JavaScript Refresher

In case it's been some time...

Maximilian Schwarzmüller

This course section is optional!

It's recommended if you haven't used JavaScript in a while or if you don't have a lot of JavaScript experience

This section does not replace a JavaScript course!

But it will revisit crucial JavaScript concepts needed for building React apps

JavaScript Refresher

In case it's been some time since you last worked with JavaScript

- Core Syntax & Rules
- Essential, Modern JavaScript Features
- Key JavaScript Features Used In React Apps

JavaScript Can Be Executed In Many Environments

In the Browser
(i.e., as part of websites)

JavaScript code can be included in any website

The code then executes inside the browser (i.e., on the machine of the website visitor)

On any Computer
(e.g, server-side code)

Thanks to Node.js or Deno, JavaScript code can be executed outside of the browser, too

The code then executes directly on the machine

On mobile Devices
(e.g., via embedded websites)

With extra technologies like Capacitor or React Native, you can build mobile apps based on JavaScript

The code then executes on the mobile device

Adding JavaScript Code To A Website

Between <script> Tags

```
<script>
alert('Hello')
</script>
```

Can quickly lead to unmaintainable & complex HTML files

Typically only used for very short scripts

Via <script> Import

```
<script src="script.js"></script>
```

Separates HTML & JavaScript code

Maintaining complex JS-powered apps becomes easier

JavaScript code is just plain text!

How Code Is Executed

Code is read top to bottom,
left to right

JavaScript Code Consists Of “Statements”

Statement

The “thing” that gets executed

Keywords	Identifiers	Values / Expressions
Built into JavaScript	Defined by developers	Defined by developers
Required to “tell” JavaScript that a certain feature is used	Used to identify commands (functions), variables (values) etc.	Hardcoded values or expressions that produce new values
`let`, `if`, `for`, …	`alert()`, `age`, …	‘Hello world’, `5 - 3`, …

Keywords

Keywords enable language features

```
let, const, if, for, function, …
```

Identifiers

Identifiers identify “things”

Variables

Functions
(“Commands”)

Parameters

Property

JavaScript code is case-sensitive!

Identifiers Must Follow Certain Rules & Recommendations

#1

Must not contain whitespace or special characters (except $ and _)
Valid: $userName, age, user_name, data$, ...
Invalid: %userName, age/, user name, ...

#2

May contain numbers but must not start with a number
Valid: user3, us3r, ...
Invalid: 3user, 11players, ...

#3

Must not clash with reserved keywords
Valid: user, age, data, ...
Invalid: let, const, if, ...

#4

Should use camelCasing
Recommended: userName, isCorrect, ...
Uncommon: user_name, iscorrect, ...

#5

Should describe what the "thing" it identifies contains or does
Recommended: userName, isCorrect, loadData, ...
Uncommon: userDataPoint, correctness, dataLoader, ...

Semicolons are optional
(in most cases)!

Whitespace is ignored in many cases!

Use it to format your code & improve readability

But avoid adding too much whitespace

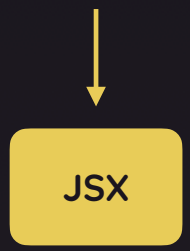
React projects use a build process

The code **you write** is **not** the code that gets **executed** (like this) in the browser

Your code is **transformed** before it's handed off to the browser

React Projects Use A Build Process

1. Raw, unprocessed React code **won't execute** in the browser



JSX is not a default JavaScript feature

2. In addition, the code would **not be optimized for production** (e.g., not minified)

React projects require a build process that transforms your code

create-react-app, vite etc. give you such a build process (no custom setup or tweaking needed)

It’s All About Data & Values!

Your tweet is data

The loaded tweets in the feed are data

...

Your location is data

The calculated route is data

...

There Are Different Types Of Values

String

Text values

Wrapped with single or double quotes

Can also be created with backticks (`)

```
"Hello World"

'Max'

`Hi there`
```

Number

Positive or negative

With decimal point (float) or without it (integer)

```
5
-23

3.14
-8.12
```

Boolean

True or false

A simple “Yes” or “No” value type

Typically used in conditions

```
true

false
```

Null & undefined

“There is no value”

undefined: Default if no value was assigned yet

null: Explicitly assigned by developer (reset value)

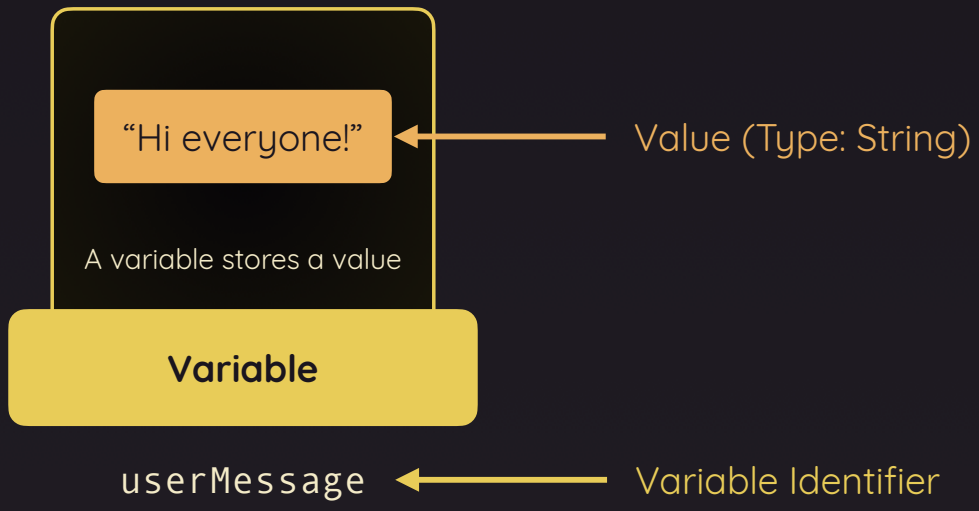
```
undefined

null
```

Additionally → **Objects**

Variables store Values

Variables Are Data Containers



Why Use Variables?

