THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

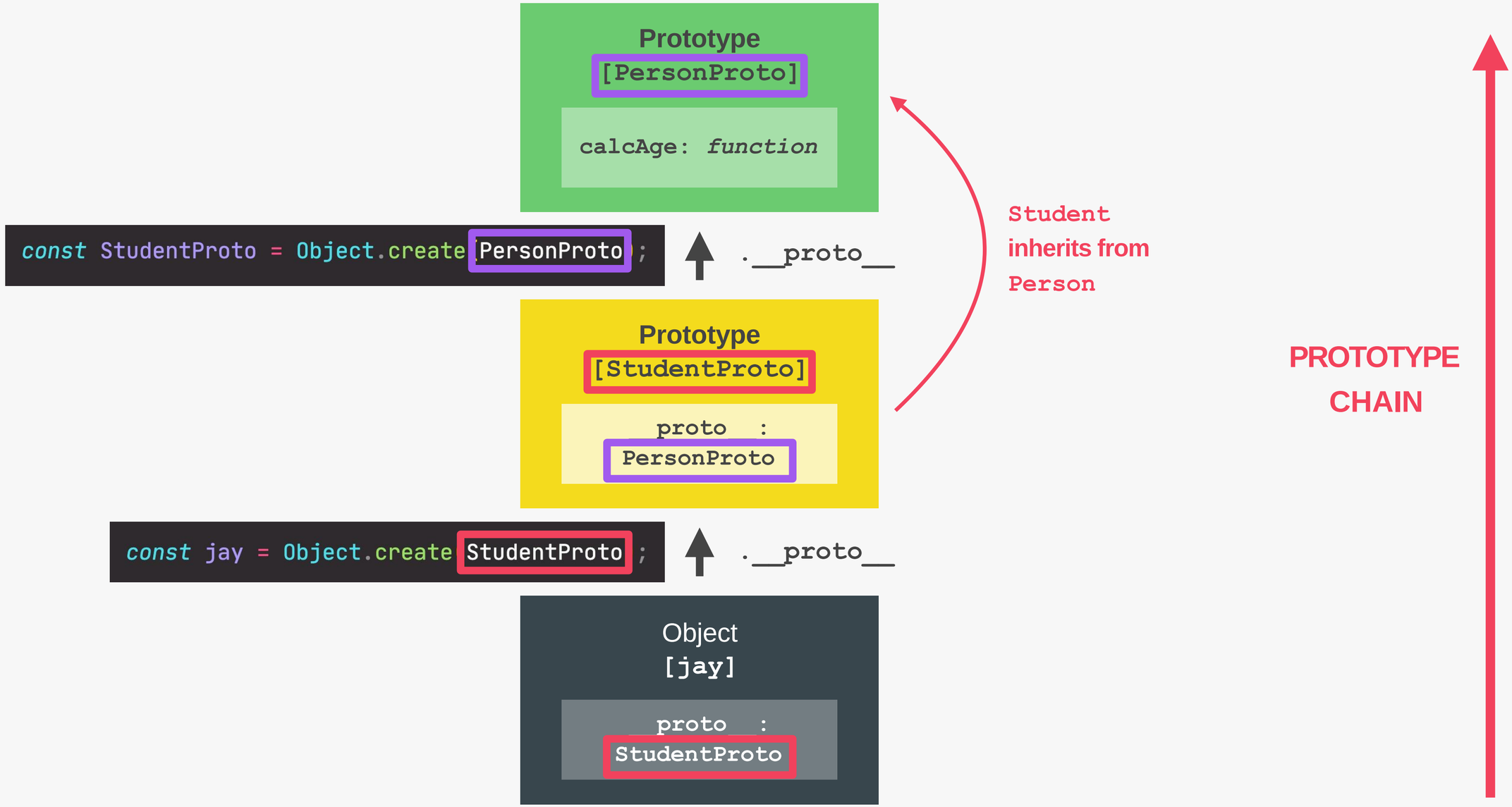
SECTION

OBJECT ORIENTED PROGRAMMING (OOP) WITH JAVASCRIPT

LECTURE

INHERITANCE BETWEEN "CLASSES": OBJECT.CREATE

INHERITANCE BETWEEN “CLASSES”: OBJECT.CREATE



THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

SECTION

OBJECT ORIENTED PROGRAMMING (OOP) WITH JAVASCRIPT

LECTURE

ES6 CLASSES SUMMARY

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 browser. “Fake” alternative: _ instead of #)

