Control Flow Cheat Sheet

Here are some notes on what's been covered in this unit. Feel free to copy this and extend it to make your own cheat sheet.

Logical Operators and Booleans

Comparison and Equality Operators

Operator	Meaning	True expressions
===	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

Undefined and Null

- 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

Below are the exact rules Boolean operators follow when dealing with non-Boolean input values.

falsey	*truthy*
`false`	`true`
`0`	All numbers except `0`
Empty Strings ("")	All non-empty strings
`undefined, null, and NaN` (Not a number", as special type of numeric value)	Pretty much everything else

- NOT(`!`): If the value is *truthy*, return `false`; if the value is *falsey*, return `true`.
- OR (`||`): Return the first *truthy* value; if both values are *falsey*, return the last *falsey* value. OR is nicknamed the "default operator" (can you explain why?)
- AND (`&&`): Return the first *falsey* value; if both values are *truthy*, return the last *truthy* value. AND is nicknamed the "guard operator" (can you explain why?)

AND operator `&&`

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

OR operator `||`

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

NOT operator `!`

Condition	Result
true	false
false	true

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

Conditionals

if...else Statement Syntax

```
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`, each additional condition will only be checked if all of the prior conditions have failed.

Switch and Ternary Operators

switch Statement Syntax

```
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 Operator Syntax

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:

```
const temperature = 55;

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

typeOfExercise;
// => "Go for a run outside."
```

Arrays

What is 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 `const 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'`

Additional Array Features

- 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.

while Loops

- `while` loops can run indefinitely, so long as the condition remains true.
- A loop's condition is re-evaluated each time the code block finishes running.

The syntax for a while loop looks like this:

```
while (someConditionIsTrue) {

  //A block of code.

}
```

for Loops

- A `for` loop will generally run a fixed number of times, not indefinitely.
- The three parameters for a `for` loop, in order, are (1) an initialization, (2) a condition, and (3) a final expression.

```
for (initialization; condition; finalExpression) {
  // A block of code.
}
```

Iterating over arrays

- `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.

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

- To change the way that you iterate through the array, just change the settings of your `for` loop.

Logical Operators and Booleans

1. Create a variable `flavor` and assign it the value to `"chocolate"`.

```
const flavor = "chocolate";
```

2. Create a variable `numberScoops` and assign it the value `3`.

```
const numberScoops = 3;
```

3. Create a variable `outsideTemperature` and assign it the value `75`.

```
const outsideTemperature = 75;
```

4. Create a variable `price` and assign it the value `3.5`.

```
const price = 3.50;
```

5. Create a variable `buyIceCream` and set it equal to an expression that incorporates the following:

    - `price` is *less than or equal to* 2.5 **OR**
    - `outsideTemperature` is *greater than or equal to* 70 **AND**
    - `flavor` is *equal to* `"chocolate"` **AND**
    - `numberScoops` is *greater than* `1`

```
Const buyIceCream = price <= 2.5 || (outsideTemperature >= 70 && flavor === "chocolate"
 && numberScoops > 1);
```

6. Type `buyIceCream;` into the console and hit the return/enter key to see whether or not you should buy that delicious chocolate ice cream cone under the given conditions.

```
buyIceCream;
```

Conditionals

1. If `x` is evenly divisible by both 3 and 5 (for example, 0 or 15), set `result` to `"fizzbuzz"`.

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

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

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

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

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

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

```
let result;
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;
}
```

5. To test your code, set a value for `x` in the editor and click "Run." Then type the variable name `result` into the right pane (the console) and hit enter/return. Did you get the `result` you expected? Try out several different values for `x`, just to be sure.

```
let result;
const 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;
}
```

Switch And Ternary Operators

Part 1 - Switch Statements

1. Declare a variable `favoriteMovie` and assign it the value `"star wars"`.

```
const favoriteMovie = "star wars";
```

2. Declare a variable `moviePhrase`. It should have no initial value.

```
const moviePhrase;
```

3. Write out a switch statement for the conditions that are written in pseudo code.

```
const favoriteMovie = "star wars"; // Try assigning different values to favoriteMovie to
  make sure everything is working.
  let moviePhrase;
  switch (favoriteMovie) {
     case "jaws":
       moviePhrase = "You're gonna need a bigger boat.";
       break;
     case "the shining":
       moviePhrase = "All work and no play makes Jack a dull boy.";
       break;
     case "star wars":
       moviePhrase = "Do. Or do not. There is no try.";
       break;
     case "forrest gump":
       moviePhrase = "Life was like a box of chocolates.";
       break;
     case "back to the future":
       moviePhrase = "Where we're going, we don't need roads.";
       break;
     default:
       moviePhrase = "Great movie choice!";
  }
```

Part 2 - Ternary Operators

1. Declare a variable `hasEmptySquares` and assign it the value `false`.

```
const hasEmptySquares = false;
```

2. Declare a variable `answersAreCorrect` and assign it the value `true`.

```
const answersAreCorrect = true;
```

3. Declare a variable `message`. It should have no initial value.

```
let message;
```

4. Write a ternary statement for the conditions that are written in pseudo code.

```
const hasEmptySquares = false;
const answersAreCorrect = true;
  let message;

  message = hasEmptySquares === false && answersAreCorrect === true ? "Puzzle complete!" :
  "Not quite!";
```

Arrays

1. Create a `contacts` array. This array should contain the names `"Matt Smith"`, `"Sam Davis"`, and `"Ashley Jones"`.

```
const contacts = ["Matt Smith", "Sam Davis", "Ashley Jones"];
```

2. Create a variable `dad` and assign it the value of the second element (item) in the array, `"Sam Davis"`.

```
const dad = contacts[1];
```

3. Update the first element in the array. The new value should be `"Mark Williams"`.

```
contacts[0] = "Mark Williams";
```

4. Use the `pop()` method to remove the last element from the array (`"Ashley Jones"`).

```
contacts.pop();
```

5. Use the `push()` method to add `"Michelle Johnson"` to the end of the array.

```
contacts.push("Michelle Johnson");
```

6. Create a variable `numberOfContacts` and assign it the value of the length of the contact array.

```
const numberOfContacts = contacts.length;
```

Loops

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

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 using the command `console.log(result);`.

```
  let result;
const 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);
  }
```