ES6: NEXT GEN JS

Peoria JS and Web Professionals

July, 2018

GENERATORS

Note: We use an asterisk when we define our function. We also use yield instead of return. This is somewhat like a continue in a loop. The function is "paused" until it is called again and the next item is yielded.

Reference: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Generator

Code snippet: let's display some numbers

```
16 ▼ function* someNumbers(){
17       yield 27310;
18       yield 50780;
19       yield 53323;
20       yield 54051;
21       yield 62310;
22   }
```

Note: if you define a variable within a generator, the scope of the variable is that function. However, the function is not finished until the final call. Therefore, I recommend using **let** instead of **const** in such a function (particularly if you are making changes to a variable and yielding the results).

Result when viewed in the browser console is shown below. Note we have to make a final call to the function to return done:true.

Code snippet: let's loop through an array with a generator.

```
const fossils = [
    {accession:'12345',age:'Jurassic', species:'dragonfly'},
    {accession:'34621',age:'Creataceous', species:'water
    strider'},
    {accession:'92731',age:'Permian', species:'dragonfly'},
    {accession:'68390',age:'Jurassic', species:'beetle'},
    {accession:'38556',age:'Miocene', species:'ant'}
    ];

function* loopThrough(myArr){
    for (const item of myArr){
        yield item;
    }
}

const fossilList = loopThrough(fossils);

console.log(fossilList.next());
```

Result when viewed in the browser console is shown below.

Note: you can append .value to the function .next() call if you don't want to see the done status.

Reference file: **09Generators.html**

Your mission: build a generator function of your choice. Verify all is working in the console log.