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!

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

```
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`

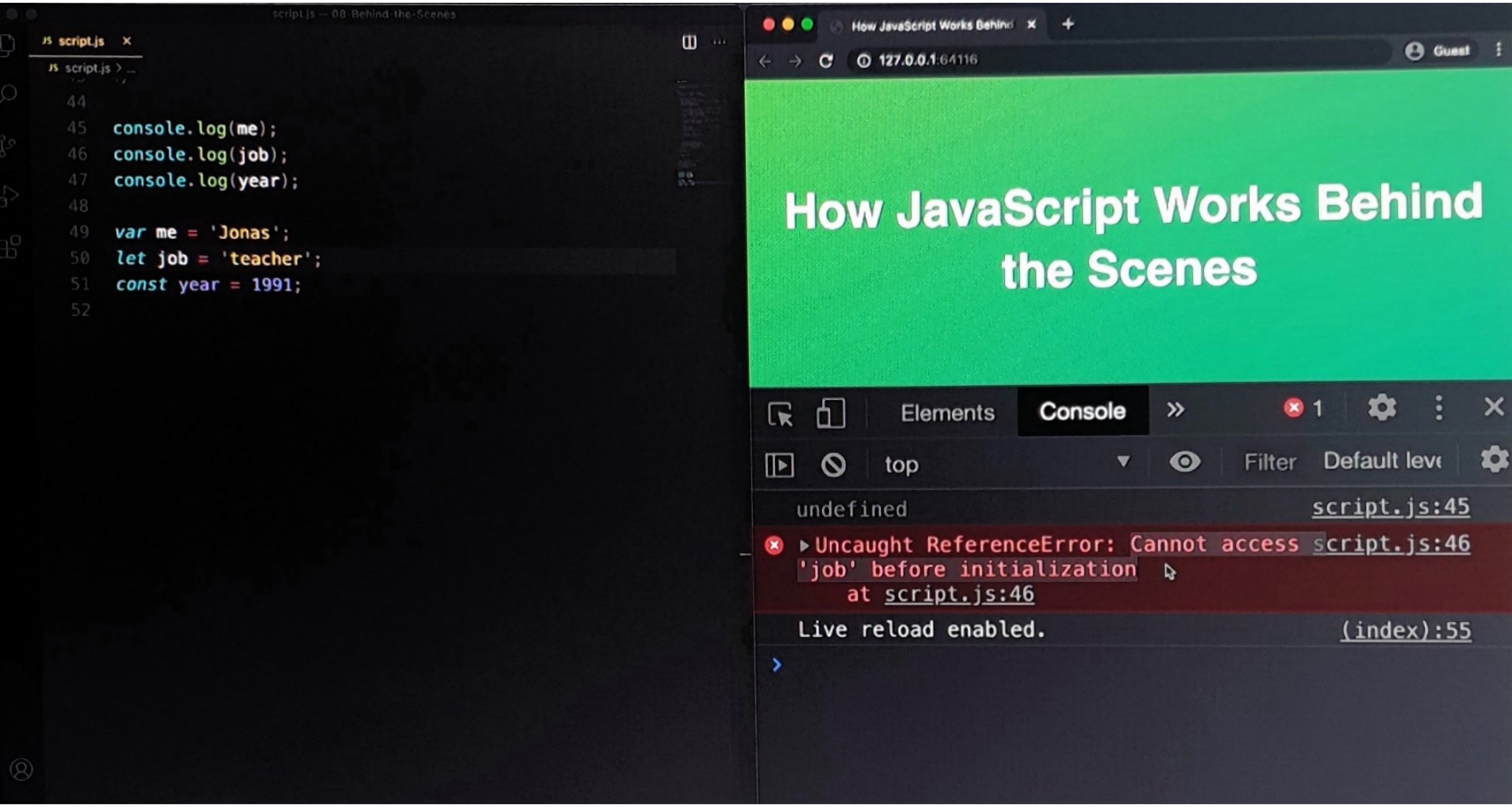
`ReferenceError: x is not defined`

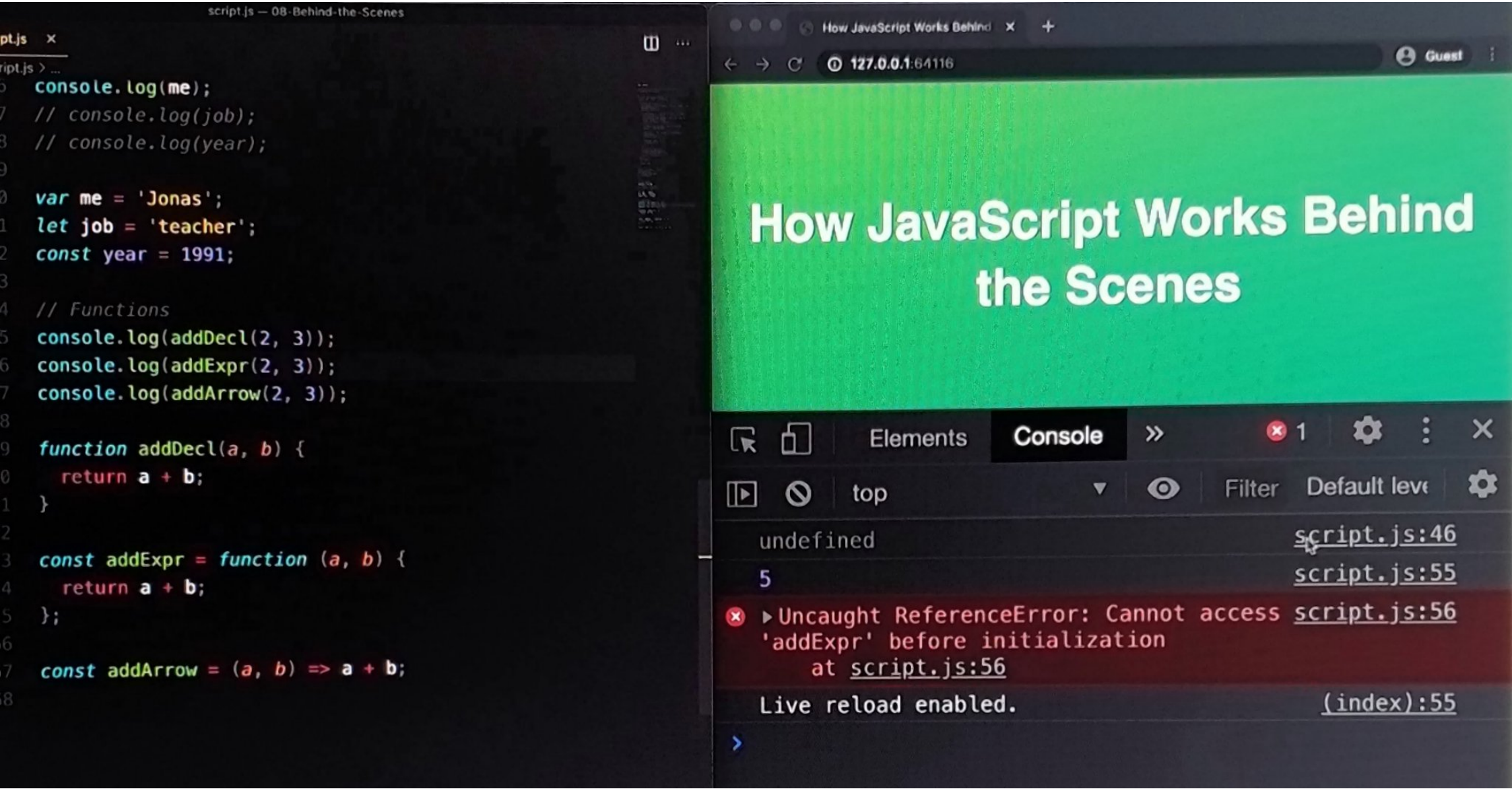
WHY HOISTING?

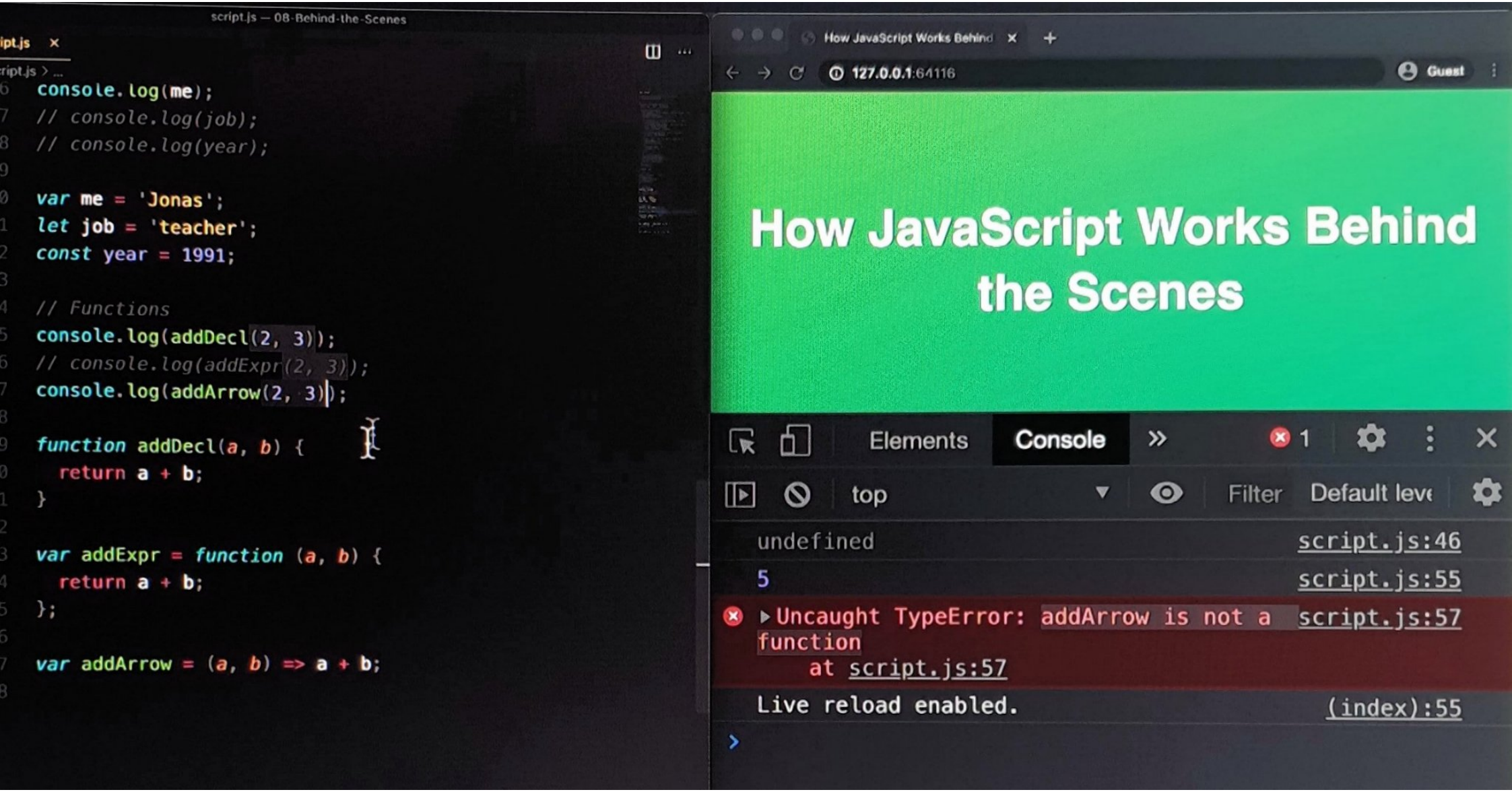
- **Using functions before actual declaration;**
- `var` hoisting is just a byproduct.

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**


script.js — 08-Behind-the-Scenes
script.js
script.js > ...
44
45 console.log(me);
46 console.log(job);
47 console.log(year);
48
49 var me = 'Jonas';
50 let job = 'teacher';
51 const year = 1991;
52
How JavaScript Works Behind
127.0.0.1:64116
Guest
How JavaScript Works Behind the Scenes
