

Web Applications

Introduction: organization, exam, communications

Enrico Masala

Antonio Servetti



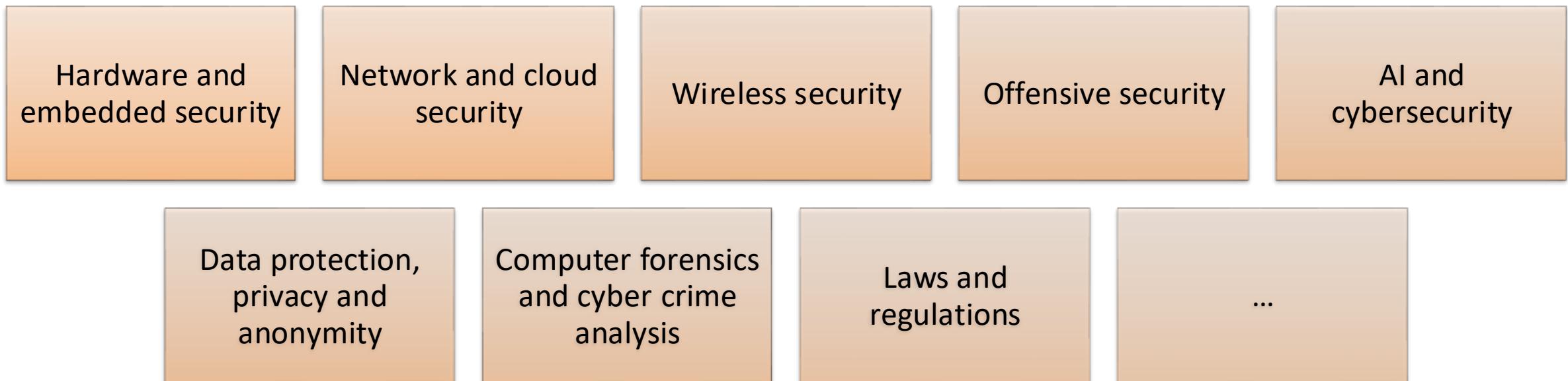
Goal

- Understanding web architectures
- Understanding and mastering web application design and development
- Gaining in-depth knowledge of the JavaScript language and ecosystem
- Becoming familiar with one of the most popular JavaScript frameworks (React) with special focus on the front-end
- Some attention on **basic** security aspects in web applications
- Official course description:
[https://didattica.polito.it/pls/portal30/gap.pkg_guide.viewGap?
p_cod_ins=01GYOWQ&p_a_acc=2026&p_header=S&p_lang=EN](https://didattica.polito.it/pls/portal30/gap.pkg_guide.viewGap?p_cod_ins=01GYOWQ&p_a_acc=2026&p_header=S&p_lang=EN)



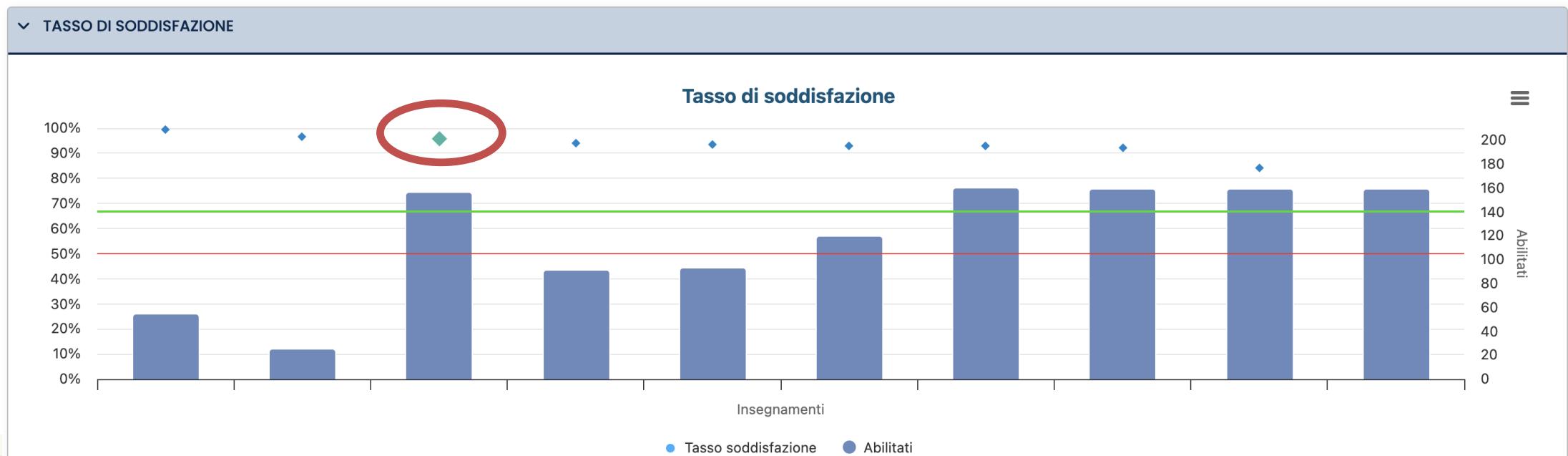
The Bigger Picture

- “Web applications” is a **basic** course in “Cybersecurity Engineering” course of study
 - Many applications nowadays are web-based: important to know their technologies
- The course of study includes many other courses regarding cybersecurity



