Breaking Javascript

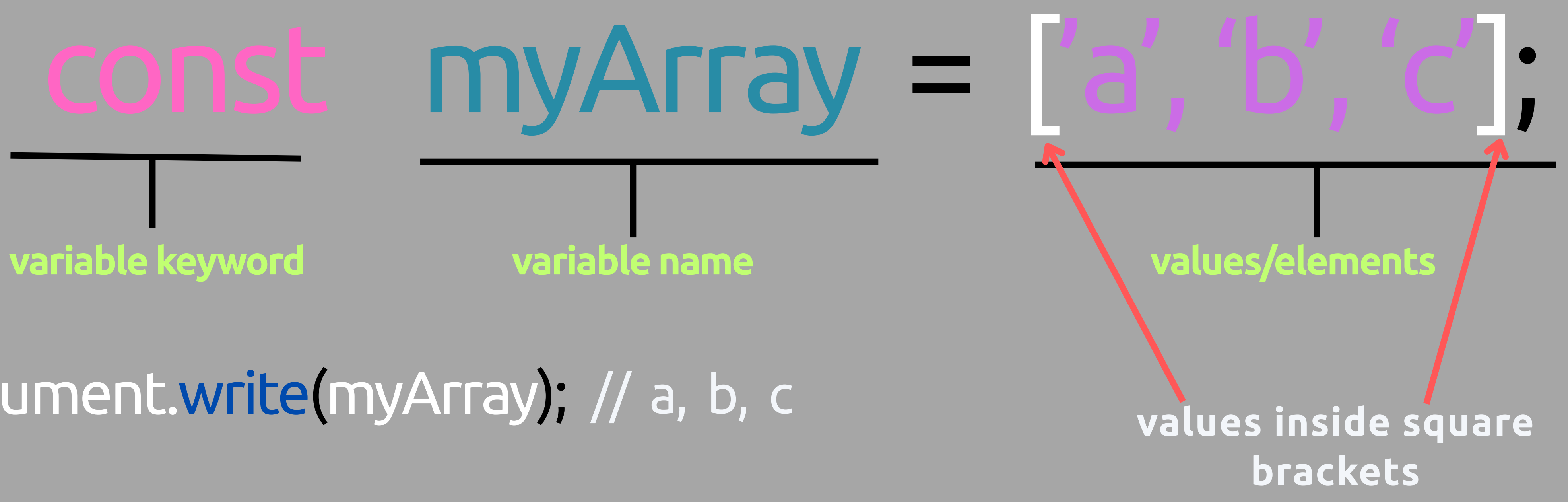
Arrays , Operators

by Shaun Kriel

Arrays :

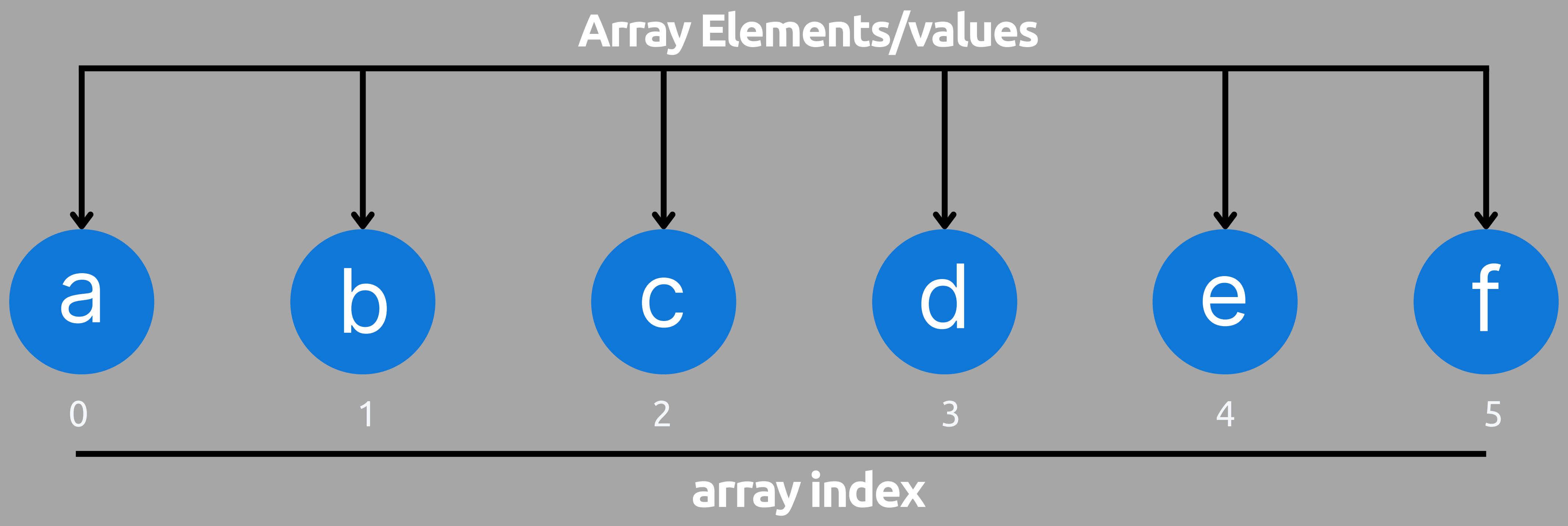
- special type of variable that can hold more than one value.
- used when working with lists or set of values that relate to each other.
- can store various data types - strings, numbers, objects and even other arrays.

