Introduction

Why JavaScript?

Today, programming has grown in to one of the biggest career fields out there. Countless languages, frameworks, libraries and other extensions are created every day! So, out of all the options, why should you choose JavaScript? JavaScript is a programming language that is one of the core technologies of the World Wide Web.

Uses

Web Development: JavaScript is used to create dynamic and interactive web content like applications and browsers. JavaScript is the most used programming language in the world, used as a client-side programming language by 97.0% of all websites. The push of a button, the changing of colors, every click, every like and every comment on every website is possible thanks to JavaScript.

Game Development: JavaScript is frequently used to create in-browser games. Developers use JavaScript to create 2D and 3D puzzles, role-playing games, racing games, platform games, and more.

What will you learn?

In this course, you will learn the fundamentals of the most modern version of JavaScript. We will start from the most basic concepts and move on to more complicated topics. We will be working in the console for now, because making websites with JavaScript requires you to be fully familiar with the fundamentals of coding. Once you learn those, you can move on to the next courses which teach JavaScript more in depth and teach you how to use it in the browser to make websites. Have fun and take your time!

Hello World!

In the programming world, once a new apprentice wants to learn the magic of code, they are introduced with a program called "Hello, World!". This is one of the simplest programs you can write with any language.

The program consists of printing "Hello, World!" to the screen. In JavaScript, we can print words, numbers and data with the command console.log ().

For example, if we want to print "Hello, World!" we would write:

```
console.log('Hello, World!');
```

When we want to print text, we have to write the text in the parentheses of console.log (). We also have to wrap the text with ' ' or " " so the compiler that runs the program knows that the text we wrote isn't code, but actual text we want to print.

Challenge

Now, it's your turn. Write the command and press "Run Code". Remember, the program has to print EXACTLY "Hello, World!" in order to be correct.

'Clean Code' and its meaning

Imagine you are a professional developer in a team of 15 developers. You are all working on one website. What would happen if every developer started writing code with their own principles and in their own distinct way? Chaos, of course! That's why it's important not just to write code, but to write clean code.

Clean code is code that is easy to read and understand. Any developer should be able to look at your code, read it, understand it easily and be able to continue it. As a beginner don't stress too much about this, but it's better to start early. During the course you will learn some principles that will help you make sure your code is always clean. Believe it or not, you can already learn some principles:

Keep you code DRY

By DRY, we mean Don't Repeat Yourself. You will learn more about this later.

Use semicolons

In most programming languages, at the end of every line of code you have to put a semicolon. In JavaScript, it is not necessary to put a semicolon at the end of lines, but it is a good practice to put them. Some developers choose not to. Whatever you choose, stick to one and don't mix the two.

Variables

In JavaScript, you will work with a lot of data. Variables are containers for data. Every variable has a name and a value. We can get the value of the variable by using it's name.

An example for variables is:

```
let someNumber = 5;
const randomText = 'Hello, World!';
var lightIsOn = false;
```

Now, in order to use a variable in our application we have to declare the variable. In JavaScript we can declare variables with these three key words: var, let, const.

You can declare variables by writing:

let/const/var variable Name = variable Value

Const

We use const to declare a constant piece of data that we know won't change. **For example: const PI = 3.14159.** Once we declare a constant with a value, we cannot change it's value. If we try to change it, the program will crash.

Let & var

These two are used to declare variables that can change their value. There are differences between let and var, but we haven't gotten that far yet. For now, all you need to know is that var is very outdated and should be avoided. You should only use let. **For example: let itemPrice = 10.25.** In order to change the value of the variable we write: **itemPrice = 11.50.**

Using variables

When you want to get the value from a variable, we can just use the name. Let's print a variable:

```
const PI = 3.14159;
console.log(PI);
```

This will output: **3.14159.** Notice how we didn't use **' '** or **" "** in **console.log ()** because we want to print the value of the variable PI, and not the actual text 'PI'. **console.log('PI')** would output **PI.**

Data types

There are lots of types of data you will encounter while programming. **15, 3.1415,** true, 'Hello, World!' are all different types of data.

In JavaScript, we declare all variables with **let** and **const,** but it's still very valuable to know the different data types:

Number

We have already worked with numbers. There are integers, which are whole numbers without decimals. Floats are numbers with decimal points. In JavaScript, all types of numbers are just called **Number.**

let age = 41;

String

A string is what we usually call text. An example for a string would be: 'Hi, my name is Christophe!', or **'teleportation123**'. A single character is called a char: 'a'. For example:

let name = 'John';

const lastName = 'Doe';

let myChar = 'A';

Boolean

Booleans have two values. True or false. Booleans are like a switch that can be turned on or off. For example:

let isFriday = true;

let isTwoOClock = false;

Undefined & Null

Undefined is a variable that has not been assigned a value: let my Variable;

Null is an intentional empty value: **let unknown Value = null;**

There are two more types of data called **Objects and Arrays**. We will cover Arrays later in this course. If you want to learn about Objects, which are very important in JavaScript, you can check out my next course called **Object Oriented Programming in JavaScript.**

Type conversion

While working with variables, you can change the data type of the variable. This will be useful later on when working with web apps and forms. There are different ways to convert from data type to data type:

Converting to a String

Converting values to a string looks like this:

```
let value = '';

value = String(55); // Value is: '55'.
value = String(true); // Value is: 'true'.
value = String(5 + 20); // Value is: '25'.
```

We can also use the **toString()** method to convert to a String:

```
let number = 25;
let string = number.toString();
```

Converting to a Number

Similar to converting to a string, converting to a number looks like this:

```
let value = 0;
value = Number('5'); // Value is: 5.
value = Number(true); // Value is: 1.
value = Number(false); // Value is: 0.
value = Number(null); // Value is: 0.
value = Number('Hello, World!'); // Value is: NaN.
```

Whenever converting to a number, we can get the value **NaN. NaN** means Not a Number and we get **NaN** whenever we try to turn something that can't be a number into a number.

