LECTURE 17

Midterm Review

No new topic today -- a consolidated checkpoint across everything from "what is the web" to your first Express server.

CSC336 Web Technologies

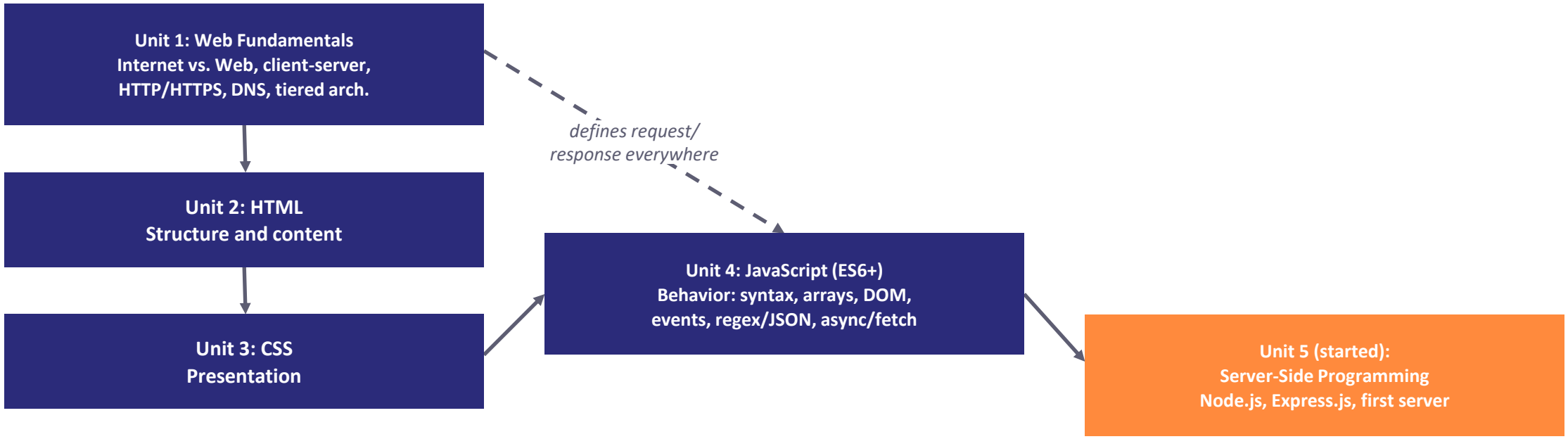
OVERVIEW

In This Review

1. Concept map -- how Units 1 through 5 (so far) connect
2. Unit 1 recap: Foundations of the Web
3. Unit 2 recap: Markup Languages (HTML)
4. Unit 3 recap: Styling with CSS
5. Unit 4 recap: Modern JavaScript (ES6+)
6. Unit 5 recap (so far): Server-Side Programming
7. Self-check questions and study tips

CONCEPT MAP

How the Units Connect



The client-server, request-response model from Unit 1 never went away -- fetch() on the client (Unit 4) now calls the server you just learned to build (Unit 5).

Foundations of the Web

- The Internet is the global network; the Web is one service (HTML, URLs, HTTP) running on top of it
- Client-server request-response model: the server does nothing until asked
- A URL breaks into scheme, host, port, path, query string, and fragment
- HTTP defines the request/response rules; HTTPS adds encryption (TLS)
- DNS translates domain names into IP addresses
- Standards bodies: W3C, WHATWG, ECMA, IETF
- Static vs. dynamic content; MPA vs. SPA vs. PWA
- Tiered architecture: 2-tier, 3-tier, n-tier -- know where client, server, and database sit

Common Gotcha

WARNING

Students often say "the Internet" when they mean "the Web," and confuse a web server (software) with the physical machine it runs on. Be precise with this vocabulary on the exam.

Markup Languages (HTML)

- Elements: opening tags, content, closing tags -- some are void (self-closing), like <img> and
- Semantic HTML (<header>, <nav>, <main>, <article>, <section>, <footer>) describes MEANING, not appearance
- Forms (<form>, <input>, <label>, <select>, <textarea>, <button>) collect user input
- Every <input> should be paired with a <label> (via for/id) for accessibility
- HTML5 added structural/multimedia elements and new input types with built-in validation

Common Gotcha

WARNING

"Semantic" does not mean "styled differently by default" -- most semantic elements look like a plain until you add CSS. Their value is in meaning, not looks.

