Computed Object Keys

using string keys to access their corresponding objects, and behavior of the toString method in Javascript

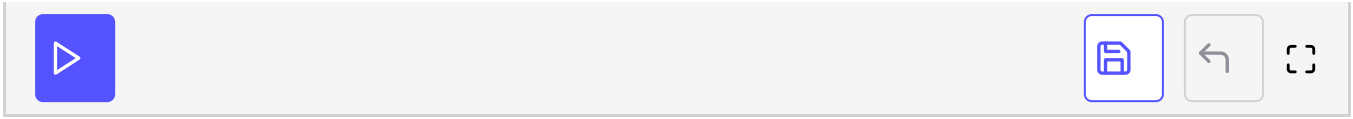
In JavaScript, objects are associative arrays (hashmaps) with String keys. We will refine this statement later with ES6 symbols, but so far, our knowledge is limited to string keys.

It is now possible to create an object property inside the object literal using the bracket notation:

```
let arr = [1,2,3,4,5];

let experimentObject = {
  [ arr.length ]: 2,
  [ arr ]: 1,
  [ {} ]: 3,
  arr

}
console.log(experimentObject)
```



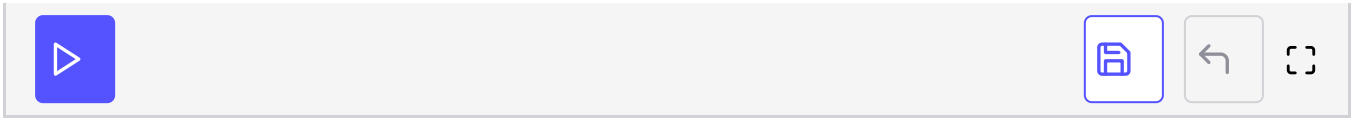
The object will be evaluated as follows:

```
{
  "5": 2,
  "1,2,3,4,5": 1,
  "[object Object]": 3,
  "arr": [1,2,3,4,5]
}
```

We can use any of the above keys to retrieve the above values from `experimentObject`. Consider the following snippet to print these values:

```
console.log(experimentObject.arr)                  // [1,2,3,4,5]
console.log(experimentObject[ 'arr' ])             // [1,2,3,4,5]
console.log(experimentObject[ arr ])               // 1
```

```
console.log(experimentObject[ arr.length ])        // 2
console.log(experimentObject[ '[object Object]' ]) // 3
console.log(experimentObject[ experimentObject ])  // 3
```



Conclusion:

- Arrays and objects are converted to their `toString` values.
- `arr.toString()` equals the concatenation of the `toString` value of each of its elements, joined by commas.
- The `toString` value of an object is `[object Object]` regardless of its contents.
- When creating or accessing a property of an object, the respective `toString` values are compared.

Now, let's talk about the various operations of objects in the next lesson.