

Arrays Iteration

iteration - motivation

array of numbers that you want to round
to the nearest whole number

```
const decimals = [1.1, 1.6, 2.8, 0.4, 3.5, 1.6];
```

```
decimals[0] = Math.round(decimals[0]);
```

```
decimals[1] = Math.round(decimals[1]);
```

```
decimals[2] = Math.round(decimals[2]);
```

```
decimals[3] = Math.round(decimals[3]);
```

```
decimals[4] = Math.round(decimals[4]);
```

```
decimals[5] = Math.round(decimals[5]);
```

What if we have 100 numbers we want to
round? Or 1,000?

for loops

One of the most common ways to loop is with a for loop.

```
const decimals = [1.1, 1.6, 2.8, 0.4, 3.5, 1.6];
```

```
for (let i = 0; i < decimals.length; i++) {  
  decimals[i] = Math.round(decimals[i]);  
}  
console.log(decimals);
```

For-in loops

Another method used to iterate through the loop is to use a for-in loop.

```
const decimals = [1.1, 1.6, 2.8, 0.4, 3.5, 1.6];
```

```
for (let i in decimals) {  
  decimals[i] = Math.round(decimals[i]);  
}  
console.log(decimals);
```

while loops

```
let decimals = [1.1, 1.6, 2.8, 0.4, 3.5, 1.6];  
let j = 0;
```

```
while (j < decimals.length) {  
  decimals[j] = Math.round(decimals[j]);  
  j++;  
}  
console.log(decimals);
```

do while loops

```
const decimals = [1.1, 1.6, 2.8, 0.4, 3.5, 1.6];
```

```
var p = 0;
```

```
do {
```

```
  decimals[p] = Math.round(decimals[p]);
```

```
  p++;
```

```
} while (p < decimals.length);
```

```
console.log(decimals);
```

looping over strings

Since strings have a length property, we always know at what point to stop looping, just like with arrays.

```
const name = 'elie';
```

```
for (let t = 0; t < name.length; t++) {  
  console.log(name[t]);  
}
```

```
// e
```

```
// l
```

```
// i
```

```
// e
```

Exiting out of loops: break

Sometimes we want to exit a loop before it has finished. To do that, we use the word `break`

```
for (let q = 0; q < 5; q++) {  
  if (Math.random() > 0.5) {  
    console.log('Breaking out of the loop when q is ' + q);  
    break;  
  }  
  else {  
    console.log(q);  
  }  
}
```


Skipping Iterations: continue

We can also skip the current iteration and continue the loop at the next step in the iteration by using the word `continue`

```
for (let r = 0; r < 5; r++) {  
  if (Math.random() > 0.5) {  
    console.log('Skipping the console.log when r is ' + r);  
    continue;  
  }  
  console.log(r);  
}
```