JONAS.IO
SCHMEDTMANN

Subscribe here

THE COMPLETE JAVASCRIPT COURSE

📖 TABLE OF CONTENTS: THEORY L

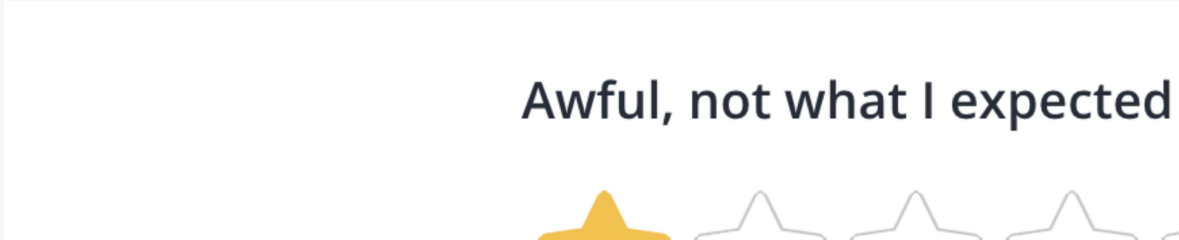
WELCOM

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

SOME QUICK CONSIDERATIONS BEFOR

✌️ **This course is for all of you!** So please don't w
progressing too slow, or too fast for you. To m
watch the course with slower or faster playba



SOME QUICK CONSIDERATIONS BEFOR

👩‍💻 **You need to code along with me!** You will lear

code **YOURSELF**!

SOME QUICK CONSIDERATIONS BEFOR

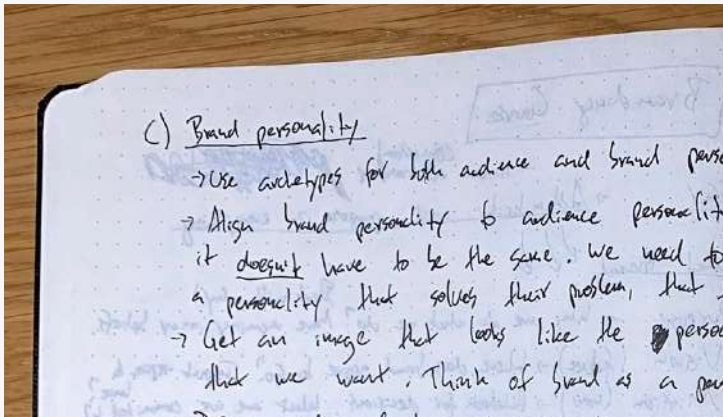
🤓 **Try all the coding challenges!** Try to do your b

up if you can't figure it out! Just rewatch the l

and move on.

SOME QUICK CONSIDERATIONS BEFOR

If you want the course material to stick, take



SOME QUICK CONSIDERATIONS BEFOR

😱 **If this is your first time ever programming, ple**

everything at the beginning. ***Just don't think "***

SOME QUICK CONSIDERATIONS BEFOR

😅 **In the first sections of the course, don't bothe**

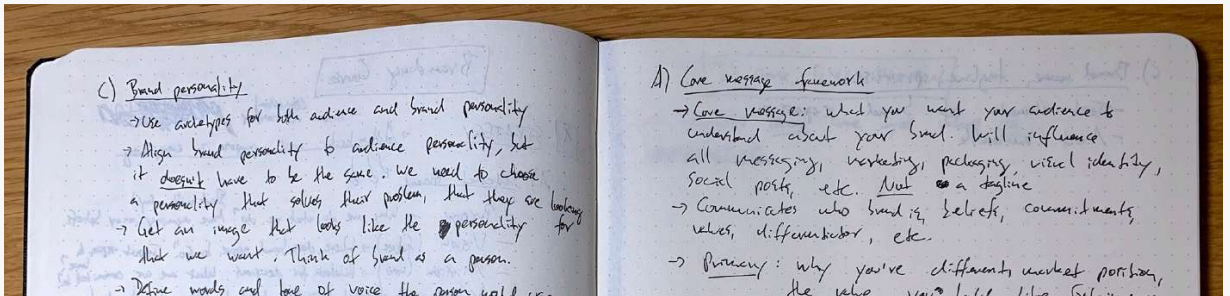
stress about efficient code, or fast code, or cl

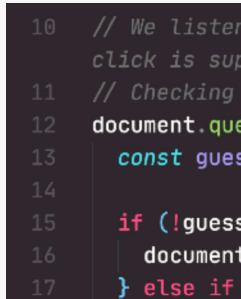
understand the **WHY** later in the course.

SOME QUICK CONSIDERATIONS BEFOR

⏳ **Before moving on from a section, make sure t**

we wrote, review your notes, review the projec

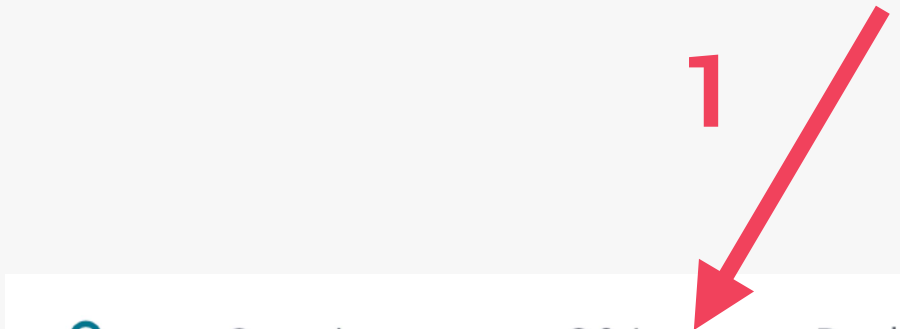




SOME QUICK CONSIDERATIONS BEFOR

!? **If you have an error or a question, start by tryi**

it, check the Q&A section. If that doesn't help,



SOME QUICK CONSIDERATIONS BEFOR

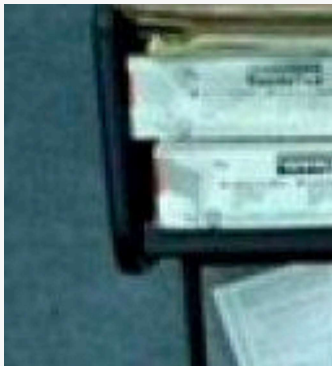
I recorded this course on a Mac, but everythin

work on your computer, it's **NOT** because you'

SOME QUICK CONSIDERATIONS BEFOR

😍 **Most importantly, have fun!** It's so rewarding t

frustrated, stop whatever you're doing, and co



JAVA

FUNDA

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

WHAT IS JAVASCRIPT?

JAVAS

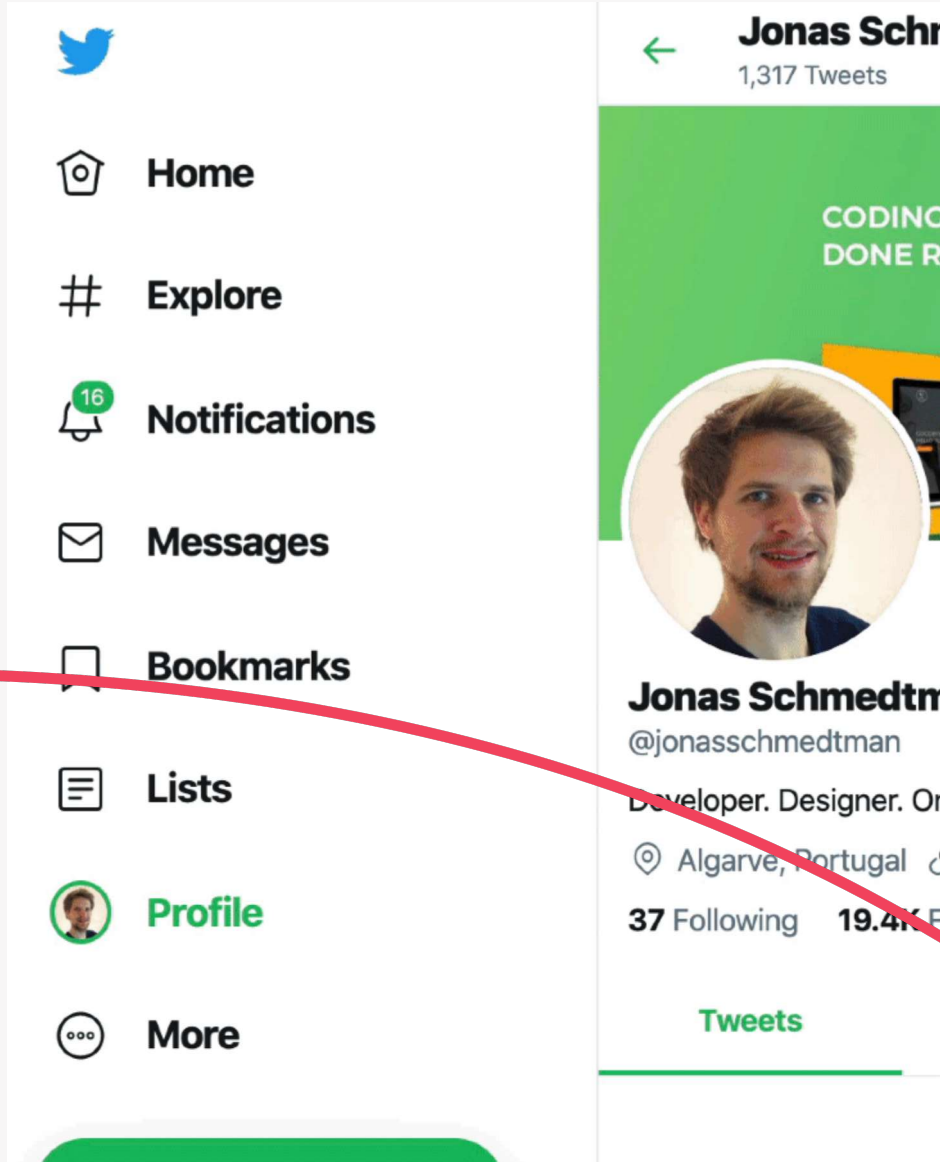
OBJECT-O

Based on objects, for

THE ROLE OF JAVASCRIPT IN WEB

EXAMPLE OF DYNAMIC EFFECTS / V

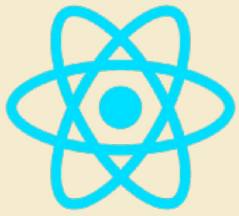
Show spinner + loading data in the background



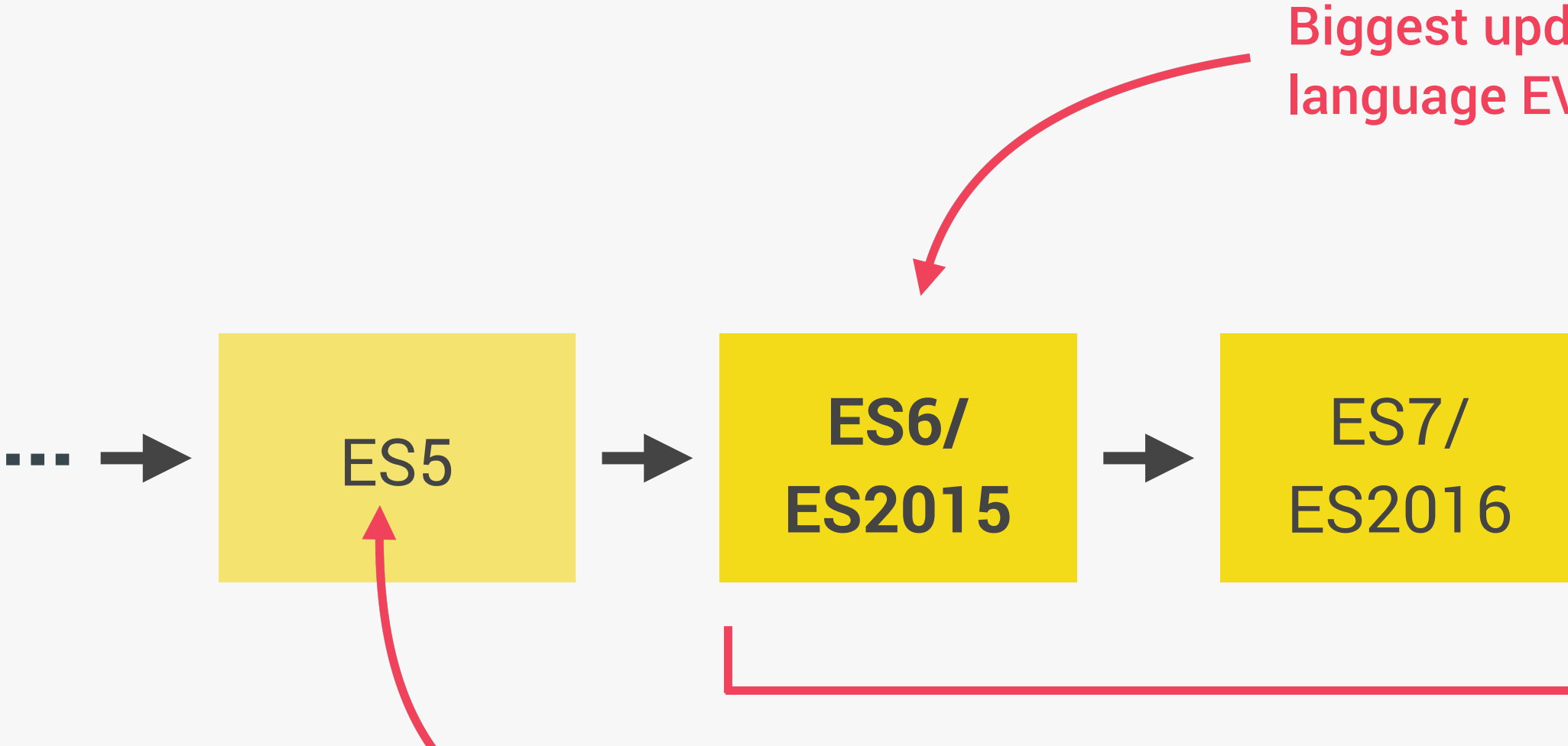
THERE IS NOTHING YOU CAN'T DO W

THIS COURSE

Dynamic effects and web applications in the browser 😍



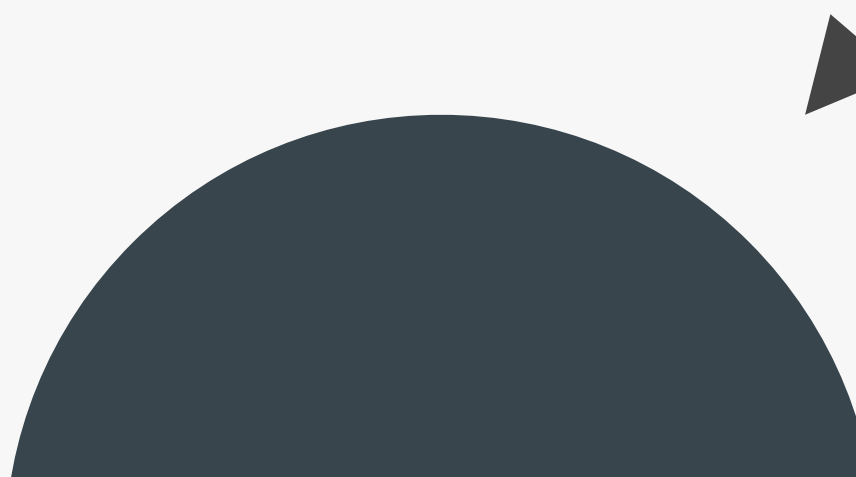
JAVASCRIPT RELEASES... (MORE ABO



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

OBJECTS AND PRIMITIVES



THE 7 PRIMITIVE DATA TYPES

1. **Number:** Floating point numbers 👉
2. **String:** Sequence of characters 👉 U
3. **Boolean:** Logical type that can only be
4. **Undefined:** Value taken by a variable th
5. **Null:** Also means 'empty value'

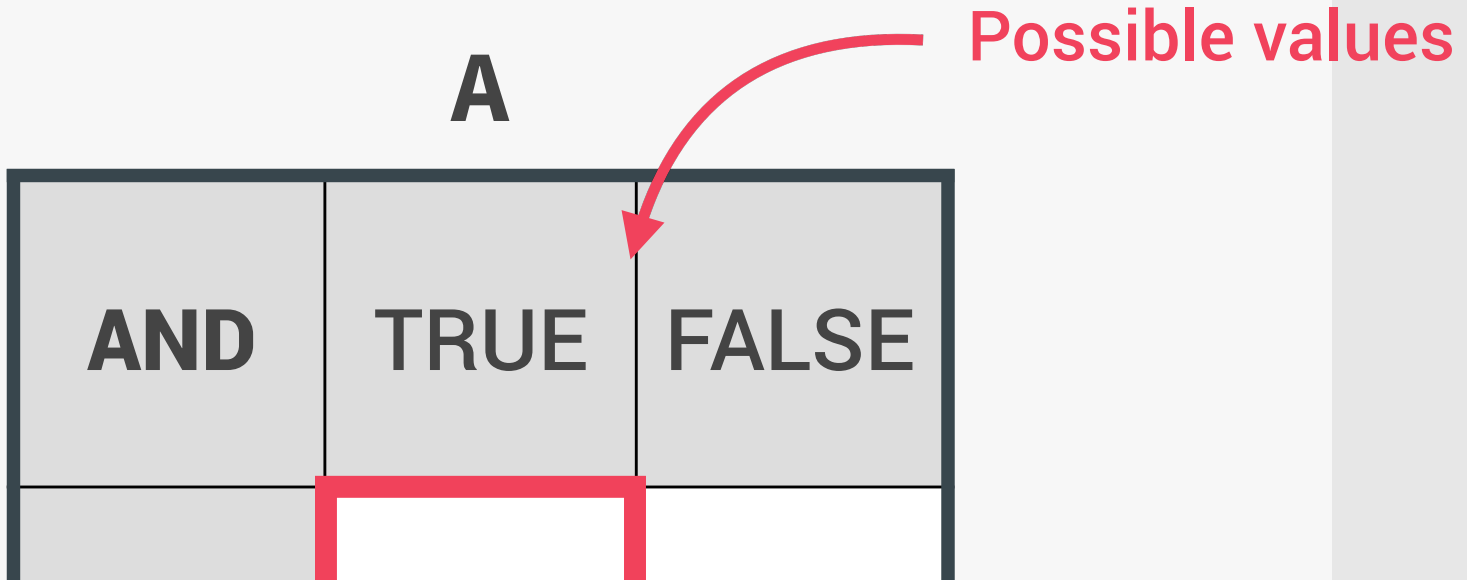
JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BASIC BOOLEAN LOGIC: THE AND, OR

A AND B

*"Sarah has a driver's license **AND** good vision"*



AN EXAMPLE 👩‍💻

BOOLEAN VARIABLES

👉 **A**: Age is greater or equal 20

👉 **B**: Age is less than 30

LET'S USE OPERATORS!

