JavaScript: Novice to Ninja Notes (Chap 2-4)

A block is a series of statements that are collected together inside curly braces:

```
{
// this is a block containing 2 statements
const message = 'Hello!';
alert(message);
}
```

Blocks do not need to be terminated by a semicolon.

Const variables cannot be reassigned (throws an error), but non-primitive data types, arrays, functions, and objects using const are immutable.

To escape an invalid string character, use a backslash '\'

ES6 Template Literals are created using the backtick and are super-powered strings. Include JS variables by using this format: ${variable}

Memory Leaks

A **memory leak** occurs when a program retains references to values that can no longer be accessed in its memory. This means that memory is being used to store values that are no longer required by the program, effectively wasting system resources.

Memory leaks can cause problems by gradually reducing the overall memory available, which can cause the program, or even the entire system, to run more slowly.

Most modern programming language, including JavaScript, employ various dynamic memory management techniques such as **garbage collection**, which is the process of automatically removing items from memory that are no longer required by the program. Some languages, such as C++, require the programmer to manually manage memory by removing items from memory once they are finished with.

Arrow Functions:

```
const add = (x,y) => x + y;
```

! (Logical NOT)

Placing the `!` operator in front of a value will convert it to a Boolean and return the opposite value. So truthy values will return `false`, and falsy values will return `true`. This is known as *negation*:

```
!true; // negating true returns false
<< false

!0; // 0 is falsy, so negating it returns true
<< true
```

You can use double negation (`!!`) to find out if a value is truthy or falsy (it is a shortcut to using the `Boolean` function we employed earlier because you are effectively negating the negation):

```
!!'';
<< false
```

Removing Values from Arrays

The `delete` operator will remove an item from an array:

```
delete avengers[3];
<< true
```

Ternary Operator

A shorthand way of writing an `if ... else` statement is to use the ternary operator, `?`, which takes three operands in the following format:

```
condition ? (//code to run if condition is true) : (//code to run if condit
```

Here's the example for testing if the variable `n` is odd or even, rewritten to use the ternary operator:

```
const n = 5;
n%2 === 0 ? console.log('n is an even number') : console.log('n is an odd r
<< 'n is an odd number'
```

Chapter Summaries:

Chapter Summary

- Comments are ignored by the program, but make your program easier to read and understand
- Data types are the basic building blocks of all JavaScript programs.
- There are six primitive data types: strings, symbols, numbers, Booleans, undefined and null.
- Non-primitive data types, such as arrays, functions and objects, all have a type of 'object'.
- Variables point to values stored in memory and are declared using the `const` or `let` keywords.
- Values are assigned to variables using the = operator.
- Strings and numbers have various properties and methods that provide information about them.
- Symbols are unique, immutable values.
- Boolean values are either `true` or `false`.
- There are only seven values that are `false` in JavaScript and these are known as 'falsy' values.

Chapter Summary

- Arrays are an ordered list of values
- Multidimensional arrays are arrays that contain other arrays
- Arrays have lots of methods that can be used to manipulate items in the array
- Sets are new in ES6 and are ordered lists of non-duplicate values
- Maps are new in ES6 and are ordered lists of key-value pairs
- We can use an `if` and `else` statement to control the flow of code
- The `switch` statement can be used instead of multiple `if` and `else` statements
- A `while` loop and `do ... while` loop can be used to repeat a block of code

that provide information about them.
- Symbols are unique, immutable values.
- Boolean values are either `true` or `false`.
- There are only seven values that are `false` in JavaScript and these are known as 'falsy' values.
- Data types can be converted into other data types.
- Type coercion is when JavaScript tries to convert a value into another data type in order to perform an operation.
- Logical operators can be used to check if compound statements are true or false.
- Values can be compared to see if they are equal, greater than or less than other values.

- The `switch` statement can be used instead of multiple `if` and `else` statements
- A `while` loop and `do ... while` loop can be used to repeat a block of code while a condition is still true
- A `for` loop works in a similar way to a `while` loop, but has a different syntax
- A `for-of` loop can be used to iterate over an array
- Sets and maps are enumerable, so can also be looped over using a `for-of` loop

Chapter Summary

- Functions are first-class objects that behave the same way as other values.
- Function literals can be defined using the `function` declaration, or by creating a *function expression* by assigning an anonymous function to a variable.
- All functions return a value. If this is not explicitly stated, the function will return `undefined`.
- A parameter is a value that is written in the parentheses of a function declaration and can be used like a variable inside the function's body.
- An argument is a value that is provided to a function when it is invoked.
- The `arguments` variable is an array-like object that allows access to each argument provided to the function using index notation.
- The rest operator can be used to access multiple arguments as an array.
- Default arguments can be supplied to a function by assigning them to the parameters.
- Arrow functions are a new shorthand notation that can used for writing anonymous functions in ES6.
- Function declarations can be invoked before they are defined because they are hoisted to the top of the scope, but function expressions cannot be invoked until after they are defined.
- A *callback* is a function that is provided as an argument to another function.