Previous Year End-of-course Questionnaire

- Feedback from previous years (CPD / JCT): high satisfaction rate
- Most criticism about the need for better connection to cybersecurity topics
 - It is already (partially) addressed since it is an introductory course to web applications, paying attention to SOME cybersecurity aspects (e.g., server side)



What We Will Learn

JavaScript as a language

- ECMAScript ES6
- Language constructs
- In-depth semantics
- Functional, Asynchronous, Modular, ...



The browser ecosystem

- HTML, CSS, page structure
- DOM
- JavaScript in the browser
- Events, Properties, Handlers, APIs



Single Page Applications

- Server-side with node (focus on security)
- API development
- Backend storage
- Sessions and Authentication



React framework

- Components, Properties, State
- JSX
- Hooks
- Router



Calendar... At a Glance!

1. Intro to JS: basics, objects, functions
2. Intro to JS: async programming, callbacks, DB interaction + Intro to Web
3. Server-side with Express; API design
4. HTML, CSS, Bootstrap
5. JS: modules and other topics, + JS in the browser
6. Intro to React
7. React: props and state
8. React: context, life cycle, forms
9. React router
10. Data fetching and client-server interaction (in React)
11. Authentication

Course Organization

- Classes
 - 6 (or 3) h/week
 - Lectures + Exercises (*mixed*)
- Laboratories (room 8i and 9T)
 - 1.5 or 1.5+1.5 h/week (using class slots)
 - 3 Lab groups (see later for the split)
 - Starting 2nd week
- **Detailed schedule week-by-week**
 - <https://github.com/polito-WA-2026/.github/blob/main/profile/SCHEDULE.md>

	MO	TU	WE	TH	FR
08:30				R3	
10:00				R3	
11:30					
13:00	R3				9T
14:30	R3	8i			
16:00		8i			
17:30					

Classes

- In person, (mostly) in rooms with power outlets at the desks
 - bring your own computer, if possible, if you want to try the examples/exercises
- Video-recorded and made available soon after each class
 - *not* streamed live
- We will try to make live examples (live coding etc.) whenever possible during lectures

Laboratories

- Starting March 3, 2026
- In rooms with power outlets at the desks
 - **No computers are available in the room, bring your own**
- Text online, some days in advance
- Exercises to be done during Lab hours
- Solution will be posted on GitHub
 - 1 week (or less) after the end of each lab

Laboratories

- You will build a simple project during the labs
 - Step by step, following the course topics
- Some labs will last one week, others will span multiple weeks
- We will start with three slots, divided by last name:
 - AA-FA : group #1
 - FB-NZ : group #2
 - OA-ZZ : group #3

Learning Material

- Everything is on GitHub:
<https://github.com/polito-WA-2026>
- Course website
 - Slides
 - Full schedule
 - Links and supplementary material
 - Examples, exercises, labs, exams, ...
- Video lectures (screencasts)
 - YouTube - <https://www.youtube.com/playlist?list=PLuZyhAOPm9pN6rlqyan1PYJ1iE5pYyUDQ>
 - Portale della Didattica (download only)



Web Applications (2025/2026)

Material for the course of Web Applications (in English) for the [Master Degree in Cybersecurity](#) at Politecnico di Torino, Italy.
Teachers: E. Masala, A. Servetti

Course information

[Detailed schedule](#), [exams and rules](#), official [syllabus](#), resources and software, [sample text of exams from past edition](#)

Available material and repositories specific to the course

- Home: github.com/polito-WA-2026
- Lecture recordings (YouTube)
- Course materials
- Lecture examples
- Lab code and solutions



Communications



- We will use **Telegram** for the main communications about the course
 - Among students, with teachers, etc.
 - Announcements and official information, and Q&A (using “topics” in Telegram)
- Feel free to contact the teachers for feedback and questions
 - **questions** of general interest (including exam) must be posted in the group, so that everybody can see the answer. NB: Do not exchange suggestions to solve the exam.
- Link to the Telegram group: https://t.me/+suu_OQRLD6c1MDA0
 - Any nickname is ok, but tell who you are for personal issues (especially in DM)
- Emails can be an **alternative** for slower, more articulated, and private individual communications (also, good to preserve info)

About the Exam

Oral discussion is **NOT** on
the day (T_0) of the exam!!!



