Ejercicio guiado: Función de flecha con valores por defecto

{desafío} latam_

Ejercicio guiado

Función de flecha con valores por defecto

Se solicita sumar tres números ingresados por el usuario, pero si el usuario no ingresa alguno de los tres números solicitados, la operación de suma debe adquirir por defecto el valor de cero para el valor no enviado.

Siendo el extracto del código HTML:

```
<h4>Implementando funciones con
ES6</h4>
<p>
   <label>Ingrese el primer número:
</label>
   <input type="text" id="num1">
</p>
<p>
   <label>Ingrese el segundo número:
</label>
   <input type="text" id="num2">
</p>
<p>
   <label>Ingrese el tercer número:
</label>
   <input type="text" id="num3">
</p>
<button id="sumar">Sumar</button>
<div>
     <p>El resultado es: <span
id="resultado"></span></p>
</div>
```

{desafío} latam_

Ejercicio guiado

Función de flecha con valores por defecto

En consecuencia, para resolver este ejemplo se debe:

Paso 1: Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos: index.html y script.js. Seguidamente en el archivo index.html debes escribir la estructura básica de un documento HTML con el extracto del código indicando en el enunciado y el llamado al archivo externo script.js, como se muestra a continuación:

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport"
content="width=device-width, initial-scale=1.0">
        <title>ES6</title>
</head>
<body>
   <h4>Implementando funciones con ES6</h4>
   <p>
   <label>Ingrese el primer número: </label>
   <input type="text" id="num1">
   </p>
   <p>
   <label>Ingrese el segundo número: </label>
   <input type="text" id="num2">
   </p>
   <p>
   <label>Ingrese el tercer número: </label>
   <input type="text" id="num3">
   </p>
   <button id="sumar">Sumar</button>
   <div>
        <p>El resultado es: <span
id="resultado"></span></p>
   </div>
   <script src="script.js"></script>
</body>
</html>
```

Ejercicio guiado

Función de flecha con valores por defecto

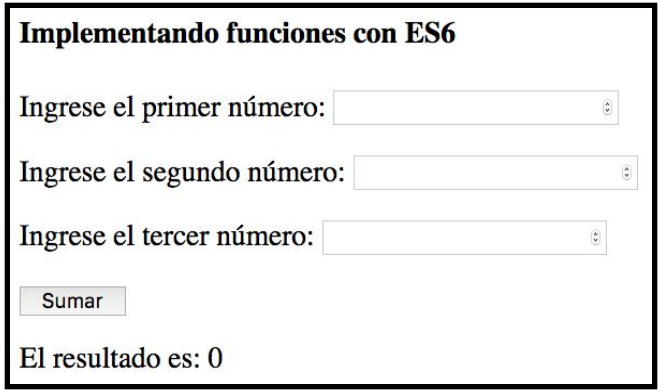
Paso 2: En el archivo script.js, se debe primeramente capturar los elementos DOM del botón y el área de resultado. Luego agregar un listener al botón para poder capturar los valores ingresados por el usuario, es decir, los tres números. Posteriormente, se llama a la función enviando el valor ingresado por el usuario, en el caso de no recibir ningún valor, se debe enviar el valor “undefined” para que la función pueda tomar los valores por defectos en los parámetros.

```
let sumar = document.getElementById('sumar');
let resultado = document.getElementById('resultado');

sumar.addEventListener('click',()=>{
      let num1 = document.getElementById('num1').value;
      let num2 = document.getElementById('num2').value;
      let num3 = document.getElementById('num3').value;
      resultado.innerHTML = sumando(parseInt(num1) || undefined,
parseInt(num2) || undefined, parseInt(num3) || undefined);
});

