08 - Make an Alarm Clock

OUTCOMES

1. Use css, conditionals and other basic code to make an alarm clock
2. Upload your finished product to github and github pages

TIME

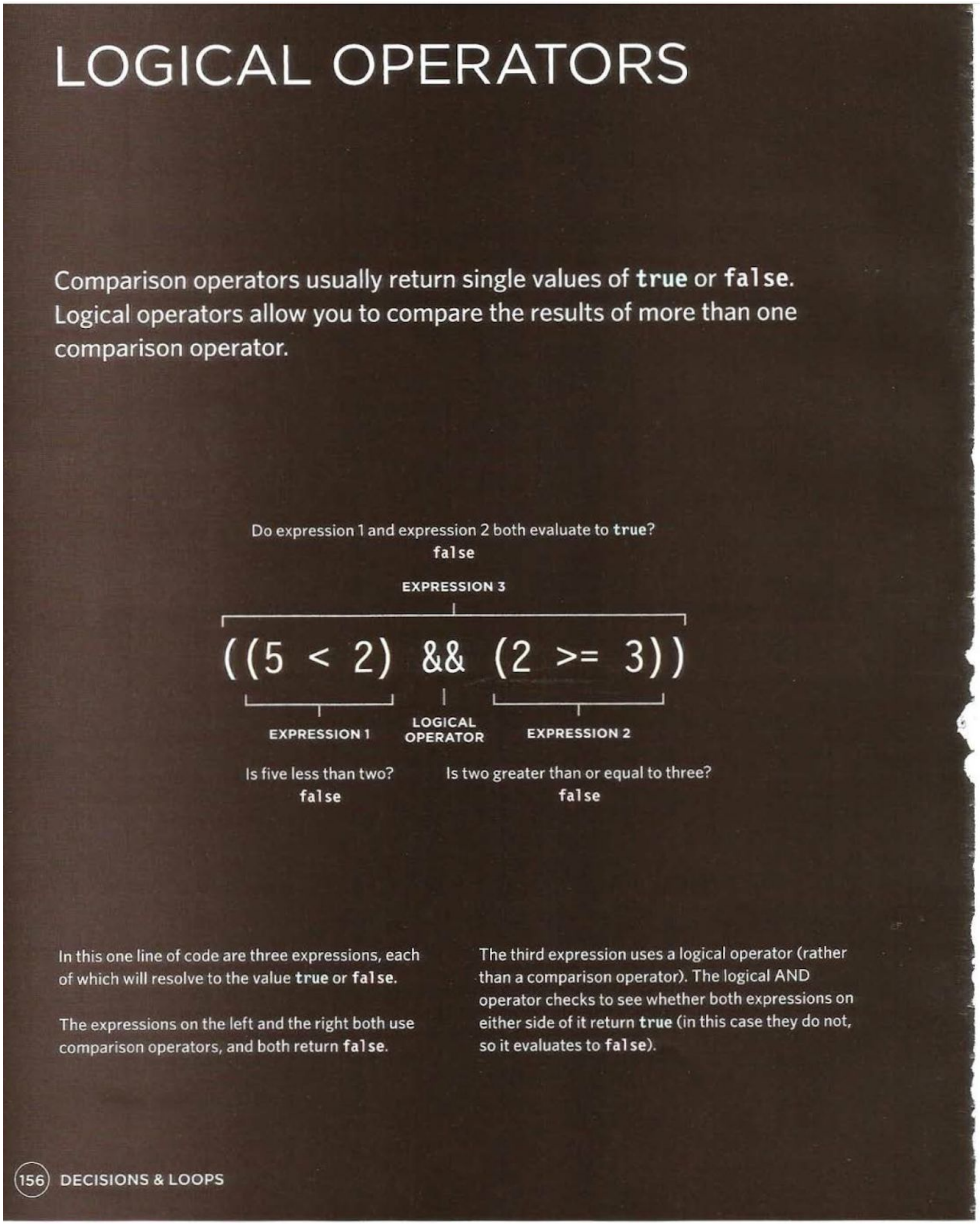
- 2 - 4 hours

MATERIALS

- Code Editor
- Web Browser - Chrome
- Pen and paper for notes and to do's

LEARNING

1. 📘Reading: Logical Conditionals/Operators -- All images from Duckett



LOGICAL OPERATORS

Comparison operators usually return single values of **`true`** or **`false`**. Logical operators allow you to compare the results of more than one comparison operator.

