JavaScript 1 - Module 2

Strings and Logic

Arrays

Objects

Functions

Solutions for JS1 Lesson 2.1 Strings and Logic

Exercise 1

a) Convert the string "Down the freeway rolls a wayward beer can" to UPPER CASE.

b) Convert the string "YOU SHOULDN'T HAVE TO SELL YOUR SOUL" to lower case.

Console log out the results.

```
// Exercise 1
console.log("Exercise 1");

// a
let str1 = "Down the freeway rolls a wayward beer can";
let upper = str1.toUpperCase();
console.log(upper);

//
let str2 = "YOU SHOULDN'T HAVE TO SELL YOUR SOUL";
let lower = str2.toLowerCase();
console.log(lower);
```

Exercise 2

Given the string "Blåbærsyltetøy":

a) Extract the letters from start position 3 to potition 5 (including the 5), into a new variable.

b) Extract the last 8 letters, into a new variable.

c) Extract the first three letters, into a new variable.

Console log out the results.

```
// Exercise 2
console.log("Exercise 2");

const norwegianString = "Blåbærsyltetøy";

// a
let sliced1 = norwegianString.slice(3, 6);
console.log(sliced1); // bær

// b
let sliced2 = norwegianString.slice(-8);
console.log(sliced2); // syltetøy

// c
let sliced3 = norwegianString.slice(0, 3);
console.log(sliced3); // Blå
```

Exercise 3

Given the string "Blåbærsyltetøy":

a) Extract the letter at position 2 into a new variable.

b) What is the unicode number for 'æ', 'ø', and 'å', respectivly?

Console log out the results.

```
// Exercise 3
console.log("Exercise 3");

// a
let myChar = norwegianString[2];
console.log(myChar); // å

// b
let unicode1 = norwegianString.charCodeAt(4);
let unicode2 = norwegianString.charCodeAt(12);
let unicode3 = norwegianString.charCodeAt(2);

console.log (unicode1, unicode2, unicode3); // 230 248 229

// PS! Might also have used:
console.log ("æ".charCodeAt(0), "ø".charCodeAt(0), "å".charCodeAt(0));
```

Exercise 4

Given the strings "Shout, shout!" and "Let it all out.", merge the two with a space between them, *without* using the string concatinator operator (`+`):

a) Using a string method.

b) Using a template literal.

Console log out the results.

```javascript
// Exercise 4
console.log("Exercise 4");

let shout1 = "Shout, shout!";
let shout2 = "Let it all out.";

// a
let concated = shout1.concat(" ", shout2);
console.log(concated);

// b
let stringified = `${shout1} ${shout2}`;
console.log(stringified);
```

Exercise 5

Given the string (template):

```
`London calling to the faraway towns
Now war is declared and battle come down
London calling to the underworld
Come out of the cupboard, you boys and girls
London calling now don't look to us
Phony Beatlemania has bitten the dust
London calling see we ain't got no swing
'Cept for the ring of that truncheon thing.`
```

a) What is the index of the first occurance of "London calling"?

a) What is the index of the last occurance of "London calling"?

```
// Exercise 5

console.log("Exercise 5");

const lyrics = `London calling to the faraway towns
Now war is declared and battle come down
London calling to the underworld
Come out of the cupboard, you boys and girls
London calling now don't look to us
Phony Beatlemania has bitten the dust
London calling see we ain't got no swing
'Cept for the ring of that truncheon thing.`;

const searchFor = "London calling";
let first = lyrics.indexOf(searchFor);
let last = lyrics.lastIndexOf(searchFor);

console.log("First occurance at index " + first); // 0

console.log("Last occurance at index " + last); // 229
```

