Table of Contents

1 In the previous lesson, we looked at `if` statemants:

```
var age = prompt( "How old are you?" );
age = Number.parseInt(age);
if ( age > 18 ) {
 alert("You are over 18 years old." );
}
```

2 The `if` statement allows us to **control** when to display the alert that says “You are over 18 years old.” If the user enters an age greater than 18 `(age > 18)` we alert the user, **otherwise** no alert is shown.

3 There are other ways we can control how code flows.
There is a way to control how many times a section of code repeats.
It is called a **while Loop**.

4 Here is an example of one scenario you can use it in.

```
var num = prompt( "Please enter a number greater or equal to 5." );
num = Number.parseInt( num );
while ( number < 5 ) {
 alert( "You entered a number less than 5. Please try again." );
 num = prompt( "I said, enter a number greater or equal to 5!" );
}
alert( "Thank you!" );
```

5 Try the code above, and investigate whether it loops or not. Can you get away with entering a number less than 5? Or will it repeat forever until you put a number greater than or equal to 5?

6 Try to make your own **while loop**. Ask the user how many tickets they’d like to buy to an event, and make sure they can’t buy more than four tickets.

7 While Loops are usually used with **counter variables**. Counter variables are just regular variables, like we’ve seen before, but they are used in a special way, for a special purpose.

8 Let's look at an example.

```
var counter = 0;
while ( counter < 5 ) {
 alert( "The counter variable equals " + counter );
counter = counter + 1;
}
alert( "Done!" );
```

First we make a variable called **counter** and we set it to zero. Then, in the **while loop**, we display an alert and then we **increment** the value of **counter** by one. So we are using the counter variable as a way to keep track of how many times we want to **repeat**. It only stops after the counter variables becomes 5.

9 Try to make a **while loop** that loops 9 times.

10 **While Loops** can be used in many more ways, and we will come back to them after we talk about **Arrays**.

1 **Arrays** are another type of value, like **Strings**, **Numbers**, **Booleans**… but arrays are special because they are collections of other values. An **Array** is a list of values that can help you keep lots of pieces of information together.

2 Here is an example of some names of students, using only **Strings**.

```
var firstStudent = "Ruiwen";
var secondStudent = "Alli";
var thirdStudent = "Lillie";
var fourthStudent = "Rebecca";
var fifthStudent = "Michael";
```

If we had many students, we would have to **define** many student name variables!

3 Here's how we can do something similar but using an array.

```
var students = [ "Ruiwen", "Alli", "Lillie", "Rebecca", "Michael" ];
alert( students );
```

All the names are in one variable! They are all in one **array, with five elements**.

4 You can mix different **value types** inside an array.

```
var phoneSpecs = [ "Android", 6.2, false, true, "Galaxy" ];
alert( phoneSpecs );
```

5 Try to make your own **array** with any information. Maybe a list of movies and TV shows you like.

6 **Arrays** can even be empty!

```
var iAmEmpty = [ ];
```

There is nothing inside this array.

7 There is a way to find out how big an **array** is. We use the `.length` property of an **array** to find out how many elements are inside of it.

8 Try this, but before you do, can you guess what the answer in the **alert()** will be?

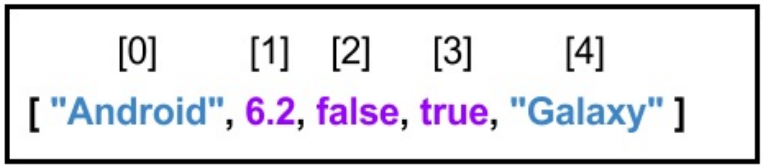
```
var animals = [ "Camel", "Bao", "Orangutan", "Rabbit",
     "Kangaroo", "Eagle", "Mouse", "Blue Jay", "Crocodile"    ];
alert( animals.length );
```

9 So now that we have an **array** of items inside it, and we know how to find out how big it is. But how do you select an individual item out?

10 With **arrays**, items are accessed by using an **index**. These are just numbers… 0, 1, 2, 3, 4,... etc. Something important to note here is that **the first index is 0**.