syntax :



```
const myArray = ['a', 'b', 'c'];
```

```
document.write(myArray); // a, b, c
```



Accessing values in an array by using the array index above

```
document.write( myArray[0] ); // a
document.write( myArray[1] ); // b
document.write( myArray[2] ); // c
```

Array Methods:

```
const  myArray = ['a', 'b', 'c'];
```

length = lenght/ size of an array

```
document.write( myArray .length); // 3
```

toString( ) = converts array to string of array values.

```
document.write( myArray .toString( ) ); // a, b, c
```

pop( ) = removes the last element from an array.

```
document.write( myArray .pop( ) ); // c
```

push( ) = adds new element to end of an array.

```
document.write( myArray .push(  d ) );      // ['a', 'b', 'c', 'd']
```

shift( ) = removes first array element & "shift" all other elements to a lower index.

```
document.write( myArray .shift(  ) );// ['b', 'c', 'd']
```

unshift( ) = adds new element to array(beginning) & 'unshifts' older elements.

```
document.write( myArray .unshift( f ) );// ['f','b', 'c', 'd']
```

reverse( ) = reverse an array.

```
document.write( myArray .unshift( f ) );// ['d', 'c', 'b', 'f']
```

indexOf( ) = return index of first match.

```
document.write( myArray .indexOf( 2 ) ); // b
```

Array Methods cont. :

slice( ) = returns selected elements in an array, as a new array.

```
document.write( myArray .slice( 1, 3 ) );// ['c', 'b']
document.write( myArray .slice( 0, 3 ) );// ['d', 'c', 'b']
```

splice( ) = similar to slice but this takes **OUT** a selected element.

```
document.write( myArray .slice( 0, 3 ) );// [' f '] removed d, c, b
```

toSpliced( ) = new method creates a **new** array, keeping original **unachanged**.

```
e.g.    let fruits = ['apple', 'pear', 'kiwi', 'grapes'];
document.write( fruits.toSpliced(0 , 2) ); // ['kiwi', 'grapes']
```

concat( ) = creates a new array by merging (concatenating) existing arrays. Joining arrays end-to-end.

```
e.g.    let fruits = ['apple', 'pear', 'kiwi', 'grapes'];
        let veg = ['carrot', 'peppers', 'peas', 'butternut'];
        let freshProduce = fruits.concat(veg);
document.write( freshProduce);
// ['apple', 'pear', 'kiwi', 'grapes', 'carrot', 'peppers', 'peas', 'butternut']
```

Changing values in an array

```
document.write( myArray[2]  = 'd');// d
document.write( myArray );// ['a', 'b', 'd']
```

Operators:

** allow programmers to create a single value from one or more values

Assignment operator:

assign a value to a variable

```
color = ' red ';
```

Arithmetic operator:

perform basic math

sum = 4 * 4;	multiplication - multiplies two values
sum = 4 + 4;	addition - adds two values
sum = 4 - 4;	subtraction - subtracts two values
sum = 4 / 4;	division - divides two values
sum = 4 ++ 4;	increment - adds one to current number
sum = 4 - - 4;	decrement - subtracts one from current number
sum = 4 % 4;	modulus - divides two values & returns the remainder

