**Programación functional**

**Aprenda sobre la programación funcional**

La programación funcional es un estilo de programación donde las soluciones son funciones simples y aisladas, sin efectos secundarios fuera del alcance de la función:INPUT -> PROCESS -> OUTPUT

La programación funcional se trata de:

**Funciones aisladas: no depende del estado del programa, que incluye variables globales que están sujetas a cambios.**

**Funciones puras: la misma entrada siempre da la misma salida**

**Funciones con efectos secundarios limitados: cualquier cambio o mutación en el estado del programa fuera de la función se controla cuidadosamente**

**// Function that returns a string representing a cup of green tea**

**const prepareTea = () => 'greenTea';**

**/\***

**Given a function (representing the tea type) and number of cups needed, the**

**following function returns an array of strings (each representing a cup of**

**a specific type of tea).**

**\*/**

**const getTea = (numOfCups) => {**

**const teaCups = [];**

**for(let cups = 1; cups <= numOfCups; cups += 1) {**

**const teaCup = prepareTea();**

**teaCups.push(teaCup);**

**}**

**return teaCups;**

**};**

**// Only change code below this line**

**const tea4TeamFCC = getTea(40);**

**// Only change code above this line**

**Comprender la terminología de programación funcional**

El equipo de FCC tuvo un cambio de humor y ahora quiere dos tipos de té: té verde y té negro. Dato general: Los cambios de humor de los clientes son bastante comunes.

Con esa información, tendremos que volver a visitar la getTeafunción del último desafío para manejar varias solicitudes de té. Podemos modificar getTeapara aceptar una función como parámetro para poder cambiar el tipo de té que prepara. Esto lo hace getTeamás flexible y le da al programador más control cuando cambian las solicitudes del cliente.

Pero primero, cubramos algo de terminología funcional:

Las devoluciones de llamada son las funciones que se deslizan o pasan a otra función para decidir la invocación de esa función. Es posible que los haya visto pasar a otros métodos, por ejemplo, en filter, la función de devolución de llamada le dice a JavaScript los criterios sobre cómo filtrar una matriz.

Las funciones que se pueden asignar a una variable, pasar a otra función o devolver desde otra función como cualquier otro valor normal, se denominan funciones de primera clase . En JavaScript, todas las funciones son funciones de primera clase.

Las funciones que toman una función como argumento, o devuelven una función como valor de retorno, se denominan funciones de orden superior .

Cuando las funciones se pasan o se devuelven desde otra función, las funciones que se pasaron o se devolvieron se pueden llamar lambda .

**// Function that returns a string representing a cup of green tea**

**const prepareGreenTea = () => 'greenTea';**

**// Function that returns a string representing a cup of black tea**

**const prepareBlackTea = () => 'blackTea';**

**/\***

**Given a function (representing the tea type) and number of cups needed, the**

**following function returns an array of strings (each representing a cup of**

**a specific type of tea).**

**\*/**

**const getTea = (prepareTea, numOfCups) => {**

**const teaCups = [];**

**for(let cups = 1; cups <= numOfCups; cups += 1) {**

**const teaCup = prepareTea();**

**teaCups.push(teaCup);**

**}**

**return teaCups;**

**};**

**// Only change code below this line**

**const tea4GreenTeamFCC = getTea(prepareGreenTea, 27);**

**const tea4BlackTeamFCC = getTea(prepareBlackTea, 13);**

**// Only change code above this line**

**console.log(**

**tea4GreenTeamFCC,**

**tea4BlackTeamFCC**

**);**

**Comprender los peligros del uso del código imperativo**

La programación funcional es un buen hábito. Mantiene su código fácil de administrar y lo salva de errores furtivos. Pero antes de llegar allí, veamos un enfoque imperativo de la programación para resaltar dónde puede tener problemas.

En inglés (y muchos otros idiomas), el tiempo imperativo se usa para dar órdenes. De manera similar, un estilo imperativo en la programación es aquel que le da a la computadora un conjunto de declaraciones para realizar una tarea.

A menudo, las declaraciones cambian el estado del programa, como actualizar variables globales. Un ejemplo clásico es escribir un forbucle que proporcione instrucciones exactas para iterar sobre los índices de una matriz.

En contraste, la programación funcional es una forma de programación declarativa. Le dices a la computadora lo que quieres que haga llamando a un método o función.

JavaScript ofrece muchos métodos predefinidos que manejan tareas comunes, por lo que no necesita escribir cómo debe realizarlas la computadora. Por ejemplo, en lugar de usar el forciclo mencionado anteriormente, puede llamar al mapmétodo que maneja los detalles de iterar sobre una matriz. Esto ayuda a evitar errores semánticos, como los "Errores desactivados por uno" que se trataron en la sección Depuración.

Considere el escenario: está navegando por la web en su navegador y desea realizar un seguimiento de las pestañas que ha abierto. Intentemos modelar esto usando un código simple orientado a objetos.

Un objeto Ventana se compone de pestañas y, por lo general, tiene más de una Ventana abierta. Los títulos de cada sitio abierto en cada objeto Ventana se mantienen en una matriz. Después de trabajar en el navegador (abrir nuevas pestañas, fusionar ventanas y cerrar pestañas), desea imprimir las pestañas que aún están abiertas. Las pestañas cerradas se eliminan de la matriz y se agregan nuevas pestañas (para simplificar) al final.

El editor de código muestra una implementación de esta funcionalidad con funciones para tabOpen(), tabClose()y join(). La matriz tabses parte del objeto Ventana que almacena el nombre de las páginas abiertas.