sumando = (a=0, b=0, c=0) => a + b + c;
```

Ejercicio guiado

Función de flecha con valores por defecto

Paso 3: Al ejecutar el código anterior y hacer un clic sobre el botón “sumar” sin ingresar ningún dato solicitado, el resultado sería:



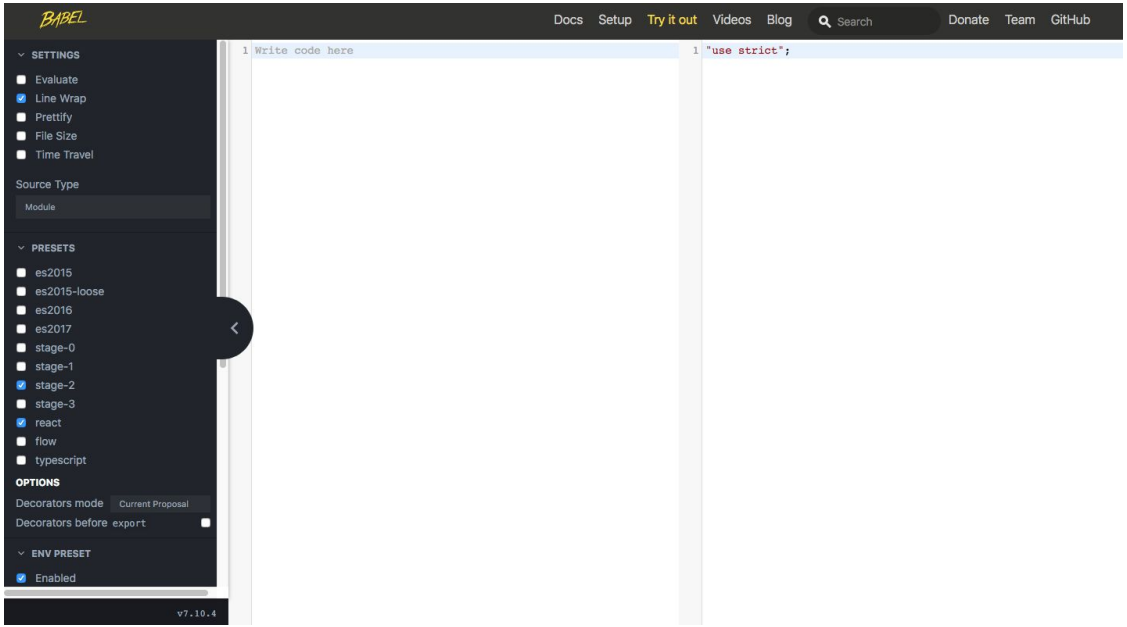
/* Try out con Babeljs.io */

Ejercicio guiado: Try out con Babeljs.io

{desafío} latam_

Ejercicio guiado

Try out con Babeljs.io



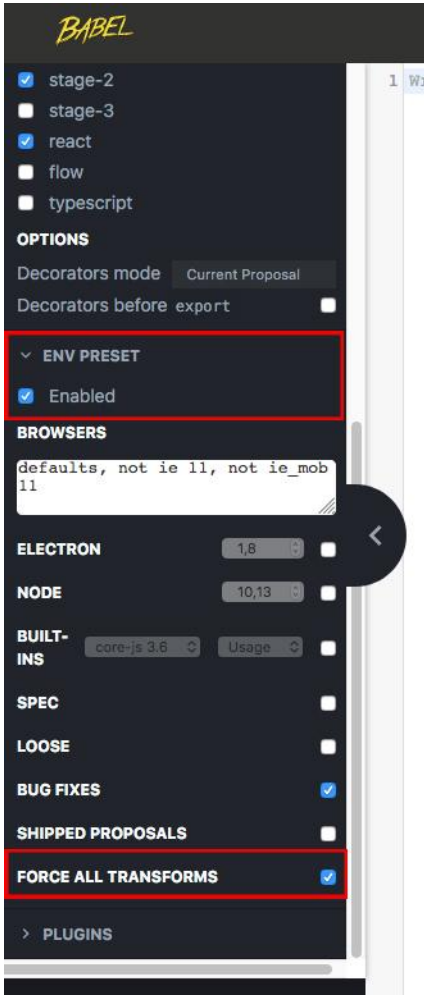
Paso 1: Ir a la web oficial: https://babeljs.io/repl, donde encontrarás un entorno listo para recibir ES6 y devolver ES5. En donde te dará la bienvenida una página con dos editores de texto y un panel al lado izquierdo. El panel no nos interesa por ahora, solo queremos transpilar un poco de código, como se muestra en la imagen.

{desafío} latam_

Ejercicio guiado

Try out con Babeljs.io

Paso 2: En el panel de opciones en el lado izquierdo de la pantalla, se debe verificar que las opciones de “ENV PREST” y “FORCE ALL TRANSFORMS” estén habilitadas o seleccionadas. De no estar alguna de ellas, se deben seleccionar para habilitar.



{desafío} latam_

Ejercicio guiado

Try out con Babeljs.io

Paso 3: Copiaremos el siguiente texto que tiene código en ES6, el cual, tiene diferentes estructuras repetitivas con for y for...of, para mostrar una secuencia de números, en el panel izquierdo. Automáticamente, obtendremos código escrito con estándar ES5:

