Direct and Indirect Function Execution

Direct Function Execution

```
doSomething()
```

Parentheses

You (= the developer) call a function immediately.

Indirect Function Execution

```
setTimeout(doSomething, 1000);
```

NO Parentheses

Function is handed off to be eventually executed by “someone else”/ in the future.

ACADEMIND

Summary – Direct vs Indirect Execution (Callbacks)

Functions can be executed **in two ways** in JavaScript: By "**directly calling**" a function or "**indirectly**" by setting a function up to be called eventually (**callback function**).

You call a function **directly** by **adding parentheses** after the function name: `doSomething()`

A function is executed **indirectly** by **passing the name** of the function to "someone" who will eventually call the function (hence the name "**callback function**").

Direct function execution runs the function code immediately, indirect execution does not.

Arguments can easily be passed to **direct** function calls (`doSomething(1, 2, 3)`).

For **indirect function calls**, a (anonymous) **wrapper function** or **`bind()`** can be used to "pre-configure" it.