Javascript in web dev use basics:

Adding inline JS to HTML document:

```
<script>
    var date = new Date();
    document.body.innerHTML = “<h1>Today is: “
+ date + “</h1>”
</script>
```

Linking external JS file in HTML:

```
<script src=”script.js” ></script>
```

Add defer in the script tag to make sure the script is executed only after verything else is loaded.

How to write JavaScript?

- JavaScipt is Case sensitive.
- Use camelCase.
- Whitespace matters only to humans.
- End each statement with a semicolon.
- Use comments liberally (// or /*...*/)

Variables:

```
var a;
var A = 5;
var currentTime;
var unit_4;
var $container;
```

To avoid global scope always declare your variables.

Data types in JS:
Numeric, undefined, String, Boolean, null, Symbol.

If-else example:

```
if ( a > b) {
    Do something.
} else {
    Do something else.
}
```

Ternary operator:

```
a == b ? console.log(“Match”) : console.log(“No match”)
```

//General ternary syntax: **condition ? True : false**

Array syntax:

```
var example_array = [“1”, “abc”, 2, 7];
```

Property : Meta info about the object.
Method : Function that belongs to the object.

```
var pens;
pens = ["red", "blue", "green", "orange"];

console.log("Before: ", pens);

// PROPERTIES:
// Get a property of an object by name:
// console.log("Array length: ", pens.length);

// METHODS:
// Reverse the array:
// pens.reverse();

// Remove the first value of the array:
// pens.shift();

// Add comma-separated list of values to the front of the array:
// pens.unshift("purple", "black");

// Remove the last value of the array:
// pens.pop();

// Add comma-separated list of values to the end of the array:
// pens.push("pink", "prussian blue");
```

// Find the specified position (pos) and remove n number of items from the array. Arguments: pens.splice(pos,n):
// pens.splice(pos, n) // Starts at the seccond item and removes two items.

// console.log("After: ", pens);

// Create a copy of an array. Typically assigned to a new variable:
// var newPens = pens.slice();
// console.log("New pens: ", newPens);

// Return the first element that matches the search parameter after the specified index position. Defaults to index position 0. Arguments: pens.indexOf(search, index):
// var result = pens.indexOf(search, index);
// console.log("The search result index is: ", result);

// Return the items in an array as a comma separated string. The separator argument can be used to change the comma to something else. Arguments: pens.join(separator):
// var arrayString = pens.join(separator);
// console.log("String from array: ", arrayString);

Operator Types

Unary

A *unary* operator requires a single operand, either before or after the operator, following this format:

```
operand operator
operator operand
```

For example, in the expression `a++`, `++` is a unary operator.

Binary

A *binary* operator requires two operands, one before the operator and one after the operator, following this format:

```
operand1 operator operand2
```

For example, in the expression `a + b = c`, `+` is a binary operator.

Ternary

There is one *ternary* operator, the conditional operator. For example, in the expression `a ? b : c`, the use of `?` and `:` in this manner constitutes the ternary operator.

Arithmetic Operators

An arithmetic operator takes numeric values (either literals or variables) as its operands and returns a single numeric value. The standard arithmetic operators are addition (`+`), subtraction (`-`), multiplication (`*`), and division (`/`). Other arithmetic operators are remainder (`%`), unary negation (`-`), unary plus (`+`), increment (`++`), decrement (`--`), and exponentiation (`**`).

1. Addition (+)

We use this operator in the form `operand1 + operand_2`. For example:

```
2 + 3 // evaluates to 5
4 + 10 // evaluates to 14
```

2. Subtraction (-)

We use this operator in the form `operand1 - operand2`. For example:

```
3 - 2 // evaluates to 1
4 - 10 // evaluates to -6
```

3. Multiplication (*)

We use this operator in the form `operand1 * operand2`. For example:

```
3 * 2 // evaluates to 6
```

```
4 * 10 // evaluates to 40
```

4. Division (/)

We use this operator in the form `operand1 / operand2`. For example:

```
6 / 3 // evaluates to 2
3 / 2 // evaluates to 1.5
4 / 10 // evaluates to 0.4
```

5. Remainder (%)

We use this operator in the form `operand1 % operand2`. For example:

```
6 % 3 // evaluates to 0
3 % 2 // evaluates to 1
4 % 10 // evaluates to 4
```

6. Exponentiation (**)

We use this operator in the form `operand1 ** operand2`. This operator is a part of ECMAScript2016 feature set. For example:

```
2 ** 3 // evaluates to 8
3 ** 2 // evaluates to 9
5 ** 4 // evaluates to 625
```

7. Unary Negation (-)

We use this operator in the form `-operand`. For example:

```
-4 // evaluates to -4
-(-5) // evaluates to 5 (not --5)
```

8. Unary Plus (+)

We use this operator in the form `+operand`. For example:

```
+4 // evaluates to 4
+(-4) // evaluates to -4
```

9. Increment (++)

We use this operator in the prefix and postfix forms, forms `++operand` and `operand++`. The prefix form, `++operand`, increments the operand by 1

and then returns the value of the operand. The postfix form, `operand++`, returns the value of the operand and *then* increments the operand's value by 1.

10. Decrement (- -)

We use this operator in the prefix and postfix forms, forms --operand and operand--. The prefix form, --operand, decrements the operand by 1.

and then returns the value of the operand. The postfix form, operand--, returns the value of the operand and then decrements the operand's value by 1.