Order of execution:

Mutliplication and division are performed ***before*** addition or subtraction

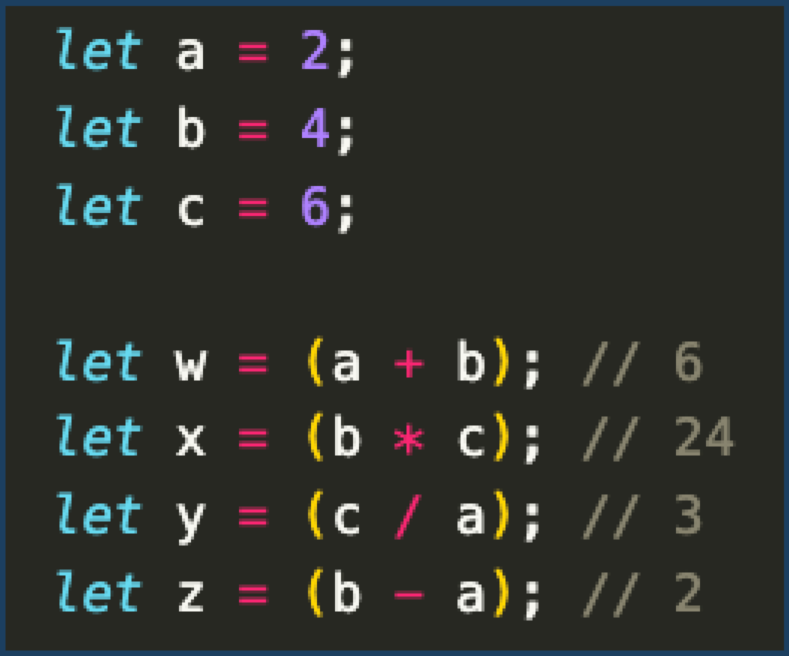
Numbers are calculated left to right: `total = 8 + 2 + 5;   // 15`

But the following is different: `total = 8 + 2 * 5;   // 18 not 50`

** because * and / happen before + and - **

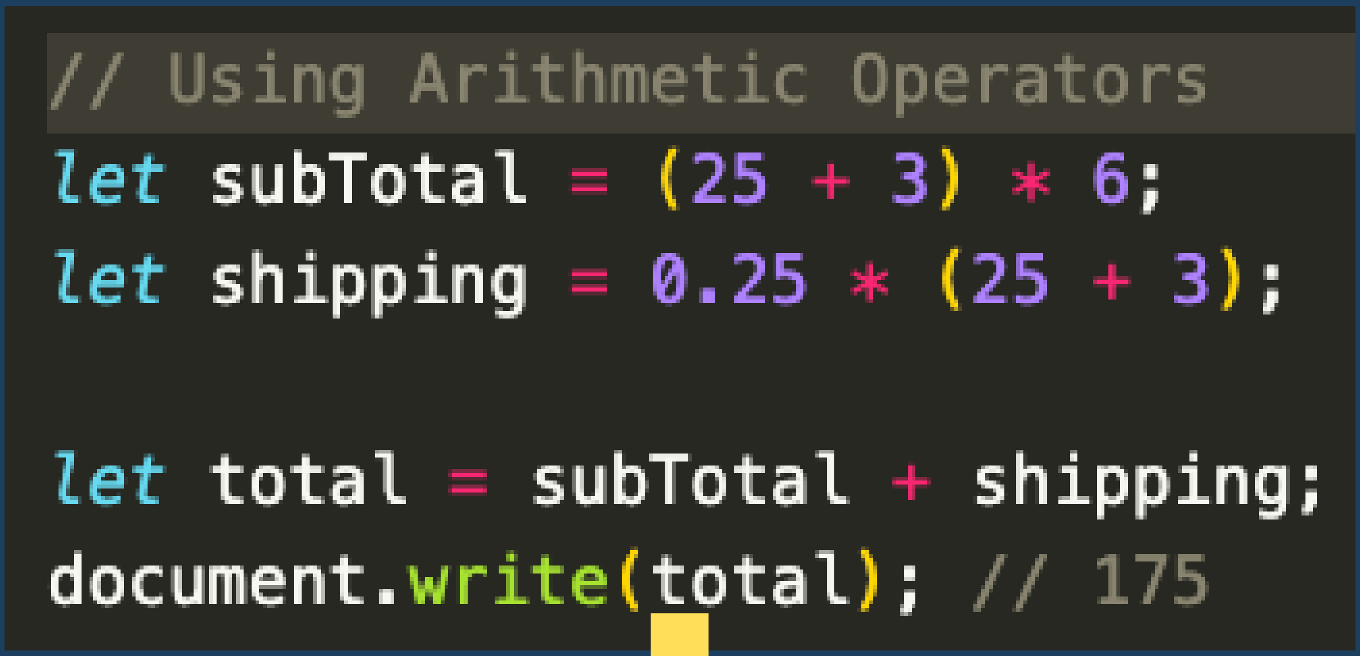
You can do the following: `total = (8 + 2) * 5;   // 50`