Exercise 6

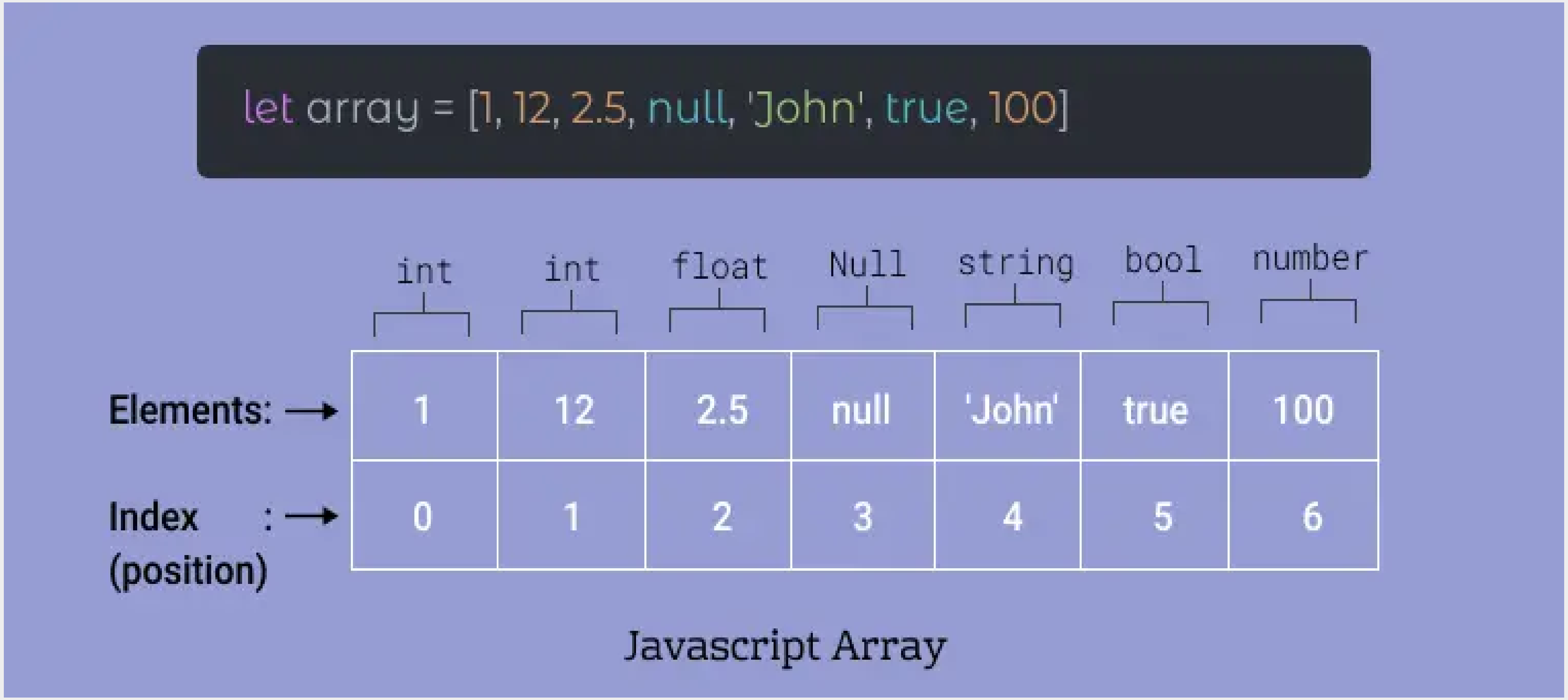
Given the following code, what is logged out (try first without coding it):

```
let sum = 35;
if (sum % 10 == 0){
    console.log("a");
} else if (sum % 2 == 1){
    if (sum % 5 == 0 && sum % 2 == 0){
        console.log("b");
    } else if (sum % 5 == 0){
        console.log("c")
    } else {
        console.log("d")
    }
} else {
    console.log("e")
}
```

Level 2: Is it possible to get output 'b'? Explain.

```
// Exercise 6

console.log("Exercise 6");

let sum = 35;
if (sum % 10 == 0){
    console.log("a");
} else if (sum % 2 == 1){
    if (sum % 5 == 0 && sum % 2 == 0){
        console.log("b");
    } else if (sum % 5 == 0){
        console.log("c");
    } else {
        console.log("d");
    }
} else {
    console.log("e");
}

// sum = 35 gives output 'c'

// Level 2:
// (sum % 2 == 1) and (sum % 2 == 0) cannot both be true,
// thus we cannot reach console.log("b");
```

Arrays



```
let array = [1, 12, 2.5, null, 'John', true, 100]
```

Elements: →	1	12	2.5	null	'John'	true	100
Index (position) : →	0	1	2	3	4	5	6

Javascript Array

Source

JavaScript Arrays

JavaScript arrays are used to store multiple values in a single variable.

Arrays are generally described as "**list-like objects**"; they are basically single objects that contain multiple values stored in a list.

Array objects can be stored in variables and dealt with in much the same way as any other type of value, the difference being that we can access each value inside the list individually, and do super useful and efficient things with the list, like loop through it and do the same thing to every value.

