Array Destructuring

...

In ES5, if you wanted to assign elements of an array to variables a, b and c you would do this:

```
let array = [“Bonfire Night”,“Christmas”, “New Year”];
```

In ES5, you would do this to get the elements out of the array:

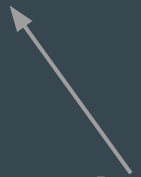
```
let a = array[0] ;
```
“Bonfire Night”

```
let b = array[1];
```
“Christmas”

```
let c = array[2];
```
“New Year”

Array destructuring simplifies this process

```
let array = [“Bonfire Night”,“Christmas”, “New Year”];
```



```
let [a,b,c] = array;
```

OR

```
let [nov5,25dec,1jan] = array;
```

Javascript maps the 1st variable to the 1st element, second variable to 2nd element, etc.

```
console.log(a,b,c) =
```
“Bonfire Night”, “Christmas”, “New Year”;

Leave a gap in the let[] to skip assigning an element to a variable

```
let array = [“Bonfire Night”,“Christmas”, “New Year”];
```

Javascript maps the 1st variable to the 1st array element, **skips the second array element** and maps the 2nd variable to the 3rd array element

Variable a becomes “Bonfire Night”

Variable b becomes “New Year” because we skipped “Christmas” (christmas is cancelled)

```
let [a, ,b] = array;
```
→ “Bonfire Night”, “New Year”;

You can destructure an array from a return value

```
function createArray(){
    return [10,20,30];

}
```

```
let [a,b,c] = createArray();
```
→ 10, 20, 30

or, if you wanted to skip an element, we can do as we previously did...

```
let [ ,b,c] = createArray();
```
→ 20, 30

Object Destructuring

...

Object destructuring allows you to assign properties from the return object as variables

```
function generateUser(username) = {

let password = generatePassword();

return {username, password};

}
```

To access your password in ES5, you would write this:

```
var myUser = generateUser("TNA_Man");

alert(myUser.password);
```

Object destructuring allows you to assign properties from the return object as variables

```
function generateUser(username) = {

let password = generatePassword();

return {username, password};

}
```

Whereas in ES6, you can write this:

```
let [username, password] = generateUser("TNA_Man");

alert(myPassword);
```

Object destructuring allows you to assign properties from the return object as variables

```
function generateUser(username) = {

let password = generatePassword();

return {username, password};

}
```

If you just want the password, all you have to do is ask for it on it's own:

```
let [password] = generateUser("TNA_Man");

alert(myPassword);
```

Object destructuring allows you to assign properties from the return object as variables

```
function generateUser(username) = {

let password = generatePassword();

return {username, password};

}
```

Note: the let[variable] **must** have the same name as one of the return object's variables.

Therefore, in this instance we can only type:

`let[username]`, `let[password]` or `let[username,password]`

because generateUser() returns username and password variables.