Contenido de la clase

Tiempo de lectura
7 min

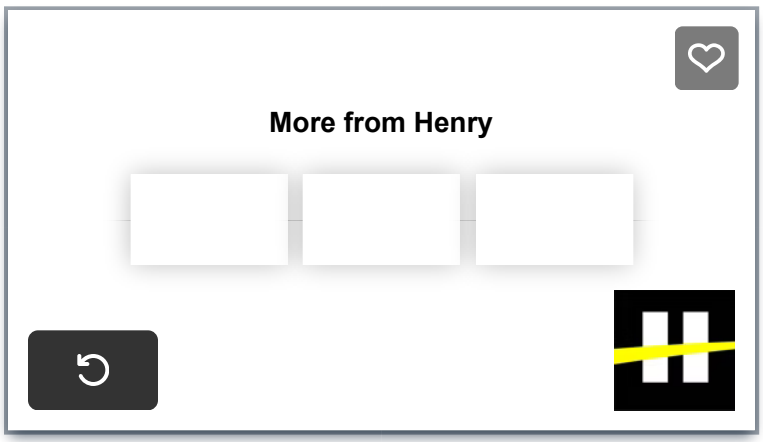
Homework 📝

JavaScript VI

Callbacks

En esta lección cubriremos:

- **Callbacks**
- **Más métodos de Arrays**
- **Introducción a la programación funcional**



Callbacks

Un concepto muy importante en Javascript es la capacidad de pasar una función como argumento a otra función. Estas funciones se denominan `callbacks` . Estas funciones pueden llamarse en cualquier momento y pasar argumentos dentro de la función. Pronto descubriremos por qué las

Dejanos tu feedback! 👍

Contenido de la clase

callback.

```
function decirHolaAlUsuario(usuario) {
  return "Hola " + usuario + "!";
}

function decirAdiosAlUsuario(usuario) {
  return "Adiós " + usuario + "!";
}

function crearSaludo(usuario, cb) {
  return cb(usuario);
}

crearSaludo("Dan", decirHolaAlUsuario); // 'Hel
crearSaludo("Dan", decirAdiosAlUsuario); // 'Go
```

Más métodos de Arrays

Ya conocemos y utilizamos métodos de matriz, `.push`, `.pop`, `.shift`, `.unshift` y `.length`. Pero hay muchos más métodos disponibles de forma nativa en un array. Los métodos de los que vamos a hablar aquí se denominan "métodos de orden superior", porque toman los callbacks como argumentos.

.forEach

`.forEach` es un bucle for integrado en cada array.

`.forEach` toma un callback como su único argumento, e itera sobre cada elemento de la matriz y llama al callback en él. El callback puede tomar dos argumentos, el primero es el elemento en sí, el segundo es el índice del elemento (este argumento es opcional).

Dejanos tu feedback! 👍

Contenido de la clase


```
// Podemos escribir el callback en los paréntes
autos.forEach(function (elemento, indice) {
  console.log(elemento);
});

// O podemos crear una instancia de una función
// Además, no necesitamos usar el argumento de
function mostrarNombres(elemento) {
  console.log(elemento);
}

// And call that function in the forEach parent
autos.forEach(mostrarNombres);
```

.reduce

`.reduce` ejecutará un bucle en nuestra matriz con la intención de reducir cada elemento en un elemento que se devuelve. Como es el primer argumento, acepta un callback que toma dos argumentos, primero un 'acumulador' (el resultado del método de reducción hasta ahora), y el segundo es el elemento en el que se encuentra actualmente. El callback debe contener siempre una declaración de devolución ("return"). `.reduce` también toma un segundo argumento opcional, que sería el acumulador de arranque ("starting accumulator"). Si no se suministra el acumulador de arranque, la reducción comenzará en el primer elemento de la matriz. `.reduce` siempre devolverá el acumulador cuando termine de recorrer los elementos.

```
const numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9];
const palabras = ["Hola,", "mi", "nombre", "es"

// Podemos escribir la función anónima directam
// Si omitimos el elemento inicial, siempre com
const suma = numeros.reduce(function (acc, elem
  return acc + elemento;
```

Dejanos tu feedback! 👍

Contenido de la clase


```
function multiplicarDosNumeros(a, b) {
  return a * b;
}

const productos = numeros.reduce(multiplicarDos

// .reduce funciona en cualquier tipo de datos.
// En este ejemplo configuramos un acumulador d
const frases = palabras.reduce(function (acc, e
  return acc + " " + elemento;
}, "Frase completa:");

console.log(suma); // 45
console.log(productos); // 362880
console.log(frases); // "Frase completa: Hola,
```

.map

`.map` se usa cuando queremos cambiar cada elemento de una matriz de la misma manera. `.map` toma una devolución de llamada como único argumento. Al igual que el método `.forEach`, el callback tiene el elemento y el índice de argumentos opcionales. A diferencia de `.reduce`, `.map` devolverá toda la matriz.

Dejanos tu feedback! 👍

Contenido de la clase


```
function multiplicarPorTres(elemento) {
  return elemento * 3;
}

const doble = numeros.map(function (elemento) {
  return elemento * 2;
});

const triple = numeros.map(multiplicarPorTres);

console.log(doble); // [ 4, 6, 8, 10 ]
console.log(triple); // [ 6, 9, 12, 15 ]
```

Recursos adicionales

- Understanding Callback Functions and How to Use Them
- Eloquent Javascript: Higher Order Functions
- MDN: Callback function
- MDN: Array methods

Homework

Completa la tarea descrita en el archivo README

Si tienes dudas sobre este tema, puedes consultarlas en el canal *07_js-vi* de Slack

Dejanos tu feedback!👍