**// tabs es un arreglo de títulos de cada sitio abierto dentro de la ventana**

**const Window = function(tabs) {**

**this.tabs = tabs; // Mantenemos un registro del arreglo dentro del objeto**

**};**

**// Cuando unes dos ventanas en una**

**Window.prototype.join = function(otherWindow) {**

**this.tabs = this.tabs.concat(otherWindow.tabs);**

**return this;**

**};**

**// Cuando abres una nueva pestaña al final**

**Window.prototype.tabOpen = function(tab) {**

**this.tabs.push('new tab'); // Abramos una nueva pestaña por ahora**

**return this;**

**};**

**// Cuando cierras una pestaña**

**Window.prototype.tabClose = function(index) {**

**// Cambia solo el código debajo de esta línea**

**const tabsBeforeIndex = this.tabs.slice(0, index); // get the tabs before the tab**

**const tabsAfterIndex = this.tabs.slice(index + 1); // get the tabs after the tab**

**this.tabs = tabsBeforeIndex.concat(tabsAfterIndex); // Únelos juntos**

**// Cambia solo el código encima de esta línea**

**return this;**

**};**

**// Vamos a crear tres ventanas del navegador**

**const workWindow = new Window(['GMail', 'Inbox', 'Work mail', 'Docs', 'freeCodeCamp']); // Tu casilla de correo, drive y otros sitios de trabajo**

**const socialWindow = new Window(['FB', 'Gitter', 'Reddit', 'Twitter', 'Medium']); // Sitios sociales**

**const videoWindow = new Window(['Netflix', 'YouTube', 'Vimeo', 'Vine']); // Sitios de entretenimiento**

**// Ahora completa la apertura de la pestaña, cierre, y otras operaciones**

**const finalTabs = socialWindow**

**.tabOpen() // Abre una nueva pestaña para memes de gatos**

**.join(videoWindow.tabClose(2)) // Cierra la tercera pestaña en videoWindow y une**

**.join(workWindow.tabClose(1).tabOpen());**

**console.log(finalTabs.tabs);**

**Evitar efectos secundarios utilizando programación funcional**

Si aún no te has dado cuenta, el problema en el desafío anterior estaba en la llamada splicede la función tabClose(). Desafortunadamente, splicemodifica el arreglo original a la que se llama, por lo que la segunda llamada a ella utilizo un arreglo modificado, y dio resultados inesperados.

Este es un pequeño ejemplo de un patrón mucho mayor: se llama una función en una variable, arreglo o un objeto, y la función modifica la variable o algo en el objeto.

Uno de los principios fundamentales de la programación funcional es no cambiar las cosas. Los cambios conducen a errores. Es más fácil evitar errores sabiendo que las funciones no cambian nada, incluyendo los argumentos de la función o cualquier variable global.

El ejemplo anterior no tenía operaciones complicadas, pero el método splicemodificó el arreglo original y dio como resultado un error.

Recuerda que en la programación funcional, cambiar o alterar cosas se denomina mutación , y el resultado es conocido como efecto secundario . Una función, idealmente**,** debe ser una función pura , lo que significa que no provoca ningún efecto secundario.

Intentemos dominar esta disciplina y no alterar ninguna variable u objeto en nuestro código.

**// La variable global**

**let fixedValue = 4;**

**function incrementer() {**

**// Cambia solo el código debajo de esta línea**

**return fixedValue + 1 ;**

**// Cambia solo el código encima de esta línea**

**}**

**Pasa argumentos para evitar la dependencia**

**externa en una funcion**

El último reto fue un paso más cerca de los principios de la programación funcional, pero todavía falta algo.

No alteramos el valor global de la variable, pero la función incrementerno funcionaría sin que la variable global fixedValueestuviera allí.

Otro principio de programación funcional es declarar siempre sus dependencias de forma limpia. Esto significa si una función depende de que una variable u objeto esté presente, después pasa esa variable u objeto directamente a la función como argumento.

Este principio tiene varias consecuencias positivas. La función es más fácil de probar, se sabe exactamente lo que necesita, y no surgió nada más en su programa.

Esto puede darte más confianza cuando cambias, eliminas o agregas código nuevo. Sabrás lo que se puede o no se puede cambiar y, puedes ver dónde están las posibles trampas.

Finalmente, la función siempre produciría el mismo resultado para el mismo conjunto de entradas, sin importar qué parte del código la ejecuta.

**// La variable global**

**let fixedValue = 4;**

**// Cambia solo el código debajo de esta línea**

**function incrementer(value) {**

**return value + 1;**

**// Cambia solo el código encima de esta línea**

**}**

**Refactoriza variables globales por fuera de funciones**

Hasta ahora, hemos visto dos principios distintos para la programación funcional:

No alteres una variable u objeto: crea nuevas variables y objetos y devuélvelos, si es necesario, desde una función. Pista: usar algo como const newArr = arrVar, donde arrVares un arreglo, simplemente creará una referencia a la variable existente y no una copia. Así que cambiaría un valor en newArrcambiaría el valor en arrVar.

Declara parámetros de función: cualquier cálculo dentro de una función depende sólo de los argumentos pasados ​​a la función y no en ningún objeto o variable global.

Añadir uno a un número no es muy emocionante, pero podemos aplicar estos principios cuando trabajamos con arreglos u objetos más complejos.

**// the global variable**

**var bookList = ["The Hound of the Baskervilles", "On The Electrodynamics of Moving Bodies", "Philosophiæ Naturalis Principia Mathematica", "Disquisitiones Arithmeticae"];**

**/\* This function should add a book to the list and return the list \*/**

**// New parameters should come before bookName**

**// Add your code below this line**

**function add(arr, bookName) {**

**let newArr = [...arr]; // Copy the bookList array to a new array.**

**newArr.push(bookName); // Add bookName parameter to the end of the new array.**

**return newArr; // Return the new array.**

**}**

**/\* This function should remove a book from the list and return the list \*/**

**// New parameters should come before the bookName one**

**// Add your code below this line**

**function remove(arr, bookName) {**

**let newArr = [...arr]; // Copy the bookList array to a new array.**

**if (newArr.indexOf(bookName) >= 0) {**

**// Check whether the bookName parameter is in new array.**

**newArr.splice(newArr.indexOf(bookName), 1); // Remove the given paramater from the new array.**

**return newArr; // Return the new array.**

**}**

**}**

**var newBookList = add(bookList, 'A Brief History of Time');**

**var newerBookList = remove(bookList, 'On The Electrodynamics of Moving Bodies');**

**var newestBookList = remove(add(bookList, 'A Brief History of Time'), 'On The Electrodynamics of Moving Bodies');**

**console.log(bookList);**

**Usa el método "map" para extraer datos de un arreglo**

Hasta ahora hemos aprendido a utilizar funciones puras para evitar efectos secundarios en un programa. Además, hemos aprendido el valor de tener una función que solo depende de sus argumentos de entrada.

Este es solo el principio. Como su nombre indica, la programación funcional se centra en una teoría de funciones.

