General Assembly

STUDY GUIDE

CONDITIONALS

Operator Meaning

Symbol	Meaning	Example
===	Equality	(2 * 5) === 10
!==	Inequality	'10' !== 10
>	Greater than	20 > 10
>=	Greater than or equal to	10 >= 10
<	Less than	10 < 30
<=	Less than or equal to	10 <= 10

Assigning Variables

Remember:

» When a variable is created but is not assigned a value, it will be evaluated as undefined.

» We can assign a variable the value null as a way to "reset" the value of a variable to "nothing."

Logical Operators

» **AND (&&):** Return the first falsey value; if both values are truthy, return the last truthy value. AND is nicknamed the "guard operator"

» **OR (||):** Return the first truthy value; if both values are falsey, return the last falsey value. OR is nicknamed the "default operator"

» **NOT(!):** If the value is truthy, return false; if the value is falsey, return true.

Remember: We can use parentheses to change the order of operations for logical operators, just like we do in mathematics.

AND Operator in Action

Condition 1	Condition 2	Result
true	true	true
true	false	false
false	true	false
false	false	false

OR Operator in Action

Condition 1	Condition 2	Result
true	true	true
true	false	true
false	true	true
false	false	false

NOT Operator in Action

Condition	Result
true	false
false	true

Conditionals

If... Else Statements

```
if (condition1) {
// Code to be executed if condition1 is true
} else if (condition2) {
// Code to be executed if condition1 is false and condition2 is true
} else if (condition3) {
// Code to be executed if condition1 and condition2 are false and condition3 is true
} else {
// Code to be executed if condition1, condition2, and condition3 are false
}
```

With "else if", the last statement, each additional condition will only be checked if all of the prior conditions have failed.

Switch Statements

```
switch (expression) {
case value1:
// Code to be executed if expression === value1
break;
case value2:
// Code to be executed if expression === value2
break;
default:
// Code to be executed if expression is different from both value1 and value2
}
```

Ternary Statement

A ternary statement is a one-line shorthand for an if...else statement. The syntax for a ternary statement looks like this:

```
condition ? result1 : result2;
```

Example:

```
let temperature = 55;
const typeOfExercise = temperature >= 45 ? "Go for a run outside." : "Go to the gym.";
typeOfExercise;
// => "Go for a run outside."
```

Arrays

What's an array?

- An array is an ordered list of values; these values can be strings, booleans, numbers... even other arrays.
- The values within an array, called elements, are accessed by their position (via a value called an index) within the array.
- An array can be defined by enclosing a list of values within square braces, like so: let myArray = ['a','b','c','d'].
- To retrieve the value at some index i from an array, add [i] to the end of the array (e.g. myArray[2]).
- To edit the value at some index i, simply act as if you were assigning a variable (e.g. myArray[1]= 'f').

In addition to storing a set of values, arrays also have a number of in-built properties and functions that they can use.

- .length gives you the length of the array you call it on.

» .push() adds a new element to the end of an array, and returns that element.

» .pop() removes the last element in an array, and returns that element.

Loops

Loops are used to tell our programs to take repeated action. A loop's condition is re-evaluated each time the code block finishes running.

While Loops

while loops can run indefinitely, so long as the condition remains true.

For Loops

A **for loop** will generally run a fixed number of times, not indefinitely. The syntax of a for loop is:

The three parameters for a for loop are:
1. An initialization
2. A condition
3. A final expression

for loops are an easy way to iterate through an array. The following will execute an arbitrary function someFunction for every element in array myArray, from left to right. To change the way that you iterate through the array, just change the settings of your for loop.

```
for (let i = 0; i < myArray.length; i += 1) {
someFunction(myArray[i]);
}
```

FizzBuzz Solution

In the FizzBuzz exercise for the conditionals lesson, we wrote code that took an input, x, and set a new result value according to a specific set of rules.

» If x is evenly divisible by 3 and 5 (for example, 15 or 30), set result to "fizzbuzz".

```
if (x % 3 === 0 && x % 5 === 0) {
result = "fizzbuzz";
}
```

» Otherwise, if x is evenly divisible by 3 (for example, 3, 6, or 9), set result to "fizz".

```
if (x % 3 === 0 && x % 5 === 0) {
result = "fizzbuzz";
} else if (x % 3 === 0) {
result = "fizz";
}
```

» Otherwise, if x is evenly divisible by 5 (for example, 5 or 10), set result to "buzz".

```
if (x % 3 === 0 && x % 5 === 0) {
result = "fizzbuzz";
} else if (x % 3 === 0) {
result = "fizz";
} else if (x % 5 === 0) {
result = "buzz";
}
```

» If x is not evenly divisible by either 3 or 5 (for example, 7), set result to x.

```
if (x % 3 === 0 && x % 5 === 0) {
result = "fizzbuzz";
} else if (x % 3 === 0) {
result = "fizz";
} else if (x % 5 === 0) {
```

```
result = "buzz";
} else {
result = x;
}
```

» To test your code, set a value for x and run the program. Did you get the result you expected? Try out several different values for x to be sure.

```
let x = 5; // Test using different values for x
if (x % 3 === 0 && x % 5 === 0) {
result = "fizzbuzz";
} else if (x % 3 === 0) {
result = "fizz";
} else if (x % 5 === 0) {
result = "buzz";
} else {
result = x;
}
```

Ready for More?

This time, your challenge is to loop through every number from 1 to max, applying those exact same rules to each number and, before ending the loop, printing the result out in the console.

Solution

```
let max = 20; // Test out different values for max
for (let x = 1; x <= max; x += 1) {
if (x % 3 === 0 && x % 5 === 0) {
result = "fizzbuzz";
} else if (x % 3 === 0) {
result = "fizz";
} else if (x % 5 === 0) {
result = "buzz";
} else {
result = x;
}
console.log(result);
}
```