Getter method

Setter method (use _ to set property with same name as method, and also add getter)

Static method (available **only** on **class**. Can **not** access instance properties nor methods, only static ones)

Creating new object with **`new`** operator

```
class Student extends Person {
  university = 'University of Lisbon';
  #studyHours = 0;
  #course;
  static numSubjects = 10;

  constructor(fullName, birthYear, startYear, course) {
    super(fullName, birthYear);

    this.startYear = startYear;

    this.#course = course;
  }

  introduce() {
    console.log(`I study ${this.#course} at ${this.university}`);
  }

  study(h) {
    this.#makeCoffe();
    this.#studyHours += h;
  }

  #makeCoffe() {
    return 'Here is a coffe for you 👦';
  }

  get testScore() {
    return this._testScore;
  }

  set testScore(score) {
    this._testScore = score <= 20 ? score : 0;
  }

  static printCurriculum() {
    console.log(`There are ${this.numSubjects} subjects`);
  }
}

const student = new Student('Jonas', 2020, 2037, 'Medicine');
```

Parent class

Inheritance between classes, automatically sets prototype

Child class

Constructor method, called by **`new`** operator. Mandatory in regular class, might be omitted in a **child** class

- 👉 Classes are just “**syntactic sugar**” over constructor functions
- 👉 Classes are **not** hoisted
- 👉 Classes are **first-class** citizens
- 👉 Class body is always executed in **strict mode**

MAPTY APP: OOP, GEOLOCATION, EXTERNAL LIBRARIES, AND MORE!

THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

SECTION

MAPTY APP: OOP, GEOLOCATION, EXTERNAL LIBRARIES, AND MORE!

LECTURE

HOW TO PLAN A WEB PROJECT

PROJECT PLANNING

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

WHAT we will build

HOW we will build it

PLANNING STEP

Implementation of our plan using code

1. USER STORIES

mapty

👉 **User story:** Description of the application's functionality from the user's perspective.

👉 **Common format:** As a *[type of user]*, I want *[an action]* so that *[a benefit]*

Who? Example: user, admin, etc.

What?

Why?

1. As a user, I want to log my running workouts with location, distance, time, pace and steps/minute, so I can keep a log of all my running
2. As a user, I want to log my cycling workouts with location, distance, time, speed and elevation gain, so I can keep a log of all my cycling
3. As a user, I want to see all my workouts at a glance, so I can easily track my progress over time
4. As a user, I want to also see my workouts on a map, so I can easily check where I work out the most
5. As a user, I want to see all my workouts when I leave the app and come back later, so that I can keep using there app over time

2. FEATURES

mapty

USER STORIES	FEATURES
1 Log my running workouts with location, distance, time, pace and steps/minute	👉 Map where user clicks to add new workout (best way to get location coordinates) 👉 Geolocation to display map at current location (more user friendly) 👉 Form to input distance, time, pace, steps/minute
2 Log my cycling workouts with location, distance, time, speed and elevation gain	👉 Form to input distance, time, speed, elevation gain
3 See all my workouts at a glance	👉 Display all workouts in a list
4 See my workouts on a map	👉 Display all workouts on the map
5 See all my workouts when I leave the app and come back later	👉 Store workout data in the browser using local storage API 👉 On page load, read the saved data from local storage and display

