



Welcome to this session:

Task Walkthrough - JavaScript Data Types and Conditional Statements

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



Nurhaan Snyman



Rafiq Manan



Ronald Munodawafa



Tevin Pitts

Scan to report a
safeguarding concern



or email the Designated
Safeguarding Lead:
Ian Wyles

safeguarding@hyperiondev.com

Skills Bootcamp Full Stack 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. We will 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.](#)
- Find all the lecture **content** in your [Lecture Backpack](#) on GitHub.
- If you are hearing impaired, kindly use your computer's function through Google chrome to enable captions.

Learning Outcomes

- ❖ Capture and validate user input using JavaScript and HTML forms to create dynamic, interactive web applications.
- ❖ Work with different data types in JavaScript, including numbers, strings, and booleans, and convert between them when necessary.
- ❖ Apply conditional logic using if/else statements to make decisions based on user input and control the flow of the program.
- ❖ Perform arithmetic operations in JavaScript to calculate results, such as totals or percentages, using user-provided data.
- ❖ Use string manipulation and template literals to generate personalised, dynamic output based on the user's responses.
- ❖ Link external JavaScript files to HTML and update the page dynamically with values calculated using JavaScript functions.

Lecture Overview

- Presentation of the Task
- Introduction to JS
- Introduction to Variables and Data Types
- Conditional Statements
- Task Walkthrough



Data Types Task

In this task, you'll create a Compliment Generator! By asking a few fun questions, you'll generate a personalised compliment that will put a smile on the user's face. Using JavaScript, you'll work with string manipulation, variables, and template literals to make compliments based on input like the user's favourite animal, favourite hobby, or a random number! 🐱🎨

- ❖ Use `prompt()` to collect user input (e.g., favorite animal, favorite hobby).
- ❖ Generate a compliment using string concatenation or template literals.
- ❖ Ensure you're collecting data of various types (e.g. age -> number).

Conditional Statements Task

Heading out for dinner with friends? 🍴 Ever wondered how to split the bill and calculate the tip? Now, you'll create a Tip Calculator that will handle the maths for you! You'll ask for the total bill, the number of people sharing it, and the percentage tip they want to leave. Then, using JavaScript, you'll calculate how much each person needs to pay, and of course, the total tip. 📁

- ❖ Use prompts to ask for the total bill, number of people and tip percentage.
- ❖ Calculate the amount each person should pay by dividing the total bill and adding the tip percentage.
- ❖ Display the result using `alert()` or dynamically update the HTML with the final amount per person.



What operator would you use to multiply in JavaScript?

- A. /
- B. %
- C. *
- D. +

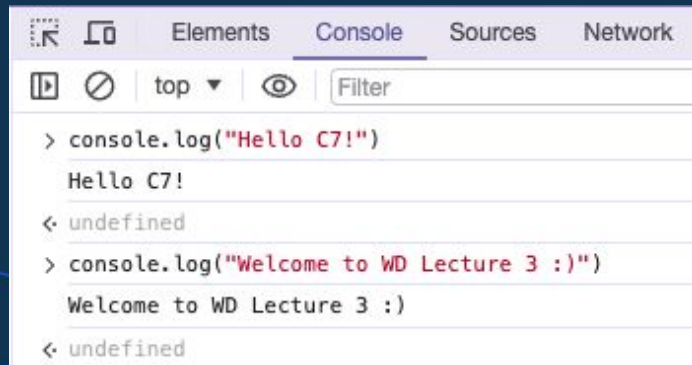
How do you prompt the user for input in JavaScript?

- A. `prompt()`
- B. `console.log()`
- C. `input()`
- D. `alert()`

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**.
- ❖ We can **link scripts** to our HTML. These scripts are written in **JavaScript**.
- ❖ Browsers have **built-in consoles** used to **debug** JavaScript code.