👉 !A `true`

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

A BRIEF HISTORY OF JAVASCRIPT

1995

👉 Brendan Eich creates the **very first**
Mocha, but already had many funda

1996

👉 Mocha changes to LiveScript and th
However, **JavaScript has almost no**

👉 Microsoft launches IE, **copying Jav**

1997

👉 With a need to standardize the lang

BACKWARDS COMPATIBILITY: DON'T

```
// ES1 Code
function add(n) {
  var x = 5 + add.arguments[0];
  return x;
}
```

1997

DO

HOW TO USE MODERN JAVASCRIPT

👩‍💻 **During development:** Simply use the late

🚀 **During production:** Use Babel to transpil
back to ES5 to ensure browser compatib

ES5

👉 Fully supported

👉 Ready to be us

MODERN JAVASCRIPT FROM THE BE

🔥 Learn **modern JavaScrip**

☝️ But, also learn how some

(e.g. `const` & `let` vs `va`

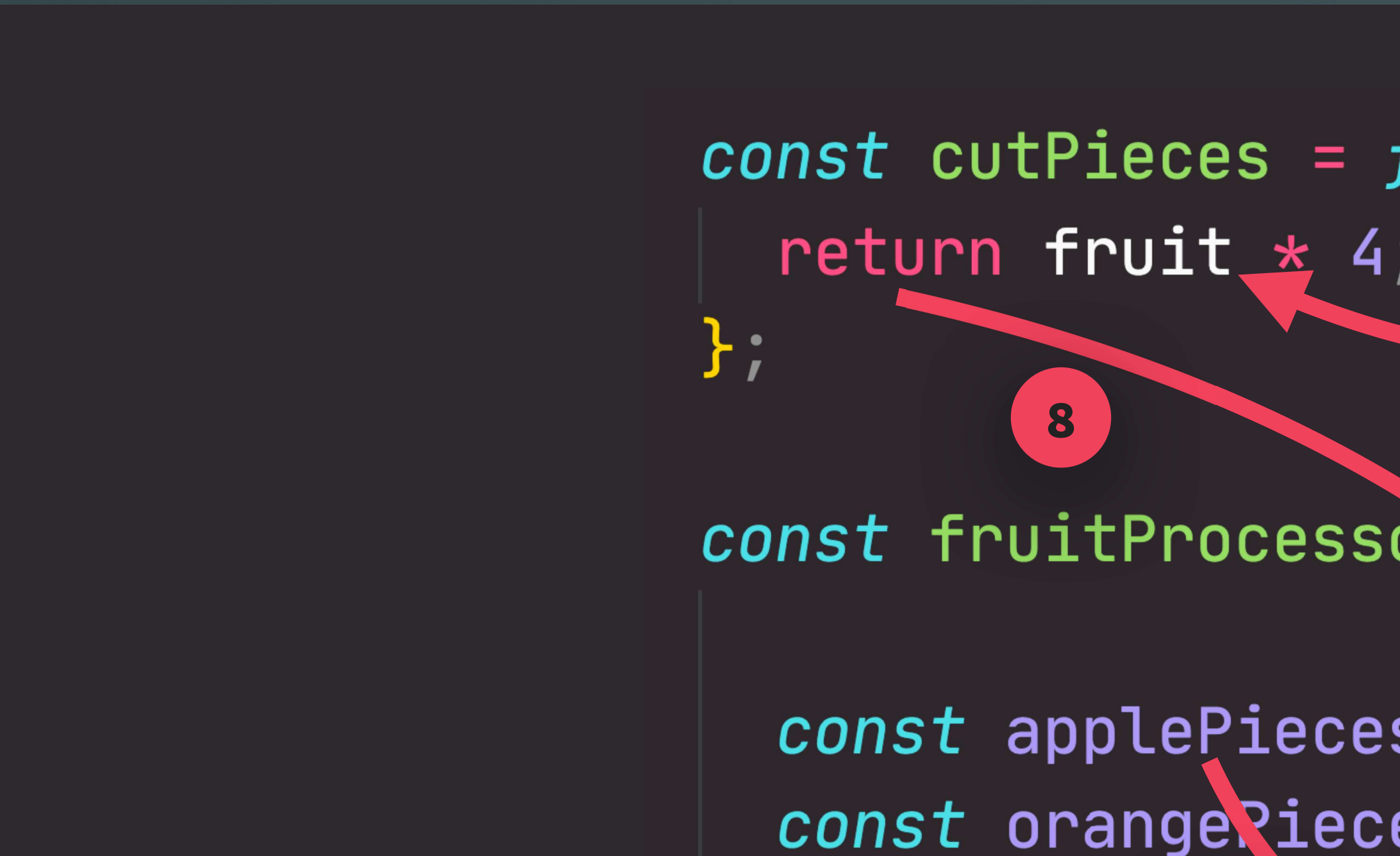
JAVA

FUNDA

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

CALLING A FUNCTION INSIDE A FUNC



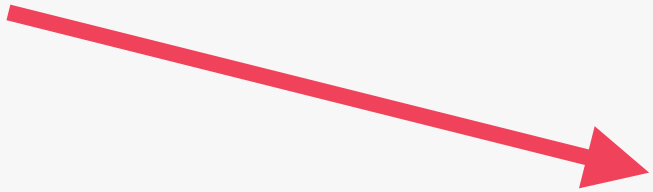
JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

FUNCTIONS REVIEW: 3 DIFFERENT FU

👉 **Function declaration**

Function that can be
used before it's declared



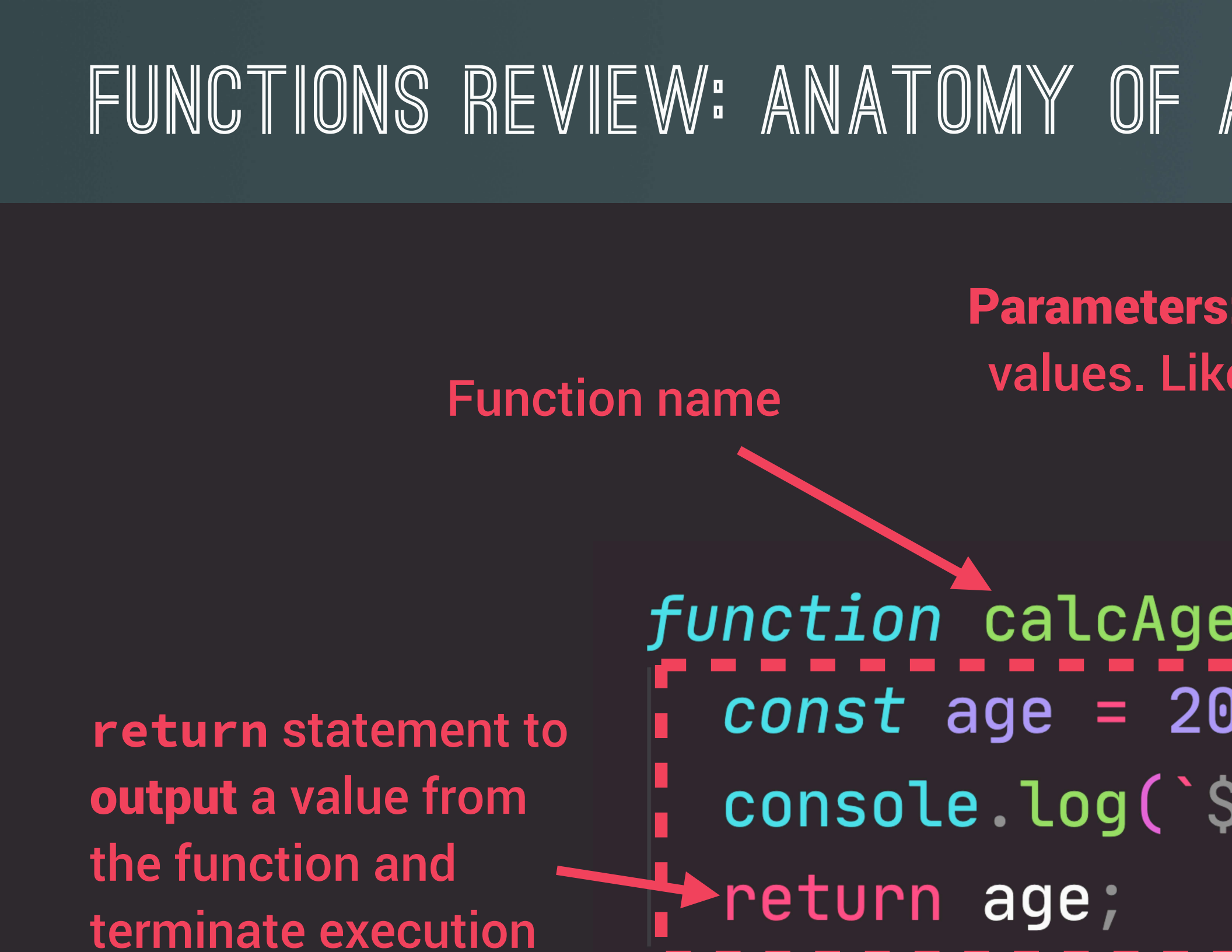
👉 **Function expression**

Essentially a function

FUNCTIONS REVIEW: ANATOMY OF A

Parameters
values. Like

Function name



`return` statement to **output** a value from the function and terminate execution

DEVELOP

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

HOW TO FAIL 🤦 AT LEARNING HOW

- He **di**
- He st
  code
- He **di**
- He **di**

HOW TO SUCCEED 🎉 AT LEARNING

💥 He **didn't have a clear goal** at the beginning of his journey

💥

⬇ **FIX**

👍 Set a **specific**, **measurable**, **realistic** and **time-based** goal

👍

👍 Know exactly **why** you are

HOW TO SUCCEED 🎉 AT LEARNING

💥 He **didn't practice coding**, and didn't come up with his own project ideas

💥

⬇ **FIX**

👍 Practicing on your own is the most important thing to do

👍

HOW TO SUCCEED 🎉 AT LEARNING

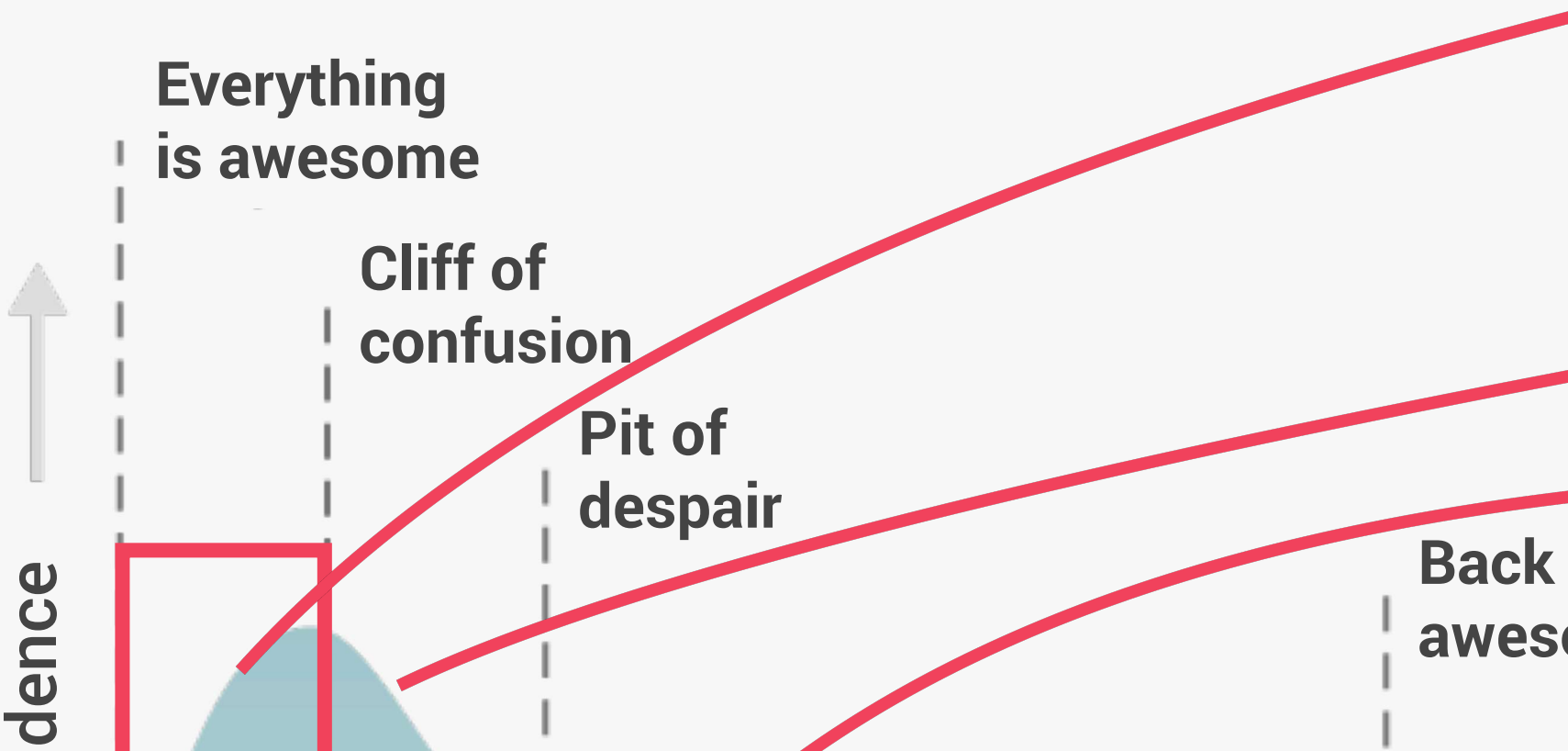
💥 He was **learning in isolat**

⬇ **FIX**

👍 Explain new concepts to
people. If you can explai
you truly understand it!

👍 Share your goals to mak

LEARNING HOW TO CODE IS HARD, B



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

HOW TO FAIL 🤦 AT SOLVING PROB

WHENEV

- 💥 He ju
- 💥 He im
- 💥 He **ge**
- 💥 He is

4 STEPS TO SOLVE ANY PROBLEM

1

Make sure you 100% understand the problem. **Ask the right questions** to get a clear picture of the problem

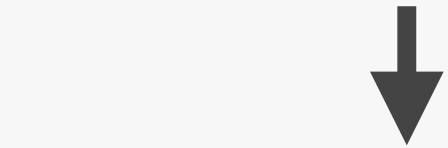
4 STEPS TO SOLVE ANY PROBLEM

1. Make sure you 100% understand the problem. **Ask the right questions** to get a clear picture of the problem

2. **Divide and conquer:** Break a big problem into smaller sub-problems.

4 STEPS TO SOLVE ANY PROBLEM

1. Make sure you 100% understand the problem. **Ask the right questions** to get a clear picture of the problem



2. **Divide and conquer:** Break a big problem into smaller sub-problems.

4 STEPS TO SOLVE ANY PROBLEM

1. Make sure you 100% understand the problem. **Ask the right questions** to get a clear picture of the problem

2. **Divide and conquer:** Break a big problem into smaller sub-problems.

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

WHAT IS A SOFTWARE BUG?

- 👉 **Software bug:** Defect or problem in a co
  Basically, any **unexpected or unintended**
  computer program is a software bug.

- 👉 Bugs are **completely normal** in software

- 👉 **Previous example:** *"We need a function that*

THE DEBUGGING PROCESS

JAVASC

BROWSE

JONAS.IO
SCHMEDTMANN

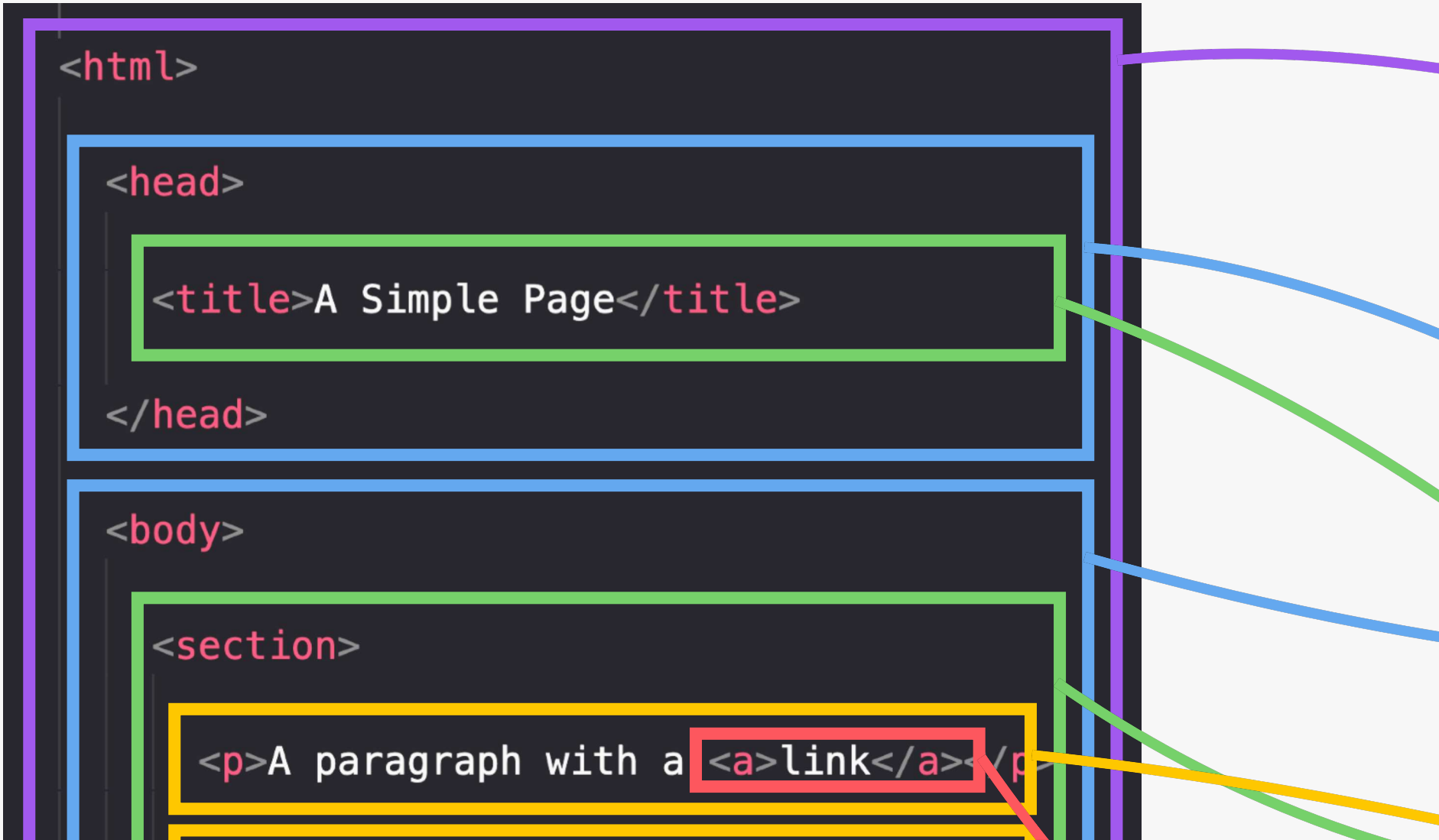
THE COMPLETE JAVASCRIPT COURSE

WHAT IS THE DOM?