11 So if you want to access the first element you have to access it this way.

```
var phoneSpecs = [ "Android", 6.2, false, true, "Galaxy" ];
var item = phoneSpecs[0];
alert( item );
```



Note how the **array's length is 5**, but the **last index is 4**. The **last index** is always **one less** than the **length of the array**.

Item	Code	Value
First item	`phoneSpecs[0];`	`"Android"`
Second item	`phoneSpecs[1];`	`6.2`
Third item	`phoneSpecs[2];`	`false`
Fourth item	`phoneSpecs[3];`	`true`
Fifth item	`phoneSpecs[4];`	`"Galaxy"`

Adding items to an array

1 Let's say we create an empty **array** and we want to add a new item to it, here's how we can do that.

```
var superheroes = [ ];
alert( "There are " + superheroes.length + " heroes in the list." );
superheroes.push( "Batman" );
superheroes.push( "Catwoman" );
alert( "There are now " + superheroes.length + " heroes in the list." );
alert( superheroes );
```

2 We started with an **empty array** and we alerted the user about how big it is, then we added two new items using the **push()** function. The **push()** function takes **one input**, it is the value of the item you want to add. The final alert shows the **array** with the new items.

Removing items from an array

3 There are also ways you can remove items from an **array**. It is slightly different than the **push()** function.
It's called the **splice()** function.

4 **splice()** takes in **two inputs**. The first one is the index at which you want to start, and the second is how many you want to remove. So you can remove many items in one go.

```
var animals = [ "Camel", "Bao", "Orangutan", "Rabbit",
              "Mouse", "Blue Jay", "Crocodile"    ];
animals.splice( 3, 1 );
alert( animals );
```

What do you see in the alert()?

5 What about the following?

```
var animals = [ "Camel", "Bao", "Orangutan", "Rabbit",
                "Mouse", "Blue Jay", "Crocodile"    ];
animals.splice( 2, 3 );
alert( animals );
```

6 Try to make different arrays and remove items from them using the splice() function.

1 **Arrays** and **while loops** make a great combination. Remember when we used a **while loop** along with a counter **variable** to print out the numbers? Like this.

```
var counter = 0;
while ( counter < 5 ) {
 alert( "The counter variable equals " + counter );
 counter = counter + 1;
}
```

2 And remember when we used an **index** value to access an item in an **array**? Like this.

```
var phoneSpecs = [ "Android", 6.2, false, true, "Galaxy" ];
var item = phoneSpecs[ 0 ];
```

3 Well now let's combine these two concepts!

```
var counter = 0;
var phoneSpecs = [ "Android", 6.2, false, true, "Galaxy" ];
while ( counter < 5 ) {
 var item = phoneSpecs[ counter ];
 alert( "The counter variable equals " + counter );
 alert( "The array item is " + item );
 counter = counter + 1;
}
alert( "Done!" );
```

4 Now let's try it with a bigger array!

```
var counter = 0;
var animals = [ "Camel", "Bao", "Orangutan", "Rabbit",
      "Kangaroo", "Eagle", "Mouse", "Blue Jay", "Crocodile"    ];
while ( counter < 5 ) {
 var item = animals[ counter ];
 alert( "The counter variable equals " + counter );
 alert( "The array item is " + item );
 counter = counter + 1;
}
alert( "Done!" );
```

5 But does it still work with a bigger **array**? Does it show us all the items? If not, why not?

Let's combine many of the concepts we learned from lessons 1, 2, and 3, into one example program that ties in many things we learned.

The program will ask the user for the number of items they want to enter in a list. We will then create an empty list, and then loop once for every item we want to insert. In every loop, we will prompt the user to input a value and we will insert it in the list. At the end we will display the list to the user.

```
var numberOfItems = prompt( "How many items do you want to insert into your new list?" );
numberOfItems = Number.parseInt( numberOfItems );
var newList = [ ];
var counter = 0;
While ( counter < numberOfItems ) {
  var newItem = prompt("Enter item number: " + counter );
  newList.push( newItem );
  counter = counter + 1;
}
alert( "Your new list is [ " + newList + " ]" );
```