Tendría sentido pasarlos como argumentos a otras funciones y devolver una función de otra función. Las funciones se consideran objetos de primera clase en JavaScript, lo que significa que pueden ser usados ​​como cualquier otro objeto. Pueden guardarse en variables, almacenarse en un objeto o pasarse como argumentos de función.

Empecemos con algunas funciones de arreglos simples, que son métodos en el prototipo de objetos del arreglo. En este ejercicio estamos usando Array.prototype.map()o más específicamente map.

El método mapiterará sobre cada elemento de un arreglo y devuelve un nuevo arreglo que contiene los resultados de llamar a la función callback en cada elemento. Esto lo hace sin cambiar el arreglo original.

Cuando se utiliza la función callback, se pasan tres argumentos. El primer argumento es el elemento actual que se está procesando. El segundo es el índice de ese elemento y el tercero es el arreglo al que se llamó el método map.

A continuación se muestra un ejemplo con el método mapen el arreglo userspara devolver un nuevo arreglo que contiene solo los nombres de los usuarios como elementos. Para que sea más fácil, el ejemplo solo utiliza el primer argumento del callback.

**const users = [**

**{ name: 'John', age: 34 },**

**{ name: 'Amy', age: 20 },**

**{ name: 'camperCat', age: 10 }**

**];**

**const names = users.map(user => user.name);**

**console.log(names);**

La consola mostraría el valor [ 'John', 'Amy', 'camperCat' ].

**// La variable global**

**const watchList = [**

**{**

**"Title": "Inception",**

**"Year": "2010",**

**"Rated": "PG-13",**

**"Released": "16 Jul 2010",**

**"Runtime": "148 min",**

**"Genre": "Action, Adventure, Crime",**

**"Director": "Christopher Nolan",**

**"Writer": "Christopher Nolan",**

**"Actors": "Leonardo DiCaprio, Joseph Gordon-Levitt, Elliot Page, Tom Hardy",**

**"Plot": "A thief, who steals corporate secrets through use of dream-sharing technology, is given the inverse task of planting an idea into the mind of a CEO.",**

**"Language": "English, Japanese, French",**

**"Country": "USA, UK",**

**"Awards": "Won 4 Oscars. Another 143 wins & 198 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMjAxMzY3NjcxNF5BMl5BanBnXkFtZTcwNTI5OTM0Mw@@.\_V1\_SX300.jpg",**

**"Metascore": "74",**

**"imdbRating": "8.8",**

**"imdbVotes": "1,446,708",**

**"imdbID": "tt1375666",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "Interstellar",**

**"Year": "2014",**

**"Rated": "PG-13",**

**"Released": "07 Nov 2014",**

**"Runtime": "169 min",**

**"Genre": "Adventure, Drama, Sci-Fi",**

**"Director": "Christopher Nolan",**

**"Writer": "Jonathan Nolan, Christopher Nolan",**

**"Actors": "Ellen Burstyn, Matthew McConaughey, Mackenzie Foy, John Lithgow",**

**"Plot": "A team of explorers travel through a wormhole in space in an attempt to ensure humanity's survival.",**

**"Language": "English",**

**"Country": "USA, UK",**

**"Awards": "Won 1 Oscar. Another 39 wins & 132 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMjIxNTU4MzY4MF5BMl5BanBnXkFtZTgwMzM4ODI3MjE@.\_V1\_SX300.jpg",**

**"Metascore": "74",**

**"imdbRating": "8.6",**

**"imdbVotes": "910,366",**

**"imdbID": "tt0816692",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "The Dark Knight",**

**"Year": "2008",**

**"Rated": "PG-13",**

**"Released": "18 Jul 2008",**

**"Runtime": "152 min",**

**"Genre": "Action, Adventure, Crime",**

**"Director": "Christopher Nolan",**

**"Writer": "Jonathan Nolan (screenplay), Christopher Nolan (screenplay), Christopher Nolan (story), David S. Goyer (story), Bob Kane (characters)",**

**"Actors": "Christian Bale, Heath Ledger, Aaron Eckhart, Michael Caine",**

**"Plot": "When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, the caped crusader must come to terms with one of the greatest psychological tests of his ability to fight injustice.",**

**"Language": "English, Mandarin",**

**"Country": "USA, UK",**

**"Awards": "Won 2 Oscars. Another 146 wins & 142 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMTMxNTMwODM0NF5BMl5BanBnXkFtZTcwODAyMTk2Mw@@.\_V1\_SX300.jpg",**

**"Metascore": "82",**

**"imdbRating": "9.0",**

**"imdbVotes": "1,652,832",**

**"imdbID": "tt0468569",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "Batman Begins",**

**"Year": "2005",**

**"Rated": "PG-13",**

**"Released": "15 Jun 2005",**

**"Runtime": "140 min",**

**"Genre": "Action, Adventure",**

**"Director": "Christopher Nolan",**

**"Writer": "Bob Kane (characters), David S. Goyer (story), Christopher Nolan (screenplay), David S. Goyer (screenplay)",**

**"Actors": "Christian Bale, Michael Caine, Liam Neeson, Katie Holmes",**

**"Plot": "After training with his mentor, Batman begins his fight to free crime-ridden Gotham City from the corruption that Scarecrow and the League of Shadows have cast upon it.",**

**"Language": "English, Urdu, Mandarin",**

**"Country": "USA, UK",**

**"Awards": "Nominated for 1 Oscar. Another 15 wins & 66 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BNTM3OTc0MzM2OV5BMl5BanBnXkFtZTYwNzUwMTI3.\_V1\_SX300.jpg",**

**"Metascore": "70",**

**"imdbRating": "8.3",**

**"imdbVotes": "972,584",**

**"imdbID": "tt0372784",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "Avatar",**

**"Year": "2009",**

**"Rated": "PG-13",**

**"Released": "18 Dec 2009",**

**"Runtime": "162 min",**

**"Genre": "Action, Adventure, Fantasy",**

**"Director": "James Cameron",**

**"Writer": "James Cameron",**

**"Actors": "Sam Worthington, Zoe Saldana, Sigourney Weaver, Stephen Lang",**

**"Plot": "A paraplegic marine dispatched to the moon Pandora on a unique mission becomes torn between following his orders and protecting the world he feels is his home.",**

