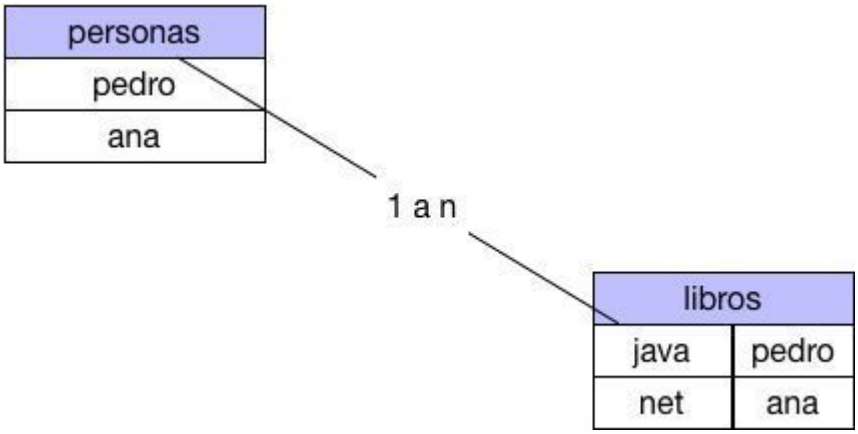
La idea de JavaScript Joins parece un poco extraña al principio . Sin embargo cada día trabajamos más con el formato JSON y leemos grupos de datos muy diversos.



Más pronto que tarde nos encontraremos con una situación que requiere realizar un join clásico de SQL pero con los datos en JSON. ¿Cómo podemos realizar esta operación?. Vamos a partir de las siguientes estructuras de datos:

```
pedro gomez 20
ana sanchez 30
```

```
introducción a java  300 pedro
programación net 300 ana
```

Estas estructuras se pueden construir en formato JSON :

```
var personas = [{
    nombre: "pedro",
    apellidos: "gomez",
    edad: 20
  },
  {
    nombre: "ana",
    apellidos: "sanchez",
    edad: 30
  }
];

var libros = [{
    titulo: "introduccion a java",
    paginas: 300,
    persona_nombre: "pedro"
  },
  {
    titulo: "programacion net",
    paginas: 300,
    persona_nombre: "ana"
  }
]
```

JavaScript Joins y programación funcional

En este caso tenemos por un lado Personas y por otro lado los Libros. Ambos conceptos están relacionados ya que una persona posee varios libros y un libro es de una persona. Se trata de la clásica relación 1 a n de SQL . Esta relación en SQL se resuelve con un join.

select * from personas inner join libros on personas.nombre= libros.personas_nombre;

¿Cómo generamos la misma consulta desde JavaScript? .Vamos a verlo:

```
var resultado = libros.map(function(libro) {

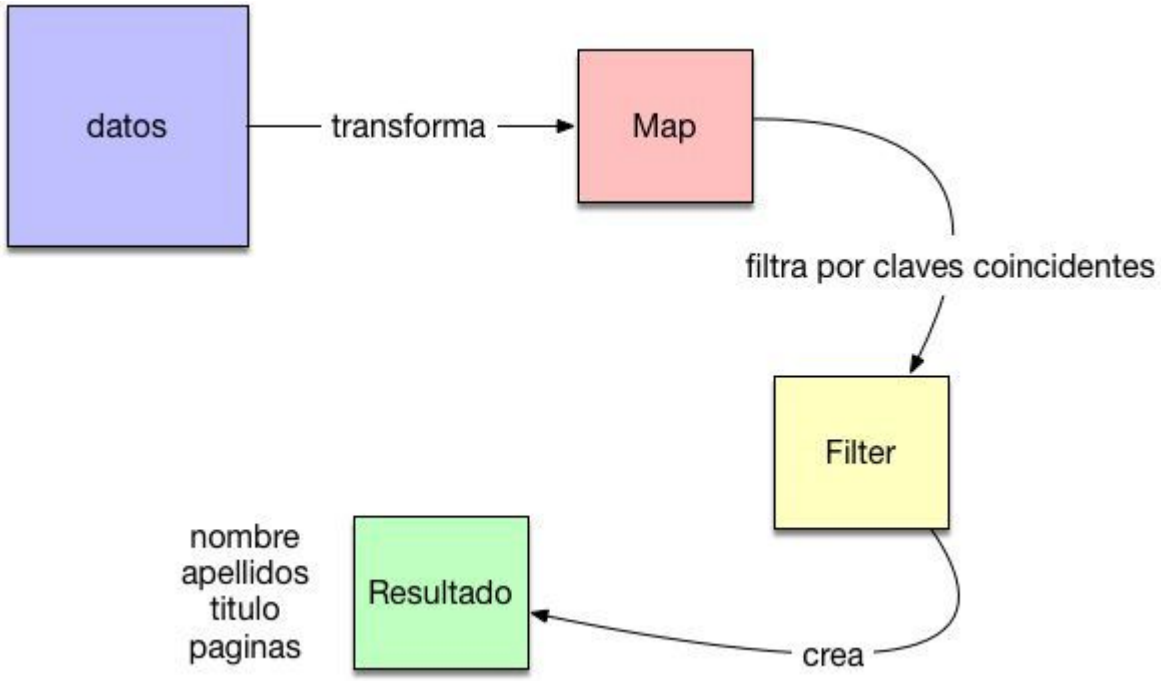
  var otraPersona = personas.filter(function(p) {

    return p.nombre == libro.persona_nombre;
  })[0];

  return {
    nombre: otraPersona.nombre,
    titulo: libro.titulo,
    paginas: libro.paginas,
    apellidos: otraPersona.apellidos
  }
});

console.log(resultado);
```

El código cuesta un poco entenderlo . En primer lugar seleccionamos el array de libros .Una vez seleccionado vamos a realizar una transformación con map que nos devuelva el conjunto de resultados que necesitamos. Esta transformación se encarga de buscar en el grupo de datos de personas aquellas en las que el nombre de la persona coincide con persona_nombre de libro. De esta forma conseguimos realizar la operación de join



Hecho esto diseñamos el conjunto de datos del resultado y los devolvemos como json.

```
[ { nombre: 'pedro',
    titulo: 'introduccion a java',
    paginas: 300,
    apellidos: 'gomez' },
  { nombre: 'ana',
    titulo: 'programacion net',
    paginas: 300,
    apellidos: 'sanchez' } ]
```

Acabamos de realizar un join con JavaScript y programación funcional. Poco a poco nos acostumbraremos a ver más los JavaScript joins en código.

Otros artículos relacionados:

1. JavaScript Observers vs Arrays
2. El concepto de JavaScript Spread operator
3. JavaScript Benchmarks y Benchmark.js