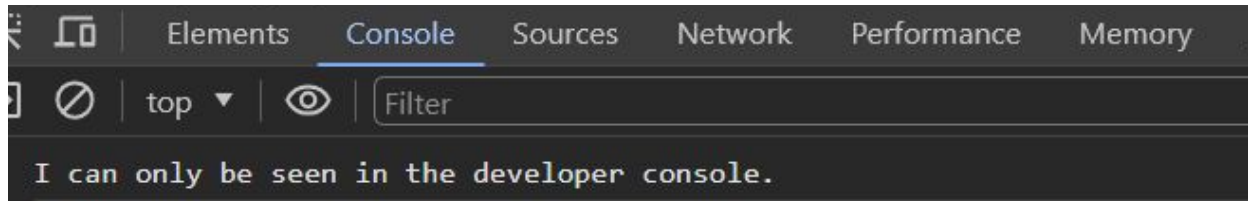
The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays two log entries: 'Hello C7!' and 'Welcome to WD Lecture 3 :)'. Each log entry is preceded by a prompt character '>' and followed by 'undefined'. The console interface includes a toolbar with icons for opening the console, disabling it, and a dropdown menu set to 'top'. A search filter box is also present.

```
> console.log("Hello C7!")
Hello C7!
< undefined
> console.log("Welcome to WD Lecture 3 :)")
Welcome to WD Lecture 3 :)
< undefined
```

Linking scripts

- ❖ The HTML `<script>` tag is used to add JavaScript to the document.

```
<script>  
  console.log("I can only be seen in the developer console.");  
</script>
```



Linking scripts

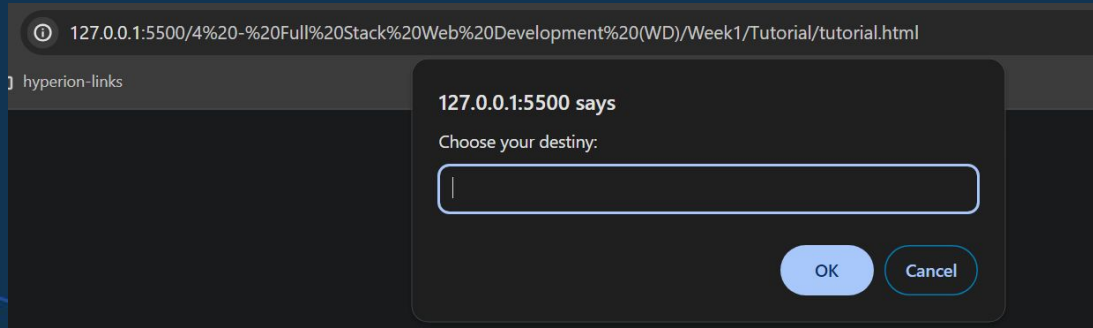
- ❖ We can also link HTML documents with external JavaScript files using the src attribute.

```
</script>  
<script src="scripts/hello.js"></script>  
</body>
```

Functions: prompt

- ❖ **prompt**: a function that shows a little dialog box asking for user input.

```
<script>  
  prompt("Choose your destiny:");  
</script>
```



Functions: console.log

- ❖ Most JavaScript systems (including all modern web browsers and Node.js) provide a **console.log** function that writes out its **arguments** to some text **output** device.
- ❖ In browsers, the output lands in the **JavaScript console**. This part of the browser interface is hidden by default, but most browsers open it when you open **Developer Tools or the Inspect view**.

```
let sum = 5 + 5;  
console.log(sum); // 10  
console.log("Value of sum: ", sum); // Value of sum: 10
```


Variables

Symbols used to represent values stored in the computer's memory

- ❖ The special word (keyword) **let** indicates that this program is going to **define a variable**.
- ❖ It is followed by the **name** of the variable, **"=" operator** and a **value/expression**.

```
let num1 = 5;  
let sum = 5+5;
```

- ❖ After a variable has been defined, its **name** can be used in expressions.

```
console.log("Your number is: ")  
console.log(num1)
```

Data Types

- ❖ Every **value** has a **type** that determines its **role**.
- ❖ Some values are numbers, some values are pieces of text and so on.

```
6           // Number
"Hi"        // String
true        // Boolean
[1, 2, 3, 4] // Array
```