**"Language": "English, Spanish",**

**"Country": "USA, UK",**

**"Awards": "Won 3 Oscars. Another 80 wins & 121 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMTYwOTEwNjAzMl5BMl5BanBnXkFtZTcwODc5MTUwMw@@.\_V1\_SX300.jpg",**

**"Metascore": "83",**

**"imdbRating": "7.9",**

**"imdbVotes": "876,575",**

**"imdbID": "tt0499549",**

**"Type": "movie",**

**"Response": "True"**

**}**

**];**

**// Cambia solo el código debajo de esta línea**

**const ratings = watchList.map(item => ({**

**title: item["Title"],**

**rating: item["imdbRating"]**

**}));**

**// Cambia solo el código encima de esta línea**

**console.log(JSON.stringify(ratings));**

**Implementa un mapa en un prototipo**

Como has visto anteriormente, al aplicar Array.prototype.map(), o simplemente map(), el método mapdevuelve un arreglo de la misma longitud que el arreglo dentro del que fue llamado. Esto tampoco altera el arreglo original, siempre y cuando su función callback no lo haga.

En otras palabras, mapes una función pura, y su salida depende únicamente de sus entradas. Además, toma otra función como argumento.

Puedes aprender mucho sobre el método mapsi implementas tu propia versión. Se recomienda utilizar un bucle foro Array.prototype.forEach().

**Array.prototype.myMap = function (callback) {**

**const newArray = [];**

**// Only change code below this line**

**this.forEach((element, index, originalArr) =>**

**newArray.push(callback(element, index, originalArr))**

**);**

**// Only change code above this line**

**return newArray;**

**};**

**Use el método de "filter" para extraer datos de un arreglo**

Otra función útil de los arreglos es Array.prototype.filter()o simplemente filter().

filterLlama a una función sobre cada elemento del arreglo y devuelve un nuevo arreglo, conteniendo solo los elementos para los cuales la función devolvió un valor de verdadero - Es decir, un valor que devuelve truesi paso al constructor Boolean(). En otras palabras, filtra el arreglo, severamente en la funcion que se le pasa. Al igual que map, hace esto sin necesidad de modificar el arreglo original.

La función callback acepta tres argumentos. El primer argumento es el elemento actual que se está procesando. El segundo es el índice de ese elemento y el tercero es el arreglo sobre el que se llamó al método filter.

A continuación se muestra un ejemplo en el que se utiliza el método filteren el arreglo userspara devolver un nuevo arreglo que contiene sólo a los usuarios menores de 30 años. Para que sea más fácil, el ejemplo solo utiliza el primer argumento de la función callback.

**const users = [**

**{ name: 'John', age: 34 },**

**{ name: 'Amy', age: 20 },**

**{ name: 'camperCat', age: 10 }**

**];**

**const usersUnder30 = users.filter(user => user.age < 30);**

**console.log(usersUnder30);**

La consola muestra el valor [ { name: 'Amy', age: 20 }, { name: 'camperCat', age: 10 } ].

**// La variable global**

**const watchList = [**

**{**

**"Title": "Inception",**

**"Year": "2010",**

**"Rated": "PG-13",**

**"Released": "16 Jul 2010",**

**"Runtime": "148 min",**

**"Genre": "Action, Adventure, Crime",**

**"Director": "Christopher Nolan",**

**"Writer": "Christopher Nolan",**

**"Actors": "Leonardo DiCaprio, Joseph Gordon-Levitt, Elliot Page, Tom Hardy",**

**"Plot": "A thief, who steals corporate secrets through use of dream-sharing technology, is given the inverse task of planting an idea into the mind of a CEO.",**

**"Language": "English, Japanese, French",**

**"Country": "USA, UK",**

**"Awards": "Won 4 Oscars. Another 143 wins & 198 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMjAxMzY3NjcxNF5BMl5BanBnXkFtZTcwNTI5OTM0Mw@@.\_V1\_SX300.jpg",**

**"Metascore": "74",**

**"imdbRating": "8.8",**

**"imdbVotes": "1,446,708",**

**"imdbID": "tt1375666",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "Interstellar",**

**"Year": "2014",**

**"Rated": "PG-13",**

**"Released": "07 Nov 2014",**

**"Runtime": "169 min",**

**"Genre": "Adventure, Drama, Sci-Fi",**

**"Director": "Christopher Nolan",**

**"Writer": "Jonathan Nolan, Christopher Nolan",**

**"Actors": "Ellen Burstyn, Matthew McConaughey, Mackenzie Foy, John Lithgow",**

**"Plot": "A team of explorers travel through a wormhole in space in an attempt to ensure humanity's survival.",**

**"Language": "English",**

**"Country": "USA, UK",**

**"Awards": "Won 1 Oscar. Another 39 wins & 132 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMjIxNTU4MzY4MF5BMl5BanBnXkFtZTgwMzM4ODI3MjE@.\_V1\_SX300.jpg",**

**"Metascore": "74",**

**"imdbRating": "8.6",**

**"imdbVotes": "910,366",**

**"imdbID": "tt0816692",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "The Dark Knight",**

**"Year": "2008",**

**"Rated": "PG-13",**

**"Released": "18 Jul 2008",**

**"Runtime": "152 min",**

**"Genre": "Action, Adventure, Crime",**

**"Director": "Christopher Nolan",**

**"Writer": "Jonathan Nolan (screenplay), Christopher Nolan (screenplay), Christopher Nolan (story), David S. Goyer (story), Bob Kane (characters)",**

**"Actors": "Christian Bale, Heath Ledger, Aaron Eckhart, Michael Caine",**

**"Plot": "When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, the caped crusader must come to terms with one of the greatest psychological tests of his ability to fight injustice.",**

**"Language": "English, Mandarin",**

**"Country": "USA, UK",**

**"Awards": "Won 2 Oscars. Another 146 wins & 142 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMTMxNTMwODM0NF5BMl5BanBnXkFtZTcwODAyMTk2Mw@@.\_V1\_SX300.jpg",**

**"Metascore": "82",**

**"imdbRating": "9.0",**

**"imdbVotes": "1,652,832",**

**"imdbID": "tt0468569",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "Batman Begins",**

**"Year": "2005",**

**"Rated": "PG-13",**

**"Released": "15 Jun 2005",**

**"Runtime": "140 min",**

**"Genre": "Action, Adventure",**

**"Director": "Christopher Nolan",**

**"Writer": "Bob Kane (characters), David S. Goyer (story), Christopher Nolan (screenplay), David S. Goyer (screenplay)",**

