Welcome to the CoGrammar JavaScript Tutorial

The session will start shortly...

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

CoGrammar

Full Stack Web Development Session Housekeeping

- 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**

Full Stack Web Development Session Housekeeping cont.

- 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**

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks
Guided Learning Hours (GLH): Minimum of 15 hours
Task Completion: First four tasks

Due Date: 24 March 2024

✅ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**
Software Engineering - **13 tasks**
Web Development - **13 tasks**

Due Date: 28 April 2024

CoGrammar

Skills Bootcamp Progression Overview

✅ Criterion 3: Course Progress

Completion: All mandatory tasks, including Build Your Brand and resubmissions by study period end
Interview Invitation: Within 4 weeks post-course
Guided Learning Hours: Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship Outcome: Document within 12 weeks post-graduation
Relevance: Progression to employment or related opportunity

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar

JavaScript, Iteration, Collections and Debugging

March 2024

Lecture Overview

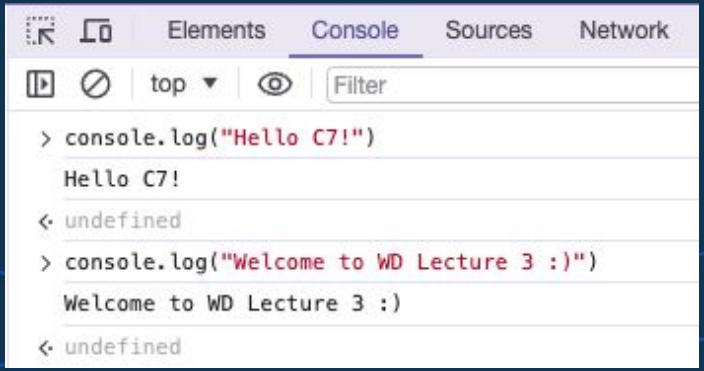
- Conditionals
- While Loops
- For Loops
- Trace Tables
- Stack Traces
- Debugging
- Handling Strings
- Arrays
- Maps

CoGrammar

JavaScript

A versatile scripting language utilised in front-end web development and server-side programming.

- We use JavaScript with HTML and CSS to transform our **static web pages** to **dynamic web pages.**
- Last week, we learnt how to **link scripts** to our HTML. These scripts are written in **JavaScript**.
- Browsers have **built-in consoles** used to **debug** JavaScript code.



CoGrammar

JavaScript

- The console is useful for **debugging** and running **code snippets**.
- To create our scripts, we will use **Visual Studio Code** and **Node.js**.
- The following extensions are also helpful when running your code:
  - **Code Runner**: allows you to run JavaScript code in VS Code by pressing **Ctrl+Alt+N**, right-clicking and pressing **"Run Code"**, or by pressing the **"Play" button**.
  - **Open in Browser**: allows you to open an HTML file which has been correctly linked to JavaScript scripts **in your default browser**.

CoGrammar

Conditional Statements

Statements that perform different actions depending on whether a condition evaluates to true or false.

- **Conditional execution** is created with the **if** keyword in JavaScript.
- We want some code to be executed **if**, and only **if**, a certain **condition** holds.
- The deciding expression is written after the **if** keyword, between parentheses, followed by the statement to execute.

```
let temperature = 10.6;
if (temperature < 20) {
  console.log("Yikes, it's too cold here");
}
```

CoGrammar

Conditional Statements

- You can use the **else keyword**, together with **if**, to create two separate, **alternative execution** paths.

```
let temperature = 10.6;
if (temperature < 20) {
  console.log("Yikes, it's too cold here.");
} else {
  console.log("Eh, I can survive.");
}
```

- If you have more than two paths to choose from, you can “chain” multiple if/else pairs together.

```
if (num < 10) {
  console.log("Small");
} else if (num < 100) {
  console.log("Medium");
} else {
  console.log("Large");
}
```

CoGrammar

Trace Tables

A technique used to test a program and predict, step-by-step, how the computer will run it.

```
let a = 10;
let b = 6;
let total = a + b;
console.log(total);
```

line	a	b	total	log
1	10			
2		6		
3			16	
4				16

CoGrammar

Stack Trace

A detailed report of function calls leading to an error.