Elements
Console
1
top
Filter
Default leve
undefined
script.js:45
Uncaught ReferenceError: Cannot access 'job' before initialization
at script.js:46
script.js:46
Live reload enabled.
(index):55


script.js — 08-Behind-the-Scenes
console.log(me);
// console.log(job);
// console.log(year);
var me = 'Jonas';
let job = 'teacher';
const year = 1991;
// Functions
console.log(addDecl(2, 3));
console.log(addExpr(2, 3));
console.log(addArrow(2, 3));
function addDecl(a, b) {
return a + b;
}
const addExpr = function (a, b) {
return a + b;
};
const addArrow = (a, b) => a + b;
How JavaScript Works Behind
127.0.0.1:64116
Guest
How JavaScript Works Behind the Scenes
Elements
Console
top
Filter
Default leve
undefined
script.js:46
5
script.js:55
Uncaught ReferenceError: Cannot access 'addExpr' before initialization
at script.js:56
script.js:56
Live reload enabled.
(index):55


script.js — 08-Behind-the-Scenes
console.log(me);
// console.log(job);
// console.log(year);
var me = 'Jonas';
let job = 'teacher';
const year = 1991;
// Functions
console.log(addDecl(2, 3));
// console.log(addExpr(2, 3));
console.log(addArrow(2, 3));
function addDecl(a, b) {
return a + b;
}
var addExpr = function (a, b) {
return a + b;
};
var addArrow = (a, b) => a + b;
How JavaScript Works Behind
127.0.0.1:64116
Guest
How JavaScript Works Behind the Scenes
Elements
Console
top
Filter
Default leve
undefined
script.js:46
5
script.js:55
Uncaught TypeError: addArrow is not a function
script.js:57
at script.js:57
Live reload enabled.
(index):55