**"Actors": "Christian Bale, Michael Caine, Liam Neeson, Katie Holmes",**

**"Plot": "After training with his mentor, Batman begins his fight to free crime-ridden Gotham City from the corruption that Scarecrow and the League of Shadows have cast upon it.",**

**"Language": "English, Urdu, Mandarin",**

**"Country": "USA, UK",**

**"Awards": "Nominated for 1 Oscar. Another 15 wins & 66 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BNTM3OTc0MzM2OV5BMl5BanBnXkFtZTYwNzUwMTI3.\_V1\_SX300.jpg",**

**"Metascore": "70",**

**"imdbRating": "8.3",**

**"imdbVotes": "972,584",**

**"imdbID": "tt0372784",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "Avatar",**

**"Year": "2009",**

**"Rated": "PG-13",**

**"Released": "18 Dec 2009",**

**"Runtime": "162 min",**

**"Genre": "Action, Adventure, Fantasy",**

**"Director": "James Cameron",**

**"Writer": "James Cameron",**

**"Actors": "Sam Worthington, Zoe Saldana, Sigourney Weaver, Stephen Lang",**

**"Plot": "A paraplegic marine dispatched to the moon Pandora on a unique mission becomes torn between following his orders and protecting the world he feels is his home.",**

**"Language": "English, Spanish",**

**"Country": "USA, UK",**

**"Awards": "Won 3 Oscars. Another 80 wins & 121 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMTYwOTEwNjAzMl5BMl5BanBnXkFtZTcwODc5MTUwMw@@.\_V1\_SX300.jpg",**

**"Metascore": "83",**

**"imdbRating": "7.9",**

**"imdbVotes": "876,575",**

**"imdbID": "tt0499549",**

**"Type": "movie",**

**"Response": "True"**

**}**

**];**

**// Cambia solo el código debajo de esta línea**

**const filteredList = watchList**

**.filter(movie => {**

**// return true it will keep the item**

**// return false it will reject the item**

**return parseFloat(movie.imdbRating) >= 8.0;**

**})**

**.map(movie => {**

**return {**

**title: movie.Title,**

**rating: movie.imdbRating**

**};**

**});**

**// Cambia solo el código encima de esta línea**

**console.log(filteredList);**

**Implementa el método filter en un prototipo**

Puedes aprender mucho sobre el método filtersi implementas tu propia versión. Se recomienda utilizar un bucle foro Array.prototype.forEach().

**Array.prototype.myFilter = function(callback) {**

**const newArray = [];**

**// Cambia solo el código debajo de esta línea**

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

**if (Boolean(callback(this[i], i, this)) === true) {**

**newArray.push(this[i]);**

**}**

**}**

**// Cambia solo el código encima de esta línea**

**return newArray;**

**};**

**Devolver parte de un arreglo mediante el método slice**

El método slicedevuelve una copia de ciertos elementos de un arreglo. Puede tomar dos argumentos, el primero da el índice de dónde comenzar el corte, el segundo es el índice de dónde terminar el corte (y no es inclusivo). Si no se proporcionan los argumentos, el valor predeterminado es comenzar desde el principio del arreglo hasta el final, la cual es una manera fácil de hacer una copia de todo el arreglo. El método sliceno muta el arreglo original, pero devuelve uno nuevo.

Por ejemplo:

**const arr = ["Cat", "Dog", "Tiger", "Zebra"];**

**const newArray = arr.slice(1, 3);**

newArraytiene el valor de ["Dog", "Tiger"].

**function sliceArray(anim, beginSlice, endSlice) {**

**// Cambia solo el código debajo de esta línea**

**return anim.slice(beginSlice, endSlice);**

**// Cambia solo el código encima de esta línea**

**}**

**const inputAnim = ["Cat", "Dog", "Tiger", "Zebra", "Ant"];**

**sliceArray(inputAnim, 1, 3);**

**Remueve elementos de un arreglo usando slice en lugar de splice**

Un patrón común al trabajar con arreglos es cuando deseas eliminar elementos y conservar el resto del arreglo. JavaScript ofrece el método splicepara esto, que toma argumentos para el índice de dónde empezar a eliminar elementos, luego la cantidad de elementos para eliminar. Si no se proporciona el segundo argumento, el valor predeterminado es eliminar elementos hasta el final. Sin embargo, el método splicemuta el arreglo original en el que se llama. Por ejemplo:

**const cities = ["Chicago", "Delhi", "Islamabad", "London", "Berlin"];**

**cities.splice(3, 1);**

Aquí splicedevluelve la "string" Londony la elimina del arreglo cities. citiestendrá el valor de ["Chicago", "Delhi", "Islamabad", "Berlin"].

Como vimos en el último desafío, el método sliceno muta el arreglo original, pero devuelve uno nuevo que puede ser guardado en una variable. Recuerda que el método slicerecibe dos argumentos para indicar el comienzo y el final del segmento (el final no es inclusivo) y devolver estos elementos en un nuevo arreglo. Usar el método sliceen lugar de spliceayuda para prevenir cualquier efecto colateral de mutar un arreglo.

**function nonMutatingSplice(cities) {**

**// Cambia solo el código debajo de esta línea**

**return cities.slice(0, 3);**

**// Cambia solo el código encima de esta línea**

**}**

**var inputCities = ["Chicago", "Delhi", "Islamabad", "London", "Berlin"];**

**nonMutatingSplice(inputCities);**

**Combina dos arreglos usando el método "concat"**

Concatenación significa unir elementos de extremo a extremo. JavaScript ofrece el método concatpara cadenas y arreglos, que funciona de la misma manera. Para arreglos, el método es llamado desde un arreglo, un segundo arrelgo es proporcionado como argumento de concat, este se añadirá al final del primer arreglo. Devuelve un nuevo arreglo, sin cambiar ninguno de los arreglos originales. Aquí hay un ejemplo:

**[1, 2, 3].concat([4, 5, 6]);**

**El arreglo devuelto seria [1, 2, 3, 4, 5, 6].**

**function nonMutatingConcat(original, attach) {**

**// Cambia solo el código debajo de esta línea**

**return original.concat(attach);**

**// Cambia solo el código encima de esta línea**

**}**

**const first = [1, 2, 3];**

**const second = [4, 5];**

**nonMutatingConcat(first, second);**

**Agrega elementos al final de un arreglo usando**

**concat en lugar de push**

La programación funcional consiste en crear y utilizar funciones no mutantes.