- Analysing the stack trace helps pinpoint the **exact location of the error**.
- **Woof, foo and bar** are the functions that were called.
  - The bottom most line shows the line number where **bar** was called, it says the function was called at line **108**.
  - From here we can see that **bar** was called first, which later called the function **foo.**
  - After that **foo** called the function **woof.** This indicates that the source of the error is **woof.**

```
Uncaught ReferenceError: barry is not defined
    at woof (practical.js:97:15)
    at foo (practical.js:101:3)
    at bar (practical.js:105:3)
    at practical.js:108:1
```

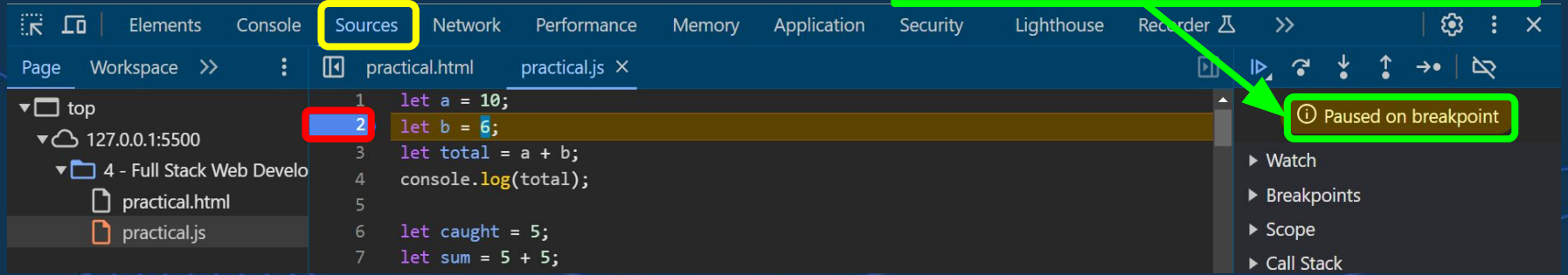
CoGrammar

Debugging JavaScript

The process of examining the program, finding the error and fixing it.

- You can set **breakpoints** for JavaScript code in the **Sources** tab in the **Developers tool**.
- JavaScript will stop executing at each **breakpoint** and lets you examine the values.

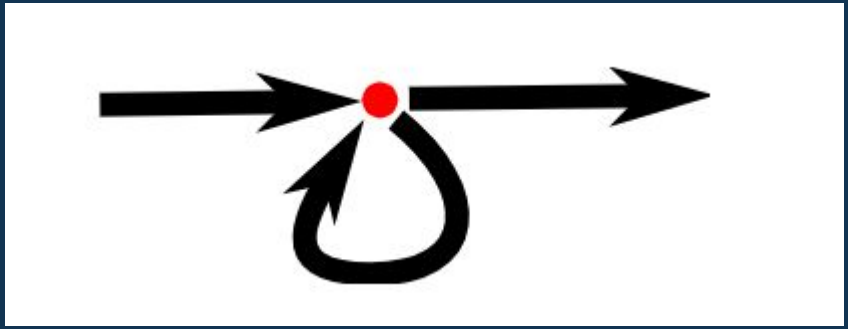
These buttons help you move around the code in debug mode.



CoGrammar

Loops

- Often in our code, we need a way to run a piece of code **multiple times**. This form of control flow is called a **loop**.
- **Looping** control flow allows us to go back to some point in the program where we were before and **repeat** it.



CoGrammar

While Loops

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

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

- **While** loops are used when you need to repeat your code until a certain **condition is met**.
- We can use **trace tables** to help us test our loops and evaluate how the computer will run the code, line by line.

CoGrammar

While Loops

```
let laps = 1,
  finish_line = 5;

// while loop from i = 1 to 5
while (laps <= finish_line) {
  console.log(laps);
  laps += 1;
}
```

laps	finish_line	laps <= finish_line	Output
1	5	true	Print 1
2	5	true	Print 2
3	5	true	Print 3
4	5	true	Print 4
5	5	true	Print 5
6	5	false	Stop

CoGrammar

Infinite While Loops

- If the **condition** of a **loop** is always true, the **loop** runs for **infinite** times.

```
// infinite while loop
while (true) {
  // body of loop
}
```