- The exam consists in an **individual** project development + oral discussion
- **Develop a web application** according to the given functional specification, using the approach/technologies seen during the course
 - React + JavaScript, Node.js + Express, SQLite
 - Different technologies/approaches will NOT be accepted and lead to exam failure, without testing the project. If in doubt, ask the teacher in advance!
- **It is NOT ONLY to develop the web applications, but being able to explain the reason of the implementation choices, tradeoffs and alternatives**
 - Today most AI tools can (partially) solve the assignment automatically, **we expect students being able to explain and reason about the assignment, at the exam**
- Assignment published about 20 days before each official exam date (deadline)
 - Different for each exam date (*details are also in specific instructions on course website*)

About the Exam: Cheating **WARNING**



- NOT following the rules lead to exam failures (already happened many times)
 - Regardless of the goodness of the project, even if it is perfect
- **Cheating will NOT be tolerated, in any form, e.g.,**
 - Copying from / developing / submitting for others
 - Using AI tools (ChatGPT/Copilot etc.) without being able to explain the code in DETAIL
- Cases may be directed to Disciplinary Board (“Commissione Disciplina”)
 - <https://www.polito.it/en/education/services-and-life-at-politecnico/student-conduct>
 - Many high-impact consequences, e.g., LOSING scholarships (unfortunately, it already happened...)
 - *Some cases of misconduct (marked with *) will be reported to the authorities (Procura della Repubblica) since they constitute alleged criminal offences



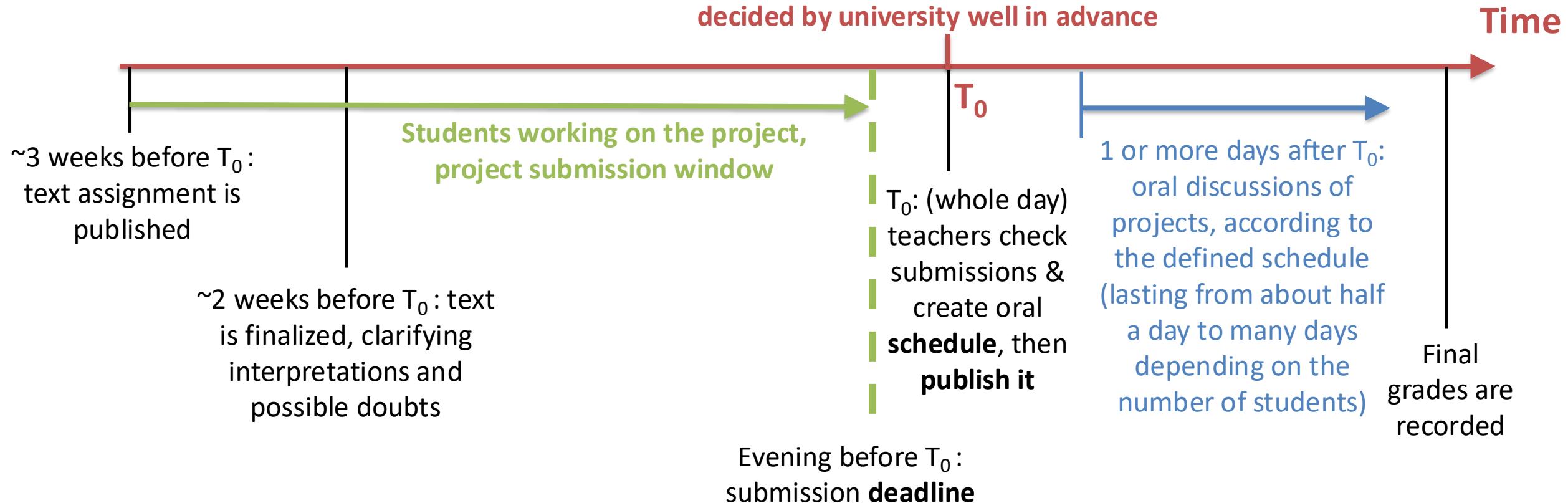
About the Exam: Timeline

Oral discussion is **NOT** on the day (T_0) of the exam!!!



- **For each** official exam date (“appello”):

T_0 : Exam date (“appello” date),
decided by university well in advance



After Project Submission

Oral discussion is **NOT** on the day (T_0) of the exam!!!



- Similarity checks will be run: excessively similar solutions will lead to exam failure without oral discussion
- A schedule for oral discussions is finalized **ON THE FIRST WORKING DAY AFTER THE DEADLINE**
 - Depends on the actual number of project submissions
 - The actual day of the oral discussion for a given student is not known until then
- Oral discussion is NOT on the day of the exam (“appello” date)!!!
 - Be careful when booking travels etc. If unsure, ask the teacher in advance!
 - All discussion **IN PRESENCE** (remote is forbidden by university regulations)

After Project Submission

- The teacher will automatically load all solutions on a Linux server
 - Done on the first working day after the deadline
 - This ensure that what is tested is what was submitted
- **STRICT** conformance to submission instruction is extremely important!!
- Check carefully library install and imports, filename upper/lowercase, TCP ports, etc.
 - Wrong settings or need for manual intervention yield some exam grade reduction
- *Complete and detailed exam rules and submission instructions in the course website (under "Exams")*

Oral Discussion



- Exam is an **INDIVIDUAL** ASSIGNMENT.
- Not able to explain any code behavior: immediately leads to exam **FAILURE**
 - In this case, no test of app. Already happened (too) many times in the past

Most common reason of EXAM FAILURE:

Suspect of external help because student cannot explain functions not included in lectures (coming from any source: AI tools, Internet, friends, etc.)

