▲ Exam Format & MCQ Trap

100 pts / 3h / 1×A4 both sides. MCQ 30pt (15Q×2) · Fill-in 10pt (5Q×2) · T/F 10pt (10Q×1) · Open 50pt (7–8Q).

MCQ TRAP (30 pts): 1–4 answers may be correct — tick EVERY correct one. 33% of past MCQs (10/30) were multi-select. Tick too few → partial/zero; tick a wrong one → score drops/zeroes. **Only tick what you can defend; don't guess-tick all.**
Past multi: V1 Q2,3,10,13,14,15 · V2 Q1,10,11,14.
"always/only/never/exclusively" ⇒ usually **FALSE**.
Time: MCQ 30m · Fill 8m · T/F 12m · Open 75m · Review 15m.

HTTP Methods

Mth	S	I	C	Use
GET	Y	Y	Y	read; 200/404
HEAD	Y	Y	Y	headers only
OPT	Y	Y	–	allowed verbs
POST	N	**N**	–	create; 201
PUT	N	Y	–	full replace
PATCH	N	**N**	–	partial upd
DELETE	N	Y	–	200/204/404

Safe = no state change. **I**dempotent = N calls=1. Safe⇒Idempotent (not vice versa). **POST & PATCH NOT idempotent.**

HTTP Status Codes
2xx success: **200** OK · **201** Created (+Location) · **204** No Content (DELETE).
3xx: 301 perm · 302 temp · 304 Not Modified.
4xx CLIENT: **400** bad syntax · **401** no auth · **403** authed-denied · **404** missing · 405 method n/a · 409 conflict · 422 unprocessable.
5xx SERVER: **500** crash · 502 bad gateway · 503 down · **504** upstream timeout.

Which 4xx?
need auth→**401** · logged in but denied→**403** · URL/id absent→**404** · syntactically broken→**400** · wrong verb→405. **500** = unhandled exception (no try/catch). **500 ≠ timeout** (504).

REST

6 constraints (Code-on-Demand OPTIONAL): **1.** Client–Server · **2.** Stateless (no session on server) · **3.** Cacheable (`Cache-Control`, ETag) · **4.** Uniform Interface (URIs, HATEOAS, self-descriptive) · **5.** Layered · **6.** Code-on-Demand (OPTIONAL).
REST = **STYLE, not protocol** (Roy Fielding, 2000).

Endpoint design rules
- Nouns, **plural**: `/users` not `/getUsers`
- Verb = HTTP method, URL = resource
- Hierarchy: `/users/42/messages`
- Filter/sort/page: `?role=admin&sort=age`
- Version: `/api/v1/...`
- **NO verb in query/path**

(e.g. `?method=delete`)
HATEOAS: OPTIONAL; responses include links for next actions. **Resource:** anything URI-addressable (noun).

Endpoint analysis (V2 Q38):
`GET /p/:p/docs/:id?method=delete`
Flaws: (1) GET must be SAFE — mutation forbidden; (2) verb-in-query = RPC anti-pattern; (3) query refines, doesn't act.
Fix: `DELETE /projects/:p/documents/:id` → 200/204; 404 absent; 401/403 auth.

POST /users/42/messages = create MSG for user 42 (NOT create user, NOT list).

HTML

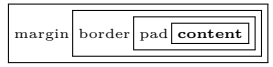
`<!DOCTYPE html>` 1st line. `<html lang>` → `<head>` (meta, title, link, script) + `<body>`.
Semantic HTML5: `header`, `nav`, `main`, `section`, `article`, `aside`, `footer` (+ `figure`, `details`, `summary`). Meaningful for a11y+SEO.
Void / empty tags (no close): `<br>` `<img>` `<hr>` `<input>` `<meta>` `<link>`.
Block (div p h1-6 section ul form): new line, full width, w/h respected, vertical margins.
Inline (span a em strong img label): in-flow, content-width, ignore w/h, only horiz margins.
inline-block: in-flow + w/h respected.
Inline CANNOT contain block.

Forms
`<form action="/x" method="post">` action=where, method=how.
`<label for="u">User</label>` `<input id="u" name="u">` — **for=id binds label**.
Input types: text, email (validates @), password (masked), number (min/max/step), date, time, datetime-local, range, color, url, tel, search, file, checkbox (multi), radio (share name=group), submit, reset, hidden, button.
Attrs: required, placeholder, pattern, min, max, step, readonly, disabled, autocomplete.
`<fieldset><legend>` group; `<select><option>`; `<textarea>`.
alt on `<img>`: (1) screen readers / a11y, (2) fallback if image fails, (3) SEO text. **NOT** for alt sources — that's `srcset`.

CSS

Rule-set: `selector { property: value; }`.

Box Model (in → out)
content → padding → border → margin



box-sizing: `content-box` (default) w/h is CONTENT only, pad+border ADD.
`border-box` w/h INCLUDES pad+border. Padding does **NOT decrease** content area (V2 Q26 FALSE).

Units
em = parent font-size (compounds). **rem** = root (`html`) font-size. **%** = parent dim.
vw/vh = 1% viewport. px absolute. em/rem BOTH relative (V2 Q4: **b**).

Selectors
`#id` id · `.cls` class · `p` type · `*` universal · `[attr=v]` attr · `:hover` pseudo-class (1:, state) · `::before` pseudo-element (2::, part).
Combinators: `A B` descendant · `A>B` child · `A+B` adjacent sib · `A~B` general sib.

