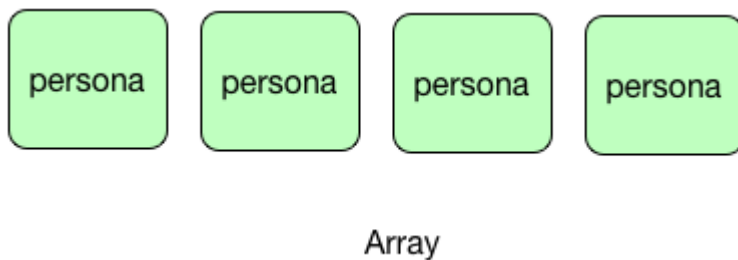


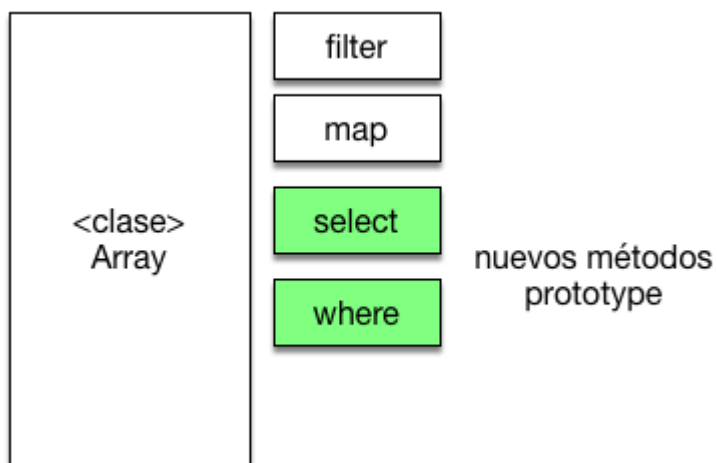
El uso de JavaScript Array Prototype es muy común y de hecho existen librerías como [lodash](#) y [underscore](#) que se apoyan intensamente en este concepto. Ahora bien ¿Cómo funciona?, ¿Cómo podemos extender la funcionalidad de un Array?. Vamos a construir un ejemplo que nos aclare la situación partiendo de una lista de personas sencilla que construimos:

```
var personas = [{
    nombre: "pedro",
    apellidos: "gomez",
    edad: 20
},
{
    nombre: "ana",
    apellidos: "sanchez",
    edad: 30
},
{
    nombre: "antonio",
    apellidos: "blanco",
    edad: 35
}, {
    nombre: "pablo",
    apellidos: "perez",
    edad: 30
}
];
```

En este caso disponemos de un array que contiene cuatro personas :



Deseamos seleccionar un subconjunto de personas utilizando una sintaxis cercana a SQL. Para ello añadiremos los métodos de select y where a la clase Array utilizando prototypes.



Vamos a ver el código:

```
Array.prototype.where = function(filtroInclusion) {  
  let resultados = [];  
  for (let i = 0; i < this.length; i++) {  
    if (filtroInclusion(this[i]))  
      resultados.push(this[i]);  
  }  
}
```

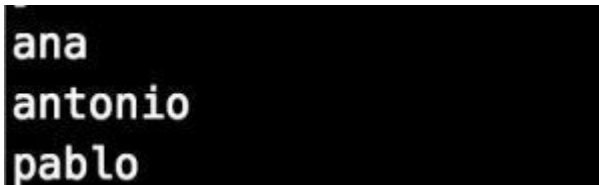
```
    }  
    return resultados;  
};
```

```
Array.prototype.select = function(seleccion) {  
    let resultados = [];  
    for (let i = 0; i < this.length; i++) {  
        resultados.push(seleccion(this[i]));  
    }  
    return resultados;  
};
```

Una vez añadidos estos dos prototipos podremos operar de una forma muy diferente con nuestro array de datos. En este caso vamos a seleccionar el nombre de todas las personas que sean mayores de 30 años. Para ello usaremos las cláusulas where y select que acabamos de construir.

```
personas.where(function(persona) {  
  
    return persona.edad >= 30;  
}).select(function(persona) {  
  
    return persona.nombre;  
  
}).forEach(function(nombre) {  
    console.log(nombre);  
  
});
```

El resultado por la pantalla será:



```
ana
antonio
pablo
```

Nos hemos acercado más a un enfoque tipo SQL . Podemos modificar el código y utilizar arrow function de JavaScript ES6 para que nuestro código se reduzca en tamaño y sea mas legible y natural.

```
personas
  .where(persona=>persona.edad>=30)
  .select(persona=>persona.nombre)
  .forEach(nombre=>console.log(nombre));
```

Esto ya es un enfoque mucho más cercano a la programación funcional con JavaScript ES6. Acabamos de utilizar JavaScript Array Prototype para extender la funcionalidad de la clase Array y añadir métodos que nos sean útiles.

Otros artículos relacionados:

1. [JavaScript Prototypes y su uso](#)
2. [JavaScript promise con jQuery](#)
3. [El uso de JavaScript for in vs for of](#)
4. [JavaScript Streams vs Promises](#)