- Anything not seen during lectures is SUSPECT by default
- It will be **systematically** questioned **IN MUCH DETAIL** before the rest

**NO STUDENT WILL PASS WITHOUT BEING ABLE
TO EXPLAIN SUSPECT CODE IN DETAIL**

Oral Discussion: Expectations

- Discussion of the submitted code / API / solution, in student's presence, running it on a Linux server, while testing their behavior
- The student must be able to explain:
 - The rationale behind the project design and why each functionality has been implemented in a certain way
 - Which checks or actions are in place to make the solution secure against the most common types of attacks
 - Possible **variants** of the solution: **extremely important nowadays that many AI tools (ChatGPT, Copilot, Codex, Cursor, etc.) can automatically “solve” simple assignments**
- NB: It is NOT a presentation of the project given by the student
- All submitted projects will be **pre-screened** for unusual (suspect) programming patterns/functions/libraries/content etc.: oral discussion will start from them



Oral Discussion and Final Grade Criteria

- Provided that the student can correctly explain every choice/part of code:
 - Evaluation of the code (both client and server): programming patterns, **security management and checks**, **data integrity checks**, code clarity, code uniformity and coherence, code placement (server/client), correct usage of React **patterns** (functional behavior, hooks, state, context, effects), etc.
 - Evaluation of the application architecture (e.g., database design and organization, HTTP **API design**, API calls, **content of API requests and responses**, organization of React components and routes, no direct DOM manipulation or browser behavior outside React), **basic usability and user-friendliness**, responses to user actions (UI update, **absence of application crashes** and unhandled exceptions), originality of the solution, etc.
 - Evaluation of the student's theoretical and practical knowledge, readiness and clarity in the replies for questions about:
 - the project design, code base, readiness and clarity in the replies
 - behavior of any function/method (especially/**including** library ones) used in the code and/or seen during lectures

Final Words about AI and Code Development

- Many automatic tools can generate code: **ChatGPT, Copilot, ...**
- Can inspire small pieces of code, can be confusing in large projects
- **Suggestion: DO NOT USE THEM for the exam, risk of exam failure!**
 - FAILURE at exam is, typically, because the student cannot explain how an (even simple) function/method (suggested or introduced by AI, even if correct ...) works
 - Very frustrating for students, since the application is often not even opened
 - You may experiment with AI tools during lectures/lab
 - In lab, we can help you review code and understand advantages/disadvantages/mistakes...
- **For the exam: start from your own lab solutions (or our published solutions)**
 - Attend (or at least) follow the labs and develop your own solutions!

Suggestions from Past Editions of the Course

- See the specific document on the course website
- Rather long but (hopefully) full of good advice
 - Quick CHECKLIST at beginning
 - Structured in sections
- Written after listening to hundreds of oral discussions ...

Web Applications

Frequent Errors in project submissions

Advice coming from previous editions of the course
v. 3.0 (academic year 2025/26)

Advice coming from previous editions of the course
v. 3.0 (academic year 2025/26)

Note:
The project can be submitted in any condition even if the points described in the following are not satisfied. These indications are not a necessary condition to pass the exam (unless otherwise stated). However, in general, addressing all these points should result in a good mark because the most common errors are typically avoided.

QUICK CHECKLIST

- Application must work correctly with more than one client, regardless of the order of client operations
- No unreasonable assumptions have been made on input data, identifiers, etc... if it's doubt, ask
- Preferably no security assumptions on the server side (client, server, both)
- Perform all the necessary authentication/authorization checks in the server
- Use the correct HTTP methods in the API
- Only use JSON components, no XML or DOM methods of any kind (including `JSON.parse()`)
- Clean and well-organized code, avoid excessive usage of `eval()`
- Only use robust components, no local or network resources must be handled (e.g., hosting impossible)
- Reasonable management of errors and error conditions
- Always provide a way to navigate the pages, without using browser back/forward buttons
- Check the project on a case-sensitive system, Windows and MacOS are NOT

APPLICATION LOGIC PROBLEMS which generally lead to loss of points on the final mark of each deliverable

- It is UNACCEPTABLE that the application does not work when two separate users log in at the same time. Such conditions will almost certainly lead to exam failure.
 - With applications are, by their nature, multi-client and multi-user, so this problem means that the student did not understand the fundamental concept underlying the course, thus he/she CANNOT PASS the exam. An example of this type of error is among students who work in a specific client without associating the unique identifier of the user that they are using with the unique identifier given to the user in the database. In these cases, think about whether it is better to keep the information in the database or in the session object itself.
- It is UNACCEPTABLE to make assumptions regarding the value of the identifier present in the database. Such assumption will probably lead to exam failure.
 - For example, if the project requires 4 users, the code must NEVER assume that their identifiers are 1,2,3,4. If done perform mathematical operations on these identifiers, e.g. to derive an address from a user id, identifiers must be considered as properties of the elements of which they are part (e.g., in the case of users, together with the name). In a generic application the identifiers could also be non-continuous and even random.

