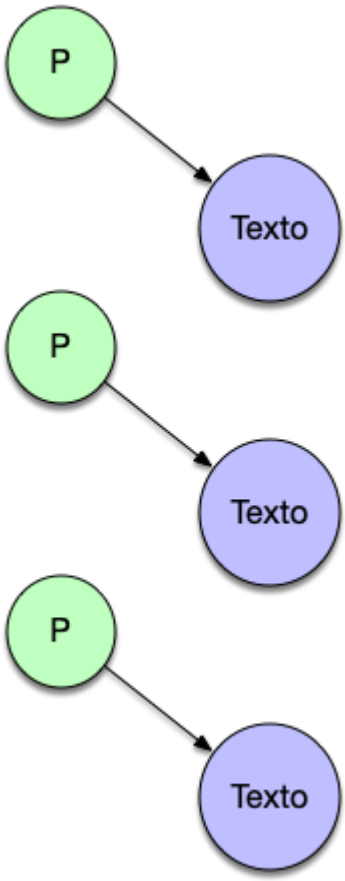
El concepto de jQuery collection y la programación funcional es a veces un gran olvidado y sin embargo en más de una ocasión nos puede ser muy útil en la programación del día a día. Vamos a ver un ejemplo sencillo en el cual deseamos sumar el contenido de una serie de párrafos . En principio es algo que con jQuery no tiene mucha miga sería suficiente con construir algo del siguiente estilo:

```
<html>

<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"
></script>
  <script type="text/javascript">
    $(document).ready(function() {

        var total=0;

        $("#sumar").click (function() {

              $("p").each(function() {

                  total+= parseInt($(this).text());
            })

            $("#resultado").html(total);

        })



    })
```

```
  </script>
</head>

<body>
  <p>
    1
  </p>
  <p>
    2
  </p>
  <p>
    3
  </p>
  <input type="button" value="sumar" id="sumar"/>
  <div id="resultado">
  </div>
</body>

</html>
```

Como podemos ver hemos recorrido la lista de nodos y los hemos sumado todo funciona correctamente.



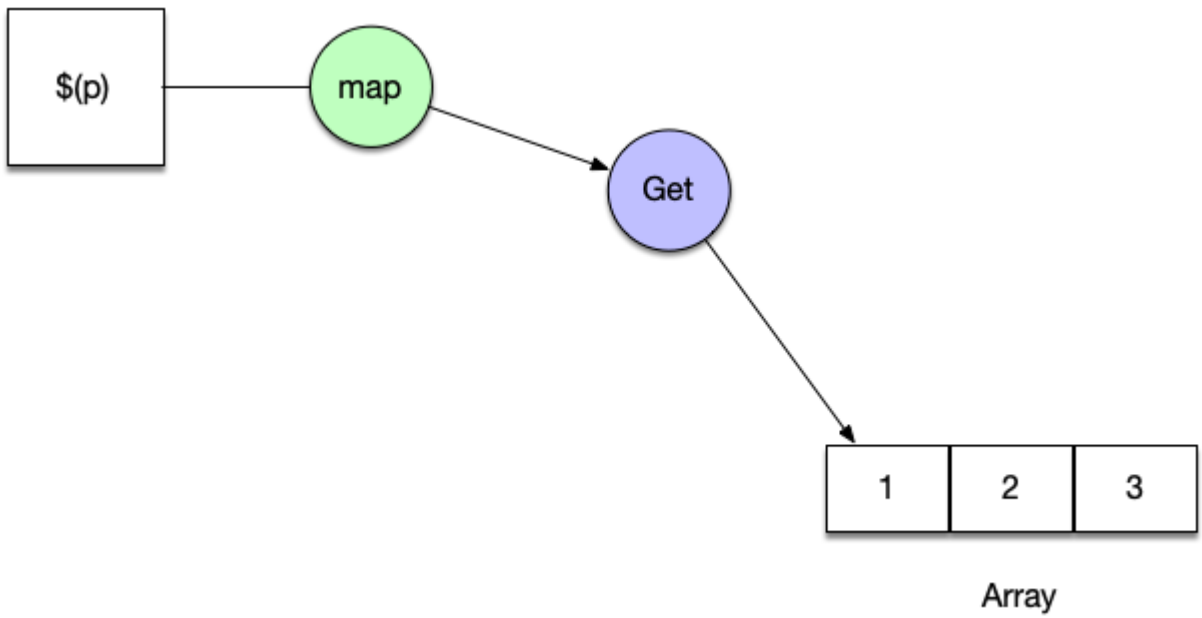
¿Ahora bien es la mejor forma de hacerlo? . Esta es una buena pregunta ya que en jQuery estamos mas que habituados a utilizar variables globales para realizar este tipo de operaciones y programación imperativa . ¿Es posible utilizar programación funcional en nuestro código y presentar una solución más flexible?. La realidad es que sí . Podemos usar la función map de jQuery y convertir la lista de nodos DOM en un sencillo array de numeros y luego usar una reducción para sumarlos . Esto nos permite aumentar la flexibilidad ya que una vez disponemos del array podríamos definir operaciones complementarias de forma sencilla

```
<html>

<head>
  <script
```

```
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"
></script>
  <script type="text/javascript">
    $(document).ready(function() {

        $("#sumar").click (function() {


            let numeros=$("p")
            .map(function() {

                return parseInt($(this).text());

            })
            .get();

            let suma=numeros.reduce(function(total, numero) {
              return total+numero;
            },0)

            $("#resultado").html(resultado);

        })




    })
  </script>
</head>
```

```
<body>
  <p>
    1
  </p>
  <p>
    2
  </p>
  <p>
    3
  </p>
  <input type="button" value="sumar" id="sumar"/>
  <div id="resultado">
  </div>
</body>

</html>
```

En este caso la función map nos ha devuelto un array de valores a nivel de jQuery con la función get nos quedamos con el array puro.



Acostumbremonos a usar más la programación funcional con las librerías o frameworks habituales ya que nos permitirá trabajar de forma más cómoda y flexible con jQuery collection en nuestro código.

Otros artículos relacionados

- jQuery on off
- jQuery Data
- $.get y $post
- jQuery map

Cecilio Álvarez Caules
Cecilio Álvarez Caules Oracle Java Certified Architech