1

Reusability

Store a value in a variable once and use it as often and in as many places as needed

2

Readability

Organize your code over several lines rather than cramming everything into a single line

Variables vs Constants

Variables

Defined via `let`

Can be **re-assigned**
(i.e., the stored value can be overwritten)

```
let age = 34;
```

Allowed → `age = 29;` ✓

Constants

Defined via `const`

Cannot be **re-assigned**
(i.e., the stored value can't be overwritten)

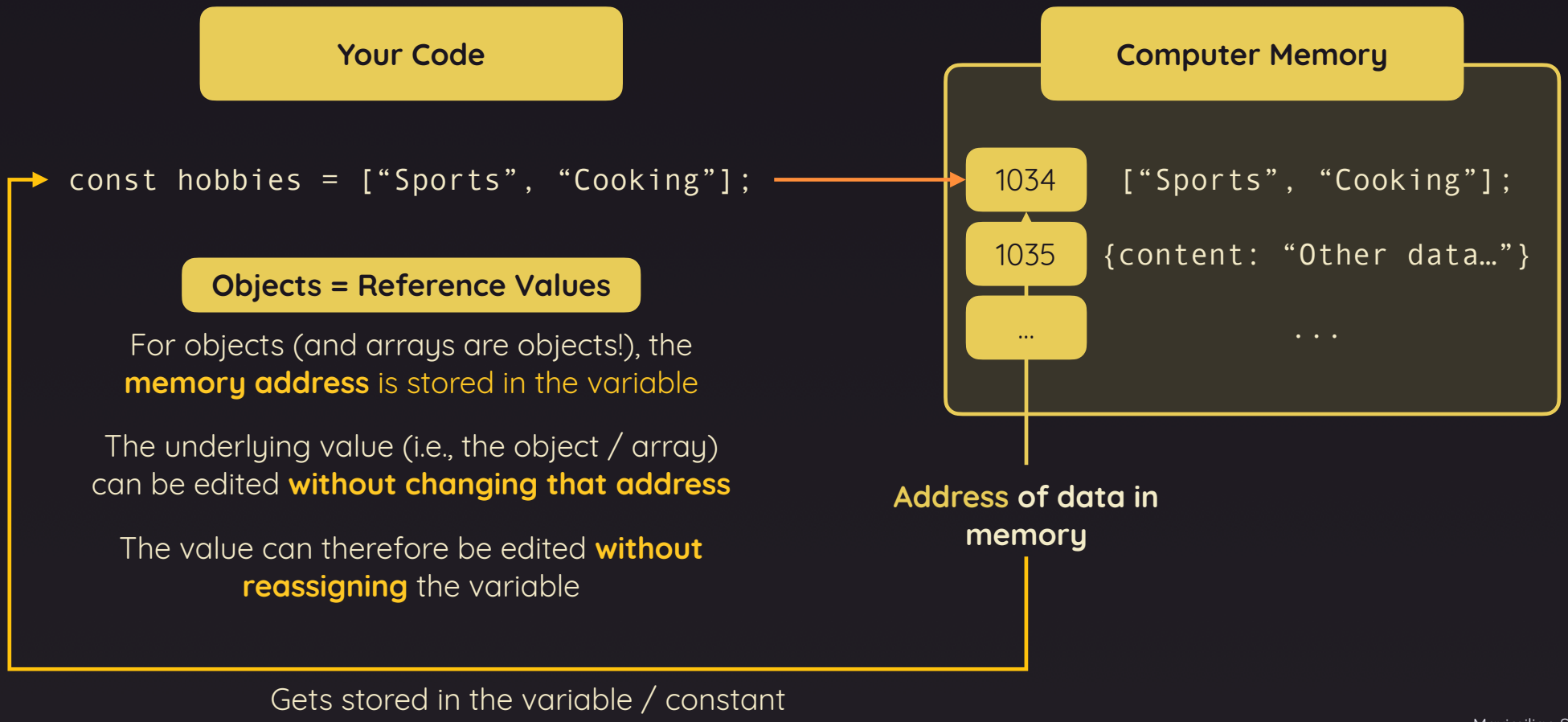
```
const age = 34;
```

✗ `age = 29;` ← Error

Values can be hardcoded

But they can also be derived via Expressions & Operators

Reference Values



Objects = Reference Values

For objects (and arrays are objects!), the **memory address** is stored in the variable

The underlying value (i.e., the object / array) can be edited **without changing that address**

The value can therefore be edited **without reassigning** the variable