**JAVASCRIPT**

Browsers have built-in consoles used to debug JS code.

**Variables**

* Define a variable using ‘let’
* Storing things in computer’s memory
* We can change the value bound to the variable by name of variable + =
* A single let statement can define multiple things, just separate with a comma
* Const can also be used instead of let, but the value cannot be changed

**Functions**

* Prompt – take user input, works in terminal and browser, in browser comes up as a text box
* Console.log – print arguments to the JS console in browser

Strings:

* Backslash inside quote indicates the character after it has a special meaning (escaping the character)
* \n = new line (don’t need to do this if using backticks)
* \t = tab

Backtick quoted strings: template literal

* Can span lines
* Can embed other values using ${number}

Modulo operator: %, shows the remainder of a division

Check for odd number (a % 2)

Result will be 0 if even

Comparison operators:

* Applying them results in a Boolean value
* == equal to, != not equal to, >=, <=, >, <

Logical operators:

* && (AND), result is true only if both the values given to it are true
* The || (OR), result is true if either the values given to it are true
* ! (NOT) flips the value given to it, !true produces false
* Use when doing multiple comparisons

Conditional statements:

* Perform different actions depending on whether a condition evaluates to true or false
* Code executed only if a certain condition holds
* The deciding expression is written after the if keyword, between parentheses, followed by the statement to execute
* Code block is only ran if condition is true

If (condition) {

Statement;

}

* Else keyword to give an alternative execution

Else {

Statement;

}

* Can chain multiple if/else pairs together (if, else if, else)

Comments:

// if in single line

/\* if multiple lines, then \*/

Increments:

Let a = 1;

a +=1;

This increases value of a by 1 (shorthand of a = a +1)

* ++ (increment by one unit)
* -- (decrease by one unit)

**Debugging**

Trace tables:

* Test a program and predict step-by-step how the computer will run it
* Tracks line by the line the value of each variable

Stack trace:

* Pinpoint exact location of error
* Puts source of error at the top

Sources tab

* In developer tool
* Set breakpoints for JS – stops executing at these and lets you examine the values
* Click on line markers to introduce breakpoints