- For loops are great for counting through a set of finite steps.

```
for (var i=0; i<10; i++){
    console.log(i);
}
```

```
0
1
2
3
4
5
6
7
8
9
```

- While loops are useful for repeating a task or set of tasks until the condition is no longer true.

```
> var entity = "Monster";
  var life =10;
  while (life !=0){
      console.log("Keep "+ entity + " on screen");
      life--;
  }
```

```
10 Keep Monster on screen
```

- While loop can also be used just like a for loop.

```
> var i = 0;
  while (i<10){
      console.log(i);
      i++;
  }

  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
```

- Do..while loop: It will execute the code at least once even if the loop fails or evaluated to false right when it starts.

```
> var entity = "Monster";
  var life =10;
  do{
      console.log("Keep " + entity + " on screen");
      life--;
  }while (life !=0);

10 Keep Monster on screen

< 1
```

```
> var entity = "Monster";
  var life =10;
  do{
      console.log("Keep " + entity + " on screen");
      life--;
  }while (life >10);
  Keep Monster on screen
< 10
```

- Arrays are just another type of variable. But a more complex and dynamic variable. Because it can store multi dimensional data, unlike a regular variable.

```
> var fruits = ["apples", "oranges", "pears"];
< undefined
```

- With the fruits array already created in the console, we can write methods by refencing the array name (fruits). The first method we will practice is the push method.

```
> fruits.push("grapes");
  4
  fruits;
< ▶ (4) ['apples', 'oranges', 'pears', 'grapes']
```

grapes is added to the array (at the end of the list)

- Pop removes the last item on the array list.

```
> fruits.pop();
< 'grapes'
> fruits;
< ▶ (3) ['apples', 'oranges', 'pears']
```

- Shift removes the first item on the array list

```
> fruits.shift();
'apples'
> fruits;
▶ (2) ['oranges', 'pears']
>
```

- Unshift adds an item to the front on the array list.

```
> fruits.unshift("apples");
3
> fruits;
▶ (3) ['apples', 'oranges', 'pears']
>
```

- Splice add or remove items on the array list. The first two integers represents: 1) position on the array items to add or remove 2) number of items to remove (optional).

```
> fruits.splice(1, 0, "grapes", "kiwi");
▶ []
> fruits;
▶ (5) ['apples', 'grapes', 'kiwi', 'oranges', 'pears']
```

- Slice remove specified number of items at the front of the array list.

```
▶ (5) ['apples', 'grapes', 'kiwi', 'oranges', 'pears']
> var favFruits = fruits.slice(2);
undefined
> favFruits;
▶ (3) ['kiwi', 'oranges', 'pears']
>
```

- Sort sorts items by alphabetically order (aà z) of the array list.

```
> fruits.sort();
▶ (5) ['apples', 'grapes', 'kiwi', 'oranges', 'pears']
>
```

- Reverse sort items in the reverse order of the array list.

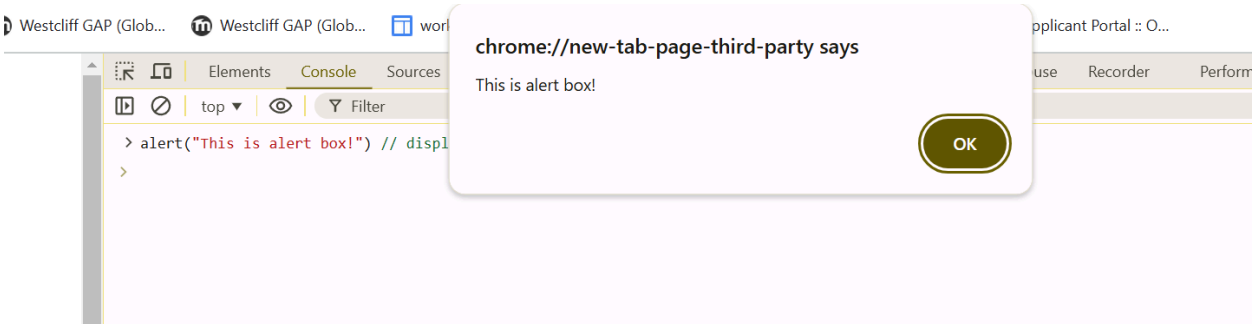
```
> fruits.reverse();
▶ (5) ['pears', 'oranges', 'kiwi', 'grapes', 'apples']
```

- Concat combines two array items into one array list. Because it will create a new list, a new array should be created to store this list.

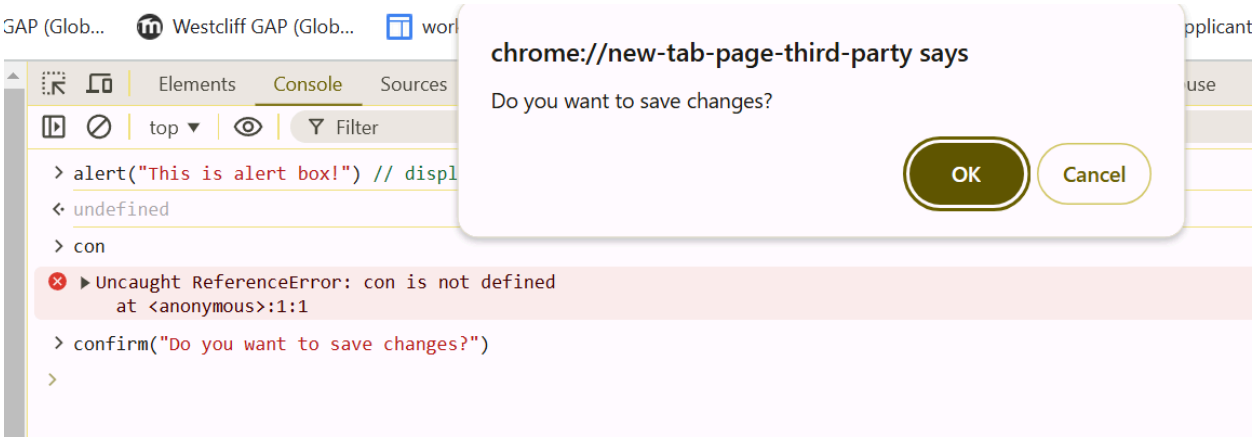
```
var beenThereList = ["New York City", "London", "Rome"];
var busketList = ["Shanghai", "Santiago"];
var myList = beenThereList.concat(busketList);
undefined
myList;
▶ (5) ['New York City', 'London', 'Rome', 'Shanghai', 'Santiago']
```

- There are three type of popups: alert, confirm and prompt.

Alert will trigger the browser window to open a small popup box that contains some message.



- Confirm will trigger the browser window to open a small popup box that contains some message that you specified and a button each to accept or to cancel.



- Prompt will trigger the browser window to open a small popup box that contains some message and an input field.