

Electronic Business Development — Lab 2 — JavaScript Basics

Contacts

Course Instructor

Dr. Nourhan Hamdi

nourhan.hamdi@giu-uni.de
M2.312

Teaching Assistant

Mr. Nour Gaser

nour.aly@giu-uni.de
M2.419

Teaching Assistant

Mr. Omar Alaa

omar.alaaeldin@giu-uni.de
M2.419

Teaching Assistant

Ms. Mariam Khalid

mariam.khalid@giu-uni.de
M2.414

Teaching Assistant

Mr. Younes Zied

younes.zied@giu-uni.de
M2.506

Teaching Assistant

Mr. Mahmoud Hani

TBD
M2.506



Lab agenda

Duration Topic

10 min	Printing, Variables & Objects
20 min	Operators & Control Flow
10 min	Functions & Callbacks
15 min	Arrays & Iteration
30 min	Async Operations
10 min	HTML, CSS & the DOM (<i>if time allows</i>)



Printing, Variables & Objects

Printing, Variables & Objects

```
const name = "Sara";
let age = 22;

// { name, age } is shorthand for { name: name, age: age } – used everywhere
const student = { name, age, gpa: 3.5, isEnrolled: true };

// name = "Ahmed";      Error: cannot re-assign const

console.log("Student info:", student);
console.log(`${name} is ${age} years old.`);
console.table(student);
console.error("Student record not found");
console.warn("GPA is close to the cutoff");
```



Control Flow

Control Flow: operators & conditionals

```
// operators
const x = 10, y = 3;
console.log("Sum:", x + y);
console.log("Greater?", x > y);
console.log("And:", x > 5 && y < 5);

// if / else + ternary
if (x % 2 === 0) {
  console.log("Even");
} else {
  console.log("Odd");
}
const status = (x > 5) ? "Big" : "Small";
console.log(status);
```

Control Flow: loops

```
// loops
for (let i = 1; i <= 3; i++) {
  console.log("For:", i);
}

let count = 0;
while (count < 3) {
  console.log("While:", count);
  count++;
  if (count === 2) break;
}
```

Before you start any activity

1. Open  **VS Code** — desktop icon, Start menu, whichever is fastest.
2. Create a new folder on your machine, then **File** → **Open Folder** into it.
3. Create one file per activity: `activity1.js`, `activity2.js`, and so on.

Keep everything in this one folder for the whole lab — it's just for practice.

✓ RUNNING YOUR CODE

Open a terminal (**Terminal** → **New Terminal**), then run a file with  **Node.js**:

```
node activity1.js
```

Save the file first, every time.  **Node.js** runs whatever is on disk, not what's shown in the editor. After every change: save, then run it again.

Tips:

- ↑ (up arrow) in the terminal reuses your last command — no need to retype `node activity1.js`.
- **Ctrl+F5** in  **VS Code** runs the open file without a debugger.



Variables & control flow

1. Create variables for your favorite movie: `title`, `releaseYear`, `watchedRecently`.
2. Create an object `movie` from those variables (try the shorthand), and print it with `console.table`.
3. Create a variable `score` — your rating of the movie, 0–100.
 - If `score` ≥ 50 → print "Pass", otherwise "Fail".
 - Use a ternary to set `status` as "Good" if `score` ≥ 80 , otherwise "Needs work".
4. Write a for loop that prints the numbers 1–5, then a while loop that prints them 5–1.



Functions & Callbacks

Functions & Callbacks

```
function greet(name) {  
    return `Hello, ${name}!`;  
}  
console.log(greet("Sara"));  
  
// Same thing, written as an arrow function – this shorter shape is everywhere in JS  
const greetArrow = (name) => `Hello, ${name}!`;  
console.log(greetArrow("Sara"));  
  
// Functions can be handed to other code to be called later – that's a callback  
setTimeout(() => {  
    console.log("This runs one second later");  
}, 1000);
```



Arrays

Arrays

```
const numbers = [1, 2, 3, 4, 5];

numbers.forEach(n => console.log("Value:", n));

const doubled = numbers.map(n => n * 2);
console.log("Doubled:", doubled);

const evens = numbers.filter(n => n % 2 === 0);
console.log("Even numbers:", evens);
```



Functions & arrays

1. Create an array `grades = [40, 55, 78, 90, 100]`.
2. Write a function `passed(grade)` that returns `true` if `grade ≥ 50`.
3. Use `grades.filter(passed)` to create a new array of passing grades — you're handing your own function to `filter` as a callback.
4. Write the same check again, this time as an inline arrow function: `grades.filter(g ⇒ g ≥ 50)`.
5. Use `map` to create a "+5 bonus" version of `grades`, and `forEach` to print the final list.



Async Operations

The one everyone finds confusing — let's take it slow



Sync vs. async: the mental model

- **Normally**, code runs top to bottom, one line at a time, in order.
- **Async** is for one thing: something that takes a long time relative to everything else — a timer, a network request — that you don't want to freeze the rest of your program waiting for.
- JavaScript kicks that slow thing off, keeps running the rest of your code, and comes back to handle its result once it's ready.

When do you use which?

- Independent logic → let it run async, don't wait on it.
- Dependent logic (you need the result before you can continue) → wait for it, with `.then()` or `await`.
- Most real code is a mix of both.

Async Operations

```
console.log("Start");

setTimeout(() => {
  console.log("Timer finished");
}, 2000);

fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then(res => res.json())
  .then(data => console.log("Fetched post:", data.title));

console.log("Still going...");

async function loadUser() {
  const res = await fetch("https://jsonplaceholder.typicode.com/users/1");
  const user = await res.json();
  console.log("Fetched user:", user.name);
}

loadUser();

console.log("End of script");
```



Trace the async script

Before running the script on the previous slide:

1. With your group, write down the order you think the `console.log` lines will print in.
2. Mark which ones you're sure about, and which ones you're just guessing.
3. Run it. Compare to your prediction — where were you right, where were you surprised?

Hint: everything **synchronous** finishes first, in order, before any async result comes back.



HTML, CSS & the DOM

Time permitting — a preview, not a deep dive

HTML, CSS & the DOM

```
<style>
  button {
    padding: 10px 20px;
    background: blue;
    color: white;
    border-radius: 6px;
  }
</style>

<button id="myBtn">Click Me</button>
<p id="message"></p>
<script>
  const btn = document.getElementById("myBtn");
  const msg = document.getElementById("message");

  btn.addEventListener("click", () => {
    msg.textContent = "Button was clicked!";
  });
</script>
```

This is just a preview — we'll properly refresh HTML, CSS, and the DOM soon before we start React.



Useful references

VIDEOS

[JavaScript in 100 seconds](#)

[What is asynchronous JavaScript code?](#)

[JavaScript Visualized — Promise Execution](#)

[JavaScript Visualized — Event Loop, Web APIs, \(Micro\)task Queue](#)

PRACTICE

[Try more simple exercises](#)

Questions?

Open floor for any questions