Creating an Array

Using an *array literal* is the easiest way to create a JavaScript Array.

```
var array_name = [item1, item2, ...];
```

Arrays consist of square brackets, `[]`, and elements that are separated by commas, `,`.

Creating an Array, cont.

Suppose we want to store a shopping list in an array:

```
let shopping = ['bread', 'milk', 'cheese', 'hummus', 'noodles'];

console.log(shopping);
```

> Spaces and line breaks are not important. A declaration can span multiple lines:

```
let shopping = [
  'bread',
  'milk',
  'cheese',
  'hummus',
  'noodles'
];
console.log(shopping);
```

Creating an Array, cont.

In the above example, each element is a string, but in an array we can store various data types — strings, numbers, objects, and even other arrays.

We can also mix data types in a single array — we do not have to limit ourselves to storing only numbers in one array, and in another only strings. For example:

```
let sequence = [1, 1, 2, 3, 5, 8, 13];

let random = ['tree', 795, [0, 1, 2]];
```

Finding the `length` of an array

You can find out the `length` of an array (how many items are in it) in exactly the same way as you find out the `length` (in characters) of a string — by using the `length` property.

```
let shopping = ['bread', 'milk', 'cheese', 'hummus', 'noodles'];

console.log(shopping.length); // should return 5
```

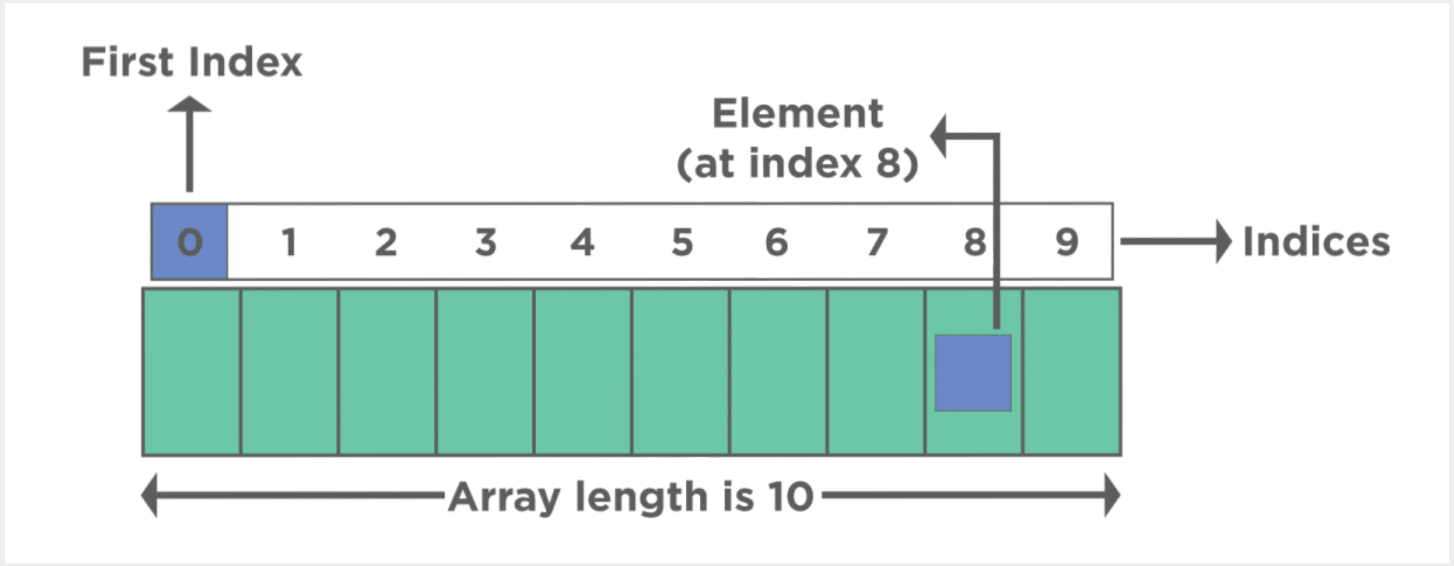


Illustration Source

Finding the `length` of an array, cont.

`Array.length` has other uses, but it is most commonly used to tell a loop to keep going until it has looped through all the items in an array.

```
let sequence = [1, 1, 2, 3, 5, 8, 13];
for (let i = 0; i < sequence.length; i++) {
  console.log(i + ": " + sequence[i]);
}
```

- Start looping at item number 0 in the array.
- Stop looping at the item number equal to the length of the array. (*An array with 7 items, only had data in position 0-6, looking for position 7 will be out of bounds.*)
- For each item, log it to the browser console.

