# DATA TYPES

```
String    :    “a string of characters”
Number    :    100
Boolean   :    true

Array     :    [1,2,3]
Object    :    {name:”John Doe”,age:23}
```

---

## Arrays [ ]

I think of arrays as a bookcase. A row of elements where each can be referenced by its position.
Example:
book 1 = Fellowship of the Ring,
book 2 = Neverending Story,
book 3 = A Clockwork Orange.

```
const bookcase = [
‘Fellowship of the Ring’,
’Neverending Story’,
’A Clockwork Orange’
]

let myArray = [];       instantiate array
                        an array always uses []

.push(x)                Adds x as the last element
.pop()                  Removes the last element

myArray.push(1)         // [1]
myArray.push(2)         // [1,2]
myArray.pop()           // [1]
```

note! .pop() returns the removed element.

We can access elements in an array by their index

```
const arr = [ ‘first’ , ‘second’ , ‘third’ ]
                 0          1           2
```

Above is an array and below we see the index of each element. Elements are separated by ,

```
arr[0]        // ‘first’
arr[1]        // ‘second’
```

We can check how many elements are in an array with the .length property.

```
arr.length    // 3
```

With the .map(x) method we can call a function x on all elements of an array.

```
function double(num){
     return num *= 2;
     };
let myArray = [1,2,3];

myArray.map(double)          // returns [2,4,6]
```

The map function does not alter the original array. We can store that in a separate variable.

```
const newArray = myArray.map(double);

// myArray = [1,2,3]
// newArray = [2,4,6]
```

note! if our array is instantiated with let we can overwrite it like so:

```
let array = [1,2,3];
array = array.map(double)
```

---

## Objects { }

I think of objects as appliances. They have properties and methods.
Example:
My coffee maker’s water tank is half full. Its brand is Nescafe and it can brew a cup of coffee.

```
const coffeeMaker = {
     water    : 0.5,
     brand    : ‘Nescafe’,
     brew     : (){
                  if(this.water > 0.25)
                  {return cupOfCoffee}
                  }
}
```

```
let myObject = {};      instantiate object
                        an object always uses {}
```

An object is different from an array. It has properties that we can reference by name.

```
const myObject = {
     name: “John”,
     age: 25
     }
myObject.name           // “John”
myObject[‘name’]        // “John”
```

We can create new properties by setting their value.

```
myObject.work = “Barista“;

myObject      // {name:”John”, age:25, work:”Barista” }
```

Objects can also have methods, which are functions on the object.

```
myObject.hello = function(){
                  console.log(‘Hello World’)
                  }
```

Notice that we use the name of the object when referencing it:
myObject

When an object refers to itself, we can use the keyword:
this

```
myObject.hello = function(){
         console.log("Hello my name is " + this.name);
         }

myObject.hello()        // "Hello my name is John"
```