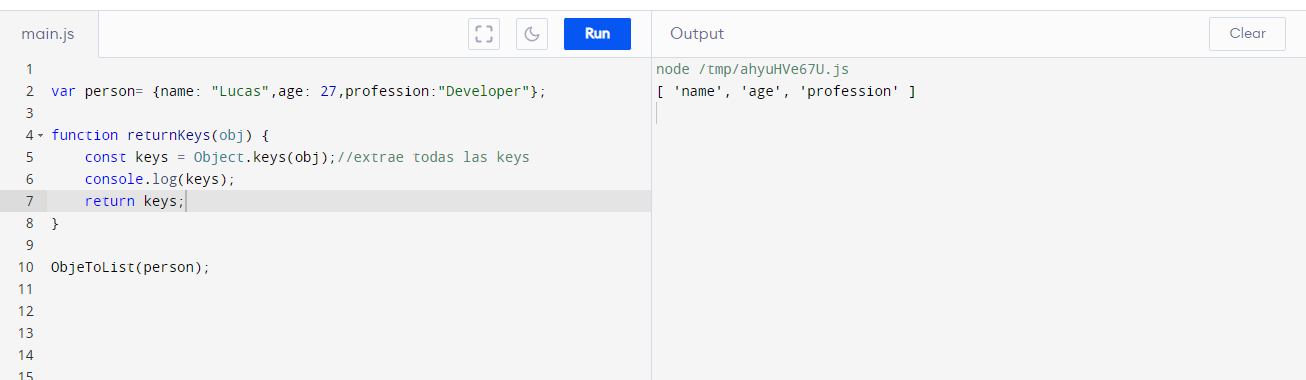
Mariángela Navarro Montero

Actividad #2

1. Escribe un programa que tome como entrada un objeto y devuelva una lista con sus propiedades. Solo puede tener como entrada un objeto y el tipo de vuelta tiene que ser un array.
2. Enumera los distintos valores que puede tener “this” y pon un ejemplo de cada uno.
   * **Contexto Global:**  this se refiere al objeto global, ya sea en modo estricto o no.

this.a = 37;

console.log(window.a); // 37

* + **Contexto de la función:**Dentro de una función, el valor de this depende de cómo la función es llamada.

function f1(){

return this;

}

f1() === window; // objeto global

* + **Constructor:**Cuando una función es usada como un constructor (con la palabra clave new), su this es enlazado al nuevo objeto en construcción, a menos que la ejecución de los resultados del constructor en el motor JavaScript encuentren una instrucción de retorno donde el valor de retorno sea un objeto.

function C(){

this.a = 37;

}

var o = new C();

console.log(o.a); // logs 37

function C2(){

this.a = 37;

return {a:38};

}

o = new C2();

console.log(o.a); // logs 38

* + **call() y apply():**Cuando una función usa la plabra clave this en su cuerpo, su valor puede ser enlazado a un objeto particular durante la ejecución del método call() or apply() que todas las funciones hereden de Function.prototype.

function add(c, d){

return this.a + this.b + c + d;

}

var o = {a:1, b:3};

add.call(o, 5, 7); // 1 + 3 + 5 + 7 = 16

add.apply(o, [10, 20]); // 1 + 3 + 10 + 20 = 34

* + **Funciones enlazadas:**f.bind(someObject) crea una nueva función con el mismo cuerpo y alcance de f, pero donde this se produce en la función original, en la nueva función esto esta permanentemente ligado al primer argumento de bind, independientemente de cómo la función está siendo utilizada.

function f(){

return this.a;

}

var g = f.bind({a:"azerty"});

console.log(g()); // azerty

var o = {a:37, f:f, g:g};

console.log(o.f(), o.g()); // 37, azerty

* + **Controlador de eventos DOM:**Cuando una función es usada como un controlador de eventos, su this es cambiado desde el elemento del evento disparado.

function bluify(e){

console.log(this === e.currentTarget); // Siempre true

console.log(this === e.target);

this.style.backgroundColor = '#A5D9F3';

}

// Consigue una lista de cada elemento en un documento

var elements = document.getElementsByTagName('\*');

for(var i=0 ; i<elements.length ; i++){

elements[i].addEventListener('click', bluify, false);