DOCUMENT OBJECT MODE

REPRESENTATION OF HTML DO

THE DOM TREE STRUCTURE



DOM !== JAVASCRIPT 🧐

DOM Methods and Properties for DOM Manipulation

For example
`document.querySelector(`

API: Application Progra

HOW JA
WORKS

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

WHAT IS JAVASCRIPT: REVISITED

JAVAS

OBJECT-O

WHAT IS JAVASCRIPT: REVISITED

JAVASCRIPT

JAVASCRIPT IS A HIGH-LEV

MULTI-PARADIGM INTE

DYNAMIC SINGLE-THREAD

DECONSTRUCTING THE MONSTER DE

High-level

Garbage-collected

Interpreted or just-in-time compiled

Multi-paradigm

Prototype-based object-oriented

DECONSTRUCTING THE MONSTER DE

High-level

Garbage-collected

Interpreted or just-in-time compiled

Multi-paradigm

Prototype-based object-oriented

DECONSTRUCTING THE MONSTER DE

High-level

Garbage-collected

Interpreted or just-in-time compiled

Multi-paradigm

Prototype-based object-oriented

DECONSTRUCTING THE MONSTER DE

High-level

Garbage-collected

Interpreted or just-in-time compiled

Multi-paradigm

Prototype-based object-oriented

DECONSTRUCTING THE MONSTER DEF

High-level

Garbage-collected

Interpreted or just-in-time compiled

Multi-paradigm

Prototype-based object-oriented

👉 **Para**

direc

1

2

DECONSTRUCTING THE MONSTER DE

High-level

Garbage-collected

Interpreted or just-in-time compiled

Multi-paradigm

Prototype-based object-oriented

Our array
inherits me
from protot

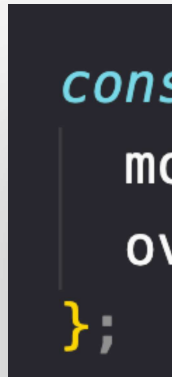
DECONSTRUCTING THE MONSTER DEF

- High-level
- Garbage-collected
- Interpreted or just-in-time compiled
- Multi-paradigm
- Prototype-based object-oriented

👉 In a la

as va

from



DECONSTRUCTING THE MONSTER DE

High-level

Garbage-collected

Interpreted or just-in-time compiled

Multi-paradigm

Prototype-based object-oriented

No
bec

DECONSTRUCTING THE MONSTER DEF

High-level

Garbage-collected

Interpreted or just-in-time compiled

Multi-paradigm

Prototype-based object-oriented

👉 **Co**
ha

👉 Ja

👉 Sc

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

WHAT IS A JAVASCRIPT ENGINE?

JS ENGINE

PROGRAM THAT **EXECUTES**

JAVASCRIPT CODE.

COMPUTER SCIENCE SIDENOTE: COMP

👉 **Compilation:** Entire code is converted into machine

Source code

STEP 1

COMPILATION

👉 **Interpretation:** Interpreter runs through the source c

MODERN JUST-IN-TIME COMPILATIO

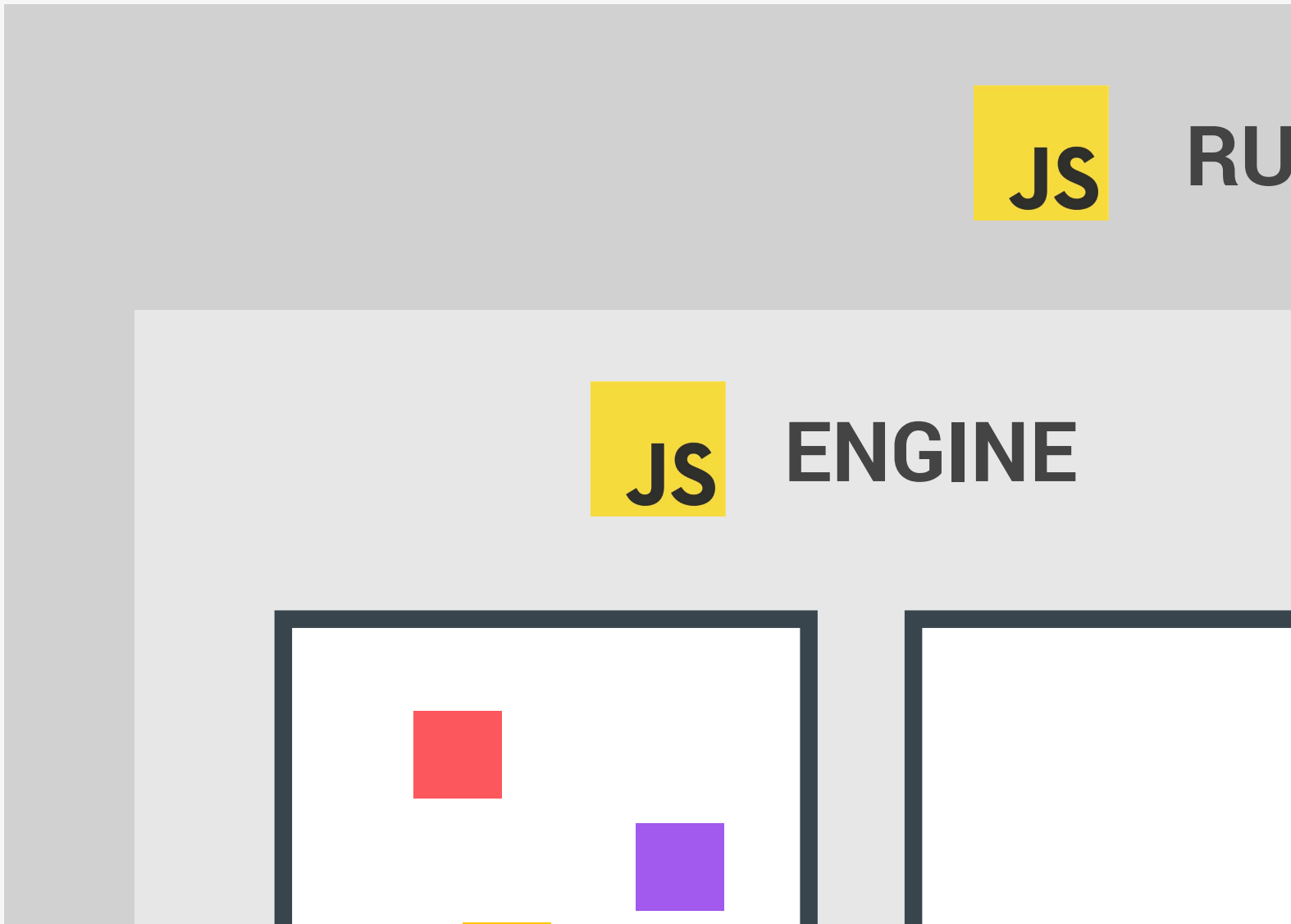
```
document.querySelector(".again").
  document.querySelector(".messag
  document.querySelector(".number
  document.querySelector(".guess"
});
```

JS ENGINE

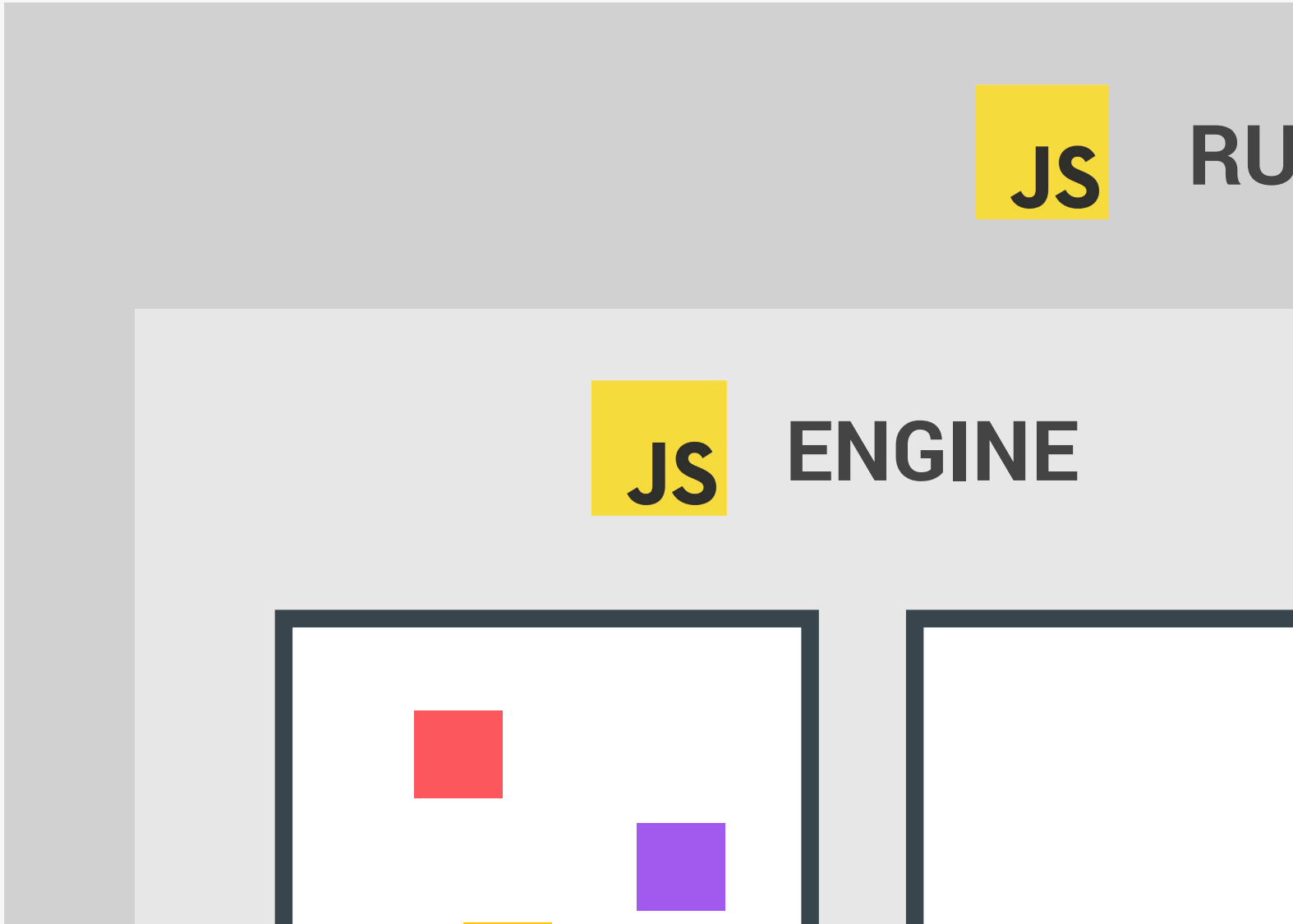
Happens in special threads that we can't access from code

Par

THE BIGGER PICTURE: JAVASCRIPT



THE BIGGER PICTURE: JAVASCRIPT



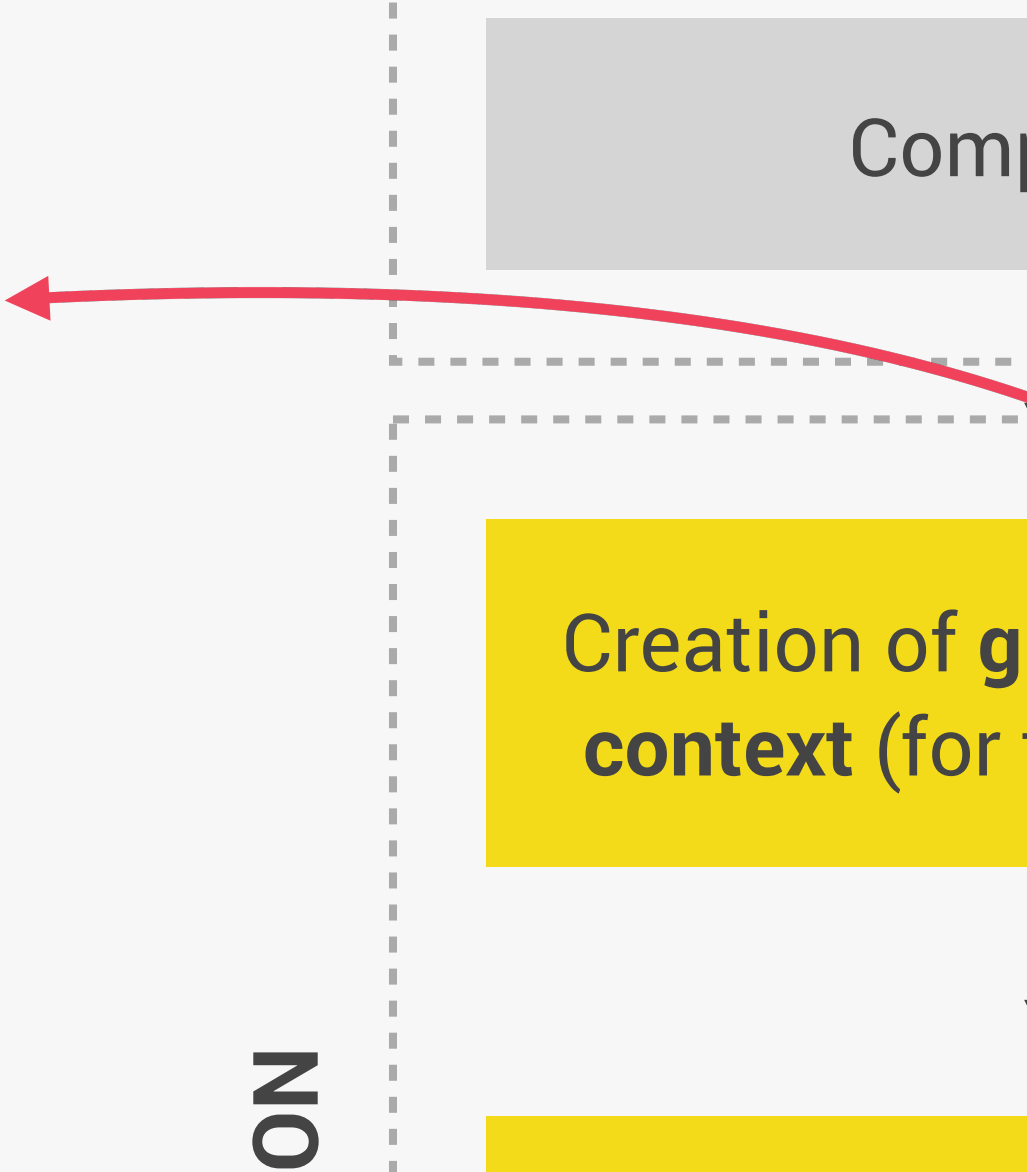
JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

WHAT IS AN EXECUTION CONTEXT?

👉 Human-readable code:

```
const name = 'Jonas';

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



EXECUTION CONTEXT IN DETAIL

WHAT'S INSIDE EXECUTION CONTEXT?

1. **Variable Environment**
   - 👉 `let`, `const` and `var` declarations
   - 👉 Functions
   - 👉 ~~`arguments`~~ object

THE CALL STACK

👉 Compiled code starts execution

```
const name = 'Jonas';

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

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

SCOPING AND SCOPE IN JAVASCRIPT

SCOPE CONCEPTS

- 👉 **Scoping:** How our program's va
  live?" or *"Where can we access a*

- 👉 **Lexical scoping:** Scoping is co

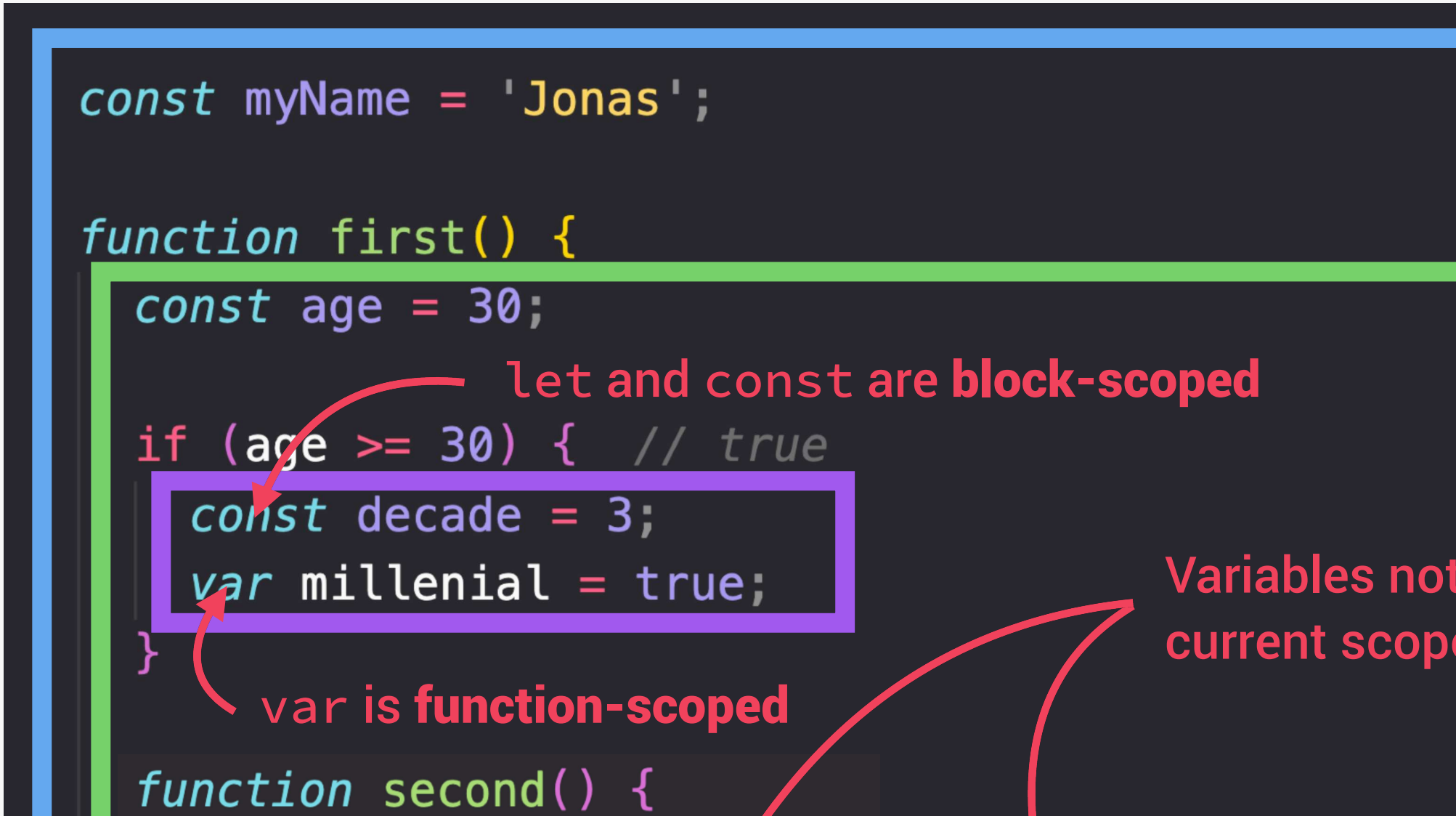
THE 3 TYPES OF SCOPE

GLOBAL SCOPE

```
const me = 'Jonas';
const job = 'teacher';
const year = 1989;
```

```
functio
  const
  const
  retur
}

console
```

THE SCOPE CHAIN



SCOPE CHAIN VS. CALL STACK

```
const a = 'Jonas';
first();

function first() {
  const b = 'Hello!';
  second();

  function second() {
    const c = 'Hi!';
    third();
  }
}

function third() {
  const d = 'Hey!';
}
```

SUMMARY 🥳

- 👉 Scoping asks the question *“Where do variables*
- 👉 There are 3 types of scope in JavaScript: the g
- 👉 Only `let` and `const` variables are block-scop
- 👉 In JavaScript, we have lexical scoping, so the code functions and blocks are written;
- 👉 Every scope always has access to all the varia

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

HOISTING IN JAVASCRIPT

👉 **Hoisting:** Makes some types of variables ac
actually declared. “Variables lifted to the top

BEHIND THE SCENES

Before execution, code is scanned for varia
property is created in the **variable environm**

HOIST

👇

TEMPORAL DEAD ZONE, LET AND CO

```
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);
}
```

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

HOW THE THIS KEYWORD WORKS

- 👉 **`this` keyword/variable:** Special variable that is cr
  Takes the value of (points to) the "owner" of the fu

- ☝️ `this` is **NOT** static. It depends on **how** the functio
  function **is actually called**.

