Principles of JavaScript

In JSHP we start with a set of fundamental principles

These tools will enable us to problem solve and communicate almost any scenario in JavaScript

— We'll start with an essential approach to get ourselves up to a shared level of understanding
— This approach will help us with the hard parts to come

What happens when javascript executes (runs) my code?

```
const num = 3;
function multiplyBy2 (inputNumber){
  const result = inputNumber*2;
  return result;
}
const name = "Will"
```

As soon as we start running our code, we create a *global execution context*

— Thread of execution (parsing and executing the code line after line)
— Live memory of variables with data (known as a Global Variable Environment)

Running/calling/invoking a function

This is not the same as defining a function

```
const num = 3;
function multiplyBy2 (inputNumber){
  const result = inputNumber*2;
  return result;
}

const output = multiplyBy2(4);
const newOutput = multiplyBy2(10);
```

When you execute a function you create a new execution context comprising:

1. The thread of execution (we go through the code **in the function** line by line)
2. A local memory ('Variable environment') where anything defined in the function is stored

We keep track of the functions being called in JavaScript with a Call stack

Tracks which execution context we are in - that is, what function is currently being run and where to return to after an execution context is popped off the stack

One global execution context, a new function execution context for every time we run a function