MÁS SOBRE FUNCIONES

Callback (llamada de vuelta)

Una función callback es una función pasada a otra función como argumento, que es invocada dentro de la función externa para completar alguna rutina o acción.

```
function greeting(name) {
    alert('Hello ' + name);
}

function processUserInput(callback) {
    const name = prompt('Please enter your name.');
    callback(name);
}

processUserInput(greeting);
```

Callback

```
function add(n1, n2) { console.log(n1 + n2); }
function sub(n1, n2) { console.log(n1 - n2); }
function mult(n1, n2) { console.log(n1 * n2); }

function f(n1, n2, callback, callback2) {
   callback(n1, n2);
   callback2(n1, 3);
}

f(5, 6, mult, add);
```

Función que retorna función

```
function magic() {
   return function calc(x) { return x * 42; };
}

const answer = magic();
answer(1337); // 56154
```

Call (llamar), apply (aplicar) y bind (enlazar)

Las funciones Call, apply y bind son todas usadas para cambiar el ámbito al que `this` hace referencia dentro de una función o un método.

Call

```
const course = {
    name: '',
    description: '',
    students: [],
    addStudents (studentName) {
        this.students.push(studentName);
        console.log(`${studentName} added to
    ${this.name} course`);
    },
    date: '12/12/2021'
};

const english = {
    name: "english course",
    description: "this is good course" ,
    students: []
};

const math = {
    name: "math course",
    description: "this is very good course" ,
    students: []
};
```

```
const addStudents =
math.addStudents;
// add("asaf") // no va a
funcionar.
// esta función va a hacer
referencia a  undefined

addStudents.call(english, "asaf")
addStudents.call(math, "Dani")
addStudents.call(english, "asaf")
addStudents.call(math, "Ron")

console.log(math);
console.log(english);
```

Call

La función call crea una nueva instancia del objeto “course” llamado “english” y luego el “this” dentro de la función `addStudents` apunta al nuevo objeto.

```
addStudents (studentName) {
    this.students.push(studentName);
    console.log(`${studentName} added to
${this.name} course`);
```

```
const english = {
   name: "english course",
   description: "this is good course",
   students: []
}
```

```
const math = {
   name: "math course",
   description: "this is very good course",
   students: []
}
```

```
const addStudents =
math.addStudents;
// add("asaf") // no va a
funcionar
// esta función va a hacer
referencia a  undefined

addStudents.call(english, "asaf")
addStudents.call(math, "Dani")
addStudents.call(english, "asaf")
addStudents.call(math, "Ron")

console.log(math);
console.log(english);
```

Apply es igual pero la función obtiene un parámetro de tipo arreglo

```javascript
const course = {
    name: '',
    description: '',
    students: [],
    addStudents(studentName) {
        this.students.push(studentName);
        console.log(`${studentName} added to
        ${this.name} course`);
    },
    date: '12/12/2021'
};

const english = {
    name: "english course",
    description: "this is good course" ,
    students: []
};

const math = {
    name: "math course",
    description: "this is very good course" ,
    students: []
};
```

```javascript
const addStudents = math.addStudents;
// add("asaf") // no va a funcionar
// esta función va a hacer
referencia a  undefined

addStudents.apply(english, ["asaf"])
addStudents.apply(math, ["Dani"])
addStudents.apply(english, ["asaf"])
addStudents.apply(math, ["Ron"])

console.log(math);
console.log(english);
studentData = ['menny']
addStudents.call(english,
...studentData)
console.log(english);
```