En el reto anterior se introdujo el método concatcomo una forma de unir varios arreglos en uno nuevo sin alterar los arreglos originales. Compare concatwith the method push. pushañade elementos al final del propio arreglo sobre el que es invocado, alterando dicho arreglo. Aquí hay un ejemplo:

**const arr = [1, 2, 3];**

**arr.push(4, 5, 6);**

arrha sido modificado a [1, 2, 3, 4, 5, 6], hecho que no cumple con los principios de la programación funcional.

concatofrece una manera de unir nuevos elementos al final del arreglo sin ningún efecto colateral.

**function nonMutatingPush(original, newItem) {**

**// Cambia solo el código debajo de esta línea**

**return original.concat(newItem);**

**// Cambia solo el código encima de esta línea**

**}**

**const first = [1, 2, 3];**

**const second = [4, 5];**

**nonMutatingPush(first, second);**

**Utiliza el metodo "reduce" para analizar datos**

Array.prototype.reduce()o simplemente reduce()es la operacion mas comun de todos para arreglos en JavaScript. Se puede resolver casi cualquier problema de procesamiento de arreglos utilizando el método reduce.

El método reducepermite formas más generales de procesamiento de arreglos y es posible mostrar que tanto filtercomo mappueden derivarse como aplicaciones especiales de reduce. El método reduceitera sobre cada elemento del arreglo y devuelve un solo valor (por ejemplo una cadena, número, objeto, arreglo). Esto se consigue mediante una función callback que se llama en cada iteración.

La función callback acepta cuatro argumentos. El primer argumento se conoce como acumulador, que recibe el valor devuelto de la función callback de la iteración anterior, el segundo es el elemento actual que se está procesando, el tercero es el índice de ese elemento y el cuarto es el arreglo sobre el que se llama a la función reduce.

Además de la función callback, reducetiene un parámetro adicional que toma un valor inicial para el acumulador. Si este segundo parámetro no se utiliza, entonces la primera iteración se omite y la segunda se pasa por el primer elemento del arreglo como acumulador.

Mira a continuación un ejemplo con reduceel arreglo userspara devolver la suma de todas las edades de los usuarios. Para hacerlo más fácil, el ejemplo sólo utiliza el primer y segundo argumento.

**const users = [**

**{ name: 'John', age: 34 },**

**{ name: 'Amy', age: 20 },**

**{ name: 'camperCat', age: 10 }**

**];**

**const sumOfAges = users.reduce((sum, user) => sum + user.age, 0);**

**console.log(sumOfAges);**

La consola muestra el valor 64.

En otro ejemplo, se puede observar cómo un objeto puede ser devuelto con los nombres de los usuarios como propiedades con las edades como valores.

**const users = [**

**{ name: 'John', age: 34 },**

**{ name: 'Amy', age: 20 },**

**{ name: 'camperCat', age: 10 }**

**];**

**const usersObj = users.reduce((obj, user) => {**

**obj[user.name] = user.age;**

**return obj;**

**}, {});**

**console.log(usersObj);**

La consola muestra el valor { John: 34, Amy: 20, camperCat: 10 }.

**// La variable global**

**const watchList = [**

**{**

**"Title": "Inception",**

**"Year": "2010",**

**"Rated": "PG-13",**

**"Released": "16 Jul 2010",**

**"Runtime": "148 min",**

**"Genre": "Action, Adventure, Crime",**

**"Director": "Christopher Nolan",**

**"Writer": "Christopher Nolan",**

**"Actors": "Leonardo DiCaprio, Joseph Gordon-Levitt, Elliot Page, Tom Hardy",**

**"Plot": "A thief, who steals corporate secrets through use of dream-sharing technology, is given the inverse task of planting an idea into the mind of a CEO.",**

**"Language": "English, Japanese, French",**

**"Country": "USA, UK",**

**"Awards": "Won 4 Oscars. Another 143 wins & 198 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMjAxMzY3NjcxNF5BMl5BanBnXkFtZTcwNTI5OTM0Mw@@.\_V1\_SX300.jpg",**

**"Metascore": "74",**

**"imdbRating": "8.8",**

**"imdbVotes": "1,446,708",**

**"imdbID": "tt1375666",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "Interstellar",**

**"Year": "2014",**

**"Rated": "PG-13",**

**"Released": "07 Nov 2014",**

**"Runtime": "169 min",**

**"Genre": "Adventure, Drama, Sci-Fi",**

**"Director": "Christopher Nolan",**

**"Writer": "Jonathan Nolan, Christopher Nolan",**

**"Actors": "Ellen Burstyn, Matthew McConaughey, Mackenzie Foy, John Lithgow",**

**"Plot": "A team of explorers travel through a wormhole in space in an attempt to ensure humanity's survival.",**

**"Language": "English",**

**"Country": "USA, UK",**

**"Awards": "Won 1 Oscar. Another 39 wins & 132 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMjIxNTU4MzY4MF5BMl5BanBnXkFtZTgwMzM4ODI3MjE@.\_V1\_SX300.jpg",**

**"Metascore": "74",**

**"imdbRating": "8.6",**

**"imdbVotes": "910,366",**

**"imdbID": "tt0816692",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "The Dark Knight",**

**"Year": "2008",**

**"Rated": "PG-13",**

**"Released": "18 Jul 2008",**

**"Runtime": "152 min",**

**"Genre": "Action, Adventure, Crime",**

**"Director": "Christopher Nolan",**

**"Writer": "Jonathan Nolan (screenplay), Christopher Nolan (screenplay), Christopher Nolan (story), David S. Goyer (story), Bob Kane (characters)",**

**"Actors": "Christian Bale, Heath Ledger, Aaron Eckhart, Michael Caine",**

**"Plot": "When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, the caped crusader must come to terms with one of the greatest psychological tests of his ability to fight injustice.",**

