CoGrammar

Welcome to this session:
Control Structures (Loops and iterations)

The session will start shortly...

Questions? Drop them in the chat.
We'll have dedicated moderators answering questions.

Safeguarding & Welfare

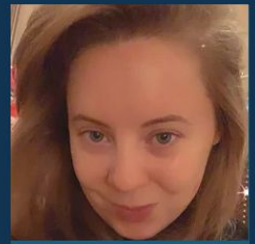
We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Rafiq Manan



Charlotte Witcher

Nurhaan Snyman

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Skills Bootcamp Cloud Web Development

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Skills Bootcamp Cloud Web Development

- For all **non-academic questions**, please submit a query: ***www.hyperiondev.com/support***
- **Report a safeguarding incident: *www.hyperiondev.com/safeguardreporting***
- We would love your feedback on lectures: Feedback on Lectures
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

Stay Safe Series:

Mastering Online Safety One week at a Time

While the digital world can be a wonderful place to make education and learning accessible to all, it is unfortunately also a space where harmful threats like online radicalization, extremist propaganda, phishing scams, online blackmail and hackers can flourish.

As a component of this BootCamp the *Stay Safe Series* will guide you through essential measures in order to protect yourself & your community from online dangers, whether they target your privacy, personal information or even attempt to manipulate your beliefs.

Don’t Take the Bait:
How to Spot Phishing Scams

- Check the Sender's Email Address
- Look for Generic Greetings
- Be Wary of Urgent Language
- Hover Over Links
- Inspect Attachments Carefully
- Look for Spelling and Grammar Errors
- Verify with the Source
- Use Multi-Factor Authentication
- Stay Informed
- Report Suspicious Emails

An HTML template typically consists of which of the following HTML elements?
(Select All That Apply)

A. <!DOCTYPE html>
B. <section>
C. <html>
D. <image>
E. <head>
F. <table>
G. <title>
H. <body>

CoGrammar

What is the purpose of CSS? (Select All That Apply)

A. Linking to JavaScript.
B. Creating Animations.
C. Improving Server Response Time.
D. Controlling Layout.
E. Styling Web Pages.

If the if statement's condition turns out to be false, which statement represents an alternative path for the flow of logic?
(Select All That Apply)

A. else
B. switch
C. return
D. break
E. else if

Learning Outcomes

- Understand and implement various looping structures in JavaScript, including for, while, and do-while loops.
- Use loops to perform repetitive tasks efficiently.
- Control loop execution using break statements.

Lecture Overview

- Loops
- For Loops
- While Loops
- Do While loops
- Break Statement
- Continue Statement
- Infinite Loops

LOOPS

- Consider a program that outputs numbers from 1 to 10.

  One way to write this is as follows:

```
console.log(1);
console.log(2);
console.log(3);
console.log(4);
console.log(5);
console.log(6);
console.log(7);
console.log(8);
console.log(9);
console.log(10);
```

LOOPS

- Although the numbers one to 10 will be printed by the code above, there are a few problems with this solution:
  - **Efficiency** - Repeatedly coding the same statements takes a lot of time.
  - **Flexibility** - What if we wanted to change the start number or end number? We would have to go through and change each line of code, adding extra lines of code where they're needed.

LOOPS

- **Scalability** - 10 repetitions are trivial, but what if we wanted 100 or even 100,000 repetitions? The number of lines of code needed would be overwhelming and very tedious for a large number of iterations.
- **Maintenance** - Where there is a large amount of code, the programmer is more likely to make a mistake
- **Feature** - The number of tasks is fixed and doesn't change at each execution.

LOOPS

- Looping control flow allows us to go back to some point in the program where we were before and repeat it.

FOR LOOPS

- The problem of outputting 1 to 10 can easily be solved by this loop. Consider the following code:

```
// Iterate through the loop 10 times
for (let i = 1; i <= 10; i++) {
  // Output the value of the variable after each iteration
  console.log(i);
}
```