Do expression 1 and expression 2 both evaluate to true?
false
EXPRESSION 3
((5 < 2) && (2 >= 3))
EXPRESSION 1
LOGICAL OPERATOR
EXPRESSION 2
Is five less than two?
false
Is two greater than or equal to three?
false

In this one line of code are three expressions, each of which will resolve to the value **`true`** or **`false`**.

The expressions on the left and the right both use comparison operators, and both return **`false`**.

The third expression uses a logical operator (rather than a comparison operator). The logical AND operator checks to see whether both expressions on either side of it return **`true`** (in this case they do not, so it evaluates to **`false`**).

156 DECISIONS & LOOPS

&&

LOGICAL AND

This operator tests more than one condition.

`((2 < 5) && (3 >= 2))`
returns **`true`**

If both expressions evaluate to **`true`** then the expression returns **`true`**. If just one of these returns **`false`**, then the expression will return **`false`**.

```
 true && true   returns true
 true && false  returns false
false && true   returns false
false && false  returns false
```

||

LOGICAL OR

This operator tests at least one condition.

`((2 < 5) || (2 < 1))`
returns **`true`**

If either expression evaluates to **`true`**, then the expression returns **`true`**. If both return **`false`**, then the expression will return **`false`**.

```
 true || true   returns true
 true || false  returns true
false || true   returns true
false || false  returns false
```

!

LOGICAL NOT

This operator takes a single Boolean value and inverts it.

`!(2 < 1)`
returns **`true`**

This reverses the state of an expression. If it was **`false`** (without the **`!`** before it) it would return **`true`**. If the statement was **`true`**, it would return **`false`**.

```
!true  returns false
!false returns true
```

SHORT-CIRCUIT EVALUATION

Logical expressions are evaluated **left to right**.
If the first condition can provide enough information to get the answer, then there is no need to evaluate the second condition.

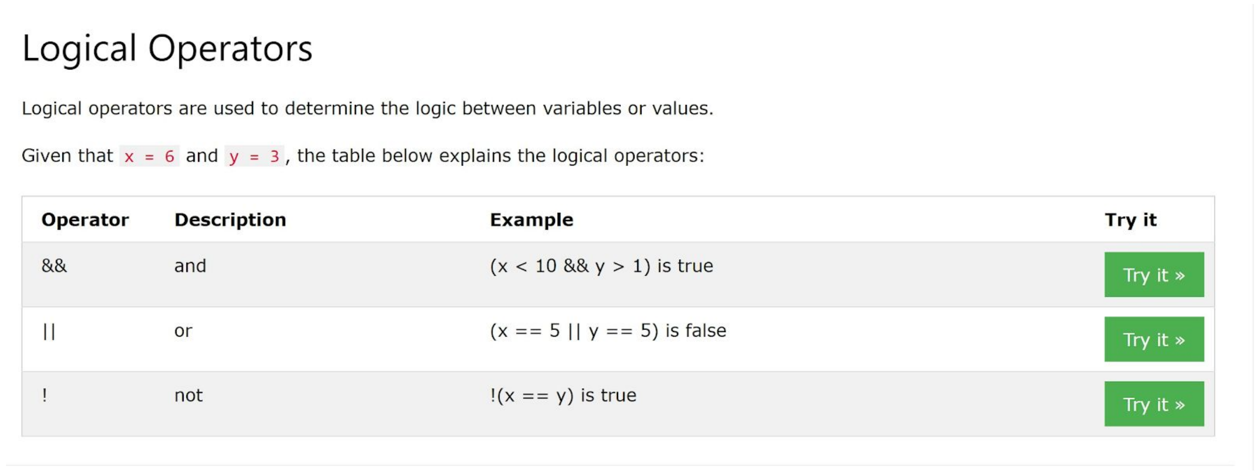
```
false && anything
    ^
      it has found a false
```