Apply es igual pero la función obtiene un parámetro de tipo arreglo

La función apply crea una nueva instancia del objeto “course” llamado “english” y la referencia de this está dentro del nuevo objeto.

```
addStudents (studentName) {
    this.students.push(studentName);
    console.log(`${studentName} added to
${this.name} course`);
```

```
const english = {
   name: "english course",
   description: "this is good course",
   students: []
};
```

```
const math = {
   name: "math course",
   description: "this is very good course",
   students: []
};
```

```
const addStudents = math.addStudents;
// add("asaf") // no va a funcionar
// esta función va a hacer
referencia a  undefined

addStudents.apply(english, ["asaf"])
addStudents.apply(math, ["Dani"])
addStudents.apply(english, ["asaf"])
addStudents.apply(math, ["Ron"])

console.log(math);
console.log(english);
studentData = ['menny']
addStudents.call(english,
...studentData)
console.log(english);
```

arreglo

bind

```
const course = {
    name: '',
    description: '',
    students: [],
    addStudents (studentName) {
        this.students.push(studentName);
        console.log(`${studentName} added to
        ${this.name} course`);
    },
    date: '12/12/2021'
};

const english = {
    name: "english course",
    description: "this is good course" ,
    students: []
};

const math = {
    name: "math course",
    description: "this is very good course" ,
    students: []
};
```

```
const addStudents = math.addStudents;
// add("asaf") // no va a funcionar
// esta función va a hacer referencia a
undefined
// bind
const addToEnglsihStudents =
addStudnents.bind(english);
addToEnglsihStudents("Lara");
addToEnglsihStudents("Lisa");
addToEnglsihStudents("Morgan");
```

bind

La función "apply" crea una nueva instancia del objeto "course" llamada "english" y luego el "this" dentro de la función "addStudents" apuntará al nuevo objeto.

```
addStudents (studentName) {
     this.students.push(studentName);
     console.log(`${studentName} added to
${this.name} course`);
}
```

```
const english = {
   name: "english course",
   description: "this is good course",
   students: []
};
```

```
const math = {
   name: "math course",
   description: "this is very good course",
   students: []
};
```

```
const addStudents = math.addStudents;
// add("asaf") // no va a funcionar
// esta función va a hacer referencia
a  undefined
// bind
const addToEnglsihStudents =
addStudnents.bind(english);
addToEnglsihStudents("Lara");
addToEnglsihStudents("Lisa");
addToEnglsihStudents("Morgan");
```

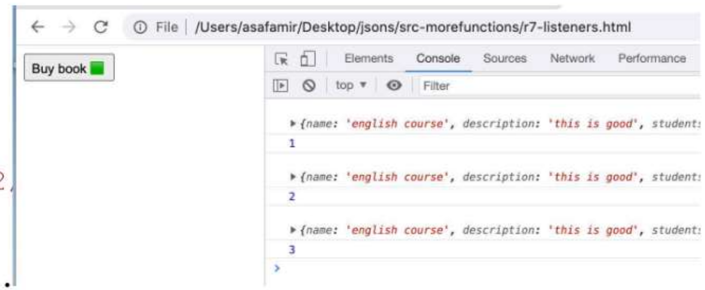
1. Crea un nuevo objeto “english” o “math”
2. Ahora puedes pasar el parámetro a la función

Bind y dom

```
<html>
    <body><button class="buy">Buy book 🟩</button></body>
</html>

const course = {
    name: '', books: 0, description: '', students: [], date: '12/12,
    addStudents(studentName) {
        this.students.push(studentName);
        console.log(`${studentName} added to ${this.name} course`);
    }
};

const english = {
    name: "english course",
    books: 0,
    description: "this is good course",
    students: []
}

english.buyCourse = function() {
    this.books++;
    console.log(this.books); // 1,2,3,,,,n
}

//document.querySelector('.buy').addEventListener('click', english.buyCourse)
document.querySelector('.buy').addEventListener('click',english.buyCourse.bind(english));
```

Agrega una función al objeto "english"

Tienes que enlazar la nueva función a la nueva instancia del objeto


File | /Users/asafamir/Desktop/jsons/src-morefunctions/r7-listeners.html
Buy book
Elements
Console
Sources
Network
Performance
top
Filter

```
▸ {name: 'english course', description: 'this is good', student:
1
▸ {name: 'english course', description: 'this is good', student:
2
▸ {name: 'english course', description: 'this is good', student:
3
```

wawiwa

¿Preguntas?