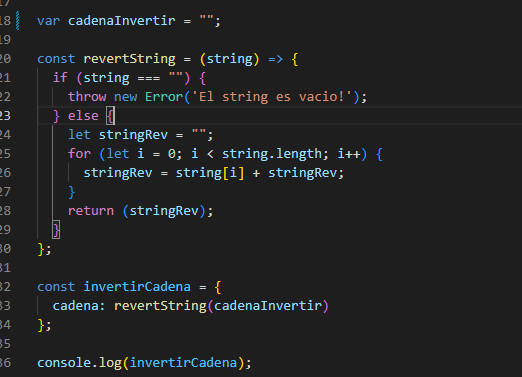
}

Extraído de:https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/this

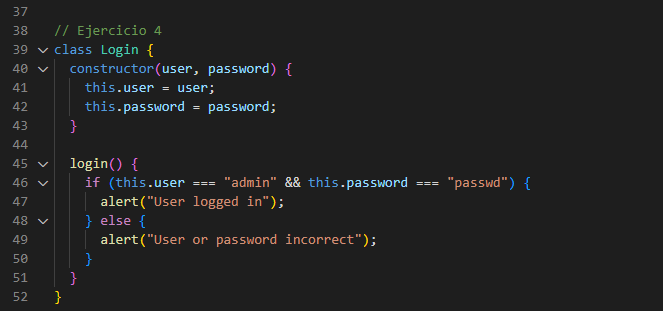
1. Crea una clase a la que llamaremos "InvertirCadena" con las siguientes propiedades:

* Un atributo llamado cadenaInvertir que sea una cadena vacía.
* Una función en formato **arrow function**, que tome el atributo cadenaInvertir e imprima en pantalla el resultado invertido. Ej "Hola mundo" quedaría "odnum aloH".
* Si el parámetro de la cadena cadenaInvertir es vacío, lance un error (throw). Ej. "". throw error.
* Ahora instancia la clase en un objeto que llamaremos invertirCadena.

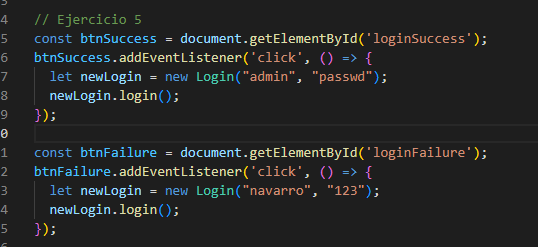
Ejecuta primero la función sin cambiar cadenaInvertir. ¿Cómo podemos hacer para que nuestro código no rompa al ejecutarse?. Ahora cambia el valor a cadenaInvertir y vuelve a llamar la función. ¿Cuál es el resultado?. Por último, intenta acceder al siguiente método invertirCadena.nuevoMetodo(). ¿Cómo podemos hacer para que no de un error?



1. Crea una clase con el formato ES6. Esta clase va a tener dos atributos, username y password y un método login() que compruebe que username tiene el valor **admin** y password el valor **passwd** y en caso positivo, lance una alerta diciendo que el usuario esté logeado, en caso contrario, que diga que el usuario o la contraseña son incorrectas.



1. En este ejercicio os voy a hacer mirar un poco de documentación extra. Y vamos a utilizar el fichero que se encuentra en activity/2\_session/index.html



1. En este ejercicio vamos a añadir asincronía al resto de botones:

* Crea dos clickListener, para los botones con id **loginSuccessAsync** y **loginFailureAsync**. En el primero llamaremos a la función loginWitUsername para que de el resultado correcto. En el segundo llamaremos a la función con parámetros incorrectos. Os dejo un par de pistas para resolver el ejercicio.
* Primero, addEventListener('click', () => {}) no acepta asincronía ya que es una **función síncrona**. Pero es posible sustituir el segundo argumento () => {} que es una función, por una función asíncrona (solo habría que añadir una palabra reservada).
* loginWithUsername devuelve una promesa, que lanza una excepción si falla, por lo que **habría que atrapar esa excepción para que nuestro programa no falle**.