Resources (fundamentals)

The screenshot shows the MDN homepage with a dark background. At the top, there's a search bar and a "Get MDN Plus" button. Below the header, a large section titled "Resources for Developers, by Developers" features a circular graphic of concentric rings with various symbols like plus signs and asterisks. A "Q" icon is positioned in the center of the rings. Below this, a sub-section titled "Documenting web technologies, including CSS, HTML, and JavaScript, since 2005." is visible. The main content area is titled "Featured Articles" and includes four cards: "CSS Introducing the CSS Cascade", "HTML <dialog>: The Dialog element", "JavaScript — Dynamic client-side scripting Asynchronous JavaScript", and "Web APIs Canvas tutorial".

Mozilla Developer Network
(MDN)
<https://developer.mozilla.org/>

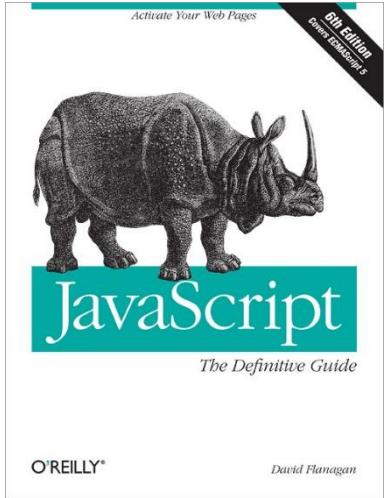
This screenshot shows the "Guides" section of the MDN website. It features a sidebar with categories: HTML, CSS, JavaScript, HTTP, Web APIs, Web Extensions, and Web Technology. The main content area is titled "MDN Learning Area" and lists several learning paths: "Learn web development", "Learn to structure web content with HTML", "Learn to style content using CSS", "Learn to run scripts in the browser", and "Developing extensions for web browsers".

The screenshot shows the React library homepage with a dark background. At the top, there's a search bar and a "Get Started" button. The main title is "React: A JavaScript library for building user interfaces". Below the title, there are three sections: "Declarative", "Component-Based", and "Learn Once, Write Anywhere". The "Declarative" section explains how React makes it painless to create interactive UIs. The "Component-Based" section discusses building encapsulated components. The "Learn Once, Write Anywhere" section highlights that React can also render on the server using Node and power mobile apps using React Native. At the bottom, there's a "LIVE JSX EDITOR" section showing a code editor with a "HelloWorld" component and a "RESULT" panel showing the output "Hello Taylor".

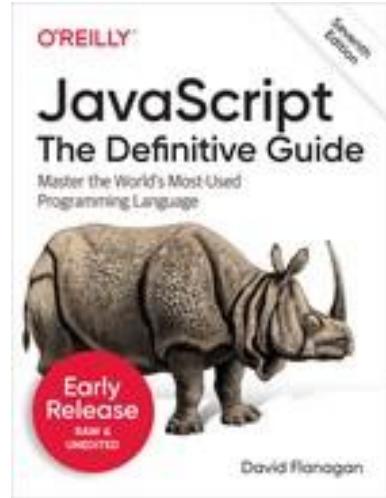
React Library
<https://reactjs.org/>

This screenshot shows the detailed documentation for the React library. The left sidebar includes sections for "MAIN CONCEPTS" (Hello World, Introducing JSX, Rendering Elements, Components and Props, State and Lifecycle, Handling Events, Conditional Rendering, Lists and Keys, Forms, Lifting State Up, Composition vs Inheritance, Thinking In React), "ADVANCED GUIDES", "API REFERENCE", "HOOKS", "TESTING", "CONCURRENT MODE (EXPERIMENTAL)", and "CONTRIBUTING". The main content area is titled "TUTORIAL" and contains chapters such as "Before We Start the Tutorial", "What Are We Building?", "Setup for the Tutorial", "Option 1: Write Code in the Browser", "Option 2: Local Development Environment", "Help, I'm Stuck!", "Overview", "What Is React?", "Inspecting the Starter Code", "Passing Data Through Props", "Making an Interactive Component", "Developer Tools", "Completing the Game", "Lifting State Up", "Why Immutability Is Important", "Function Components", "Taking Turns", "Declaring a Winner", "Adding Time Travel", "Storing a History of Moves", "Lifting State Up, Again", "Showing the Past Moves", "Picking a Key", "Implementing Time Travel", and "Wrapping Up".