Accessing and modifying array items

You can then **access** individual items in the array using bracket notation, in the same way that you accessed the letters in a string (property access).

```
let shopping = ['bread', 'milk', 'cheese', 'hummus', 'noodles'];

console.log(shopping[0]); // "bread"
console.log(shopping[2]); // "cheese"
console.log(shopping[5]); // undefined
```

> Note: Array indexes start with 0.
>
> arrayname[0] is the first element, arrayname[1] is the second element, etc.
>
> Last element in a normal array is arrayname[length-1]

Accessing and modifying array items, cont.

You can also **modify** an item in an array by giving a single array item a new value.

```
let shopping = ['bread', 'milk', 'cheese', 'hummus', 'noodles'];

shopping[0] = 'rolls';
console.log(shopping);
// shopping will now return:
// Array(5) [ "rolls", "milk", "cheese", "hummus", "noodles" ]
```

Now try this:

```
shopping[3] = 'guacamole';
shopping[5] = "pasta";

console.log(shopping);
// shopping will now return:
// Array(6) [ "bread", "milk", "cheese", "hummus", "noodles", "pasta" ]
```

Converting between strings and arrays

Often you'll be presented with some raw data contained in a big long string, and you might want to separate the useful items out into a more useful form and then do things to them, like display them in a data table. To do this, we can use the `split()` method.

In its simplest form, this takes a single parameter, the character you want to separate the string at, and returns the substrings between the separator as items in an array.

> Note: Technically `split()` a string method, not an array method, but we've put it in with arrays as it goes well here.

Converting between strings and arrays, cont.

Using `split()` to convert a break up string, and put the elements into an array:

```
let myData = 'Manchester,London,Liverpool,Birmingham,Leeds,Carlisle';
let myArray = myData.split(','); // split string it at each comma

console.log(myArray);
// Array(6) [ "Manchester", "London", "Liverpool", "Birmingham", "Leeds", "Carlisle" ]

console.log(myArray.length;);
// 6

console.log(myArray[0]);
// the first item in the array: "Manchester"

console.log(myArray[1]);
// the second item in the array: "London"

console.log(myArray[myArray.length-1]);
// the last item in the array: "Carlisle"
```

Converting between strings and arrays, cont.

You can also go the opposite way, and convert an Array (back) to a string using the `join()` method.

Continue the example from above:

```
let myNewString = myArray.join(',');

console.log(myNewString);
// "Manchester,London,Liverpool,Birmingham,Leeds,Carlisle"
```

And try another seperator:

```
let myNewString2 = myArray.join(' - ');

console.log(myNewString2);
// "Manchester - London - Liverpool - Birmingham - Leeds - Carlisle"
```

Converting between strings and arrays, cont.

Another way of converting an array to a string is to use the `toString()` method.

`toString()` is arguably simpler than `join()` as it doesn't take a parameter, but more limiting.

With `join()` you can specify different separators (as shown above), whereas `toString()` always uses a comma:

```
let dogNames = ['Rocket','Flash','Bella','Slugger'];

let str = dogNames.toString();

console.log(str); // Rocket,Flash,Bella,Slugger
```

Adding and removing array items

To add or remove an item at the end of an array we can use `push()` and `pop()` respectively.

Starting a new JS file with this Array:

```
let myArray = ['Manchester', 'London', 'Liverpool',
'Birmingham', 'Leeds', 'Carlisle'];
```

The `push()` method adds one or more new elements to an array (at the end):

```
myArray.push('Cardiff');
console.log(myArray); // Array(7) [ "Manchester", ..., "Cardiff" ]
```

```
myArray.push('Bradford', 'Brighton');
console.log(myArray); // Array(9) [ "Manchester", ..., "Bradford", "Brighton" ]
```

Adding and removing array items, cont.

The new length of the array is returned when the `push()` method call completes. If you wanted to store the new array length in a variable, you could do something like this:

```
let newLength = myArray.push('Bristol');

console.log(myArray); // Array(10) [ "Manchester", ..., "Bristol" ]
console.log(newLength); // 10
```

Adding and removing array items, cont.

The `pop()` method removes the last element from an array:

```
myArray.pop();

console.log(myArray); // "Bristol" should be gone
```

The item that was removed is returned when the method call completes. To save that item in a new variable, you could do this:

```
let removedItem = myArray.pop();

console.log(myArray); // "Brighton" should be gone

console.log(removedItem); // But not forgotten
```

Adding and removing array items, cont.