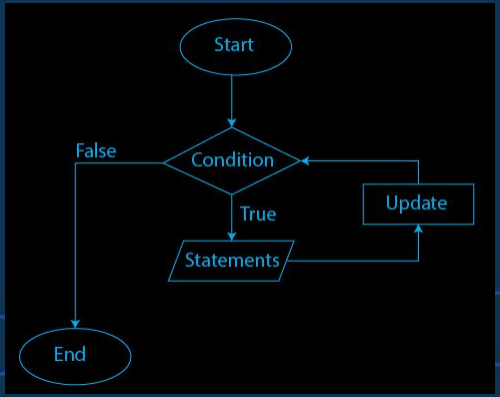
CoGrammar

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

For Loops

```
const MAX = 5;

// looping from i = 1 to 5
for (let i = 1; i <= MAX; i++) {
  console.log(`Good night`);
}
```

i	MAX	i <= MAX	Action
1	5	true	Print
2	5	true	Print
3	5	true	Print
4	5	true	Print
5	5	true	Print
6	5	false	Stop

CoGrammar

Infinite For loop

- If the test **condition** in a **for** loop is always **true**, it runs forever (until memory is full).

```
// infinite for loop
for (let i = 1; i > 0; i++) {
  console.log("I will go onnnnn foreverrrrrrr.....");
}
```

CoGrammar

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);
}
```

CoGrammar

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.

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

  console.log(i);
}
```

```
for (init; condition; update) {
    // code
    if (condition to continue) {
        continue;
    }
    // code
}

while (condition) {
    // code
    if (condition to continue) {
        continue;
    }
    // code
}
```

CoGrammar

Arrays

An array is a data structure that can store multiple values at once.

- We can create an **array** by placing elements inside an array literal **[ ]**, separated by **commas.**

```
// empty array
let colors = [];

// array of strings
let colors2 = ["red", "blue", "green"];

// array with mixed data types
let data = ["name", 1, true];
```

CoGrammar

Arrays

- Each **element** of an array is associated with a number called an **index**.
- The **index** specifies the **position** of the element inside the array.

```
let even = [2, 4, 6, 8, 10];
```

Index	0	1	2	3	4
	2	4	6	8	10

- We can use an array **index** to **access** the elements of the array.

```
console.log("even[0]: ", even[0]); // even[0]:  2
console.log("even[2]: ", even[2]); // even[2]:  6
```

CoGrammar

Arrays

- We can **add** elements to an array using **built-in** methods like **push()** and **unshift()**.
  - The **push()** method adds an element at the **end** of the array.
  - The **unshift()** method adds an element at the **beginning**.

```
even.push(12);
even.unshift(0);
```

  - We can add or **change** elements by accessing the **index** value.

```
even = [2, 4, 6, 8, 10];
even[0] = 100;
console.log(even); // [100, 4, 6, 8, 10]
```

CoGrammar

Arrays

- We can remove (and add) an element from any specified index of an array using the **splice()** method.
- The arguments are: **(position, howManyToRemove, itemsToAdd..)**

```
even = [2, 4, 6, 8, 10];
console.log(even.splice(2, 1)); // [6]
console.log(even); // [2, 4, 8, 10]
```

CoGrammar

Arrays

- We can find the length of an array using the **length** property.

```
console.log(even); // [2, 4, 8, 10]
console.log(even.length); // 4
```