Method 👉 `this` = <Object that is callin

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

JONAS.IO
SCHMEDTMANN

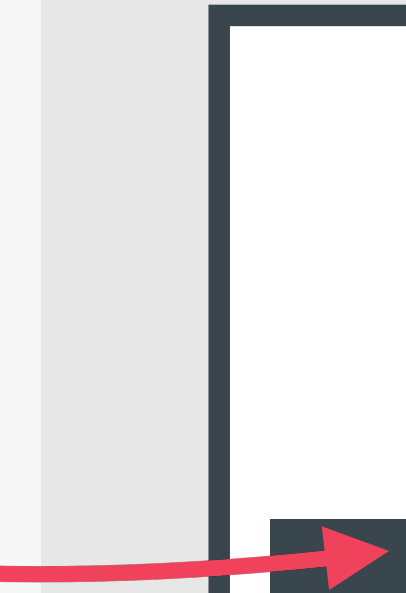
THE COMPLETE JAVASCRIPT COURSE

REVIEW: PRIMITIVES, OBJECTS AND

PRIMITIVES

- 👉 Number
- 👉 String
- 👉 Boolean
- 👉 Undefined

STORED IN



PRIMITIVE VS. REFERENCE VALUES

👉 **Primitive** values example:

```
let age = 30;
let oldAge = age;
age = 31;
console.log(age); // 31
console.log(oldAge); // 30
```

👉 **Reference** values example:

```
const me = {
```

"HOW JAVASCRIPT WORKS BEHIND

1 **Prototypal Inheritance** 👉

2 **Event Loop** 👉 Asynchro

DATA ST
MODERN

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

DATA STRUCTURES OVERVIEW

SOURCES OF DATA

1. **From the program itself:** Data written directl
2. **From the UI:** Data input from the user or data
3. **From external sources:** Data fetched for exa

ARRAYS VS. SETS AND OBJECTS VS

ARRAYS

VS.

SETS

```
tasks = ['Code', 'Eat', 'Code'];
// ["Code", "Eat", "Code"]
```

```
tasks = new Set(['Code', 'Ea
// {"Code", "Eat"}
```

- 👉 Use when you need **ordered** list of values (might contain duplicates)

- 👉 Use when you nee with **unique** values
- 👉 Use when **high-per**

A CLOSE

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

FIRST-CLASS VS. HIGHER-ORDER FU

FIRST-CLASS FUNCTIONS

- 👉 JavaScript treats functions as **first-class citizens**
- 👉 This means that functions are **simply values**
- 👉 Functions are just another **“type” of object**

👉 **Store functions in variables or properties:**

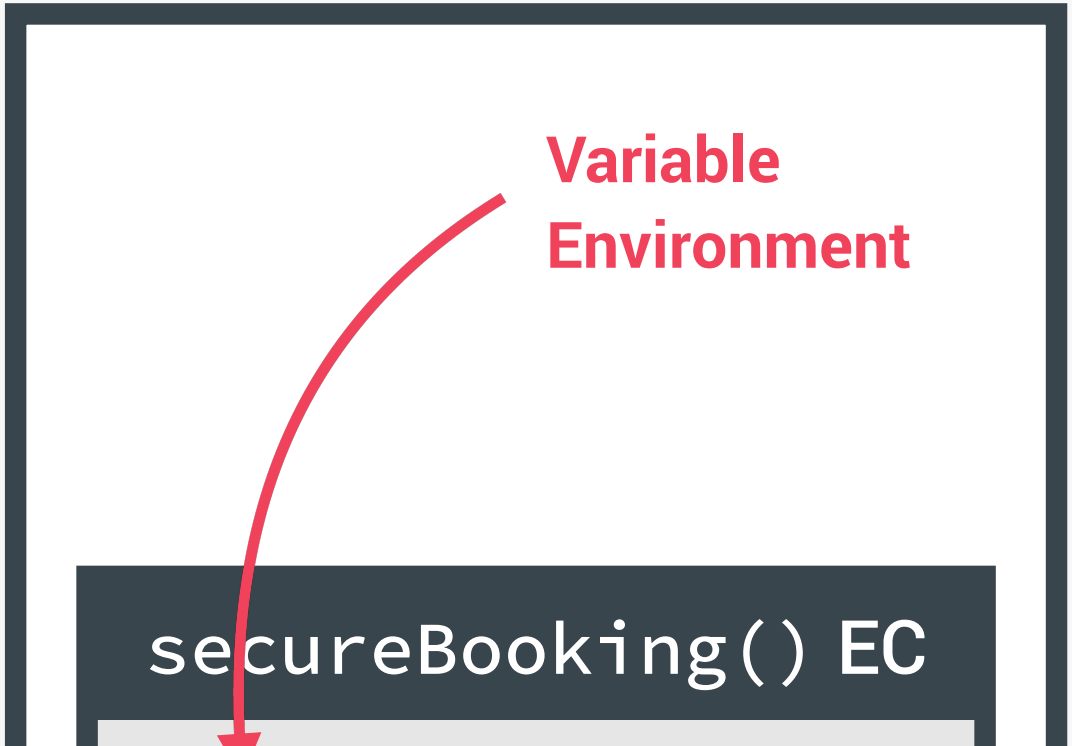
```
const add = (a, b) => a + b;
```

```
const counter = {
```

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

"CREATING" A CLOSURE




```
const secur
  let passe

  return fu
    passeng
    console
```

UNDERSTANDING CLOSURES

```
secureBooking()

  passengerCount =
```

```
const secur
  let passe

  return fu
    passeng
```

UNDERSTANDING CLOSURES

- 👉 A function has access to the variable environment (VE) o
- 👉 **Closure:** VE attached to the function, exactly as it was at

```
const secur
  let passe

  return fu
    passeng
```

CLOSURES SUMMARY 🥳

👉 A closure is the closed-over **variable environment** execution context is gone;

↓ Less formal

👉 A closure gives a function access to all the variab function keeps a **reference** to its outer scope, whi

↓ Less formal

👉 A closure makes sure that a function doesn't loos

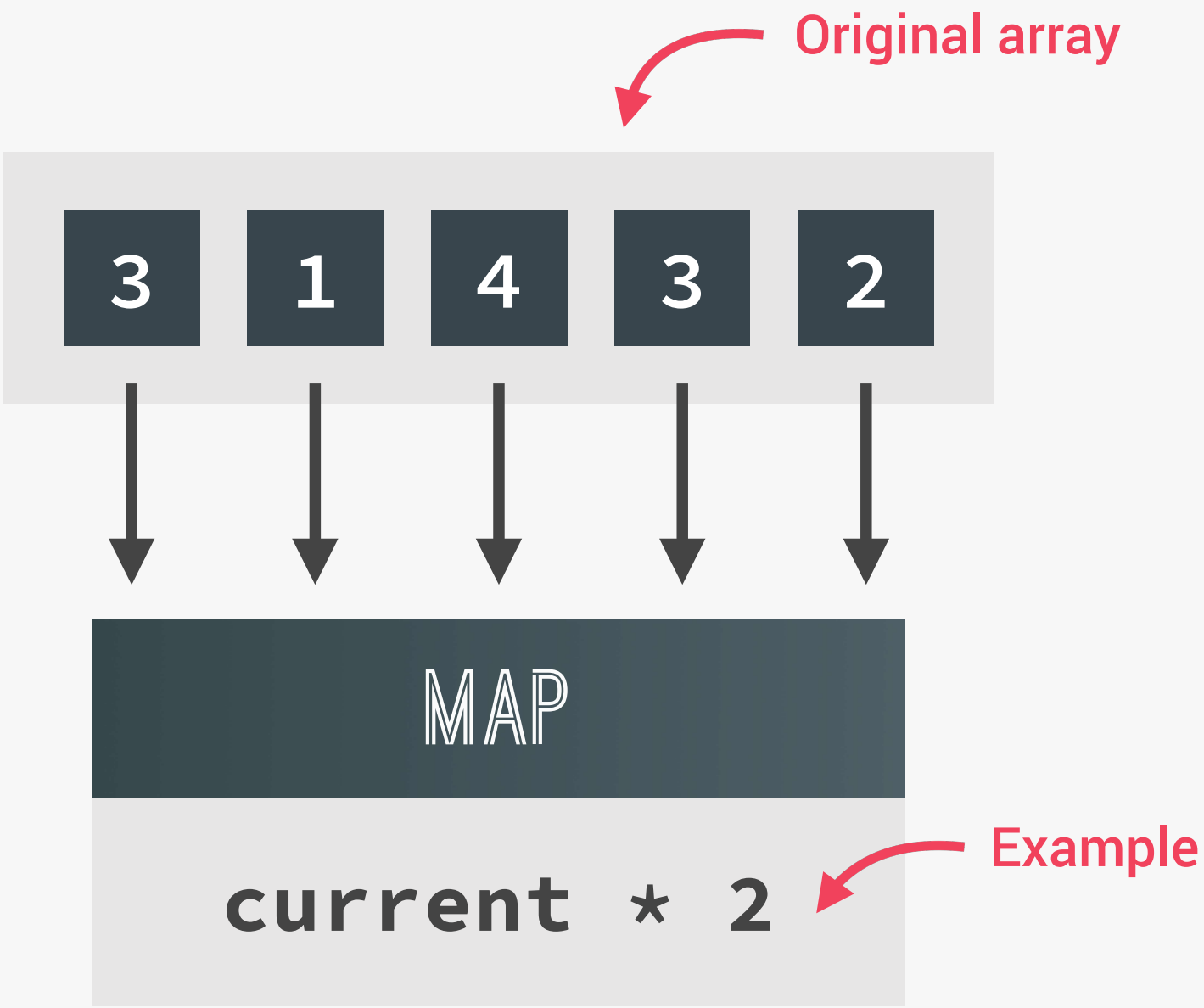
↓ Less formal

WORK

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

DATA TRANSFORMATIONS WITH MA



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

WHICH ARRAY METHOD TO USE? 🤔

To mutate original array

👉 Add to original:

- `.push` *(end)*
- `.unshift` *(start)*

👉 Remove from original:

- `.pop` *(end)*
- `.shift` *(start)*

A new array

👉 Computed from original:

- `.map` *(loop)*

👉 Filtered using condition:

- `.filter`

👉 Portion of original:

- `.slice`

ADVANC

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

REVIEW: WHAT IS THE DOM?

- 👉 Allows us to make J
- 👉 We can write JavaS

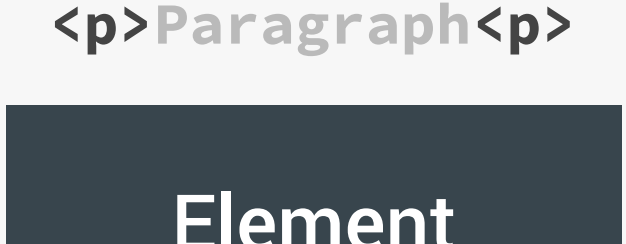
HOW THE DOM API IS ORGANIZED BE

MDN web docs
moz://a

Represented by
JavaScript object

```
.innerHTML
.classList
.children
.parentElement
.append()
.remove()
.insertAdjacentHTML()
.querySelector()
.closest()
```

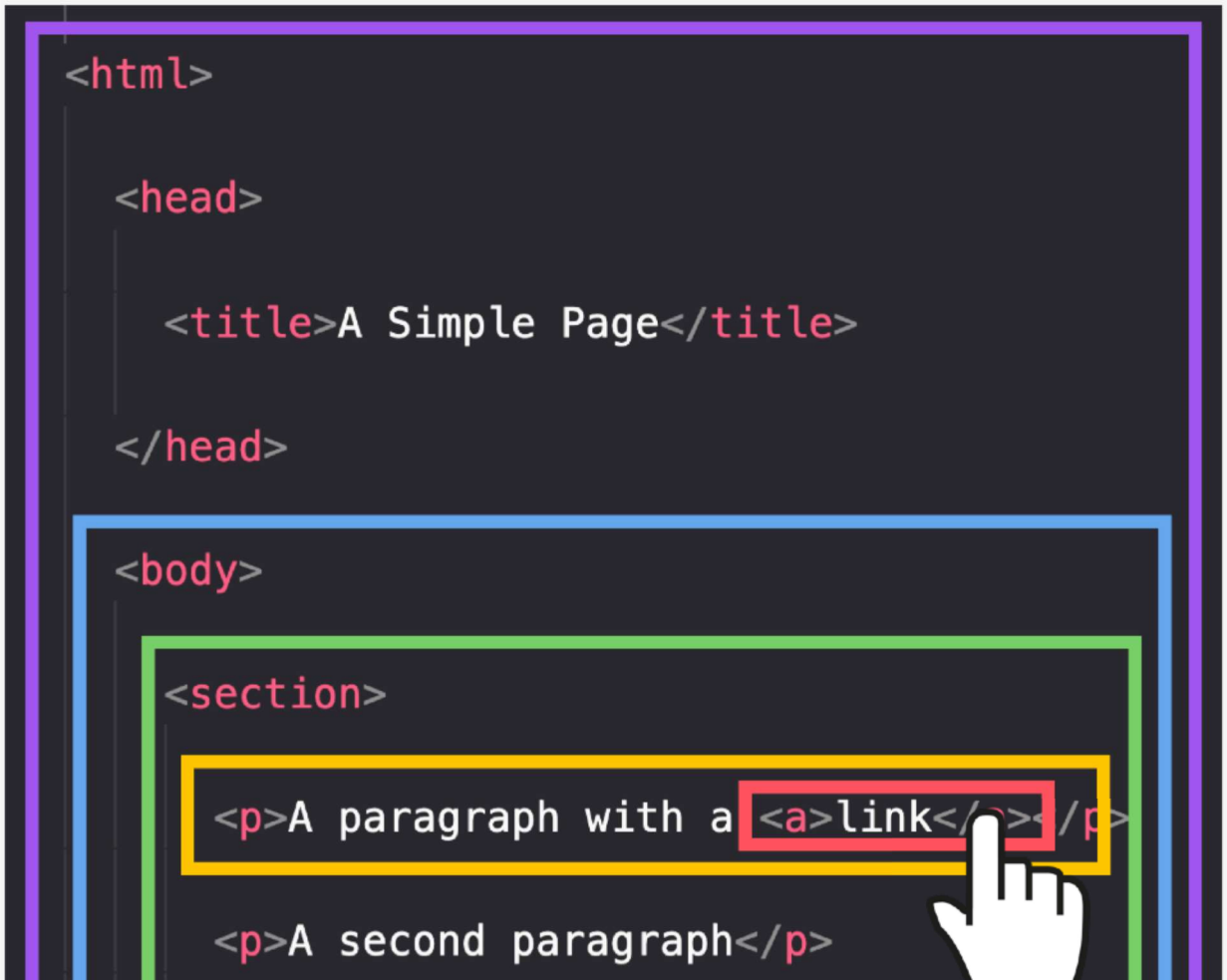
```
.textContent
.childNodes
.parentNode
.cloneNode()
```



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUBBLING AND CAPTURING



1

CAPTURING PHASE

JONAS.IO
SCHMEDTMANN

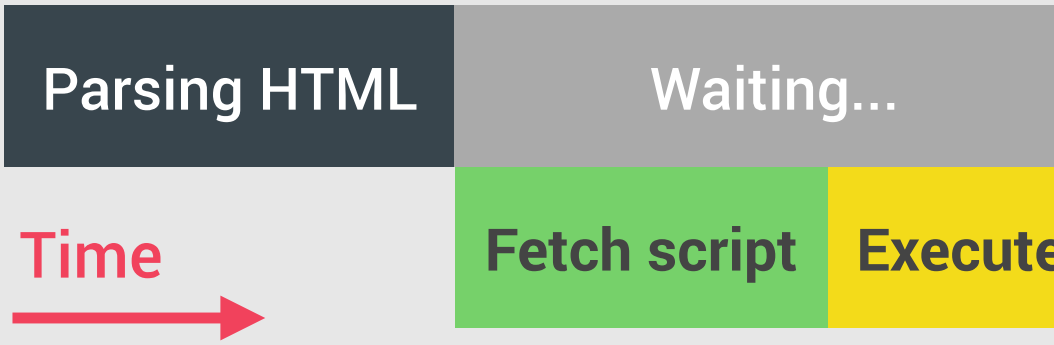
THE COMPLETE JAVASCRIPT COURSE

DEFER AND ASYNC SCRIPT LOADING

HEAD

REGULAR

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



REGULAR VS. ASYNC VS. DEFER

END OF BODY

- 👉 Scripts are fetched and executed ***after the HTML is completely parsed***
- 👉 **Use if you need to support old browsers**