Template literals

- ❖ Backtick-quoted strings, usually called **template literals**, can do a few more tricks.
- ❖ Apart from being able to span lines, they can also **embed other values**.
- ❖ When you write something inside **`${}`** in a **template literal**, its result will be **computed, converted to a string**, and included at that position.

```
`Half of 100 is ${100/2}`
```

Arithmetic Operations

- ❖ The **+** and ***** symbols are called **operators**.
- ❖ **Operators** are used to represent **operations**, the former being **addition** and the latter being **multiplication**.
- ❖ Putting an operator between two values will **apply** it to those values and produce a **new value**.
- ❖ We use **-** for **subtraction** and **/** for division.

```
console.log(100 + 4);  
console.log(4 * 11);  
console.log(100 - 10);  
console.log(100 / 10);
```

Arithmetic Operations

- ❖ The % symbol is used to represent the **remainder** operation. You'll also often see this operator referred to as **modulo**.

```
console.log(314 % 100); // 14  
console.log(144 % 12); // 0
```

Comparison Operations

- ❖ The **>** and **<** signs are the traditional symbols for “**is greater than**” and “**is less than**”, respectively.
- ❖ Applying them results in a Boolean value that indicates whether they hold true in this case.

```
console.log(3 > 2) // -> true  
console.log(3 < 2) // -> false
```

- ❖ Other similar operators are **>= (greater than or equal to)**, **<= (less than or equal to)**, **== (equal to)**, and **!= (not equal to)**.

```
console.log(4 >= 4); // true  
console.log(4 <= 5); // true  
console.log(40 == 40); // true  
console.log(100 != 100); // false
```


Logical Operators

- ❖ JavaScript supports three logical operators: **&&**, **||**, and **!**.
- ❖ The **&&** operator represents logical **AND**
 - Its result is **true** only if **both** the values given to it are **true**.
- ❖ The **||** operator denotes logical **OR**.
 - Its result is **true** if **either** the values given to it are **true**.
- ❖ **Not** is written as an **exclamation mark (!)** and it flips the value given to it.
 - **!true** produces **false** and **!false** gives **true**.

```
console.log(true && false); // false
console.log(true && true); // true
console.log(false || true); // true
console.log(false || false); // false
console.log(!true); // false
console.log(!false); // true
```


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");  
}
```



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");
}
```

Data Types Task

Who doesn't love a good compliment? 🌟 In this task, you'll create a Compliment Generator! By asking a few fun questions, you'll generate a personalised compliment that will put a smile on the user's face. Using JavaScript, you'll work with string manipulation, variables, and template literals to make compliments based on input like the user's favourite animal, favourite hobby, or a random number! 🐱🎨

- ❖ Use `prompt()` to collect user input (e.g., favorite animal, favorite hobby).
- ❖ Generate a compliment using string concatenation or template literals.
- ❖ Ensure you're collecting data of various types (e.g. age -> number).

