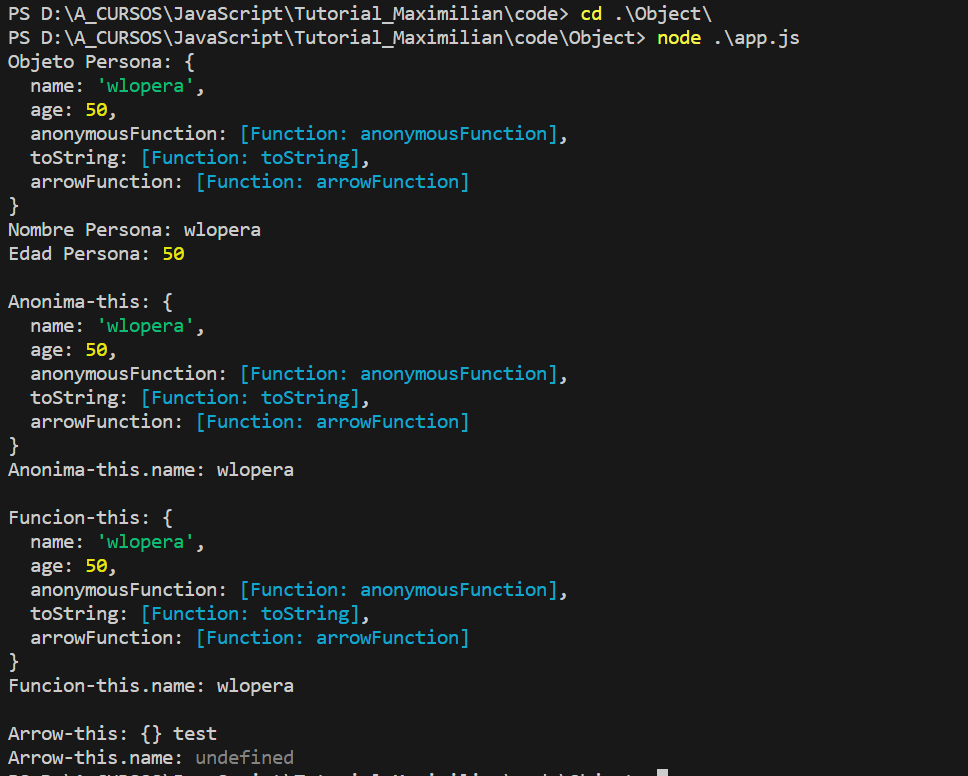
console.log();

person.toString();

console.log();

person.arrowFunction("test");

El Arrow, imprime el parámetro object pero el this es undefined



**Arreglos**

const countries = ["Argentina", "Brasil", "Colombia"]

console.log(countries)

countries.push("Dinamarca")