- 👉 Scri
exe
- 👉 Usu
wai

OBJECT

PROGRAM

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

WHAT IS OBJECT-ORIENTED PROGRA

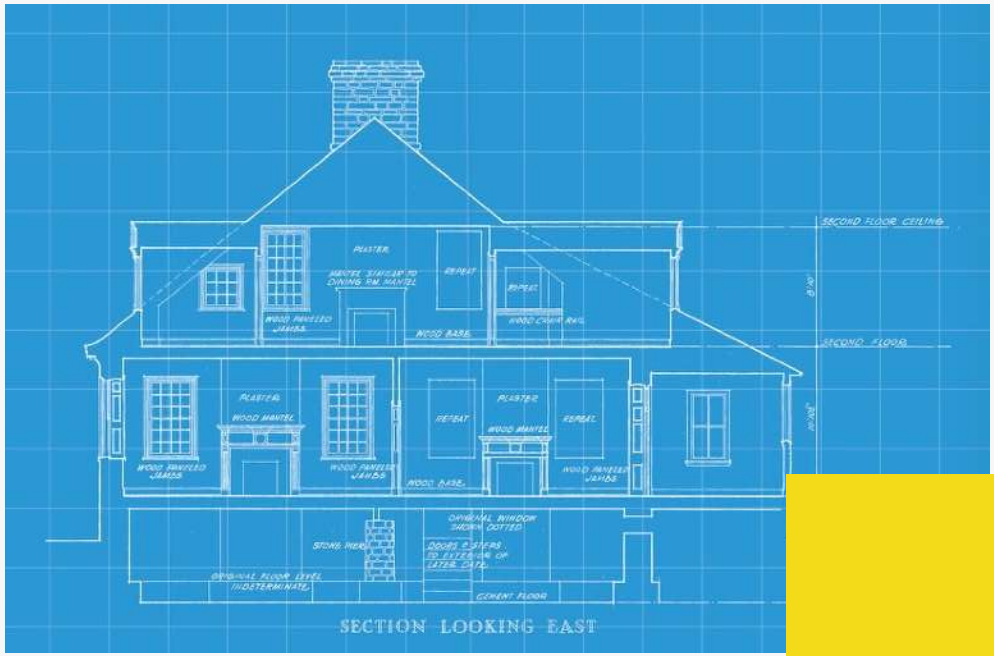
OOP

Data

```
const user = {
  user: 'jonas',
  password: 'dk23s',

  login(password) {
```

- 👉 Object-oriented programm
  concept of objects;
- 👉 We use objects to **model**
- 👉 Objects may contain data
  pack **data and the corresp**

CLASSES AND INSTANCES (TRADITIO



SECTION LOOKING EAST

Like a blueprint
which we can cr
new objects

CLASS

```
User {
```

THE 4 FUNDAMENTAL OOP PRINCIPLE

Abstraction

Encapsulation

Inheritance

The 4 f
princip
Oriente

PRINCIPLE 1: ABSTRACTION

Abstraction

Encapsulation

Inheritance

```
Phone {
  charge
  volume
  voltage
  temperatur

  homeBtn()
  volumeBtn(
  screen() {
  verifyVolt
  verifyTemp
  vibrate()
  soundSpeak
  soundEar()
```

PRINCIPLE 2: ENCAPSULATION

Abstraction

Encapsulation

Inheritance

NOT accessible from
outside the class!

STILL accessible fro
within the class!

STILL accessible fro
within the class!

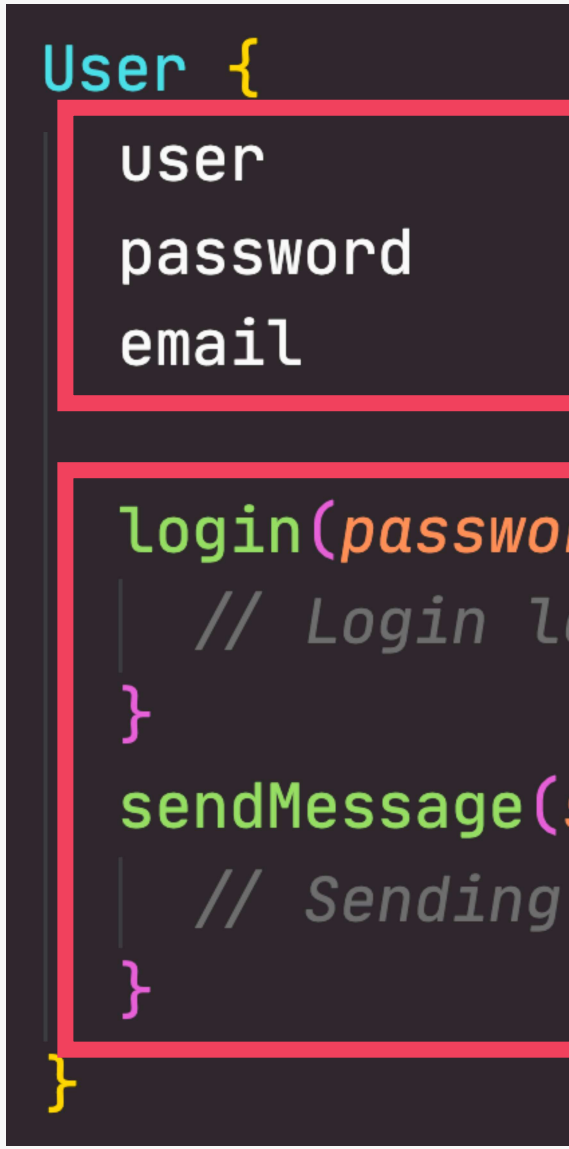
NOT accessible from

PRINCIPLE 3: INHERITANCE

Abstraction

Encapsulation

Inheritance



PRINCIPLE 4: POLYMORPHISM

Abstraction

Encapsulation

Inheritance

INHE

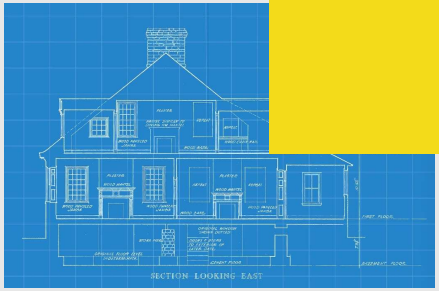
```
Admin {
  user
  password
  email
  permissions

  login(password,
    // DIFFERENT
```

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

OOP IN JAVASCRIPT: PROTOTYPES

"CLASSICAL OOP": CLASSES



Class

INSTANTIATION

Instance

3 WAYS OF IMPLEMENTING PROTOT

🤔 *"How do we actually create prototype*
prototypes? How can we create new obj

1 **Constructor functions**

👉 Technique to create objects from a func

👉 This is how built-in objects like Arrays, M

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

HOW PROTOTYPAL INHERITANCE /

Constructor function

[Person()]

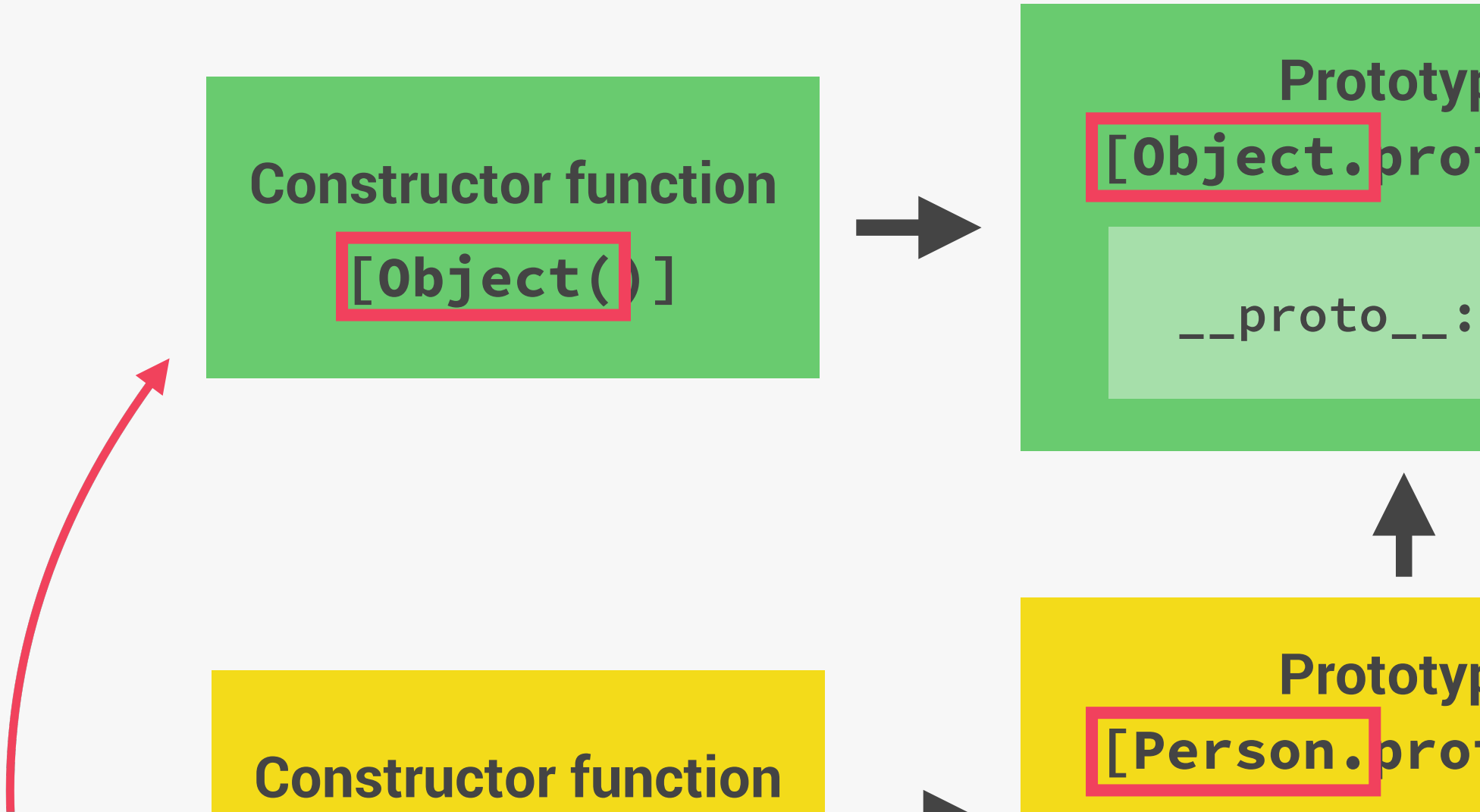
.prototype

.constructo

```
const Person = function(name, birthYear) {
  this.name = name;
  this.birthYear = birthYear;
};
```

PROTOTYPE CHAIN

THE PROTOTYPE CHAIN



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

HOW OBJECT.CREATE WORKS

Prototype

`[PersonProto]`

`calcAge: function`

```
const PersonProto = {
  calcAge() {
    console.log(2037 - t
  },
};
```

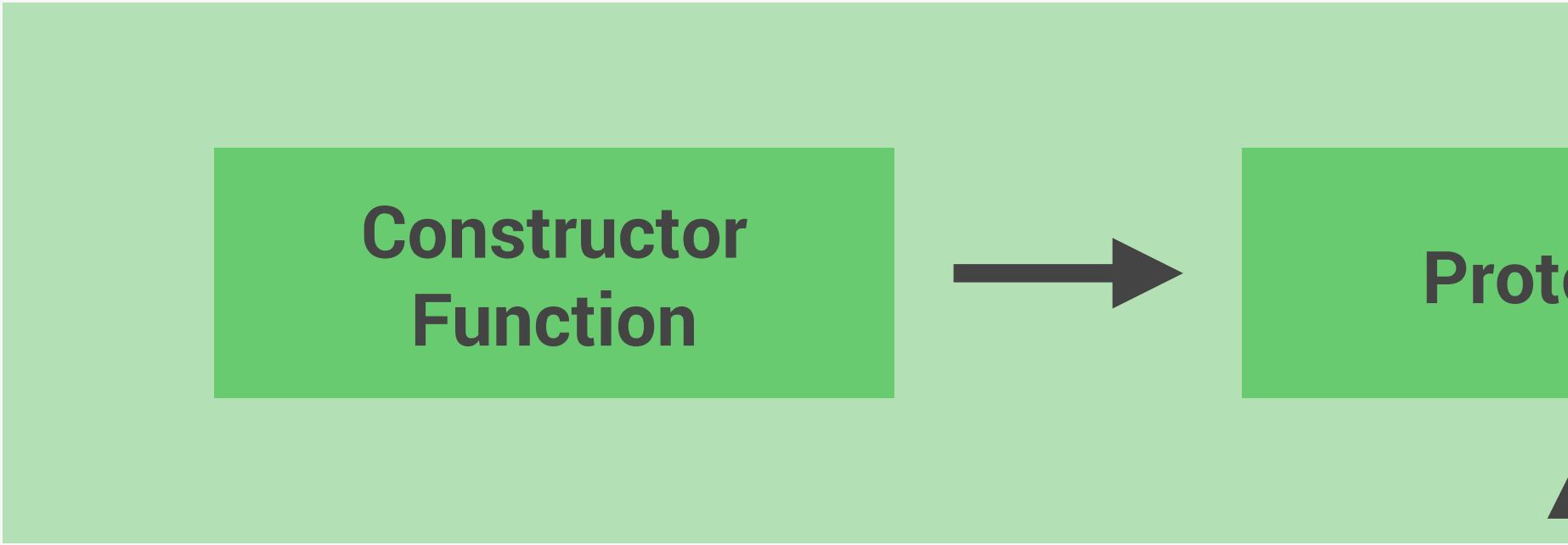
`.__proto__` ↑

```
const steven = Object.create(P
```

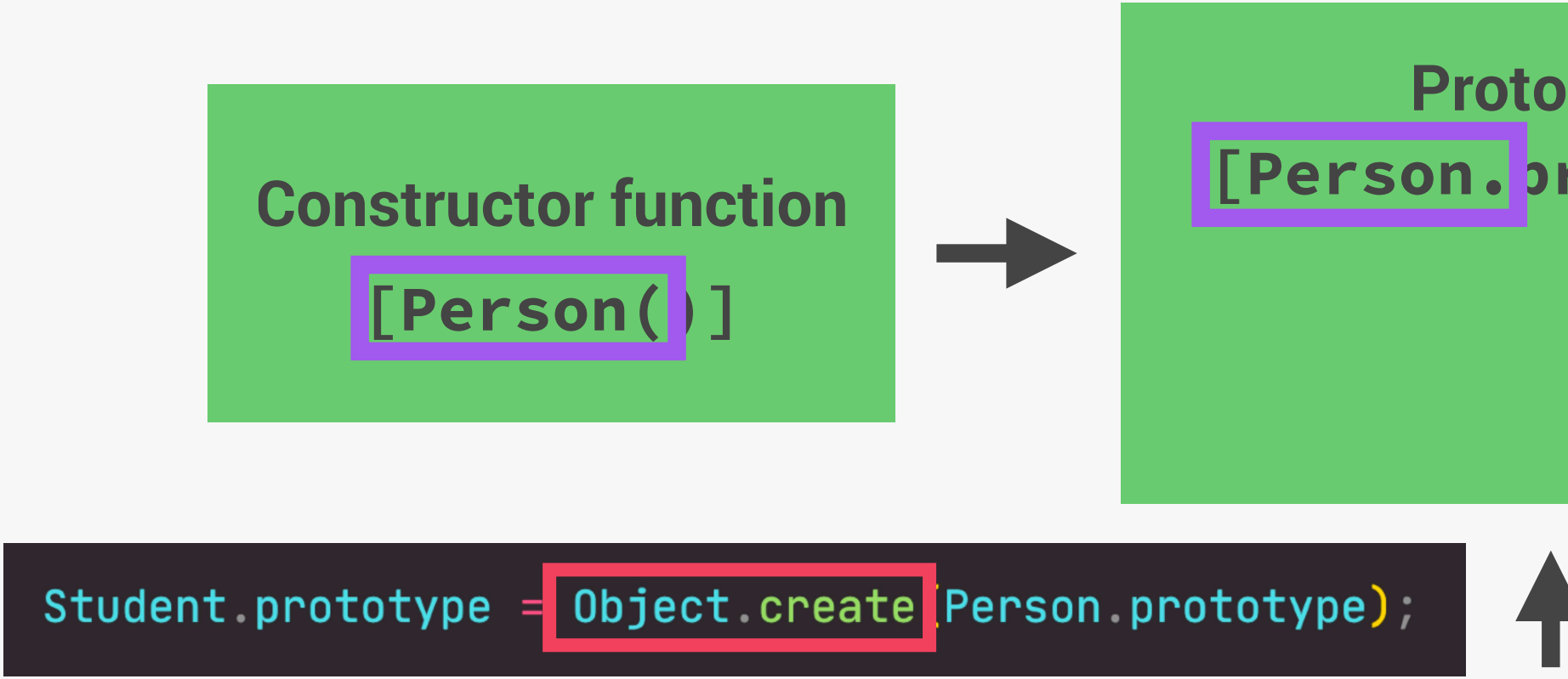
JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

INHERITANCE BETWEEN "CLASSES"