Styling with CSS

- Box model: content, padding, border, margin, in that order, moving outward -- box-sizing: border-box includes padding/border in width/height
- Positioning: static, relative, absolute, fixed, sticky
- Stacking context and z-index only work on positioned (non-static) elements
- CSS3: transitions, animations, custom properties, media queries
- Flexbox: one-dimensional layout; Grid: two-dimensional layout
- Responsive design: relative units, media queries, flexible layouts, and frameworks (Bootstrap, Tailwind)

Common Gotcha

WARNING

Margin collapsing (adjacent vertical margins combining into one) and the difference between justify-content (main axis) vs. align-items (cross axis) in Flexbox are two of the most frequently missed exam points.

Modern JavaScript (ES6+)

- ES6+ syntax: let/const (prefer over var), arrow functions, template literals, destructuring, spread/rest, classes
- Array methods: map, filter, reduce, find, forEach -- process data without manual loops
- DOM manipulation: document.querySelector and similar, changing content/attributes/classes
- Events: addEventListener, the event object, event bubbling/capturing
- Regular expressions for pattern matching; JSON.stringify/JSON.parse for data exchange
- Async JavaScript: callbacks -> promises -> async/await; fetch() for HTTP requests from JS

Common Gotcha

WARNING

async/await is still promise-based under the hood -- a function marked async always returns a promise, and await only works inside an async function (or at a module's top level). Forgetting to await a promise is one of the most common async bugs.

Server-Side Programming

- Client-side code runs in the browser and is visible/editable by the user; server-side code runs on a machine you control
- Node.js is a JavaScript runtime (built on Chrome's V8 engine) that runs JS outside the browser
- Node.js achieves high concurrency through non-blocking I/O and the event loop
- npm manages reusable packages, tracked in package.json, downloaded into node_modules
- Express.js: express() creates an app, app.get('/path', handler) defines routes, app.listen(port) starts the server

SELF-CHECK

Test Yourself -- Questions 1-6

1. 1. Explain, in your own words, the difference between the Internet and the Web.
2. 2. Draw the request-response cycle for a page with one image and one stylesheet. How many requests happen?
3. 3. What does DNS do, and why can't browsers just use domain names directly?
4. 4. In a 3-tier architecture, name the three tiers and one responsibility of each.
5. 5. What is the difference between semantic and non-semantic HTML? Give an example of each.
6. 6. Explain the box model. width:200px, padding:10px, border:2px -- rendered width under default vs. border-box?

SELF-CHECK

Test Yourself -- Questions 7-12

1. 7. When would you choose Flexbox over Grid, and vice versa?
2. 8. What is the difference between map and forEach? Why choose one over the other?
3. 9. Explain what a promise represents, and describe its three states.
4. 10. Why is client-side validation alone never sufficient for security? What must also happen on the server?
5. 11. What does "non-blocking I/O" mean, and why does it let Node.js handle many requests on one thread?
6. 12. Walk through what happens when you run node index.js on a basic Express app, from start to a browser response.

STUDY TIPS

How to Study for the Midterm

- Don't just re-read passively -- close your notes and explain each concept out loud, or write it from memory, then check yourself
- Rebuild a couple of small code examples from scratch (from memory, not copy-paste) -- typing it yourself cements it far better
- Focus extra time on the "Common gotcha" boxes -- they call out the mistakes students make most often
- Group topics by layer: what happens in the browser (HTML/CSS/client-side JS) vs. on the server (Node.js/Express)
- If a topic still feels shaky, revisit that lecture's "Try It Yourself" exercise and actually do it again

KEY TAKEAWAYS

What the Midterm Spans

1. Unit 1: the client-server, request-response model and the vocabulary (URL, HTTP/HTTPS, DNS, tiered architecture) that everything else builds on.
2. Unit 2: HTML structure, semantics, and forms -- meaning over appearance.
3. Unit 3: CSS presentation -- the box model, positioning, Flexbox/Grid, and responsive design.
4. Unit 4: modern JavaScript -- ES6+ syntax, array methods, the DOM, events, and asynchronous code with promises/async/await.
5. Unit 5 (so far): server-side programming -- Node.js's event loop, npm, and your first Express routes.
6. Mixing up what happens in the browser vs. on the server, under time pressure, is the single most common source of lost marks -- review the "Common gotcha" boxes one more time before the exam.