We can also use the **parseInt()** and **parseFloat()** functions:

```
const someNumber = parseInt('100'); // Value is: 100.
const otherNumber = parseInt('100.25'); // Value is: 100.
const anotherNumber = parseFloat('100.25'); // Value is 100.25.
```

We use **parseInt()** for whole numbers and **parseFloat() for** numbers with decimal places.

Type coersion

Type coercion is when JavaScript converts types of data for us automatically. Let's say we are trying to sum up two numbers, 5 and 6, but one of them in our application is of type String and the other of type Number. What do you think would happen then?

const firstNumber = '5';

const secondNumber = 6;

const sum = firstNumber + secondNumber;

Believe it or not, the output from this program will be: **'56'.** The output will be a string because JavaScript recognizes that one of the variables that we are adding together is a string, so it transforms the other variable into a string as well. Then, because we are adding them together, it adds **'5'** and **'6'** and gets **'56'.**

Why doesn't it turn them into numbers? Well, because when one variable is already a string, that string could contain anything. It could be a digit, random text or anything else. We cannot transform random text into a number, so that's why JavaScript decides to turn the number into a string. This is one example of type coercion. There are many more, but you will get to know them as you learn more coding.

Naming variables

Variable names should be descriptive and easy to understand. Don't be afraid to write long variable names. Now, there are some rules to keep in mind while writing names for variables.

The names of variables can only contain **letters, numbers, underscores _ and dollar signs $.** They also must not start with a number. For example:

Const number = 5;	Valid
Const you&I = ‘love’;	Invalid
Const HeUiOna4 = true	Valid
Const 4times = false	Invalid
Const $time_flies = trues;	Valid

Naming conventions

You will often write variable names that contain multiple words. How should you connect the words?

The most popular way is called **Camel Case.** It consists of starting with lowercase letters and capitalizing the first letter of every word after the first word. Valid examples of Camel Case are

- **userPassword**
- **shouldTurnOnLightSwitch**
- **secondList**
- **number.**

Some also write variable names with underscores like

- **user_password**
- **should_turn_on_light_switch**
- **second_list**
- **number**

In JavaScript, it is best to use the Camel Case naming convention, as most built-in commands are written that way.

Challenge

Clean up this code and these variables. Delete the variables that don't have valid names. Also delete the variables that aren't written in Camel Case.

```
const myNewNumber = '555-555-5555';
const _init = true;
const website = 'coddy.tech';
const thisIsAGreatCourse = true;
```

Writing 'Clean' variables

There are some principles to keep in mind when declaring and working with variables.

Here are some to keep in mind:

Only declare variables when needed. This doesn't mean that you should rarely use variables, it just means don't over do it. Declare them when and where is needed only.

Use descriptive names for variables that describe the piece of data they hold. For example, instead of writing **let a1 = '123456',** you should write **let badPassword = '123456'**.

Declare variables where you need them. Some schools these days teach you that all variables should be declared at the top of the program. That is not good practice. Declare a variable as close to the lines where you use it as possible.

Use const unless you know the value will change. For variables that you will only use once, or only in a specific place in the program and you know that you won't have to change their value, use const to declare them. Only use let if the value of the variable has to change during the program.

Working with variables

Simple math

Time to do some math!

Simple math with numbers

There are some basic operations you can easily do in JavaScript. We will use the following variables to explain all operations:

```
const num1 = 100;
const num2 = 50;
let value;
```

Addition & Subtraction

We use the + symbol to add two numbers. We use the - symbol to subtract two numbers.

```
value = num1 + num2; // Value: 150
value = num1 - num2; // Value:
```

Modulo

We use the % symbol to calculate the remainder from dividing two numbers.

```
value = num1 % num2; // Value: 0
```

Challenge

You are given two numbers num1 and num2. Print the sum, difference, product and the quotient.

Hints

Use the +, -, *, / symbols for each operation.

```
function math(num1, num2) {
    // Write your code below this line
    console.log(num1 + num2);
    console.log(num1 - num2);
    console.log(num1 * num2);
    console.log(num1 / num2);
}
```

Numbers & the Math object

The Math object

In JavaScript, there is an object called Math that can be used for more complex mathematics. It has multiple methods that we can use:

Getting the value of PI

```
value = Math.PI; // Output: 3.141592653589793
```

Rounding a number

```
value = Math.round(2.4); // Output: 2
```

Rounding down to the nearest whole number

value = Math.floor(3.67); // Output: 3

Rounding up to the nearest whole number

value = Math.ceil(2.1); // Output: 3

Calculating square root

value = Math.sqrt(64); // Output: 8

Absolute value

value = Math.abs(-5); // Output: 5

Powers (First number to the power of second number)

value = Math.pow(8, 2); // Output: 64

Minimum

```
value = Math.min(5, 2, 1, 4, 6); // Output: 1
```

Maximum

```
value = Math.max(2, 3, 1, 5, 4); // Output: 5
```

Random value

```
value = Math.random() // Output: random decimal number from 0 to 1;
```

There are other methods in the Math object, but these are the important ones for now.