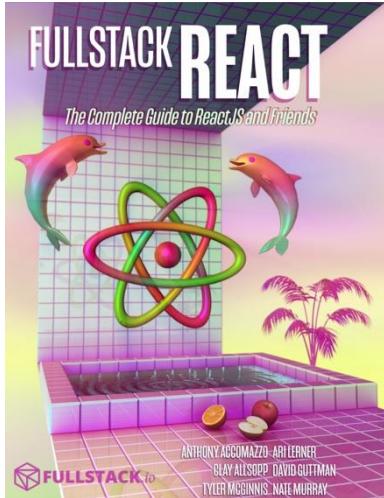
Resources (books)



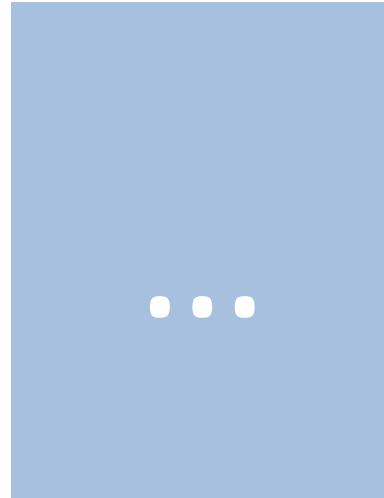
JavaScript: The Definitive Guide,
6th Edition
By David Flanagan
ISBN 978-0596805524
Release Date: May 2011
(not very updated...)



JavaScript: The Definitive Guide,
7th Edition
By David Flanagan
ISBN 978-1491952023
Release Date: July 2020

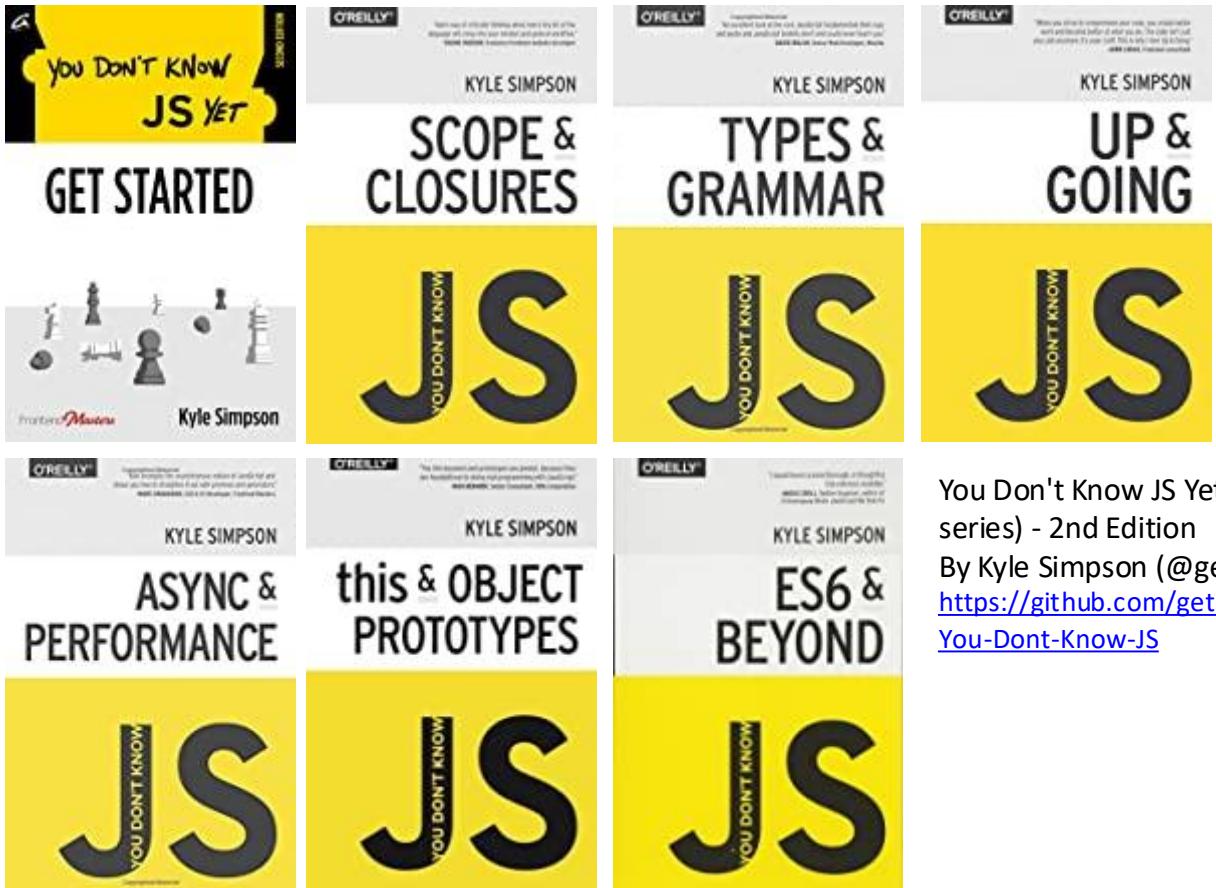


Fullstack React
By Anthony Accomazzo, Nate Murray, Ari Lerner, Clay Allsopp, David Guttman, and Tyler McGinnis
<https://www.newline.co/fullstack-react>
Release: r40 (January 2020)