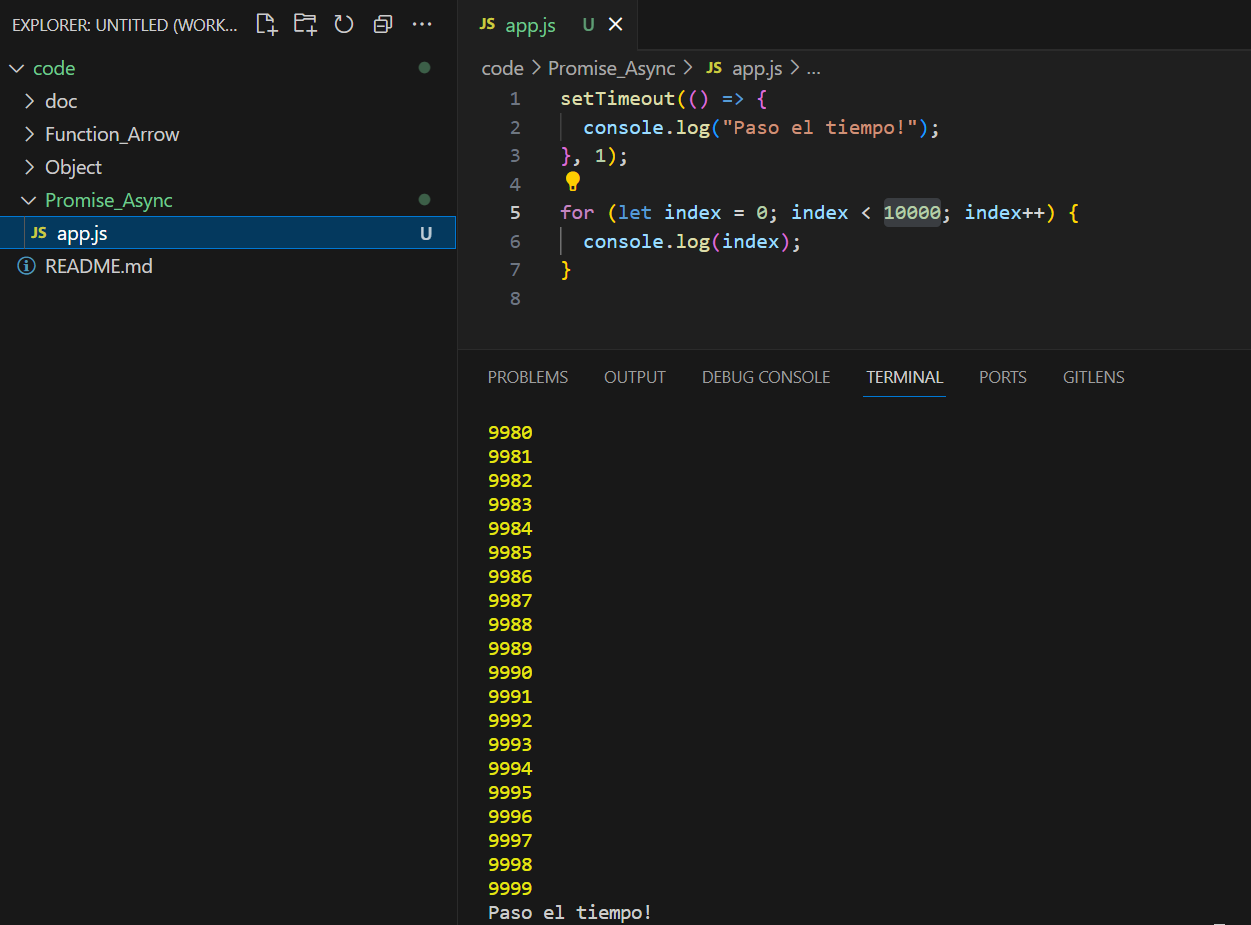
console.log(countries)



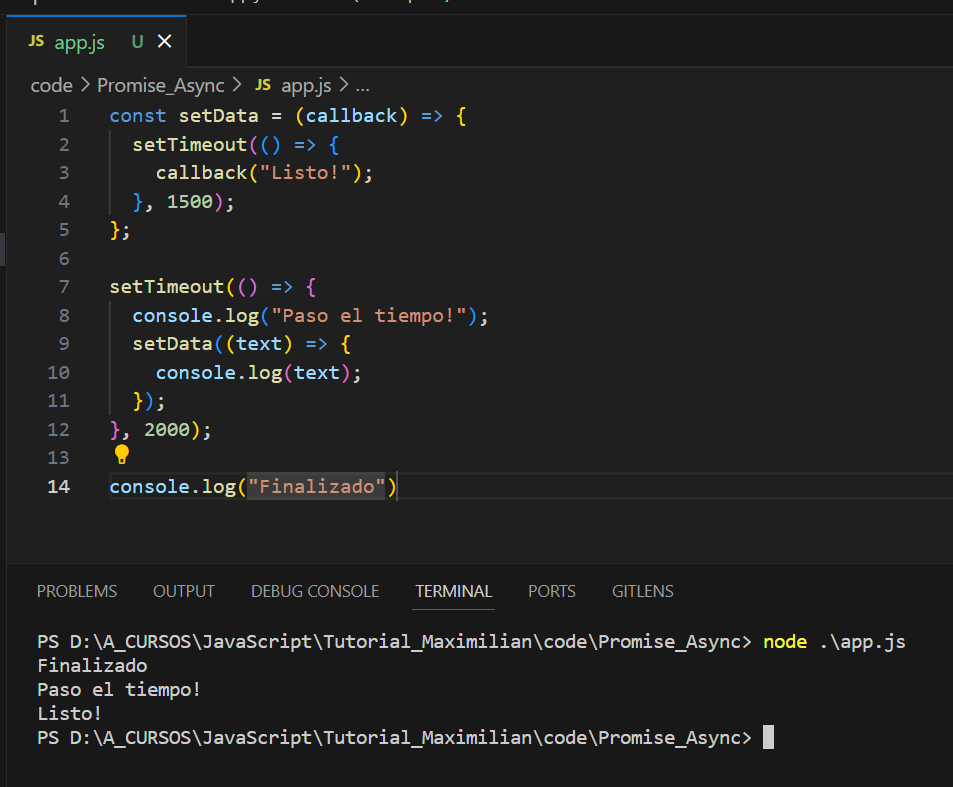
* A pesar de que el array ***countries*** es una constante puedo agregar elementos en él arreglo. Esto es porque se esta almacenando la referencia o dirección de memoria en la variable ***countries*,** el arreglo es almacenado en esa dirección y ese puntero o dirección de memoria no cambia si agregamos un nuevo elemento al arreglo, pero el arreglo si es variable.