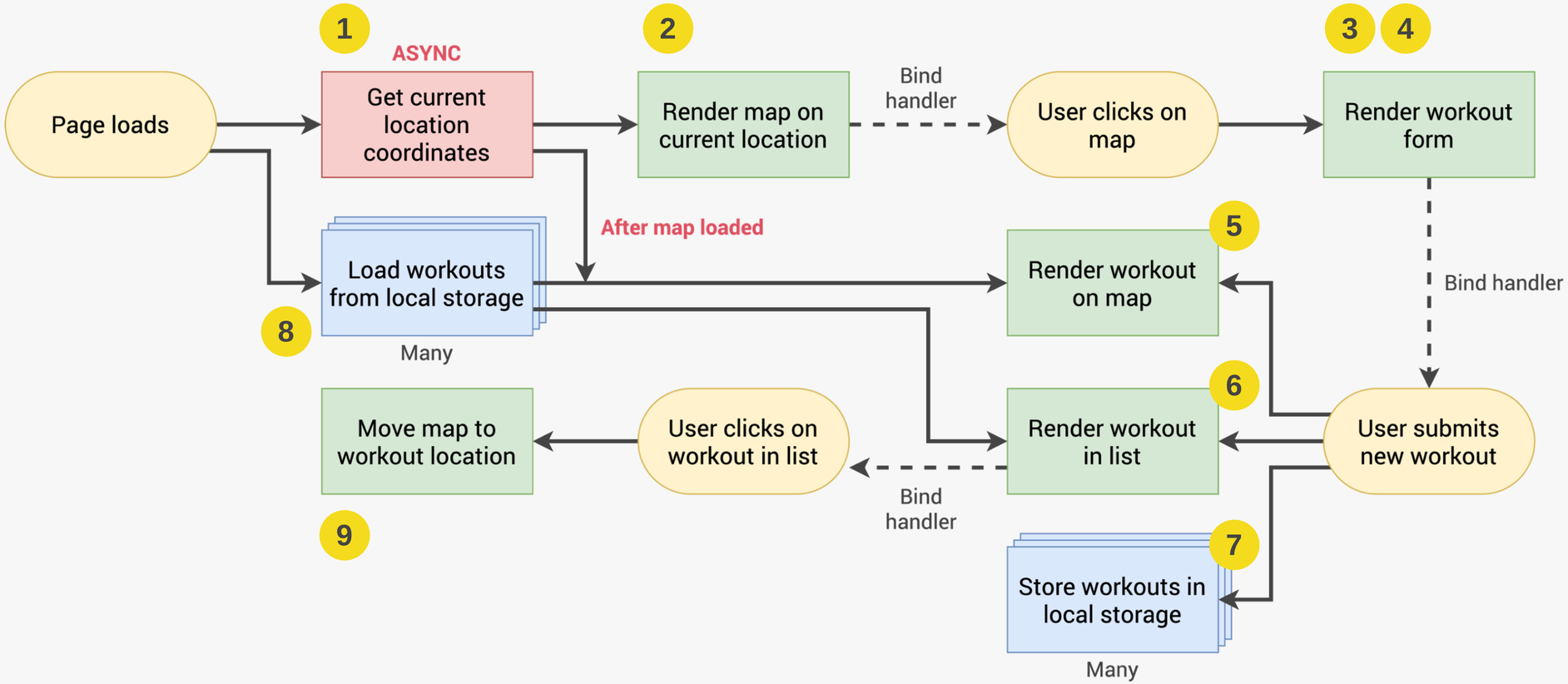
3. FLOWCHART

mapty

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
5. Display workouts in a list
6. Display workouts on the map
7. Store workout data in the browser
8. On page load, read the saved data and display
9. Move map to workout location on click

Added later



☝ In the real-world, you don't have to come with the final flowchart right in the planning phase. It's normal that it changes throughout implementation!

mapty

FOR NOW, LET'S JUST START CODING 😍