13 – Polyfills

A Polyfill is a piece of code used to provide modern JavaScript functionality to older browsers that do not support it. They allow backward compatibility.

For example, a Polyfill could be written to mimic the functionality of the .reduce() method for arrays.

The reason why Polyfills are not used exclusively is for better performance. A native implementation of a functionality will outperform a Polyfill written for the same.

**Protocol to Write Polyfills**

Step 1: Understand the signature of the functionality that you intend to implement

The .forEach() function is an Array method that applies a given callback to each element of the array. However, it does not return a new array or any values. The callback has two parameters – the value of the current index and the current index itself.

Step 2: Write the Polyfill on the basis of the signature.

if(!Array.prototype.forEach) { // if it doesn’t exist in the base class of Arrays

Array.prototype.forEach = function forEach(callback) {

for(i = 0; i < this.length; i++) {

let value = this[i];

callback(value, i);

}

}

}

**Polyfills for Important Array Methods**

1) .map()

Signature: The .map() method applies a callback to each element of the array that it is called upon and returns a new array consisting of the values returned from the callback. The callback function has two parameters – the current value of the array and the index of that value.

Writing the Polyfill:

if(!Array.prototype.map) {

Array.prototype.map = function map(callback) {

let arr = []

for(i = 0; i < this.length; i++){

let value = callback(this[i], i);

arr.push(value);

}

return arr

}

}

2) .reduce()

Signature: The .reduce() method applies a callback function to each element of the array and returns a single value at the end. The second parameter of the .reduce() method is the initial value of the accumulator, which is optional. The callback takes two parameters – the accumulator, which is the final value returned, and the current value of the array.

Writing the Polyfill:  
  
if(!Array.prototype.reduce) {

Array.prototype.reduce = function(callback, init) {

let acc = init ? init : this[0]

const startIdx = (init ? 1 : 0)

for(i = startIdx; i < this.length; i++){

acc = callback(acc, this[i])

}

return acc

}

}

3) .filter()

Signature: The .filter() method returns a new array that consists of only selective elements from the original array that pass a given test.

Writing the Polyfill:   
  
if(!Array.prototype.filter) {

Array.prototype.filter = function filter(callback) {

let arr = [];

for(i = 0; i < this.length; i++) {

if(callback(this[i])) {

arr.push(this[i]);

}

}

return arr

}

}