**Promesa y Funciones Asíncronas**

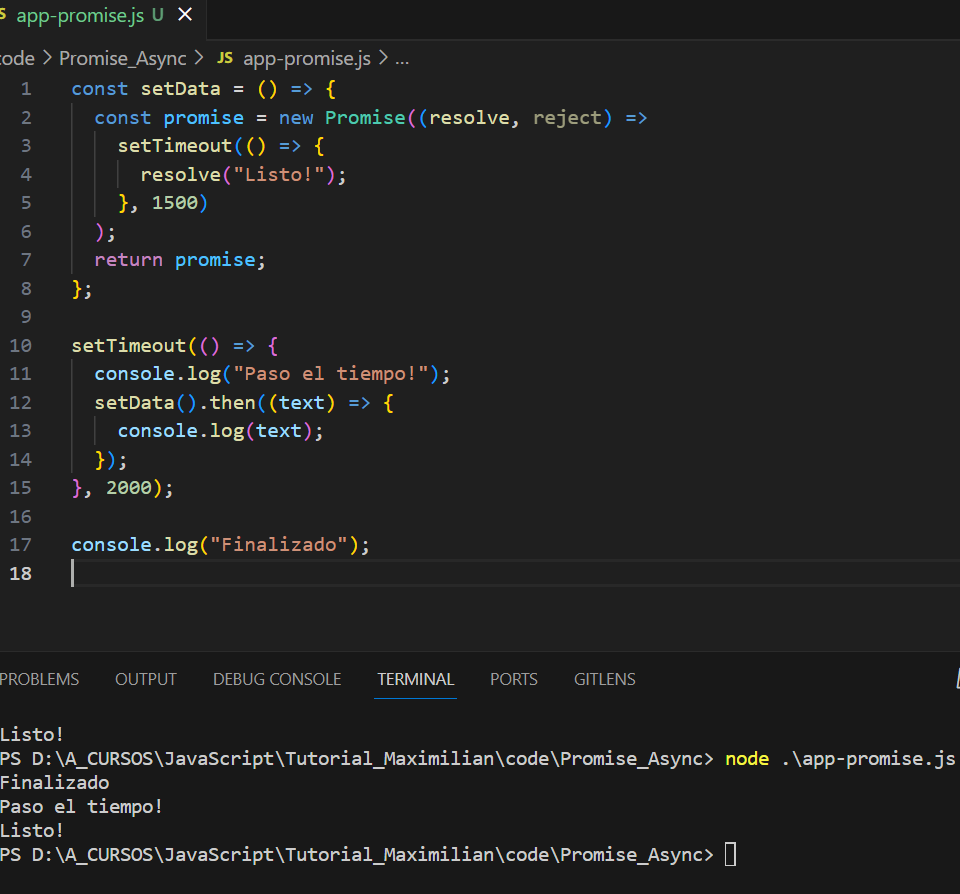
* En JS primero se ejecuta el código síncrono y al finalizar el asíncrono a menos que se utilicen promesas o uso de funciones asíncronas (async-away) o “callback”. En el siguiente ejercicio se imprimen los 10000 números y luego el setTime de 1 seg.



* Si desde una función llamo a otra función y le paso como parámetro una función (callback) este si continuo cuando la función llamada retorna la respuesta (esto es igual si las funciones son asíncronas)



* Usando una promesa



<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/constructor>

**ANEXO**

**Valores primitivos:** [**https://academind.com/tutorials/**](https://academind.com/tutorials/)

**Arreglos:** [**https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Array#**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array)