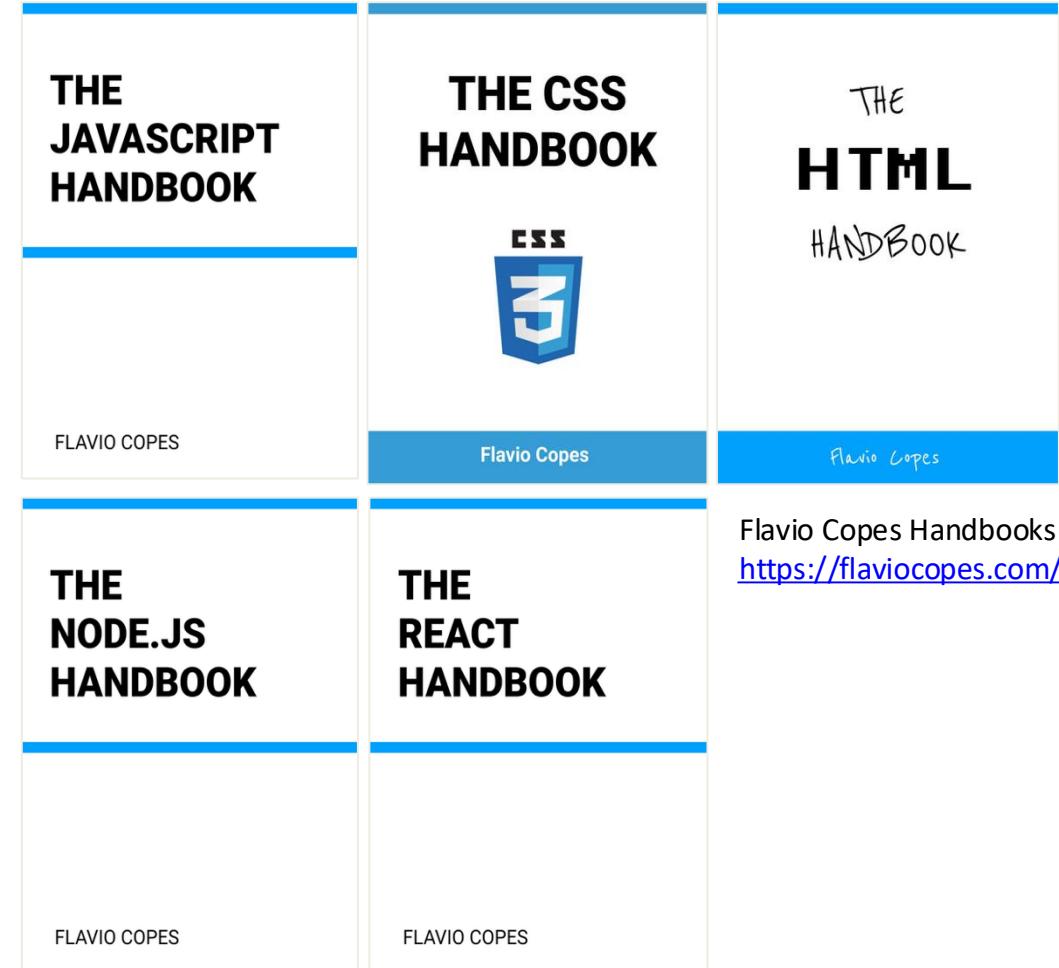


... and many others

Resources (on-line books)

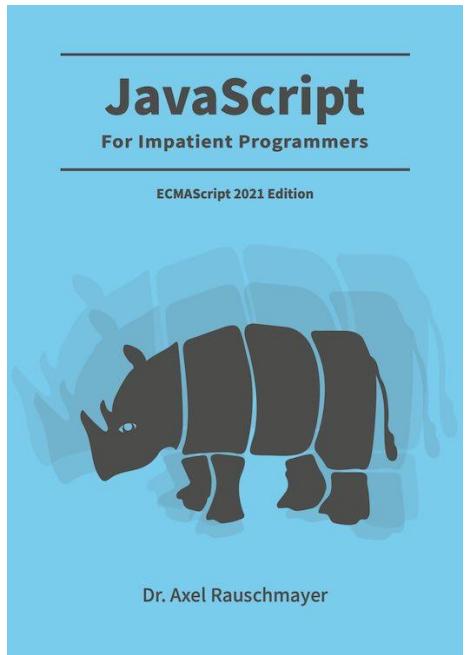


You Don't Know JS Yet (book series) - 2nd Edition
By Kyle Simpson (@getify)
[https://github.com/getify/
You-Dont-Know-JS](https://github.com/getify/You-Dont-Know-JS)