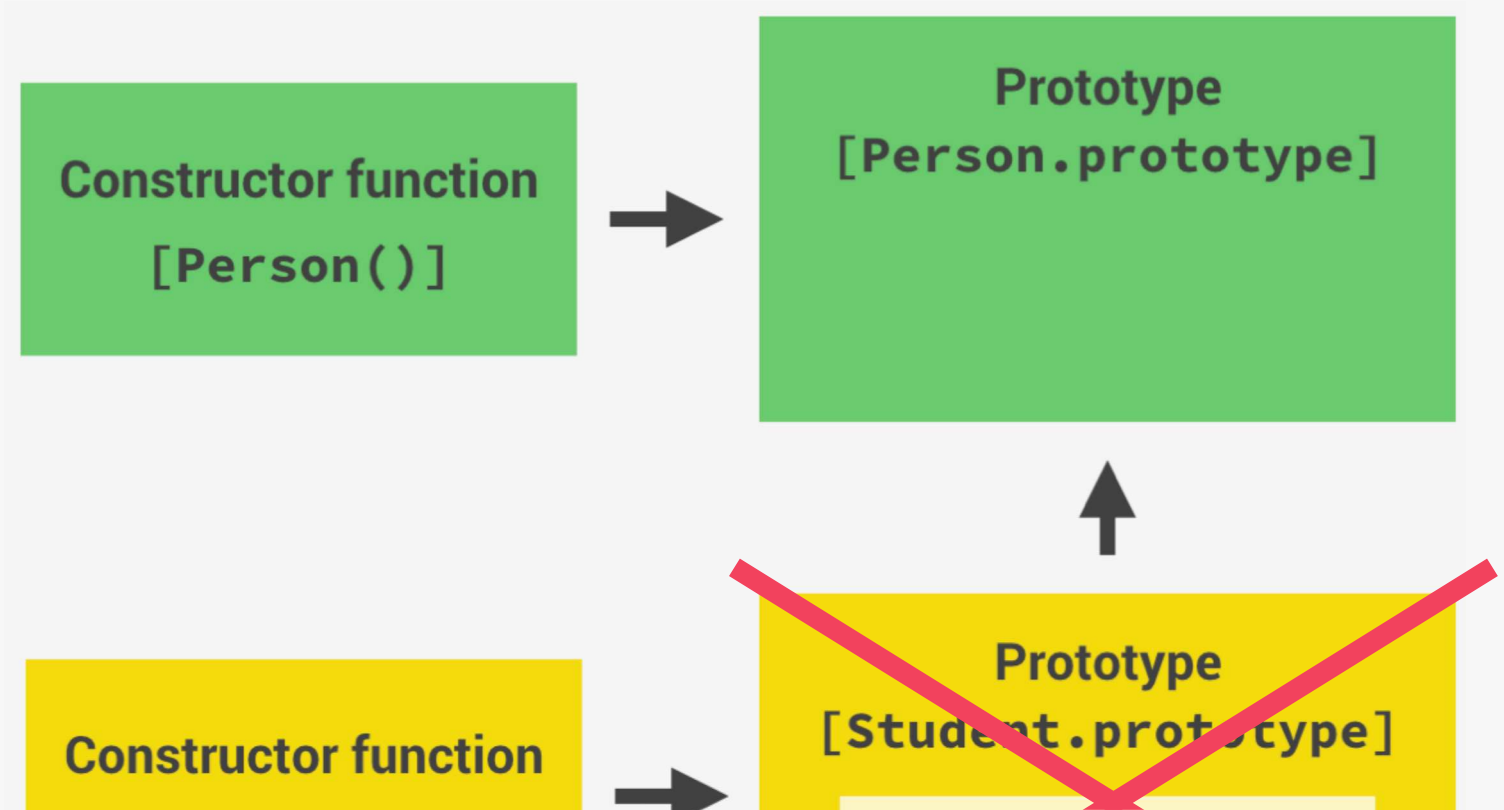


INHERITANCE BETWEEN "CLASSES"

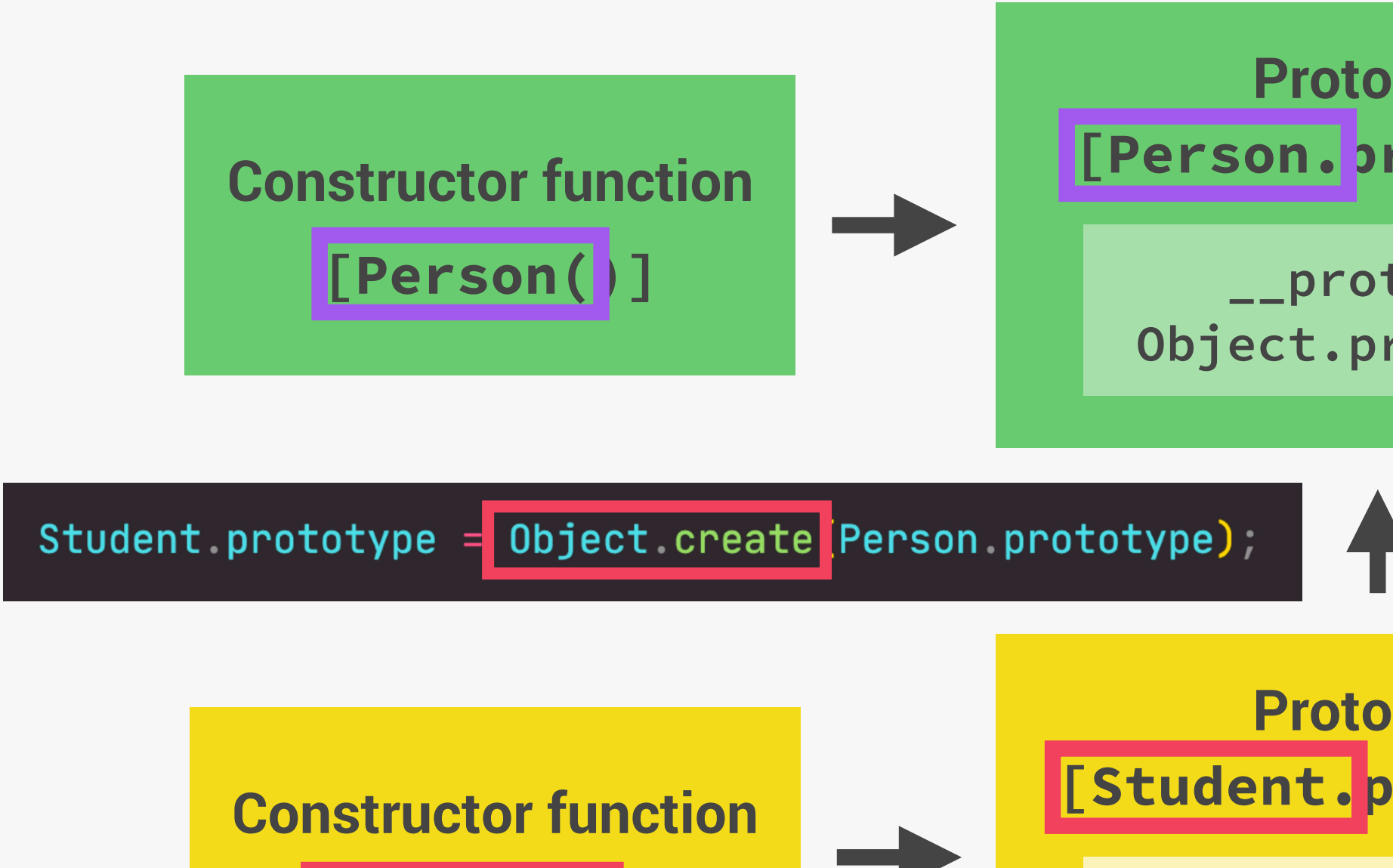


INHERITANCE BETWEEN "CLASSES"

```
Student.prototype = Object.create(Person.prototype);
```



INHERITANCE BETWEEN "CLASSES"



JONAS.IO
SCHMEDTMANN

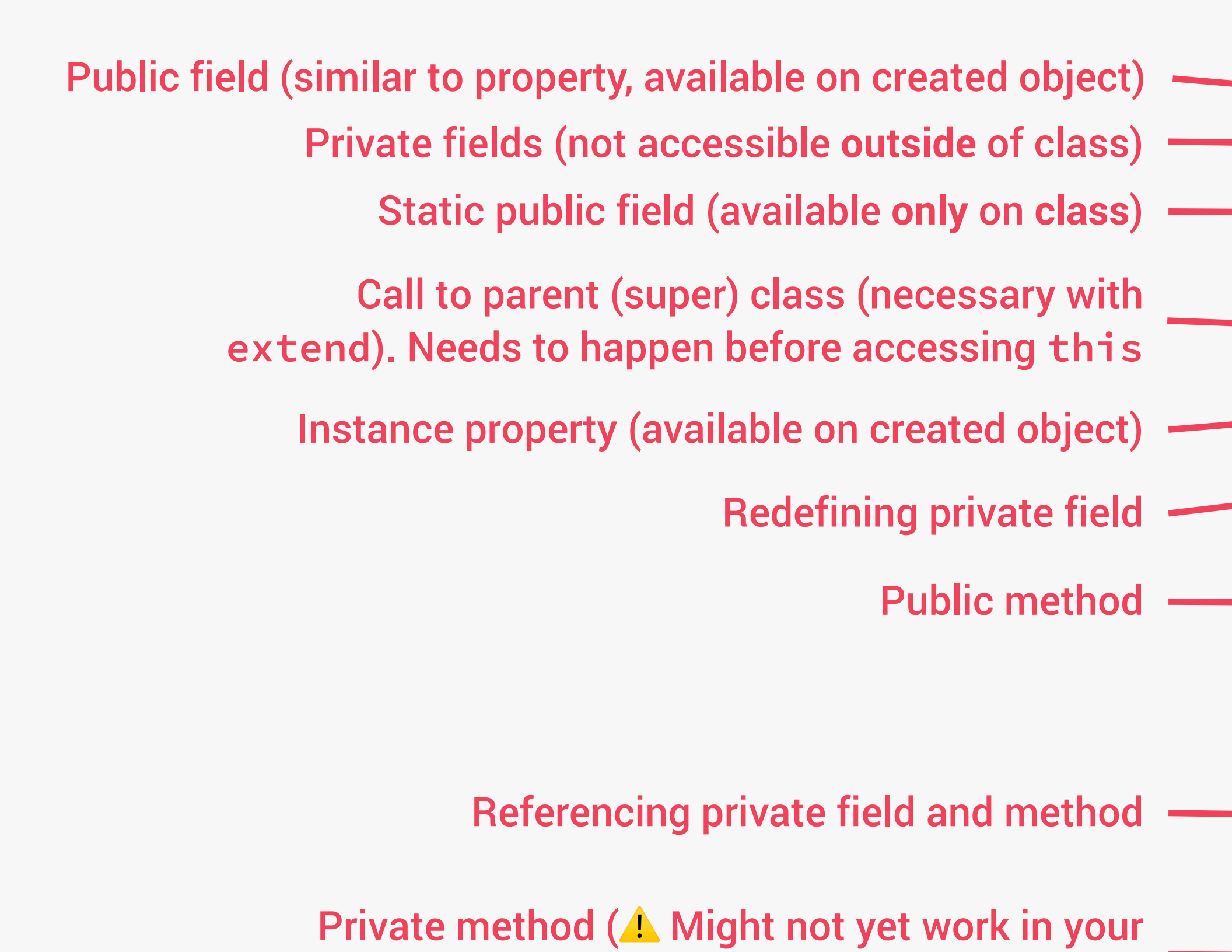
THE COMPLETE JAVASCRIPT COURSE

INHERITANCE BETWEEN "CLASSES":

```
const StudentProto = Object.create(Pe
```

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE


Public field (similar to property, available on created object)
Private fields (not accessible **outside** of class)
Static public field (available **only** on **class**)
Call to parent (super) class (necessary with `extend`). Needs to happen before accessing `this`
Instance property (available on created object)
Redefining private field
Public method
Referencing private field and method
Private method (⚠️ Might not yet work in your

MAPTY

GEOL

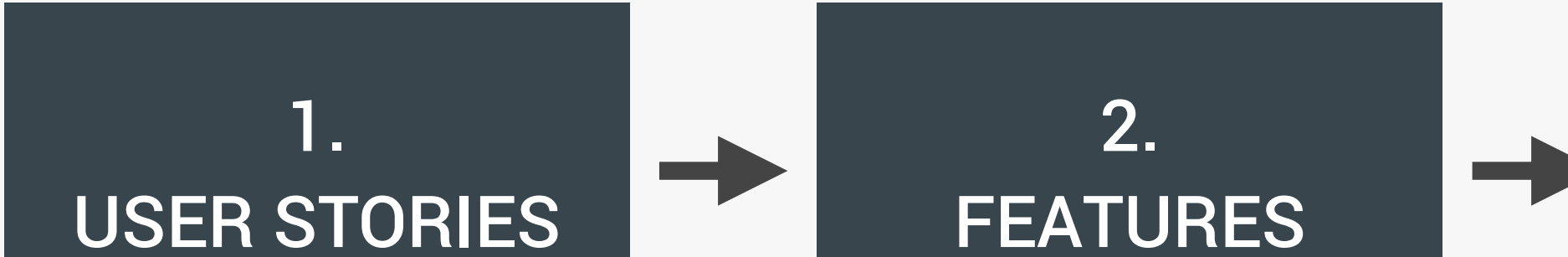
EXTENA

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

PROJECT PLANNING

Description of the application's functionality from the user's perspective. All user stories put together describe the entire application.



1. USER STORIES

👉 **User story:** Description of t

👉 **Common format:** As a *[type*

1. As a user, I want to **log my steps/minute**, so I can kee

2. As a user, I want to **log my**

2. FEATURES

USER STORIES

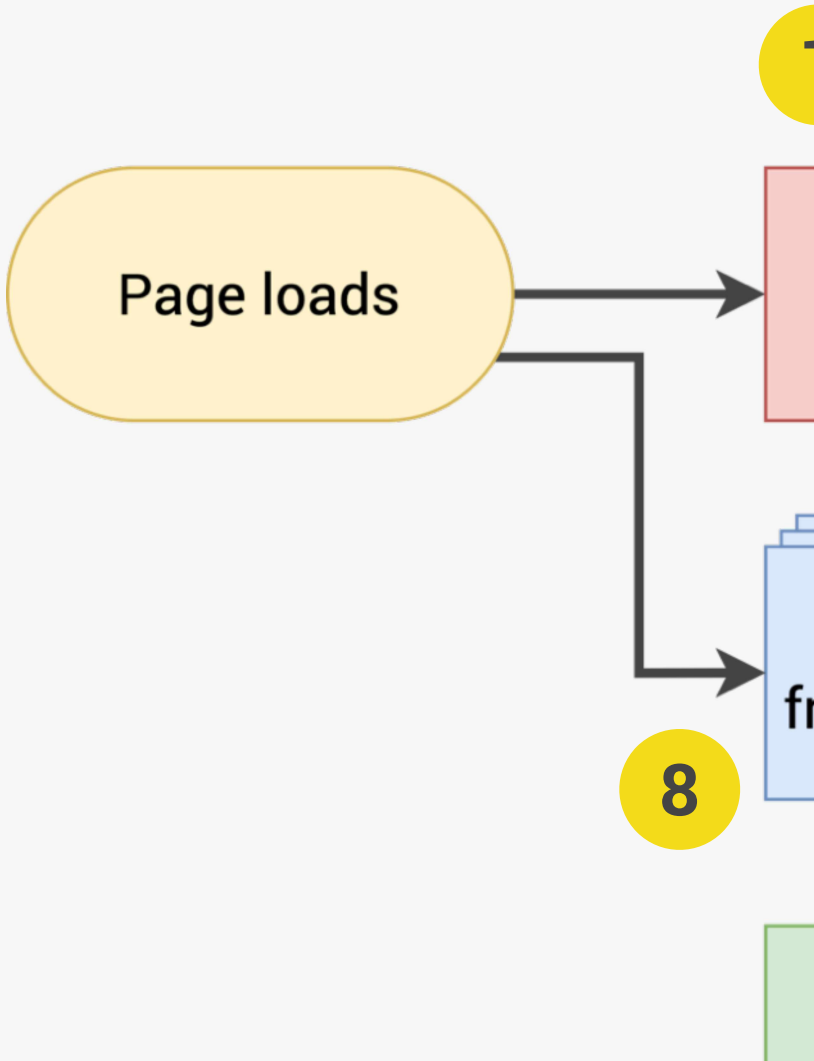
1. Log my **running** workouts with location, distance, time, pace and steps/minute

2. Log my **cycling** workouts with location, distance, time, speed and elevation gain

3. FLOWCHART

FEATURES

1. Geolocation to display map at current location
2. Map where user clicks to add new workout
3. Form to input distance, time, pace, steps/minute
4. Form to input distance, time, speed, elevation gain



4. ARCHITECTURE

FOR N

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

10 ADDITIONAL FEATURE IDEAS: CHA

- 👉 Ability to **edit** a workout;
- 👉 Ability to **delete** a workout;
- 👉 Ability to **delete all** workouts;
- 👉 Ability to **sort** workouts by a certain field (e.g
- 👉 **Re-build** `Running` and `Cycling` objects cor
- 👉 More realistic error and confirmation **messag**

ASYNC

JAVA

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

SYNCHRONOUS CODE

BLOCKING

```
const p = document
p.textContent = 'M
alert('Text set!')
p.style.color = 'r
```

ASYNCHRONOUS CODE

Asynchro

CALLBACK WILL
RUN AFTER TIMER

```
const p = document
setTimeout(functio
  p.textContent =
}, 5000);
p.style.color = 'r
```

👉 **Example:** Timer with callba

```
[1, 2, 3].map(x => x *
```

ASYNCHRONOUS CODE

Asynchro

CALLBACK WILL RUN
AFTER IMAGE LOADS

```
const img = document
img.src = 'dog.jpg';
img.addEventListener
  img.classList.add
});
p.style.width = '300
```

👉 **Example:** Asynchronous in

WHAT ARE AJAX CALLS?

AJAX

Asynchronous **J**avaScri

remote web servers in a

request data from web s

WHAT IS AN API?

API

- **A**pplication **P**rogramming **I**nterface: Piece
  that can be used by another piece of soft
  allow **applications to talk to each other**;

- There are be many types of APIs in web d

  DOM API | Geolocation API | Own Class A

JONAS.IO
SCHMEDTMANN

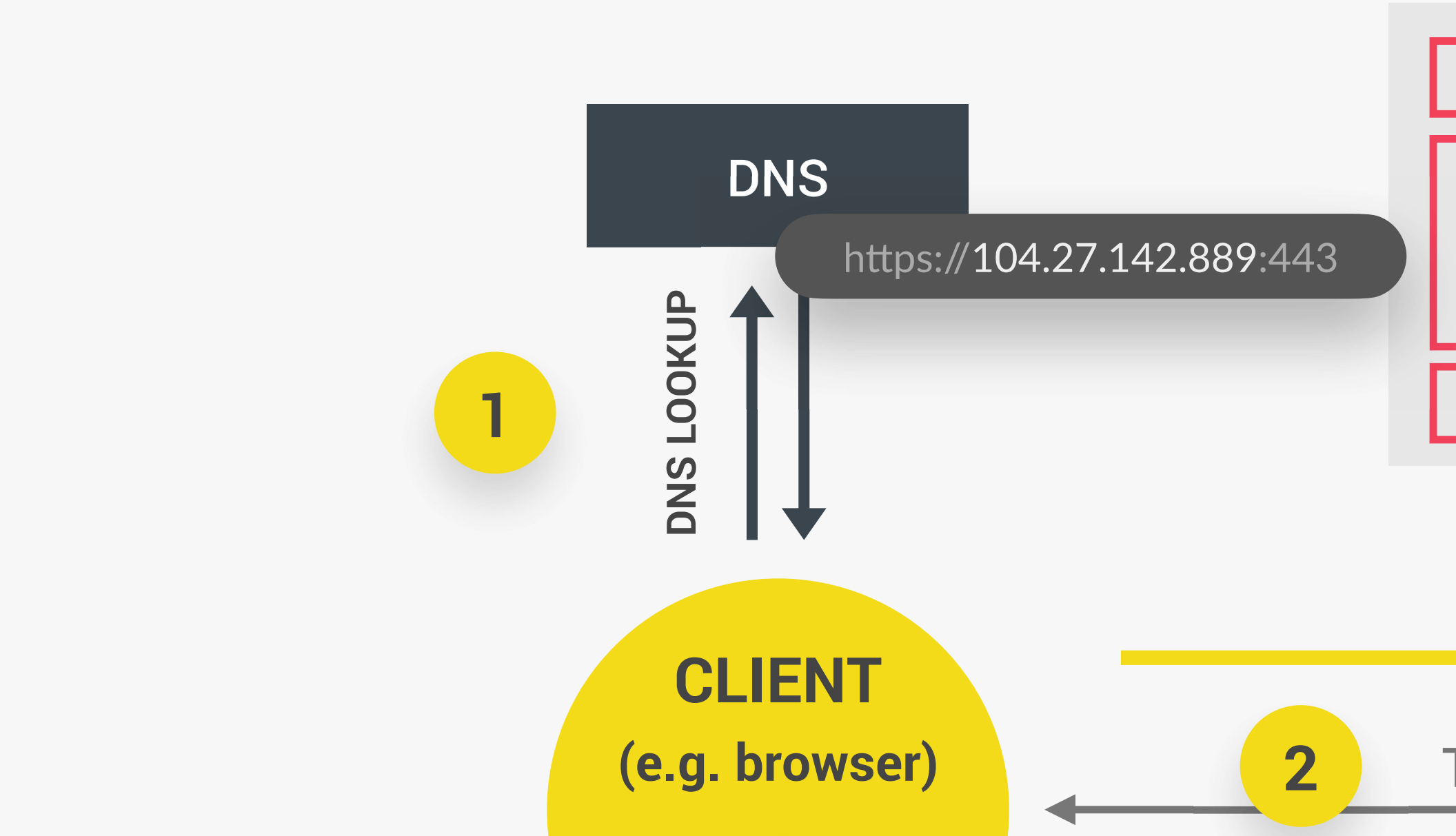
THE COMPLETE JAVASCRIPT COURSE

WHAT HAPPENS WHEN WE ACCESS

👉 **Request-respo**

CLIENT
(e.g. browser)

WHAT HAPPENS WHEN WE ACCESS



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

WHAT ARE PROMISES?