**"Language": "English, Mandarin",**

**"Country": "USA, UK",**

**"Awards": "Won 2 Oscars. Another 146 wins & 142 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMTMxNTMwODM0NF5BMl5BanBnXkFtZTcwODAyMTk2Mw@@.\_V1\_SX300.jpg",**

**"Metascore": "82",**

**"imdbRating": "9.0",**

**"imdbVotes": "1,652,832",**

**"imdbID": "tt0468569",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "Batman Begins",**

**"Year": "2005",**

**"Rated": "PG-13",**

**"Released": "15 Jun 2005",**

**"Runtime": "140 min",**

**"Genre": "Action, Adventure",**

**"Director": "Christopher Nolan",**

**"Writer": "Bob Kane (characters), David S. Goyer (story), Christopher Nolan (screenplay), David S. Goyer (screenplay)",**

**"Actors": "Christian Bale, Michael Caine, Liam Neeson, Katie Holmes",**

**"Plot": "After training with his mentor, Batman begins his fight to free crime-ridden Gotham City from the corruption that Scarecrow and the League of Shadows have cast upon it.",**

**"Language": "English, Urdu, Mandarin",**

**"Country": "USA, UK",**

**"Awards": "Nominated for 1 Oscar. Another 15 wins & 66 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BNTM3OTc0MzM2OV5BMl5BanBnXkFtZTYwNzUwMTI3.\_V1\_SX300.jpg",**

**"Metascore": "70",**

**"imdbRating": "8.3",**

**"imdbVotes": "972,584",**

**"imdbID": "tt0372784",**

**"Type": "movie",**

**"Response": "True"**

**},**

**{**

**"Title": "Avatar",**

**"Year": "2009",**

**"Rated": "PG-13",**

**"Released": "18 Dec 2009",**

**"Runtime": "162 min",**

**"Genre": "Action, Adventure, Fantasy",**

**"Director": "James Cameron",**

**"Writer": "James Cameron",**

**"Actors": "Sam Worthington, Zoe Saldana, Sigourney Weaver, Stephen Lang",**

**"Plot": "A paraplegic marine dispatched to the moon Pandora on a unique mission becomes torn between following his orders and protecting the world he feels is his home.",**

**"Language": "English, Spanish",**

**"Country": "USA, UK",**

**"Awards": "Won 3 Oscars. Another 80 wins & 121 nominations.",**

**"Poster": "http://ia.media-imdb.com/images/M/MV5BMTYwOTEwNjAzMl5BMl5BanBnXkFtZTcwODc5MTUwMw@@.\_V1\_SX300.jpg",**

**"Metascore": "83",**

**"imdbRating": "7.9",**

**"imdbVotes": "876,575",**

**"imdbID": "tt0499549",**

**"Type": "movie",**

**"Response": "True"**

**}**

**];**

**function getRating(watchList){**

**// Add your code below this line**

**const averageRating = watchList**

**// Use filter to find films directed by Christopher Nolan**

**.filter(film => film.Director === "Christopher Nolan")**

**// Use map to convert their ratings from strings to numbers**

**.map(film => Number(film.imdbRating))**

**// Use reduce to add together their ratings**

**.reduce((sumOfRatings, rating) => sumOfRatings + rating) /**

**// Divide by the number of Nolan films to get the average rating**

**watchList.filter(film => film.Director === "Christopher Nolan").length;**

**// Add your code above this line**

**return averageRating;**

**}**

**console.log(getRating(watchList));**

**Utiliza las funciones de orden superior "mapear", "filtrar" o "reducir" para resolver un problema complejo**

Ahora que has superado algunos desafíos usando funciones de orden superior como map(), filter(), y reduce(), ahora puedes aplicarlos para resolver un desafío más complejo.

**const squareList = arr => {**

**// Cambia solo el código debajo de esta línea**

**return arr**

**.filter(num => num > 0 && num % parseInt(num) === 0)**

**.map(num => Math.pow(num, 2));**

**// Cambia solo el código encima de esta línea**

**};**

**const squaredIntegers = squareList([-3, 4.8, 5, 3, -3.2]);**

**console.log(squaredIntegers);**

**Ordena un arreglo alfabéticamente con el método sort**

El método sortordena los elementos de un arreglo de acuerdo a la función callback.

Por ejemplo:

**function ascendingOrder(arr) {**

**return arr.sort(function(a, b) {**

**return a - b;**

**});**

**}**

**ascendingOrder([1, 5, 2, 3, 4]);**

**Esto devolvería el valor de [1, 2, 3, 4, 5].**

**function reverseAlpha(arr) {**

**return arr.sort(function(a, b) {**

**return a === b ? 0 : a < b ? 1 : -1;**

**});**

**}**

**reverseAlpha(['l', 'h', 'z', 'b', 's']);**

Esto devolvería el valor de ['z', 's', 'l', 'h', 'b'].

Por defecto, JavaScript ordena demasiado en el valor "Unicode" de la cadena de caracteres, lo cual puede dar resultados inesperados. Por lo tanto, se recomienda proporcionar una función callback para especificar cómo se deben ordenar los elementos del arreglo. Cuando se proporciona dicha función callback, normalmente compareFunction, los elementos del arreglo se ordenan de acuerdo al valor que devuelve la función compareFunction: Si compareFunction(a,b)devuelve un valor menor que 0 para dos elementos ay b, entonces airá antes que b. Si compareFunction(a,b)devuelve un valor mayor a 0 para dos elementos ay b, entonces birá antes que a. Si compareFunction(a,b)devuelve un valor igual a 0 para dos elementos ay b, entoncesay bse dejarán sin cambios.

**function alphabeticalOrder(arr) {**

**// Cambia solo el código debajo de esta línea**

**return arr.sort();**

**// Cambia solo el código encima de esta línea**

**}**

**alphabeticalOrder(["a", "d", "c", "a", "z", "g"]);**

**Devuelve un arreglo ordenado sin cambiar el arreglo original**

Un efecto secundario del método sortes que cambia el orden de los elementos en el arreglo original. En otras palabras, muta el arreglo en su sitio. Una forma de evitar esto es concatenar primero un arreglo vacío al que está siendo ordenado (recuerda que slicey concatdevuelven un nuevo arreglo), luego ejecute el método sort.

