JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

@JONASSCHMEDTMAN

SECTION

HOW JAVASCRIPT WORKS BEHIND THE SCENES

LECTURE

EXECUTION CONTEXTS AND THE CALL STACK

JS

WHAT IS AN EXECUTION CONTEXT?

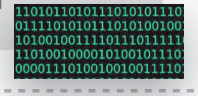
👉 Human-readable code:

```
const name = 'Jonas';

const first = () => {
  let a = 1;
  const b = second();
  a = a + b;
  return a;
};

function second() {
  var c = 2;
  return c;
}
```

Function body only executed when called!

EXECUTION

Compilation



Creation of **global execution context** (for top-level code)

NOT inside a function

Execution of **top-level code** (inside global EC)

Execution of **functions** and waiting for **callbacks**

Example: click event callback

EXECUTION CONTEXT

Environment in which a piece of JavaScript is executed. Stores all the necessary information for some code to be executed.

👉 **Exactly <u>one</u> global execution context (EC):** Default context, created for code that is not inside any function (top-level).

👉 **One execution context <u>per function</u>:** For each function call, a new execution context is created.

All together make the call stack

EXECUTION CONTEXT IN DETAIL

WHAT'S INSIDE EXECUTION CONTEXT?

1. Variable Environment
   - 👉 let, const and var declarations
   - 👉 Functions
   - 👉 ~~arguments~~ object
2. Scope chain
3. ~~this~~ keyword

NOT in arrow functions!

Generated during "creation phase", right before execution

```
const name = 'Jonas';

const first = () => {
  let a = 1;
  const b = second(7, 9);
  a = a + b;
  return a;
};

function second(x, y) {
  var c = 2;
  return c;
}

const x = first();
```

Global

```
name = 'Jonas'
first = <function>
second = <function>
x = <unknown>
```

Literally the function code

Need to run `first()` first

first()

```
a = 1
b = <unknown>
```

Need to run `second()` first

second()

```
c = 2
arguments = [7, 9]
```

Array of passed arguments. Available in all "regular" functions (not arrow)

(Technically, values only become known during execution)