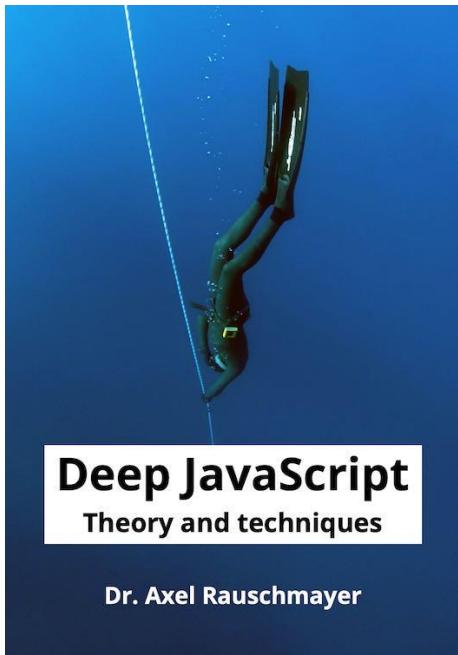
Flavio Copes Handbooks
<https://flaviocopes.com/>

Resources (on-line books)

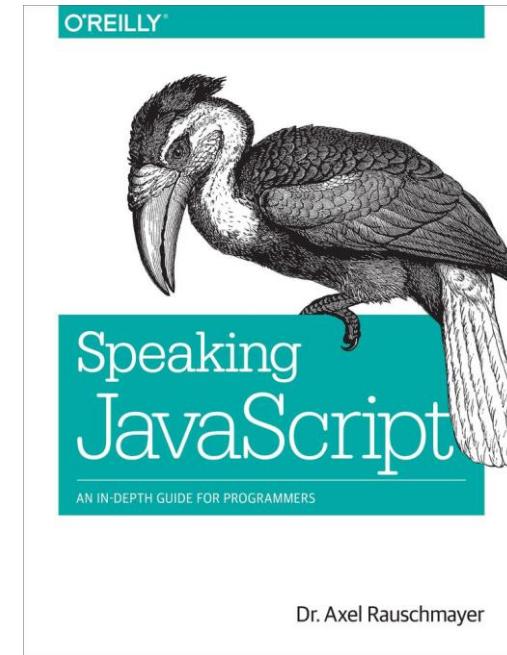


<https://exploringjs.com/impatient-js/index.html>

+

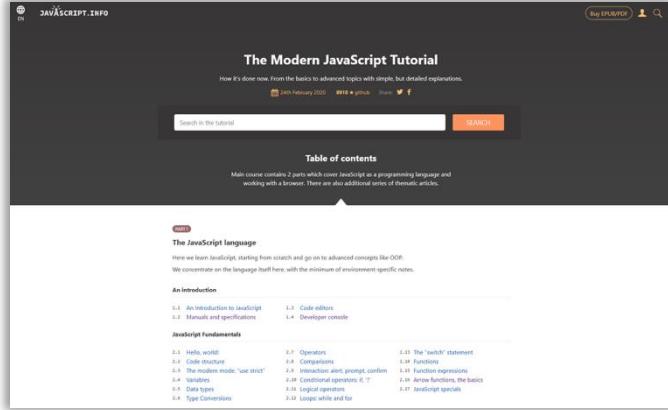


<https://exploringjs.com/deep-js/index.html>

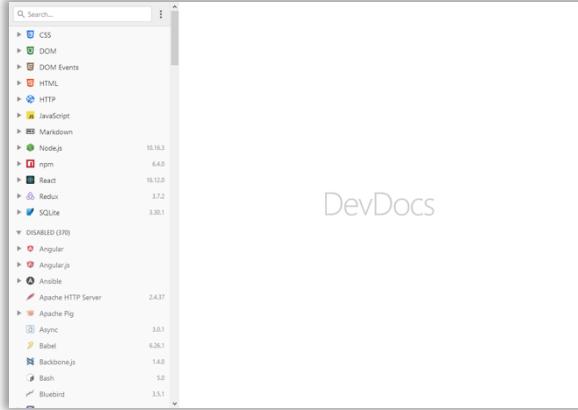


<http://speakingjs.com/>

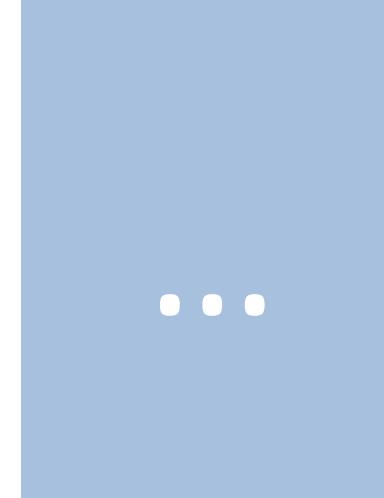
More resources...



The Modern JavaScript Tutorial
<https://javascript.info/>



DevDocs: API Documentation
Browser
<https://devdocs.io/>



... and many others



License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
 - **Share** — copy and redistribute the material in any medium or format
 - **Adapt** — remix, transform, and build upon the material
 - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
 - **Attribution** — You must give [appropriate credit](#), provide a link to the license, and [indicate if changes were made](#). You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
 - **NonCommercial** — You may not use the material for [commercial purposes](#).
 - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the [same license](#) as the original.
 - **No additional restrictions** — You may not apply legal terms or [technological measures](#) that legally restrict others from doing anything the license permits.
- <https://creativecommons.org/licenses/by-nc-sa/4.0/>