CoGrammar

Maps

CoGrammar

Maps

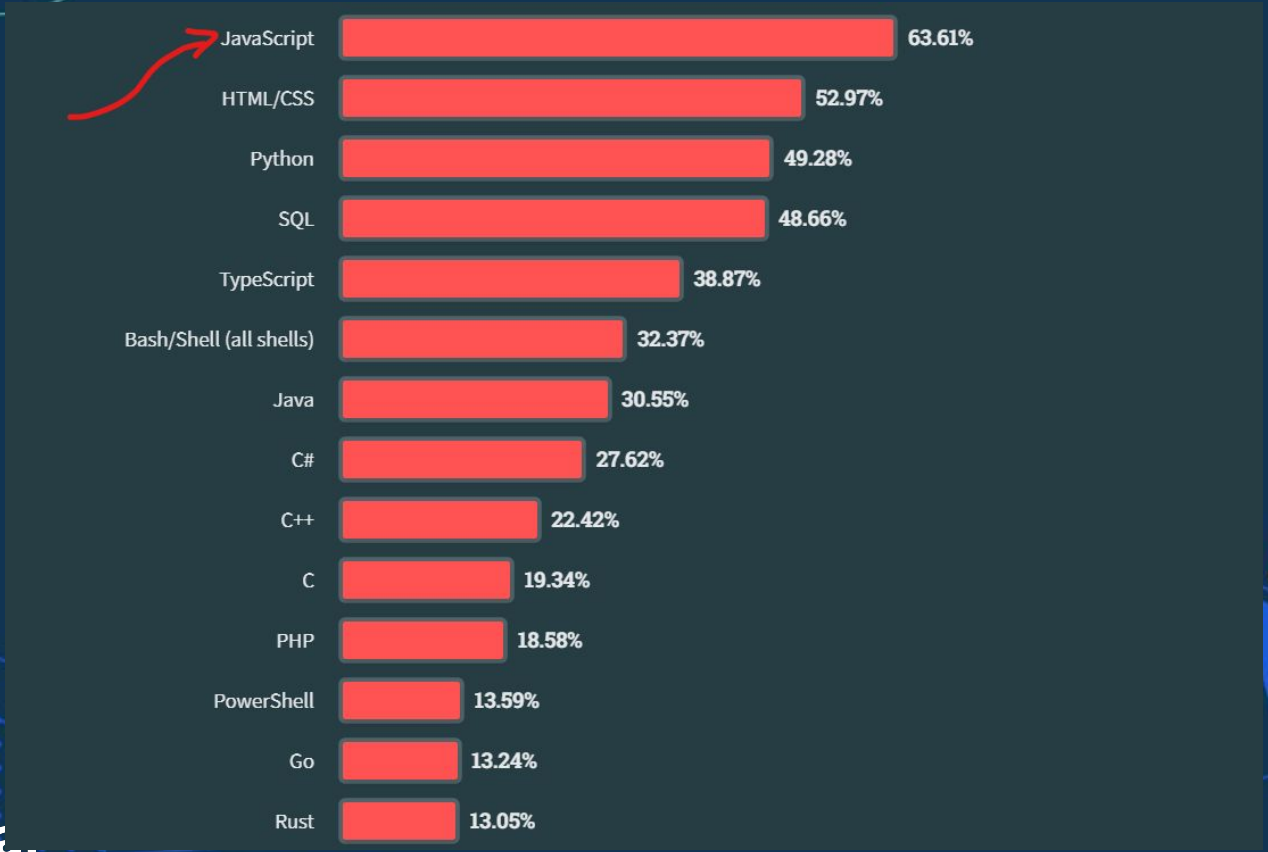
- The optional argument to the **Map()** constructor should be an **iterable** object that yields two element **[key, value] arrays**.
- In practice, this means that if you want to initialize a **map** when you create it, you'll typically write out the desired keys and associated values as an array of arrays.

Maps

- In addition to **get()** and **set()**,
  - use **has()** to check whether a map includes the specified key
  - use **delete()** to remove a key (and its associated **value**) from the map
  - use **clear()** to remove all **key/value** pairs from the map
  - use the **size** property to find out how many keys a map contains

CoGrammar

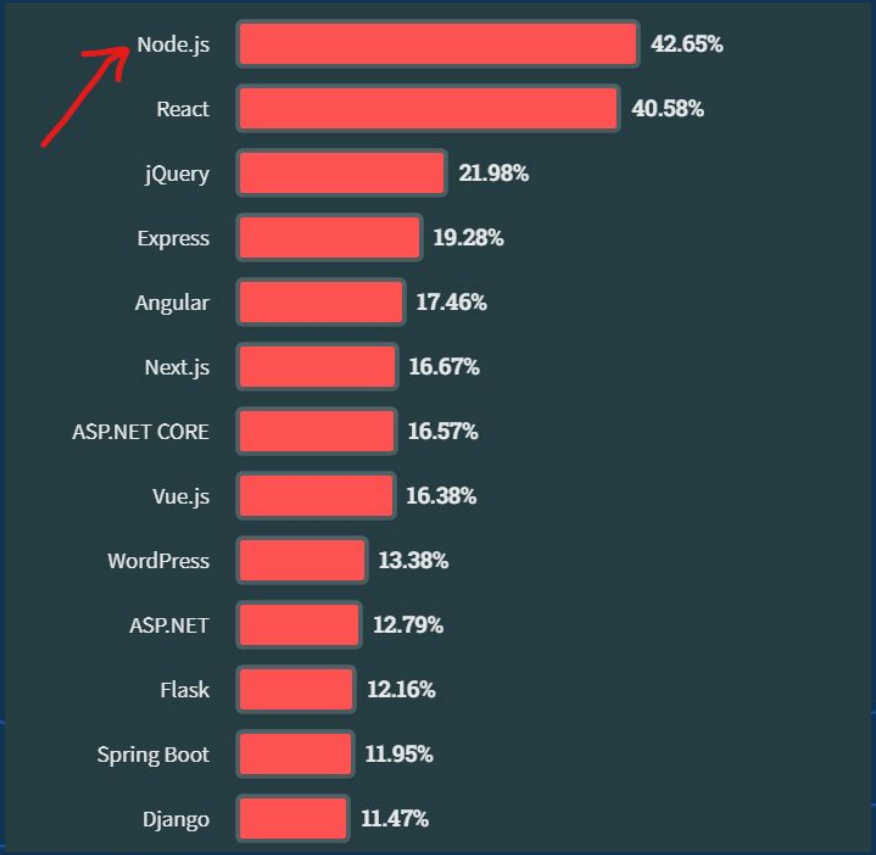
Facts (Programming Language)