👉 **Promise:** An object that is used as a p
result of an asynchronous operation.

↓ Less formal

👉 **Promise:** A container for an asynchro

↓ Less formal

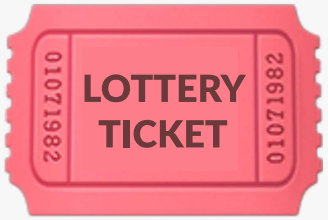
👉 **Promise:** A container for a future valu

THE PROMISE LIFECYCLE

Before the future value is available

PENDING

ASYNC TASK

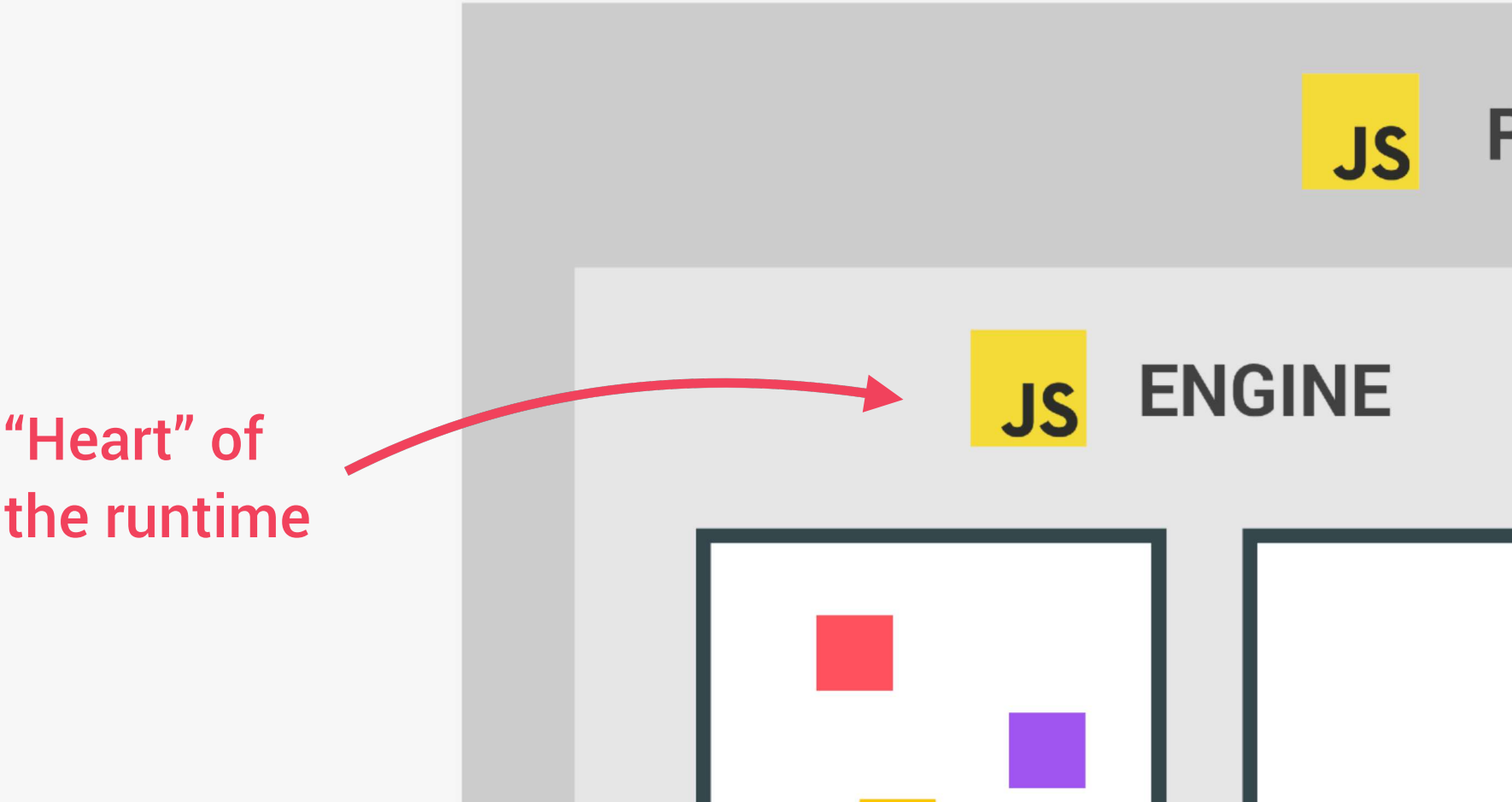


JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

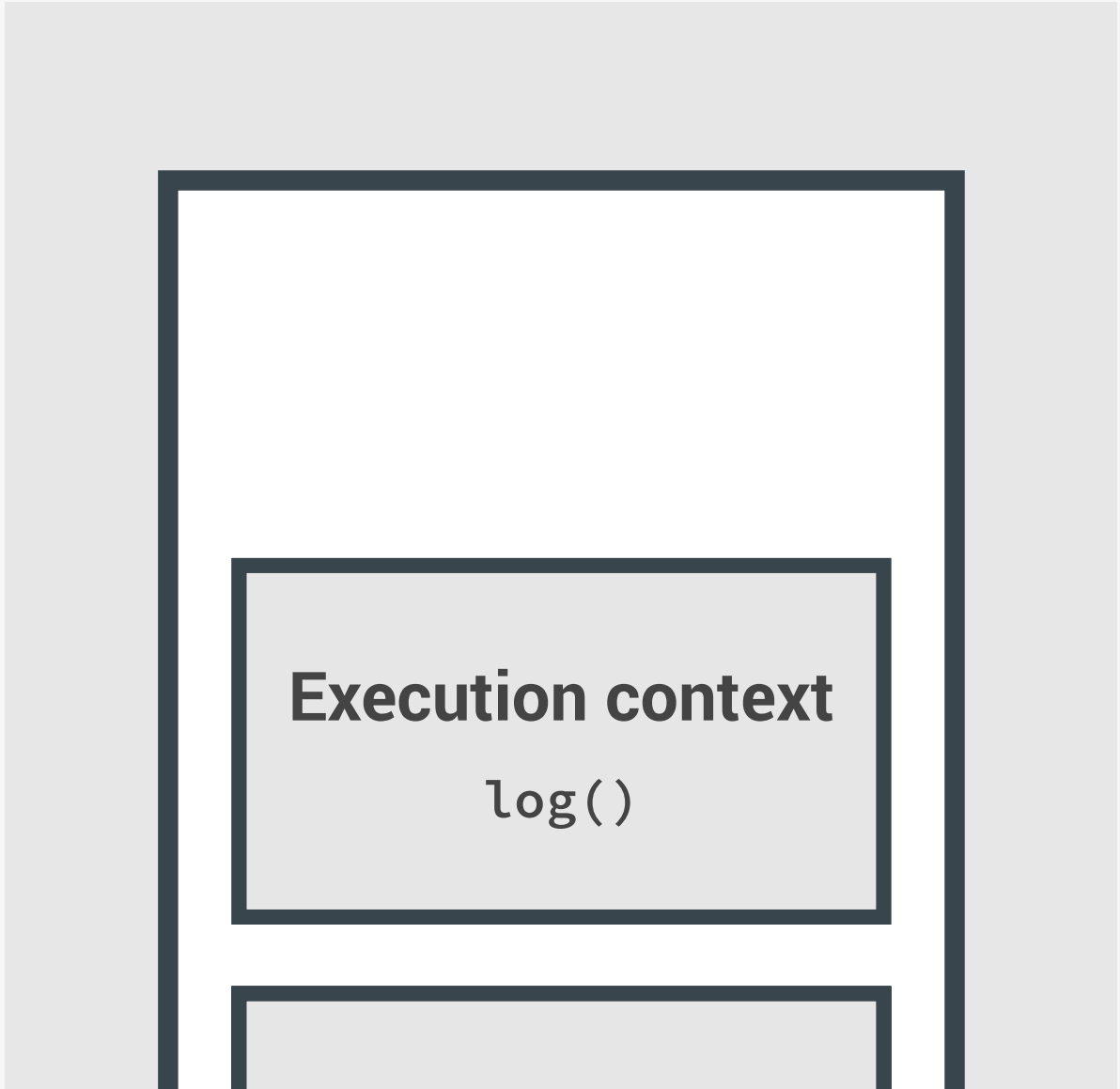
REVIEW: JAVASCRIPT RUNTIME

👉 **Concurrency model**: How JavaScript handles multiple tasks happening at the same time.

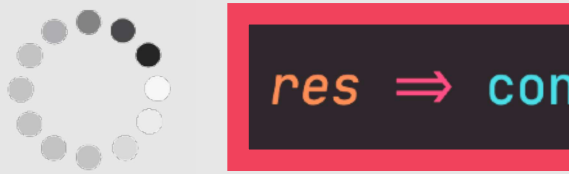


HOW ASYNCHRONOUS JAVASCRIPT

Where asynchronous tasks run



Loading image



Fetching data

MODERN

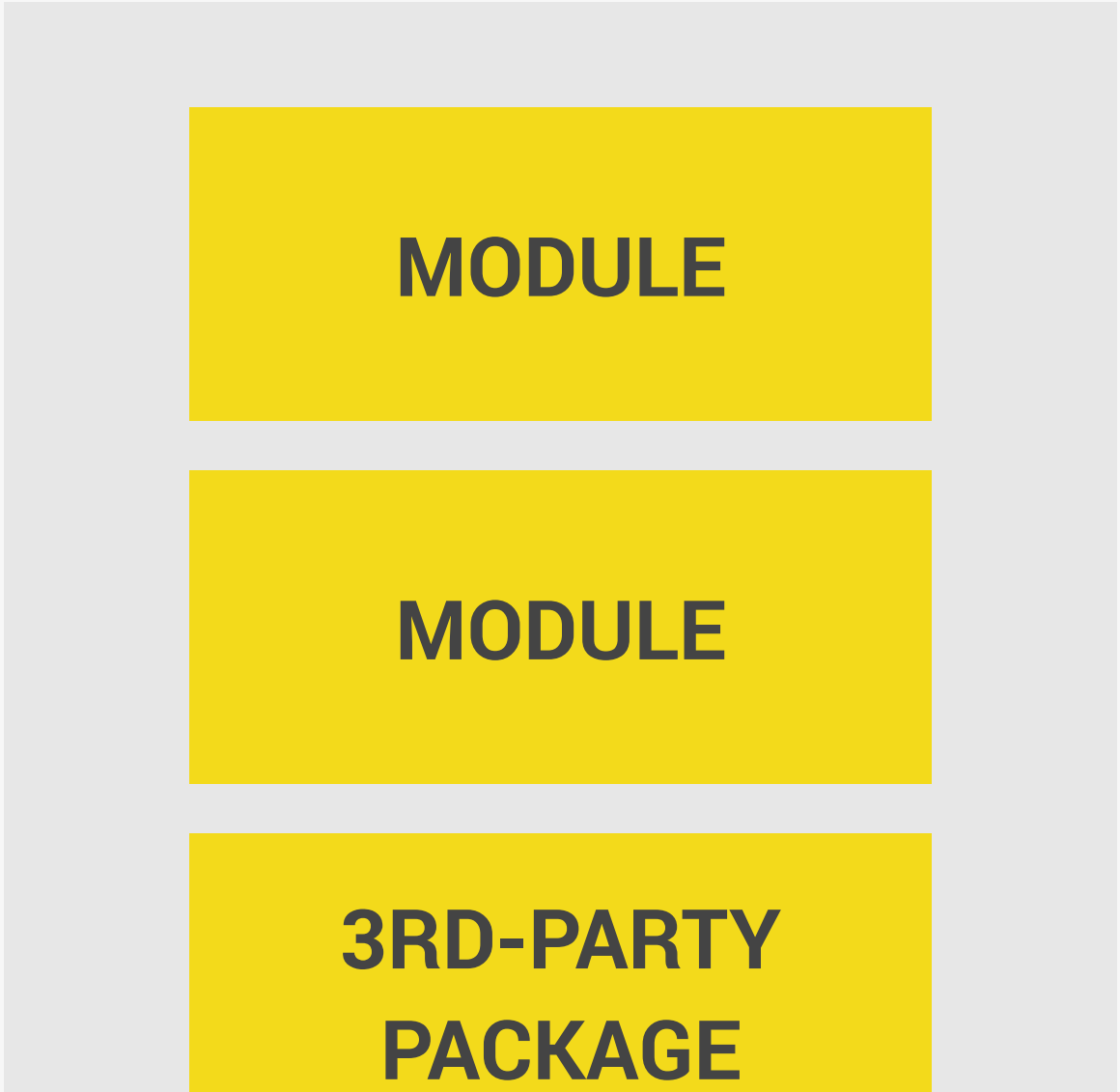
DEVE

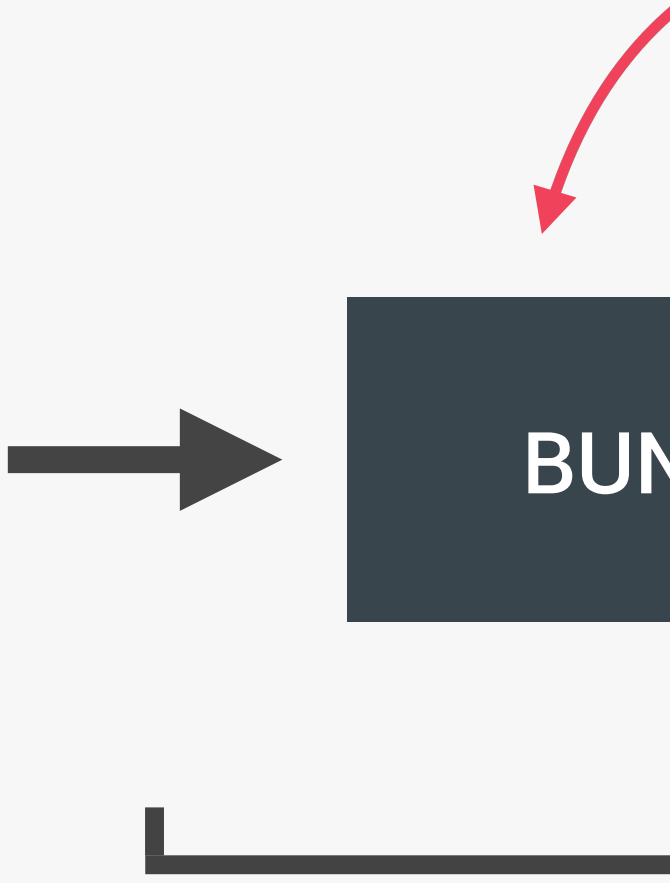
MOD

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

MODERN JAVASCRIPT DEVELOPMENT





JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

AN OVERVIEW OF MODULES

MODULE

- 👉 Reusable piece of code that **encapsulates** implementation details;
- 👉 Usually a **standalone file**, but it doesn't have to be.