`unshift()` and `shift()` work in exactly the same way as `push()` and `pop()`, respectively, except that they work on the beginning of the array, not the end.

The `shift()` method removes the first array element and "shifts" all other elements to a lower index:

```
removedItem = myArray.shift();

console.log(myArray); // "Manchester" should be gone
console.log(removedItem); // And put into removedItem
console.log(myArray[0]); // "London"
```

Adding and removing array items, cont.

The `unshift()` method adds a new element to an array (at the beginning), and "unshifts" older elements:

```
newLenght = myArray.unshift('Edinburgh');

console.log(myArray);
console.log(newLenght); // 8
console.log(myArray[0]); // "Edinburgh"
console.log(myArray[1]); // "London"
console.log(myArray[newLenght - 1]); // "Bradford"
```

Splicing an Array

The `splice()` method can be used to add new items to an array (at any position):

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.splice(2, 0, "Lemon", "Kiwi");

console.log(fruits); // Banana,Orange,Lemon,Kiwi,Apple,Mango
```

- The first parameter (2) defines the position where new elements should be added (spliced in).
- The second parameter (0) defines how many elements should be removed.
- The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be added.

Splicing an Array, cont.

The `splice()` method returns an array with the deleted items (as defined by the second parameter):

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

var removed = fruits.splice(2, 2, "Lemon", "Kiwi");

console.log(fruits); // Banana,Orange,Lemon,Kiwi
console.log(removed); // Apple,Mango
```

Merging (Concatenating) Arrays

The `concat()` method creates a new array by merging (concatenating) existing arrays:

```
var myGirls = ["Cecilie", "Lone"];
var myBoys = ["Emil", "Tobias", "Linus"];
var myChildren = myGirls.concat(myBoys);
// Concatenates (joins) myGirls and myBoys, in the new array myChildren

console.log(myChildren); // Cecilie,Lone,Emil,Tobias,Linus
```

> NOTE: The concat() method does not change the existing arrays. It always returns a new array.

Merging (Concatenating) Arrays, cont.

The concat() method can take any number of array arguments:

```
var arr1 = ["Cecilie", "Lone"];
var arr2 = ["Emil", "Tobias", "Linus"];
var arr3 = ["Robin", "Morgan"];
var myChildren = arr1.concat(arr2, arr3);
// Concatenates arr1 with arr2 and arr3

console.log(myChildren);
// Cecilie,Lone,Emil,Tobias,Linus,Robin,Morgan
```

The concat() method can also take strings as arguments:

```
var arr1 = ["Emil", "Tobias", "Linus"];
var myChildren = arr1.concat("Peter");

console.log(myChildren); // Emil,Tobias,Linus,Peter
```

Slicing an Array

The `slice()` method slices out a piece of an array into a new array.

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(1);

console.log (fruits); // All fruits are still there
console.log (citrus); // Orange,Lemon,Apple,Mango
```

This example slices out a part of an array **starting from** array element 1 ("Orange")

> NOTE: The `slice()` method creates a new array. It does not remove any elements from the source array.

Slicing an Array, cont.

The `slice()` method can take two arguments like `slice(1, 3)`.

The method then selects elements from the start argument, and up to (*but not including*) the end argument.

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(1, 3);

console.log (fruits); // All fruits are still there
console.log (citrus); // Orange,Lemon
```

> TIP: Using `slice()` without any arguments is the best way to copy an entire array:

```
var citrus = fruits.slice();
```

Back to `switch` example from Lesson 1.3

```
var today = new Date().getDay(); // Returns 0 (Sunday) - 6 (Saturday)
switch (today) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
     day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
}
console.log ("Today is " + day);
```

In stead of making a switch to "translate" the date object's week day numbering (`getDay()` returns 0 for Sunday, 1 for Monday, 2 for Tuesday, ..., to 6 for Saturday), we can use a (constant) array to hold the days:

```
const days = [
  "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
];

var today = new Date().getDay(); // getDay() returns 0-6

console.log ("Today is " + days[today]);
```

Sources and Resources

JavaScript First Steps > Arrays

JavaScript reference > Standard built-in objects > Array

JavaScript Arrays

JavaScript Array Methods

Todos

Github Classroom

JS1 Lesson 2.2 Arrays

Mollify

Read Arrays, and to the Lesson Task.