**const globalArray = [5, 6, 3, 2, 9];**

**function nonMutatingSort(arr) {**

**// Cambia solo el código debajo de esta línea**

**var localArray = arr.slice();**

**return localArray.sort((a,b) => a-b)**

**// Cambia solo el código encima de esta línea**

**}**

**nonMutatingSort(globalArray);**

**Divide una cadena en un arreglo usando el método "split"**

El método splitdividir una cadena en un arreglo de cadenas. Se necesita un argumento para el delimitador, el cual puede ser un carácter para separar la cadena o una expresión regular. Por ejemplo, si el delimitador es un espacio, se obtiene un arreglo de palabras y si el delimitador es una cadena vacía, se obtiene un arreglo de cada carácter en la cadena.

Aquí hay dos ejemplos que dividen una cadena por espacios, luego otra por dígitos usando una expresión regular:

**const str = "Hello World";**

**const bySpace = str.split(" ");**

**const otherString = "How9are7you2today";**

**const byDigits = otherString.split(/\d/);**

bySpacetendrá el valor ["Hello", "World"]y byDigitstendrá el valor ["How", "are", "you", "today"].

Dado que las cadenas son inmutables, el método splitfacilita el trabajo con ellas.

**function splitify(str) {**

**// Cambia solo el código debajo de esta línea**

**return str.split(/[\s,.-]/)**

**// Cambia solo el código encima de esta línea**

**}**

**splitify("Hello World,I-am code");**

**Combina un arreglo en una cadena usando el método "join"**

El método joinse utiliza para unir los elementos de un arreglo creando una cadena de texto. Se necesita un argumento para especificar el delimitador a utilizar para separar los elementos del arreglo en la cadena.

**Aquí hay un ejemplo:**

**const arr = ["Hello", "World"];**

**const str = arr.join(" ");**

strtendrá una cadena con valor Hello World.

**function sentensify(str) {**

**// Cambia solo el código debajo de esta línea**

**return str.split(/\W/).join(" ");**

**// Cambia solo el código encima de esta línea**

**}**

**sentensify("May-the-force-be-with-you");**

**Aplicar programación funcional para convertir cadenas a slugs de URL**

Los últimos desafíos abarcaban una serie de métodos útiles para arreglos y cadenas, que siguen los principios de la programación funcional. También hemos aprendido acerca de reduce, que es un poderoso método utilizado para reducir los problemas a formas más simples. Desde el cálculo de promedios a la ordenación, cualquier operación con arreglos puede lograrse si lo aplicamos. Recuerda que mapy filterson casos especiales de reduce.

Combinemos lo que hemos aprendido para resolver un problema práctico.

Muchos sitios de gestión de contenidos (CMS) tienen los títulos de una publicación añadidos como parte de la URL con el simple propósito de ser marcadores. Por ejemplo, si escribe una publicación titulada Stop Using Reduce, es probable que la url tenga parte de este título en ella ( .../stop-using-reduce). Puede que ya haya notado esto en el sitio de freeCodeCamp.

**// the global variable**

**var globalTitle = "Winter Is Coming";**

**// Add your code below this line**

**function urlSlug(title) {**

**return title**

**.split(" ")**

**.filter(substr => substr !== "")**

**.join("-")**

**.toLowerCase();**

**}**

**// Add your code above this line**

**var winterComing = urlSlug(globalTitle); // Should be "winter-is-coming"**

**Usa el método "every" para comprobar que cada elemento de un arreglo atienda un criterio**

El método everyfunciona con arreglos para comprobar si cada elemento pasa una prueba en particular. Devuelve un valor booleano - truesi todos los valores cumplen los criterios, falsesi no.

Por ejemplo, el siguiente código comprobaría si cada elemento en el arreglo numberses menor a 10:

**const numbers = [1, 5, 8, 0, 10, 11];**

**numbers.every(function(currentValue) {**

**return currentValue < 10;**

**});**

El método everydevolvería falseaquí.

**function checkPositive(arr) {**

**// Cambia solo el código debajo de esta línea**

**return arr.every(val => val > 0);**

**// Cambia solo el código encima de esta línea**

**}**

**checkPositive([1, 2, 3, -4, 5]);**

**Usa el método "some" para comprobar si algún elemento**

**en un arreglo cumple un criterio**

El método somefunciona con arreglos para comprobar si algún elemento pasa una prueba en particular. Devuelve un valor booleano truesi alguno de los valores cumple el criterio, falsesi no.

Por ejemplo, el siguiente código comprobará si algún elemento en el arreglo numberses menor que 10:

**const numbers = [10, 50, 8, 220, 110, 11];**

**numbers.some(function(currentValue) {**

**return currentValue < 10;**

**});**

El método somedevolverá true.

**function checkPositive(arr) {**

**return arr.some(elem => elem > 0);**

**}**

**checkPositive([1, 2, 3, -4, 5]);**

**Introducción a la currificación ya la aplicación de funciones parciales**

La aridad de una función es el número de argumentos que requiere. Currificar una función significa convertir una función de N aridades a N funciones de 1 aridad.

En otras palabras, reestructura una función, por lo que toma un argumento, luego devolverá otra función que toma el siguiente argumento, y así sucesivamente.

**A continuación un ejemplo:**

**function unCurried(x, y) {**

**return x + y;**

**}**

**function curried(x) {**

**return function(y) {**

**return x + y;**

**}**

**}**

**const curried = x => y => x + y**

**curried(1)(2)**

**curried(1)(2)devolverá 3.**

Esto es útil en tu programa si no puedes proporcionar todos los argumentos a una función al mismo tiempo. Puede guardar la llamada a cada función dentro de una variable, la cual mantendrá la referencia de la función devuelta que toma el siguiente argumento cuando este disponible. Aquí hay un ejemplo utilizando la función currificada del ejemplo anterior:

**const funcForY = curried(1);**

**console.log(funcForY(2)); // 3**

Similarmente, la aplicación de una función parcial puede describirse como aplicar algunos argumentos a la función al mismo tiempo y devolviendo una función que se aplica a más argumentos. A continuación un ejemplo:

**function impartial(x, y, z) {**

**return x + y + z;**

**}**

**const partialFn = impartial.bind(this, 1, 2);**

**partialFn(10); // 13**

**function add(x) {**

**// Cambia solo el código debajo de esta línea**

**return y => z => x + y + z;**

**// Cambia solo el código encima de esta línea**

**}**

**add(10)(20)(30);**