There is no point continuing to determine the other result. They cannot both be **`true`**.

```
true || anything
   ^
     it has found a true
```

There is no point continuing because at least one of the values is **`true`**.

https://www.w3schools.com/js/js_comparisons.asp (About half way down page)



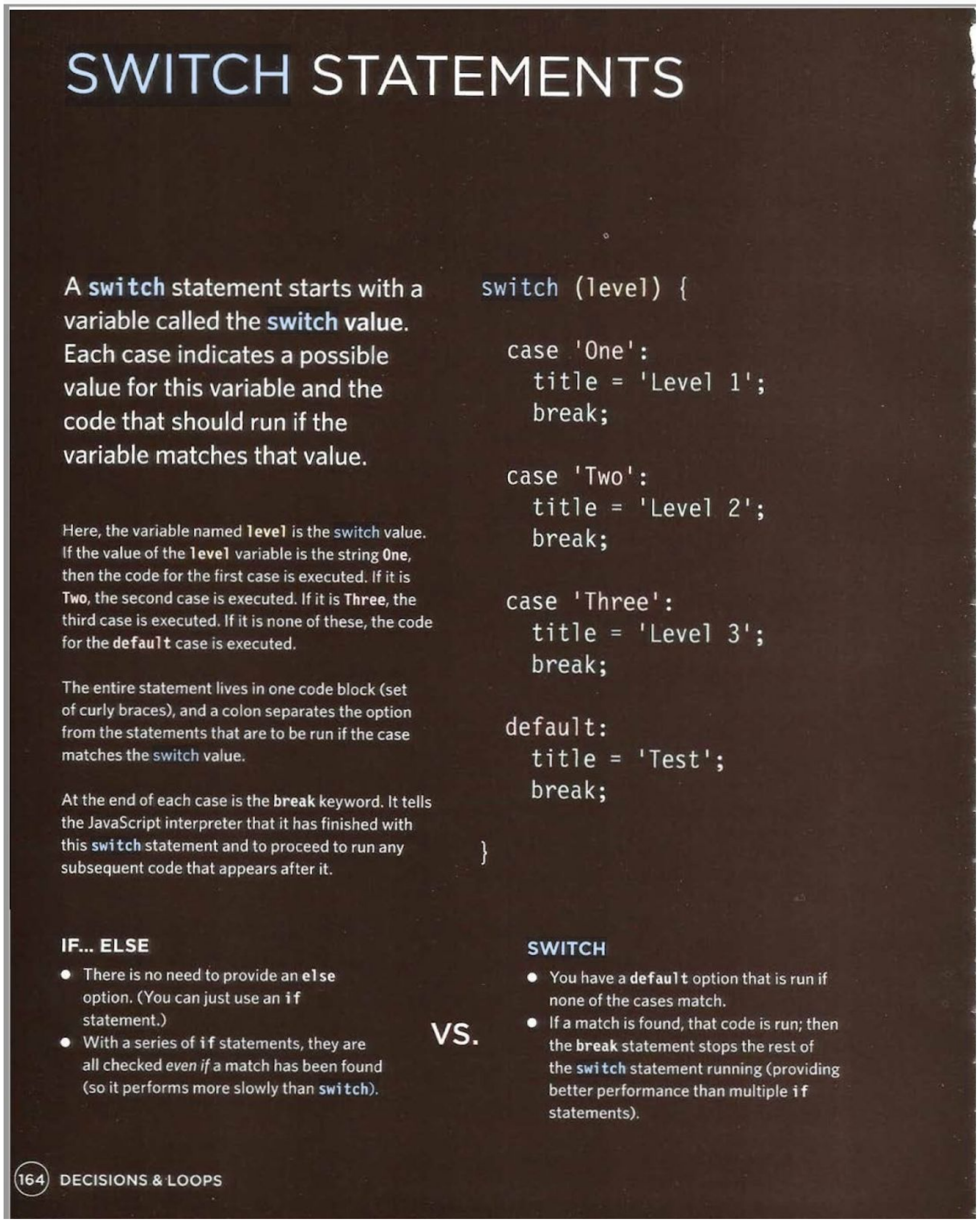
Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that `x = 6` and `y = 3`, the table below explains the logical operators:

Operator	Description	Example	Try it
&&	and	(x < 10 && y > 1) is true	Try it »
||	or	(x == 5 || y == 5) is false	Try it »
!	not	!(x == y) is true	Try it »

2. 📘Reading: Switch Statements:



SWITCH STATEMENTS

A **switch** statement starts with a variable called the **switch value**. Each case indicates a possible value for this variable and the code that should run if the variable matches that value.

Here, the variable named **level** is the switch value. If the value of the **level** variable is the string **One**, then the code for the first case is executed. If it is **Two**, the second case is executed. If it is **Three**, the third case is executed. If it is none of these, the code for the **default** case is executed.

The entire statement lives in one code block (set of curly braces), and a colon separates the option from the statements that are to be run if the case matches the switch value.

At the end of each case is the **break** keyword. It tells the JavaScript interpreter that it has finished with this **switch** statement and to proceed to run any subsequent code that appears after it.

```
switch (level) {

  case 'One':
    title = 'Level 1';
    break;

  case 'Two':
    title = 'Level 2';
    break;

  case 'Three':
    title = 'Level 3';
    break;

  default:
    title = 'Test';
    break;
}
```

IF... ELSE

- There is no need to provide an **else** option. (You can just use an **if** statement.)
- With a series of **if** statements, they are all checked *even if* a match has been found (so it performs more slowly than **switch**).

VS.

SWITCH

- You have a **default** option that is run if none of the cases match.
- If a match is found, that code is run; then the **break** statement stops the rest of the **switch** statement running (providing better performance than multiple **if** statements).

164 DECISIONS & LOOPS

🚦 W3 Schools on Switch Statements: https://www.w3schools.com/js/js_switch.asp

ACTIVITES

1. CSS style an alarm clock
2. Finish the code and use the setInterval() function to make the clock tick https://www.w3schools.com/jsref/met_win_setinterval.asp
3. Use conditional statements to hide/show the (a)0 markers and (b)am/pm
4. See teachers alarm code, finish and comment it so you have a working alarm
5. Use an mp3 file to play a sound for your alarm (code incoming for repo, teacher will notify on slack/teams channel)
6. Upload your finished product to github
7. Make sure all of your code is well commented, teacher will review

🏋 **Extra -- Reading On Functions -- for next week:** https://medium.com/better-programming/newbie-js-function-declaration-vs-function-expression-a3ae67573270

💡 **Remember to use console.log**

STUDENT INTERACTIONS

1. Initial meeting and interaction
2. Chats online and code reviews
3. Complete and work on task html templates

QUESTIONS & REFLECTION

1. How do you feel about the module so far? Do you have any concerns, feel free to contact your tutor.
2. Have you booked in for a One on One yet, please do so before the end of the week. Even just a chat and catchup is fine.

MODULE OUTCOMES

Foundation coding

These learning outcomes will enable you to:

- Debug JavaScript to eliminate errors
- Include a JavaScript library to meet project requirements
- Extend a JS library with a 3rd party plugin
- Use a range of production tools to assist in the development of a project
- Use JavaScript to manipulate the DOM
- Implement functionality of UI components with appropriate raw JavaScript and/or a library
- Write code consistently following a code style guide

- Quality assure own code by testing against industry standards
- Define deliverables based on use cases prior to production
- Write an appropriate proposal for a web project
- Set critical deadline milestones for project during the planning stage, and analyses variations from this when signing off the project