```
for (let i = 0; i < 3; i++) {
   console.log(i);
   let log = '';
   for (let i = 0; i < 3; i++) {
       log += i;
   }
   console.log(log);
}

for (let i of [1, 2, 3, 4, 5]) {
   console.log(i);
}
```

Ejercicio guiado

Try out con Babeljs.io

Paso 4: El código anterior debería producir lo siguiente en el segundo panel:

```
for (let i = 0; i < 3; i++) {
    console.log(i);
    let log = '';
    for (let i = 0; i < 3; i++) {
        log += i;
    }
    console.log(log);
}

for (let i of [1, 2, 3, 4, 5]) {
    console.log(i);
}
```

```
"use strict";

for (var i = 0; i < 3; i++) {
  console.log(i);
  var log = '';

  for (var _i = 0; _i < 3; _i++) {
    log += _i;
  }

  console.log(log);
}

for (var _i2 = 0, _arr = [1, 2, 3, 4, 5]; _i2 < _arr.length; _i2++) {
  var _i3 = _arr[_i2];
  console.log(_i3);
}
```

Ejercicio guiado: var, let y const

{desafío}
latam_

Ejercicio guiado

var, let y const

Paso 1: Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea un archivo script.js, en el cual, vamos a implementar tres funciones y dentro de ellas, se declara primeramente la variable, una función con una variable var, otra con let y una con const. Luego, en todas las funciones se aplica una estructura condicional que siempre sea verdadera y dentro de ella se vuelve a declarar la misma variable, pero con otro valor para mostrar por consola. Finalmente, fuera de las funciones se vuelve a mostrar la variable en la consola para poder observar el valor.

Ejercicio guiado

var, let y const

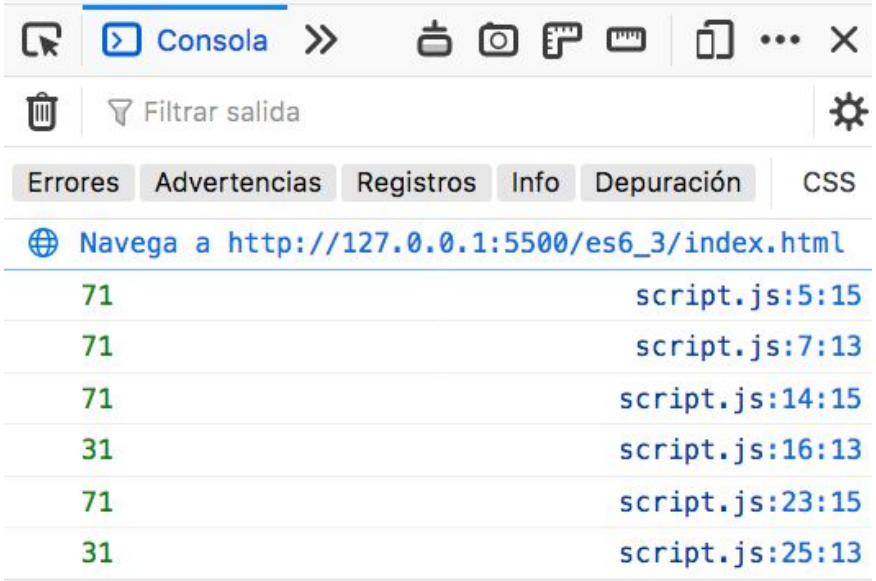
```
var pruebaVar = () => {
      var num = 31;
      if (true) {
         var num = 71;
        console.log(num);  // 71
      }
      console.log(num);  // 71
};

