Polyfills

Polyfills in JavaScript refer to code that is used to provide modern features or functionality to older browsers that do not support those features natively. These older browsers might lack support for certain JavaScript methods, APIs, or language features that have been introduced in newer versions of the language.

When a developer wants to use these modern features in their code and ensure compatibility with older browsers, they can include polyfills. Polyfills essentially "fill in" the gaps in functionality by providing custom implementations of these features that work on older browsers.

forEach

```
Array.prototype.myForEach = (callback)=>
{
  for (var i = 0; i < this.length; i++)
  {
    callback(this[i], i, this);
  }
};


let movies = ["IRON MAN", "BATMAN", "Godzilla"];

movies.myForEach((movie)=>
{
  console.log(movie);
})
```

Map

```
Array.prototype.myMap = function(cb)
{
  const arr = [];
  for(var i=0; i < this.length; i++)
  {
    arr.push(cb(this[i], i, this));
  }
  return arr;
}

let movies = ["IRON MAN", "BATMAN", "Godzilla"];

const result = movies.myMap((movie)=>
{
  return `${movie} 1`;
})

console.log(result);
```

filter

```
Array.prototype.myFilter = function(cb)
{
  const arr = [];
  for(var i=0; i < this.length; i++)
  {
    if(cb(this[i], i, this))
    {
      arr.push(this[i]);
    }
  }
  return arr;
}

let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16];
const evenNums = nums.myFilter((num)=> num %2 == 0);
console.log(evenNums);
```

find

```
Array.prototype.myFind = function(cb)
{
  for(var i=0; i < this.length; i++)
  {
    if(cb(this[i], i, this))
    {
      return this[i];
    }
  }
  return undefined;
}

let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16];
const firstEvenNum = nums.myFind((num)=> num %2 == 0);
console.log(firstEvenNum);
```

findIndex

```
Array.prototype.myFindIndex = function(cb)
{
  for(var i=0; i < this.length; i++)
  {
    if(cb(this[i], i, this))
    {
      return i;
    }
  }
  return -1;
}

let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16];
const firstEvenNumIndex = nums.myFindIndex((num)=> num %2 == 0);
console.log(firstEvenNumIndex);
```

some

```
Array.prototype.mySome = function(cb)
{
  for(var i=0; i < this.length; i++)
  {
    if(cb(this[i], i, this))
    {
      return true;
    }
  }
  return false;
}

let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16];
const containsEvenNums = nums.mySome((num)=> num %2 == 0);
console.log(containsEvenNums);
```

every

```
Array.prototype.myEvery = function(cb)
{
  for(var i=0; i < this.length; i++)
  {
    if(!cb(this[i], i, this))
    {
      return false;
    }
  }
  return true;
}

let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16];
const allEvens = nums.myEvery((num)=> num %2 == 0);
console.log(allEvens);
```

reduce

```
Array.prototype.myReduce = function(cb, initialValue)
{
  let accumulator = initialValue !== undefined ? initialValue : array[0];
  const startIndex = initialValue !== undefined ? 0 : 1;
  for(var i = startIndex; i < this.length; i++)
  {
    accumulator = cb(accumulator, this[i], i, this);
  }
  return accumulator;
}

let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16];
const sum = nums.myReduce((total, currNum)=> total + currNum, 50);
console.log(sum);
```

bind

```
Function.prototype.myBind = function(context, ...args)
{
    return (...innerArgs)=>
    {
        return this.apply(context, [...args, ...innerArgs])
    }
}

function displayFullname(...args)
{
    console.log(`${this.firstName} ${this.lastName}`);
    const sum = args.reduce((acc, curr)=> acc + curr);
    console.log(`sum is ${sum}`);
}


const display = displayFullname.myBind({firstName: 'John', lastName:
'Doe'}, 5, 6, 7);

display(8, 9, 10);//45
```