** using parentheses means thats what will be calculated first **



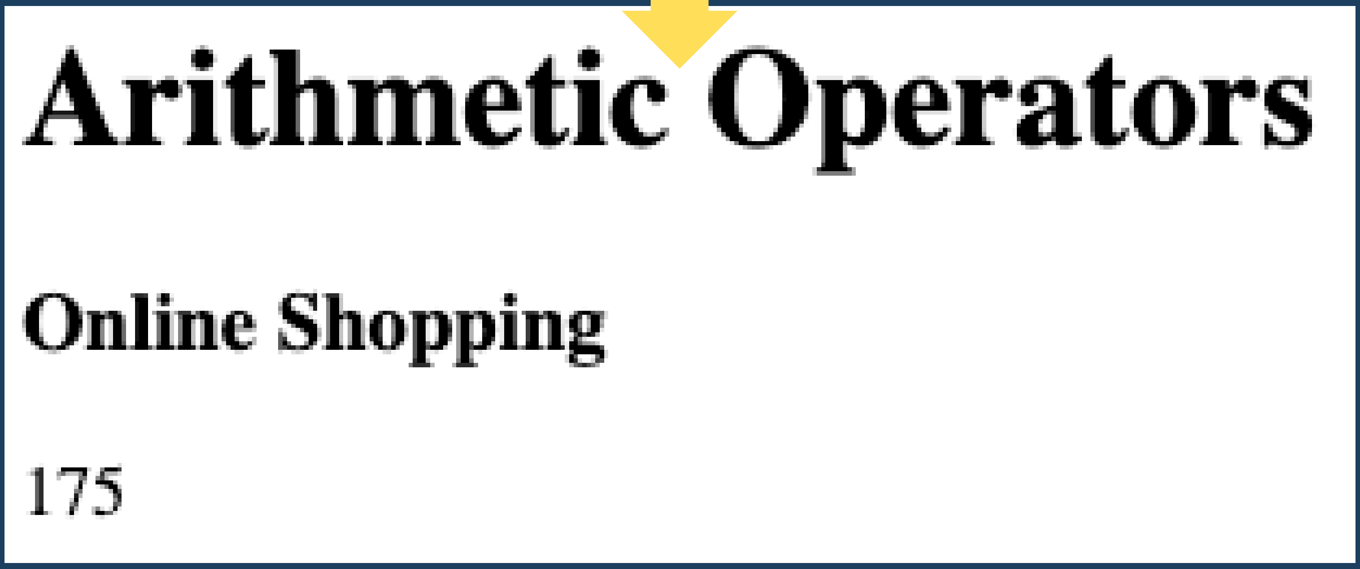
```
let a = 2;
let b = 4;
let c = 6;

let w = (a + b); // 6
let x = (b * c); // 24
let y = (c / a); // 3
let z = (b - a); // 2
```



```
// Using Arithmetic Operators
let subTotal = (25 + 3) * 6;
let shipping = 0.25 * (25 + 3);

let total = subTotal + shipping;
document.write(total); // 175
```



Arithmetic Operators

Online Shopping

175

String operator:

combine two strings

greeting = ‘ Hi ‘ + ‘ Bob ’; // Hi Bob

Comparison operators:

comapares two vlaues and returns true or false

number = 2 > 6 ; // false

<	less than
>	greater than
<=	less than equal to
>=	greater than equal to
==	equal to
!=	not equal to
!==	not equal value not equal type
===	equal to(incl type)
?	ternary operator

Logical operators:

combines expressions & return true or false

number = (5 > 3) && (2 < 4) ;

// true

&&	and
||	or
!	not