let pruebaLet = () => {
      let num = 31;
      if (true) {
         let num = 71;
         console.log(num);  // 71
      }
      console.log(num);  // 31
};

const pruebaConst = () => {
      const num = 31;
      if (true) {
         const num = 71;
         console.log(num);  // 71
      }
      console.log(num);  // 31
};

pruebaVar();
pruebaLet();
pruebaConst();
```

{desafío} latam_

Ejercicio guiado

var, let y const

Paso 2: Al ejecutar el código anterior, el resultado en la consola del navegador web debería ser:



/* Node y NPM */

Ejercicio guiado: Babel y la terminal

{desafío} latam_

Ejercicio guiado

Babel y la terminal

Utilizar Babel de forma manual (instalar y transpilar el código de ES6 a ES5) a través de la terminal de tu computador, el primer código consistirá en una serie de ciclos for y for...of para mostrar una secuencia de números, mientras que el segundo código construirá un objeto mediante una función cuando se le pasan los valores. Ahora para desarrollar sigamos los siguientes pasos:

Paso 1: Dirígete a un directorio donde quieras guardar tu código fuente, crea una carpeta llamada fullstack-entorno y entra en ella utilizando la terminal de tu sistema operativo:

```
mkdir fullstack-entorno  // se crea la carpeta
cd fullstack-entorno  // entremos dentro de la carpeta creada
```

Ejercicio guiado

Babel y la terminal

Paso 2: Una vez dentro, lo primero que haremos será inicializar nuestro gestor de paquetes NPM, a través del comando npm init -y. Luego de unos segundos, saldrá por la terminal lo siguiente:

```
{
 "name": "fullstack-entorno",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1"
},
 "keywords": [],
 "author": "",
 "license": "ISC"
}
```

Ejercicio guiado

Babel y la terminal

Paso 3: Con esto tenemos lo que necesitamos para descargar e instalar Babel en nuestro computador mediante las líneas de comando de la terminal, siendo estas:

```
npm i -D @babel/preset-env @babel/cli @babel/core
@babel/polyfill
npm i core-js
```

La primera línea instala el comando Babel, la API principal y el preset de transpilación que usarás. Hoy en día, env es el preset principal de babel y contiene instrucciones para transpilar todas las funcionalidades presentes en el lenguaje. Mientras que @babel/polyfill y core-js, instala una colección de polyfills para ser incorporados al código publicado a los navegadores. Un polyfill es un código escrito en ES5 que rellena partes de la API que han sido modificadas como parte de ES6, pero que aún no han sido implementadas por todos los navegadores.

Ejercicio guiado

Babel y la terminal

Paso 4: Una vez se haya instalado todo, con tu editor de texto de preferencia, vuelve a revisar tu package.json, el código en ese archivo en específico se vería así:

```
//package.json
{
   "name": "fullstack-entorno",
   "version": "1.0.0",
   "description": "",
   "main": "index.js",
   "scripts": {
       "test": "echo \"Error: no test specified\" && exit 1"
  },
   "keywords": [],
   "author": "",
   "license": "ISC",
   "devDependencies": {
      "@babel/cli": "^7.12.1",
      "@babel/core": "^7.12.3",
      "@babel/polyfill": "^7.12.1",
      "@babel/preset-env": "^7.12.1"
},
   "dependencies": {
       "core-js": "^3.7.0"
  }
}
```