CoGramma

StackOverflow Dev Survey

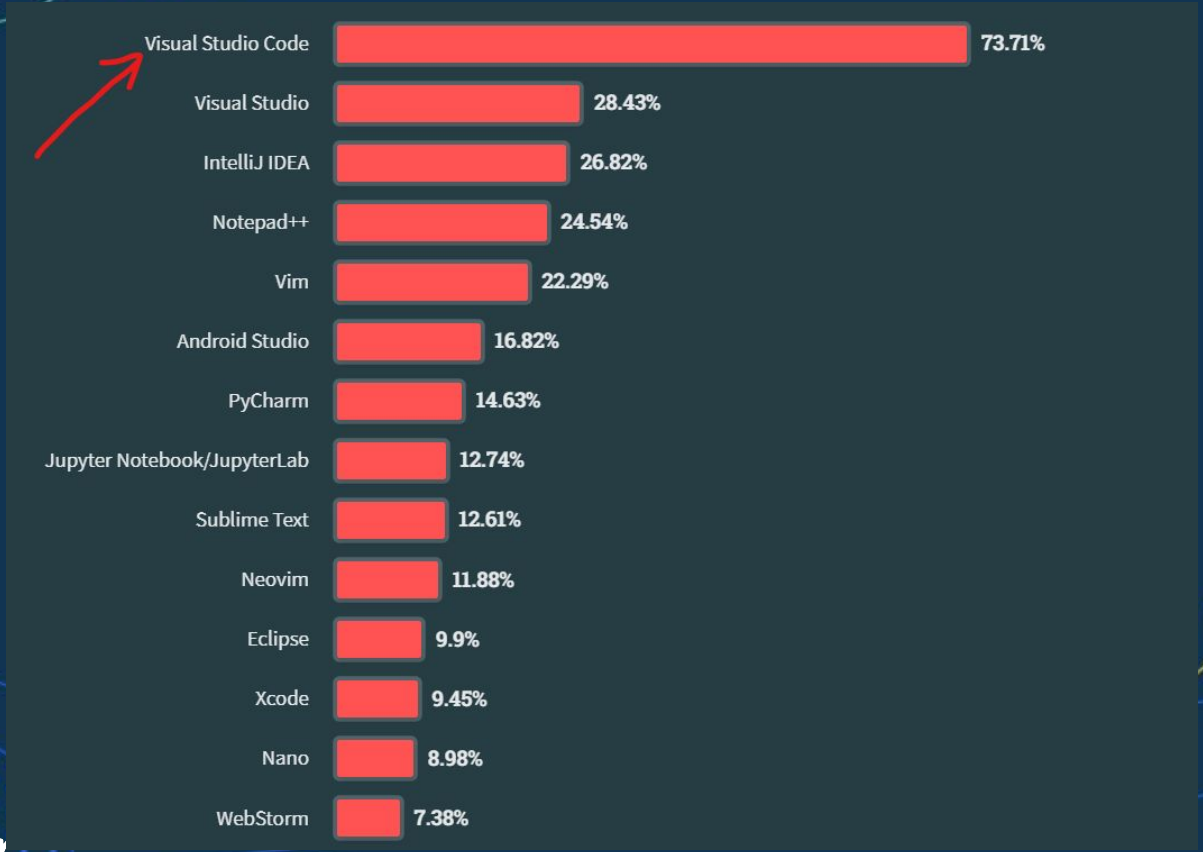
Facts (Frameworks)



StackOverflow Dev Survey

CoGrammar

Facts (IDE)



StackOverflow Dev Survey

Resources

JavaScript Documentation

- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- https://devdocs.io/javascript/
- https://www.w3schools.com/jsrEF/default.asp

StackOverflow Developer Survey

- https://survey.stackoverflow.co/2023/#technology-most-popular-technologies

NodeJS

- https://nodejs.org/en/download

CoGrammar

StackOverflow Dev Survey

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar