JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

@JONASSCHMEDTMAN

SECTION

HOW JAVASCRIPT WORKS BEHIND THE SCENES

LECTURE

VARIABLE ENVIRONMENT: HOISTING AND THE TDZ

JS

HOISTING IN JAVASCRIPT

👉 **Hoisting:** Makes some types of variables accessible/usable in the code before they are actually declared. "Variables lifted to the top of their scope".

↓ ***BEHIND THE SCENES***

Before execution, code is scanned for variable declarations, and for each variable, a new property is created in the **variable environment object**.

EXECUTION CONTEXT

- 👉 Variable environment
- ✅ Scope chain
- 👉 `this` keyword

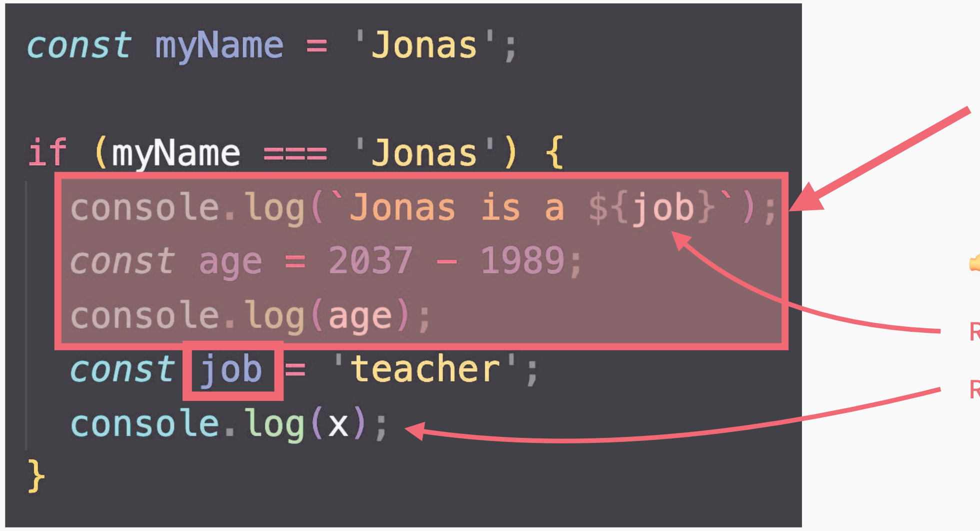
	HOISTED? 👇	INITIAL VALUE 👇	SCOPE 👇
`function` declarations	✅ YES	Actual function	Block
`var` variables	✅ YES	`undefined`	Function
`let` and `const` variables	🚫 NO	`<uninitialized>`, TDZ	Block
`function` expressions and arrows	🤷 Depends if using `var` or `let`/`const`		

In strict mode. Otherwise: function!

Technically, yes. But not in practice

Temporal Dead Zone

TEMPORAL DEAD ZONE, LET AND CONST



```javascript
const myName = 'Jonas';

if (myName === 'Jonas') {
  console.log(`Jonas is a ${job}`);
  const age = 2037 - 1989;
  console.log(age);
  const job = 'teacher';
  console.log(x);
}
```

TEMPORAL DEAD ZONE FOR **`job`** VARIABLE

- **Different kinds of error messages:**

```
ReferenceError: Cannot access 'job' before initialization
ReferenceError: x is not defined
```

WHY HOISTING?

- **Using functions before actual declaration;**
- `var` hoisting is just a byproduct.

WHY TDZ?

- **Makes it easier to avoid and catch errors:** accessing variables before declaration is bad practice and should be avoided;
- **Makes `const` variables actually work**