FOR LOOPS

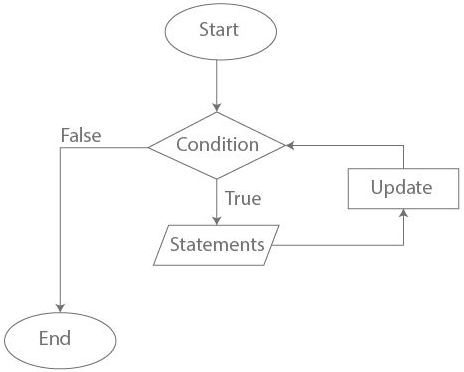
- A for loop is made up of the following steps:
  - **Declare a counter/control variable** - The code above does this when it says ***let i = 1;***. This creates a variable called ***i*** that contains the value ***1***
  - **Increase the counter/control variable in the loop** - In the for loop, this is done with the instruction ***i++*** which increases i by one with each pass of the loop
  - **Specify a condition to control when the loop will end** - The condition of the for loop is ***i <= 10***. This loop will carry on executing as long as ***i*** is less than or equal to ***10***. This loop will, therefore, execute ***10*** times

FOR LOOPS

- The screenshot below shows the syntax of **for** loops.

```
for (initialExpression; condition; updateExpression) {
  // for loop body
}
```

- For loops are used when we need to repeat our code a **set number of times**.



CoGrammar

Let's take a break

CoGrammar

WHILE LOOPS

- While loops are used when you need to repeat your code until a certain condition is met.
- A While loop is used when we don't know in advance the number of times the loop will run.
- This an example of a while loop:

```
while (condition) {
  // body of loop
}
```

WHILE LOOPS

- The problem of outputting 1 to 10 can again be resolved by while loop through this syntax:

```
// Initialise the number to start at 0.
let number = 0;

// Set a condition for the loop to repeat itself until 10 is reached
while (number < 10) {
  number++; // Increment number by 1 to ensure the output starts at 1 not 0
  console.log(number); // Output the count from 1 to 10
}
```

DO WHILE LOOPS

- The do while loop structure has the same functionality as the while loop.
- With the exception of being guaranteed to iterate at least **once** (because the condition is only checked at the end).

DO WHILE LOOPS

- Below is an example of a **do while loop** syntax:

```
// Initialise the variable with a value of -10
let counter = -10;

// Output message until the condition is met
do {
  console.log("I have run at least once!");
  counter++; // Increment the counter by 1
} while (counter <= 1); // Loop will repeat as long as the counter is <= 1

// Outputs the value of the counter once the loop ends
console.log("The result of the counter is " + counter);
```

For vs While

- A **for loop** is usually used when the number of iterations is **known**.
- The **while loop** is usually used when the number of iterations is **unknown.**
- The **do while loop** is usually used when the number of iterations is **unknown**, however, we require at least **one iteration**.

Break Statement

- The **break** statement is used to **terminate** the loop immediately when it is encountered.
- You can run a **break statement** by using the **break** keyword.
- This works for both **while** and **for** loops.

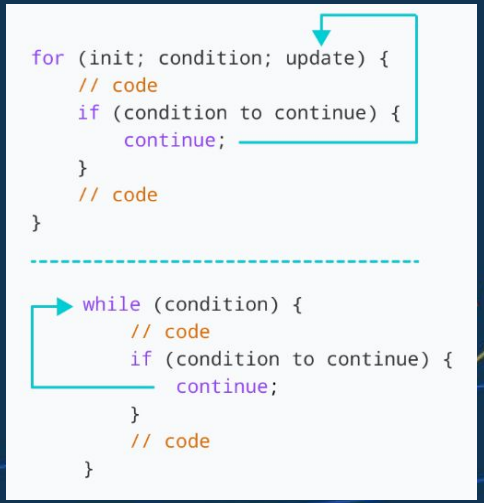
```
// program to print the value of i
for (let i = 1; i <= 5; i++) {
  // break condition
  if (i == 3) {
    break;
  }
  console.log(i);
}
```

Continue Statement

- The **continue** statement is used to **skip** the current iteration of the loop and the control flow of the program goes to the **next iteration**.
- This works for both **while** and **for** loops.

```
for (let i = 1; i <= 5; i++) {
  // condition to continue
  if (i == 3) {
    continue;
  }

  console.log(i);
}
```



INFINITE LOOPS

- A **loop** runs the risk of running forever if the condition never becomes false.
- A loop that never ends is called an infinite loop.
- Creating an infinite loop will mean that your program will run indefinitely
- An example of an infinite loop is:

```
let number = 0;

while (number < 10) {
  number--;
  console.log(number);
}
```

A while loop will execute its block of code as long as its condition evaluates to false. (True/False Statement)

A. The statement is true.
B. The statement is false.

CoGrammar

What could happen if you do not change the condition of a while loop within the body of the loop? (Select All That Apply)

A. Infinite Loop.
B. Error Message.
C. Skipping the Loop (The loop will not be executed).
D. Unresponsive Script.

Questions and Answers

CoGrammar

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education