Funciones de alto nivel para el manejo de listas

Universidad del Valle

Programa

- Referencias por índices
- push
- forEach
- filter
- map
- reduce
- slice, splice and sort

Referencias por índices

Es una manera de acceder a los elementos de una lista, sin necesidad de recorrerla. Por ejemplo para acceder al elemento 3 de la lista(con valor 8) podemos escribir

```
let data = [6, 3, 8, 3, 1, 2, 4, -1]
let x = data[2]; // Lee. Retorna 8
 data[2] = 7; // Asigna el elemento 2 a 7
```

push

Agrega un elemento al final de una lista

```
let data = [6, 3, 8, 3, 1, 2, 4, -1]
data.push(10)
```

Modifica el elemento data. No se debe usar en programación funcional

forEach

Esta función aplica un procedimiento a cada uno de los elementos de una lista. Por ejemplo para imprimir sus elementos

```
let data = [6, 3, 8, 3, 1, 2, 4, -1]
data.forEach(x => console.log("x = " + x))
```

filter

Esta función filtra los elementos de una lista para los cuales f(x) es verdadero. Por ejemplo, para filtrar los números impares del arreglo:

```
let data = [6, 3, 8, 3, 1, 2, 4, -1]
data.filter(x => {return x % 2 !== 0})
```

map

Esta función genera una nueva lista aplican la función parámetros a cada elemento de la lista. En el ejemplo se transforma la lista de números en una lista de objetos con los valores originales en el elemento x y el cuadrado de cada elemento en y

```
let data = [6, 3, 8, 3, 1, 2, 4, -1]
data.map(x => {return {x: x, y: x * x}})
>[{x:6, y:36}, {x:3, y:9}, {x:8, y:64}, {x:3, y:9},...]
```

reduce

Esta función agrega los elementos de la lista para producir una única salida. La función recibe 2 parámetros: la función de acumulación y el valor inicial del acumulador. La función de acumulación recibe 2 parámetros, el acumulador y el valor actual de la lista. En el ejemplo mostramos la función que suma todos los elementos de la lista.

```
let data = [6, 3, 8, 3, 1, 2, 4, -1]
data.reduce((acum, x) => {return acum + x}, 0);
```

```
>26
```

slice

Esta función retorna una lista donde los elementos son una copia de los elementos de la lista original. Recibe el índice donde comienza la copia(incluido) y el índice donde debe terminar(excluido). Si el segundo índice no se especifica, se copian todos los elementos hasta el final de la lista. Si no se pasan parámetros se retorna una copia completa del arreglo. En ejemplo se copian los elementos con índices 1 y 2 del arreglo:

```
let data = [6, 3, 8, 3, 1, 2, 4, -1]
data.slice(1, 3);
>[3, 8]
```

Universidad del Valle

splice

modifica el contenido de un arreglo para quitarle o agregarle elementos. Cuando se usa para quitar elementos, el primer parámetro dice dónde comenzar a quitar y el segundo cuántos elementos quitar. Cuando se usa para agregar elementos, el primero dice donde comenzar y el segundo, que elementos agregar. Retorna los elementos que quita y modifica el arreglo original

```
let data = [6, 3, 8, 3, 1, 2, 4, -1]
data.splice(1, 3);
>[3, 8, 3]  data = [6, 1, 2, 4, -1]
```

sort

Ordena un arreglo usando la función de orden que se le especifique. Ejemplo: Ordenar la lista de manera ascendente

```
let data = [6, 3, 8, 3, 1, 2, 4, -1]
```

```
data.sort((a, b) => a - b);
```

Ordenar la lista de manera descendente

```
data.sort((a, b) => b - a);
```

sort

Ordena un arreglo usando la función de orden que se le especifique. Ejemplo: Ordenar la lista alfabéticamente usando el campo msg

```
let tweets = [{msg: "hello ;)", likes: 4},
{msg: "ahello ;)", likes: 8},
{msg: "zhello ;)", likes: 2},
{msg: "bhello ;)", likes: 15}];

console.log(tweets.sort((a, b) => a.msg.localeCompare(b.msg)));
```

sort

Ordena un arreglo usando la función de orden que se le especifique. Ejemplo: Ordenar la lista de manera ascendente usando la cantidad de likes

```
let tweets = [{msg: "hello ;)", likes: 4},
{msg: "ahello ;)", likes: 8},
{msg: "zhello ;)", likes: 2},
{msg: "bhello ;)", likes: 15}];


console.log(tweets.sort((a, b) => a.likes - b.likes));
```