NATIVE JAVASCRIPT (ES6) MODULES

ES6 MODULES

Modules stored in files, **exactly one module per file.**

HOW ES6 MODULES ARE IMPORTED

```
import { rand } from './math.js';
import { showDice } from './dom.js';
const dice = rand(1, 6, 2);
showDice(dice);
```

👉 index.js

```
const rand = (
  // Random nu
};
export { rand
```

👉 math.js

Live connection,
NOT copies

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

REVIEW: MODERN AND CLEAN CODE

READABLE CODE

- 👉 Write code so that **others** can understand it
- 👉 Write code so that **you** can understand it in 1 year
- 👉 Avoid too “clever” and overcomplicated solutions
- 👉 Use descriptive variable names: **what they contain**
- 👉 Use descriptive function names: **what they do**

REVIEW: MODERN AND CLEAN CODE

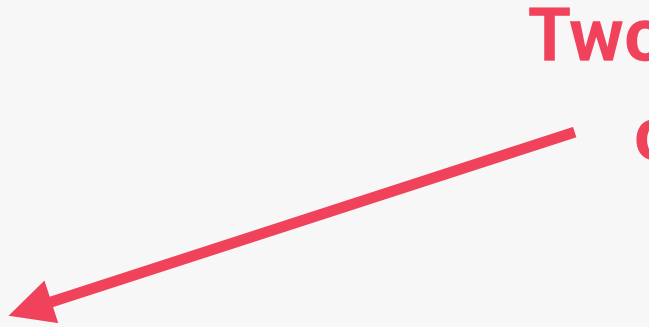
AVOID NESTED CODE

- 👉 Use early `return` (guard clauses)
- 👉 Use ternary (conditional) or logical operators instea
- 👉 Use multiple `if` instead of `if/else-if`

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

IMPERATIVE VS. DECLARATIVE CODE



IMPERATIVE

- 👉 Programmer explains **“HOW to do things”**
- 👉 We explain the computer *every single step* it has to
follow to achieve a result

FUNCTIONAL PROGRAMMING PRINCIP

FUNCTIONAL PROGRAMMING

- 👉 **Declarative** programming paradigm
- 👉 Based on the idea of writing software by combining **pure functions**, avoiding **side effects** and **mutating**
- 👉 **Side effect:** Modification (mutation) of any data **out** of the function (mutating external variables, loggin

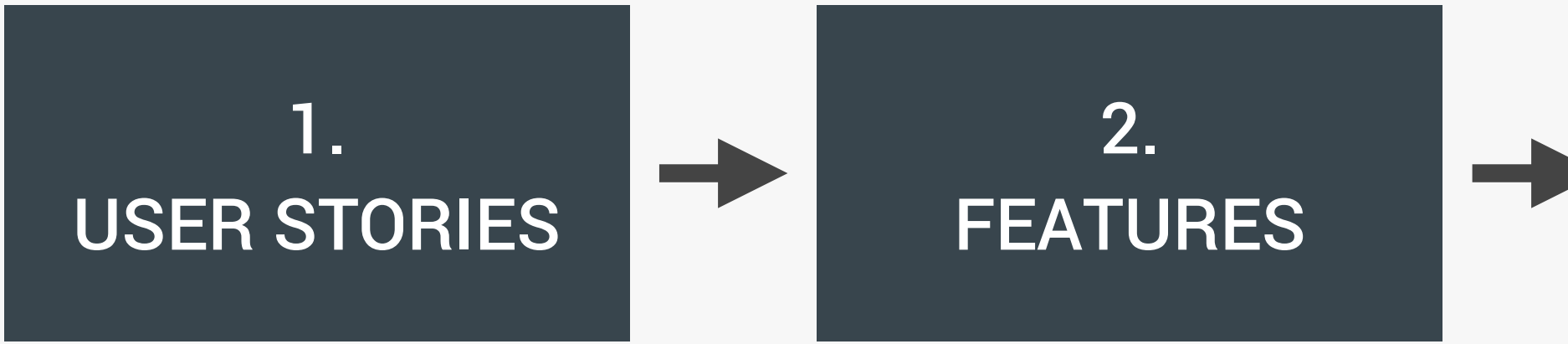
FORK

BUILDING

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

PROJECT PLANNING



1. USER STORIES

👉 **User story:** Description of t

👉 **Common format:** As a *[type*

1. As a user, I want to **search**

2. As a user, I want to be abl
meal for different number

2. FEATURES

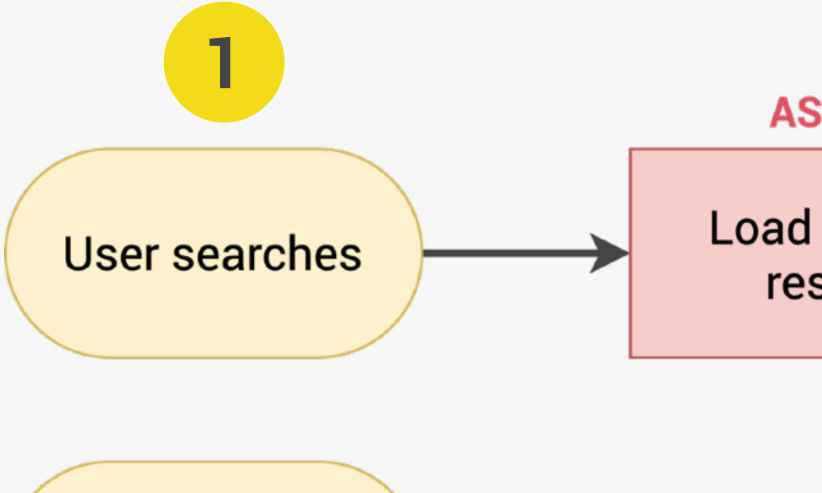
USER STORIES

1. Search for recipes
2. Update the number of servings

3. FLOWCHART (PART 1)

FEATURES

1. Search functionality: API search request
2. Results with pagination
3. Display recipe



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

WHY WORRY ABOUT ARCHITECTURE

👉 Like a house, software needs a structure: the way we **organize our code**

COMPONENTS OF ANY ARCHITECTUR

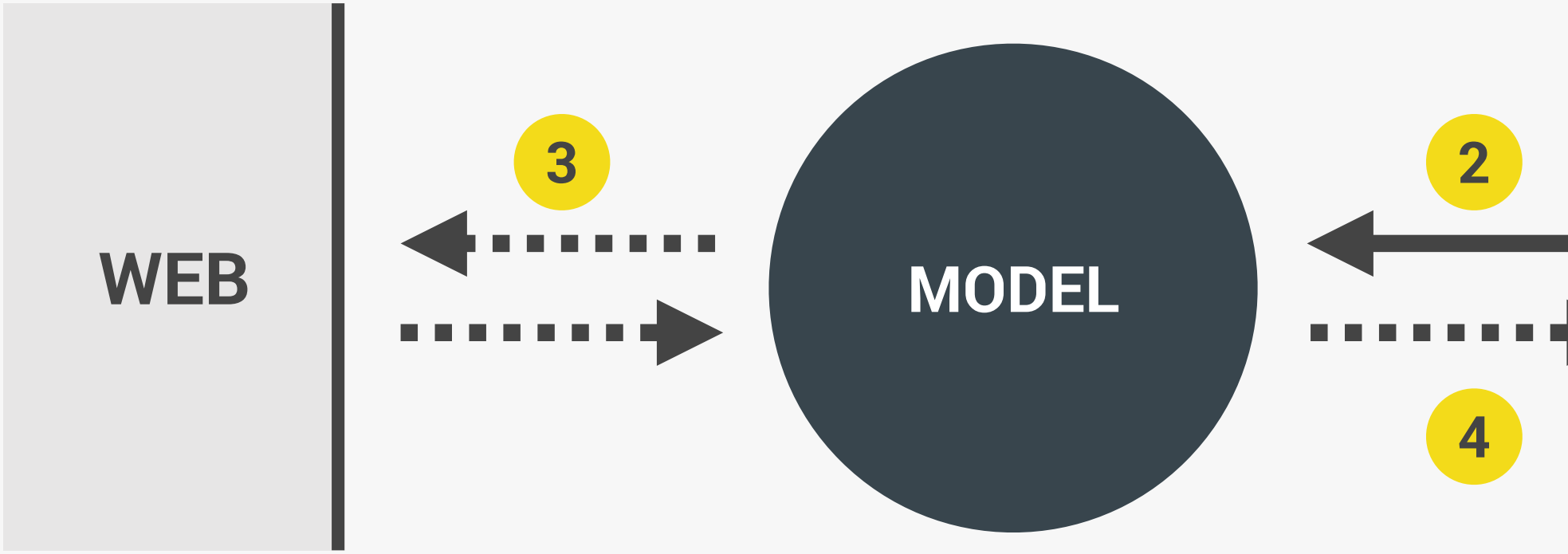
BUSINESS LOGIC

- 👉 Code that **solves the actual business problem**;
- 👉 Directly related to

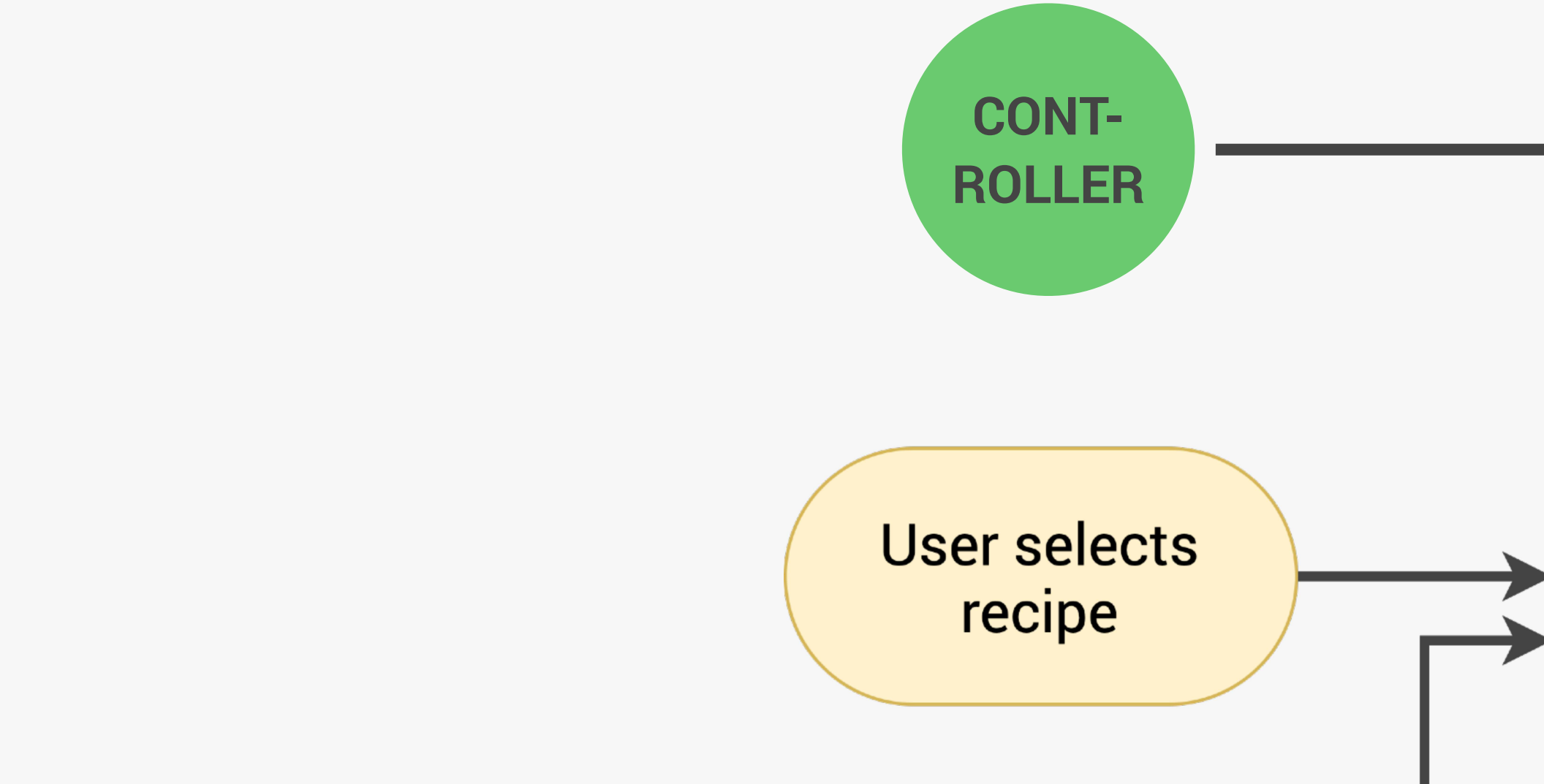
STATE

- 👉 Essentially **stores all the data** about the application
- 👉 Should be the

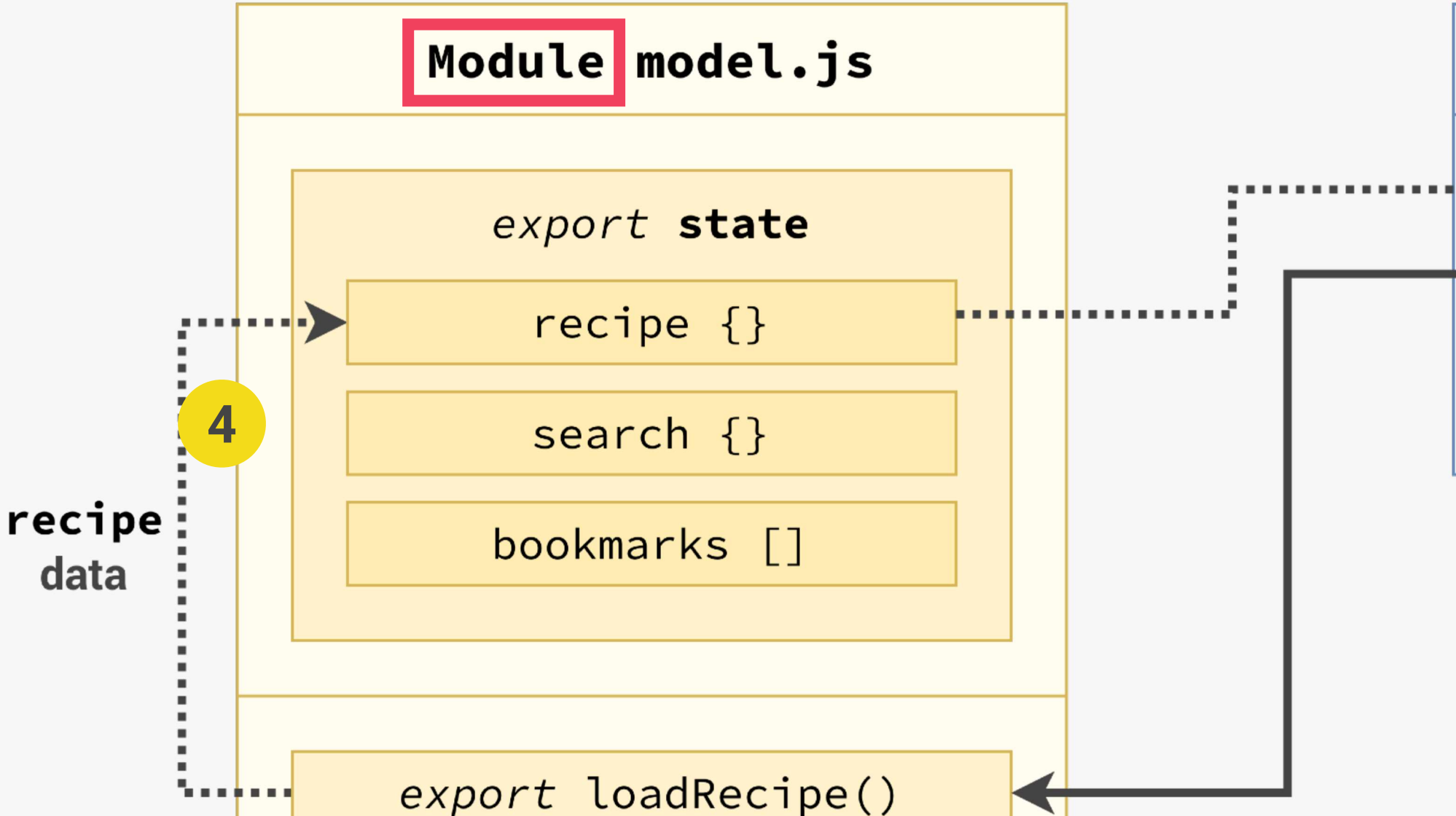
THE MODEL-VIEW-CONTROLLER (MV



MODEL, VIEW AND CONTROLLER IN R



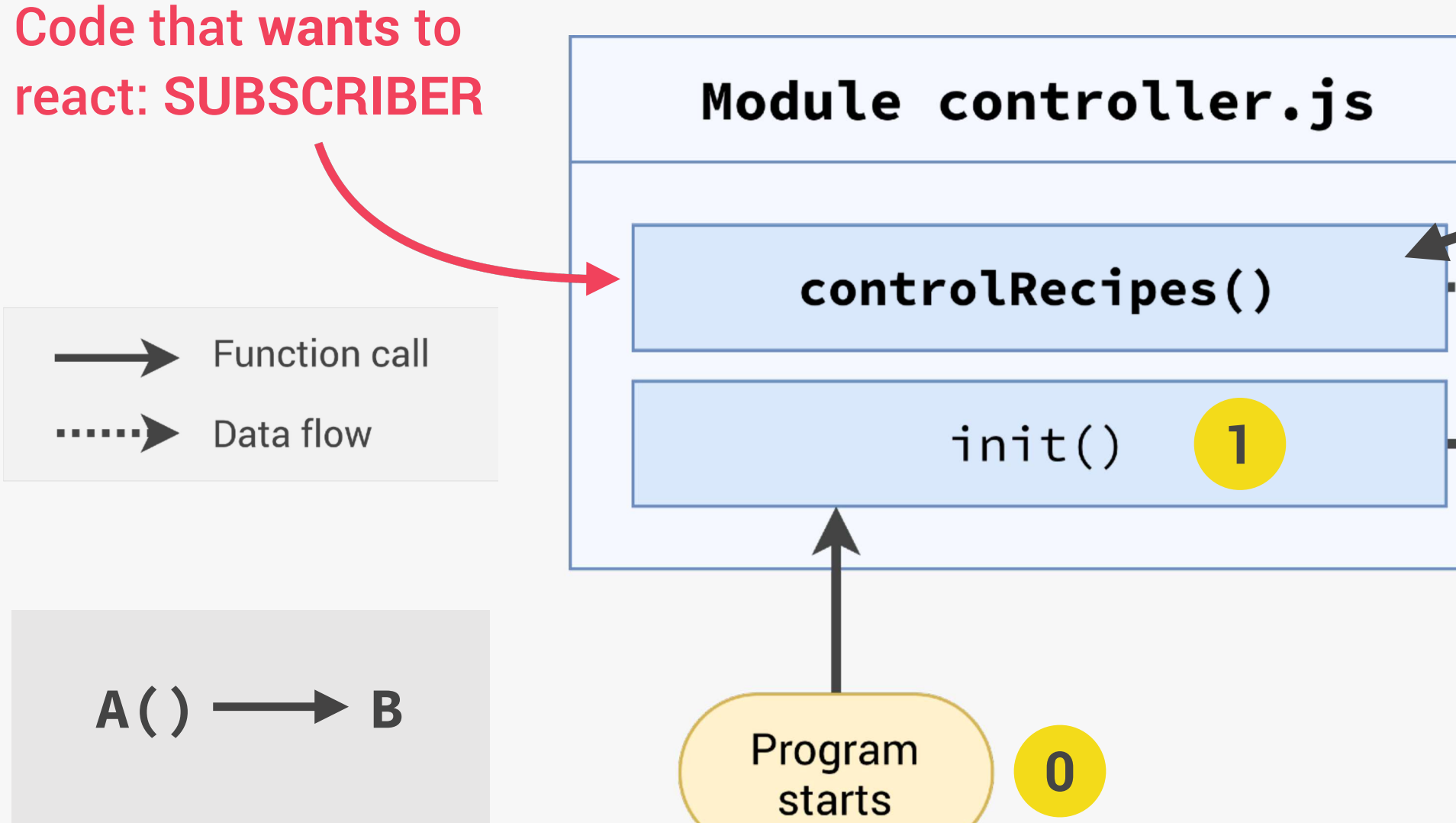
MVC IMPLEMENTATION (RECIPE DISP



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

EVENT HANDLING IN MVC: PUBLISHER



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

IMPROVEMENT AND FEATURE IDEAS:

- 👉 Display **number of pages** between the pagina
- 👉 Ability to **sort** search results by duration or nu
- 👉 Perform **ingredient validation** in view, before s
- 👉 **Improve recipe ingredient input**: separate in m than 6 ingredients;