Conditional Statements Task

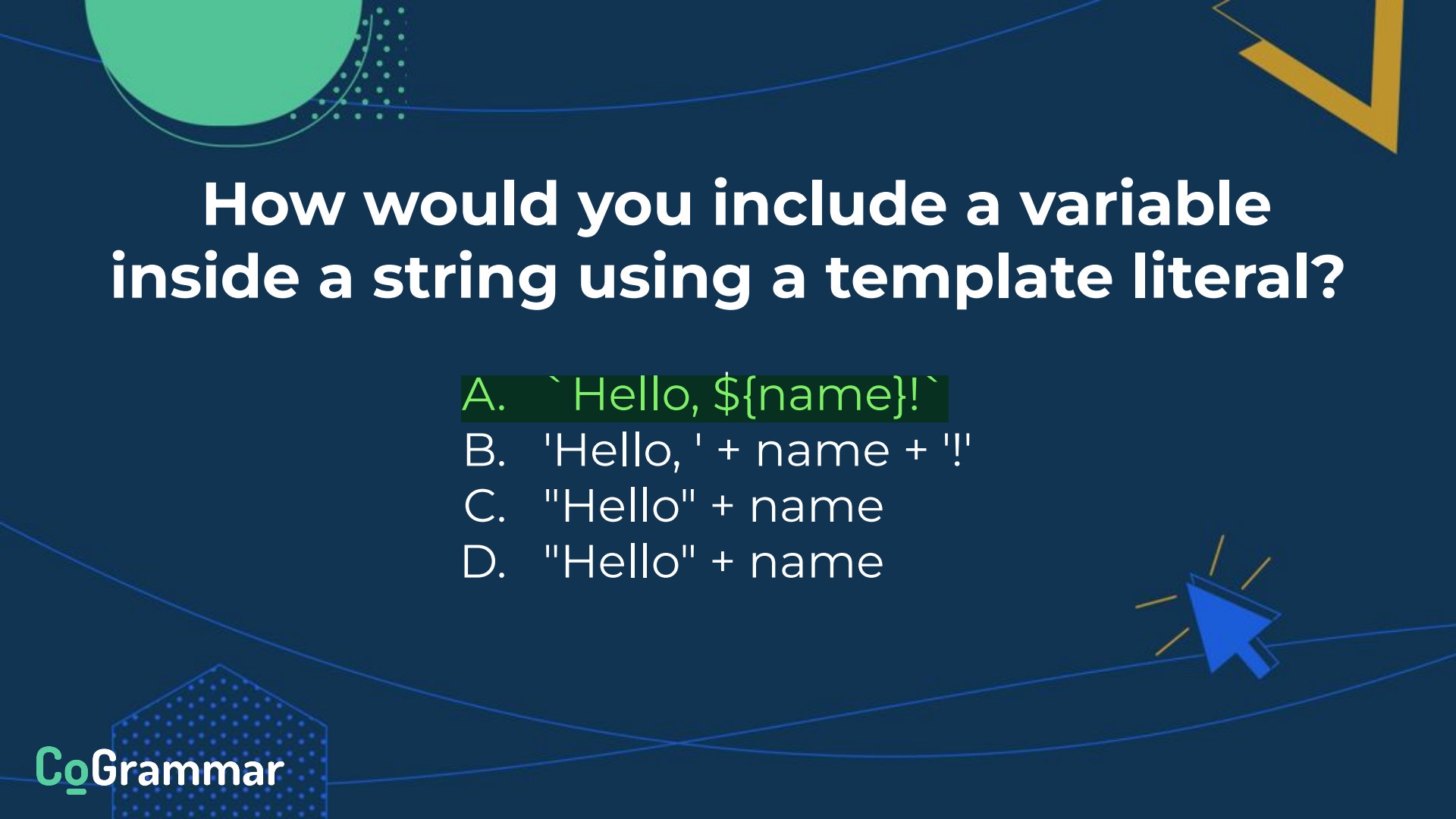
Heading out for dinner with friends? 🍴 Ever wondered how to split the bill and calculate the tip? Now, you'll create a Tip Calculator that will handle the maths for you! You'll ask for the total bill, the number of people sharing it, and the percentage tip they want to leave. Then, using JavaScript, you'll calculate how much each person needs to pay, and of course, the total tip. 📁

- ❖ Use prompts to ask for the total bill, number of people and tip percentage.
- ❖ Calculate the amount each person should pay by dividing the total bill and adding the tip percentage.
- ❖ Display the result using `alert()` or dynamically update the HTML with the final amount per person.



Which of the following correctly calculates 10% of a bill in JavaScript?

- A. `totalBill / 100`
- B. `totalBill * 0.10`
- C. `totalBill - 10`
- D. `totalBill % 0.10`



How would you include a variable inside a string using a template literal?

- A. ``Hello, ${name}!``
- B. `'Hello, ' + name + '!'`
- C. `"Hello" + name`
- D. `"Hello" + name`

CoGrammar

Q & A SECTION

**Please use this time to ask
any questions relating to the
topic, should you have any.**

Thank you for attending



CoGrammar



Department
for Education