Specificity (high → low)
inline > #id > .class/[attr]/:pc > type/::pe. Tie → last wins (source order). Cascade: **UA < User < Author < Author!important < User!important**.

Inheritance
Inherit: color, font-*, text-align, line-height. **NOT inherit:** border, margin, padding, width, background. **Direct match ALWAYS beats inheritance** (regardless of specificity — V2 Q34).

Display vs Visibility
`display:none` out of flow, no space
`visibility:hidden` invisible, KEEPS space
`display:` block/inline/flex/grid/none

Position & Flex/Grid
Position: static (default) · relative (offset, creates ctx) · absolute (vs nearest positioned) · fixed (vs viewport) · sticky (rel then fixed). Center: `top:50%;left:50%;transform:translate(-50%,-50%)`.
Flex (1D): `display:flex; flex-direction:row|col; justify-content; align-items; gap; flex:g s b.`
Grid (2D): `display:grid;grid-template-columns:1fr 2fr 1fr;gap;grid-area.`
Responsive: `<meta name="viewport" content="width=device-width">` + `@media(max-width:600px){...}`.

Specificity puzzle (V2 Q34):
`#mySect{color:purple} p{color:orange}`
`<h1>` in #mySect = **purple** (inherits from parent, no direct rule).
`<p class="sunYellow">` = **orange** (direct p-rule beats inheritance; unused class ignored).

JavaScript Core

var = function-scoped, hoisted to `undefined` (avoid).
let = block-scoped, reassign, TDZ.
const = block-scoped, no reassign (obj contents mutable). Use `const>let>var`.

== vs === / null vs undefined

`'100' == 100`	TRUE (coerces)
`'100' === 100`	FALSE
`null == undefined`	**TRUE**
`null === undefined`	**FALSE**
`[] == false`	TRUE
`NaN == NaN`	FALSE (use `Number.isNaN`)

Falsy: `false 0 " null undefined NaN`. **Else truthy.**

Hoisting / TDZ
`console.log(x); var x=1` → undefined.
`console.log(y); let y=1` → **ReferenceError** (TDZ).
`foo(); function foo(){}` works (fn decl fully hoisted).
`bar(); var bar=function(){}` → TypeError.

this Binding (bound at CALL, not def)

method call (`o.f()`)	this = o
plain call (`f()`)	global / strict: undefined
arrow fn	**no own this** (lexical)
`new C()`	this = new obj
`f.call(o)`	this = o

Arrow CANNOT use `.bind/.call/.apply` to change this. **Cannot** `new` an arrow. Arrow in Node module top: `this = module.exports` (empty) so `this.x` is `undefined`.

Callbacks / Promises / async-await / fetch
Callback = fn passed as arg, invoked later (V2 Q20).
Promise states: pending → fulfilled/rejected; `.then().catch().finally()`.
async/await = sugar on promises; `await` pauses; wrap try/catch.
fetch does NOT reject on 4xx/5xx — check `res.ok`. **axios DOES reject.**

Event Loop — Output Prediction
Sync stack first; macrotasks (`setTimeout`, I/O) AFTER stack empty. `setTimeout(fn,0)` runs LAST.

V2 Q37 trace:
```
func1{func2{setT(B,0);log C};
log A}
func3=(cb)=>{cb();log D};
func3(()=>log E); log F;
```
Order: C → A → E → D → F → B (B last via timer queue).

IIFE (V2 Q7): `(function(){...})();` runs at definition, isolates scope.
AJAX (V2 Q6): **technique** (NOT a language/library) for async send/receive WITHOUT page reload. Modern: JSON, fetch/axios.

ES6 Modern Syntax
Arrow: `(x)=>x*2`; no own this; can't be `new`-ed.
Spread/Rest: `[...a, ...b]`, `{...o, x:1}`, `fn(...args)`.
Destructure: `const {x, y=1} = obj; const [a,b] = arr;`
Template: `` `hi ${name}` ``.
Optional chain: `obj?.a?.b`.
Nullish coal: `x ?? 'def'` (only null/undef triggers; 0/'' pass). `||` triggers on ANY falsy.
Array: `.map .filter .reduce((a,x)=>a+x,0) .find .some .every .includes .sort((a,b)=>a-b)`.

DOM

What: tree, in-memory, language-neutral model of HTML built by browser at parse. Root = `document`. `DOMContentLoaded` fires when HTML parsed (before images).

JS DOM API
Select: `getElementById`, `querySelector` (CSS, 1st), `querySelectorAll`.
Read/Write: `textContent` (safe), `innerHTML` (XSS risk; get & set), `getAttribute/setAttribute`.
Structure: `createElement`, `appendChild`, `el.remove()`.
Style: `el.style.x`, `el.classList.add/remove/toggle`.
Events: `el.addEventListener('click',h)`.

Events: Capture / Bubble
Capture (root→target) → Bubble (target→root, default). 3rd arg `true`=capture. `e.preventDefault()` stops default (submit/link). `e.stopPropagation()` stops bubble. `e.target`=clicked.

"Cannot read properties of null" = querySelector fired before element exists. Fix: `defer` on `<script>`, move `<script>` to end of body, OR wrap in `DOMContentLoaded`.

Node.js / Express

Node.js (V1 Q16): JS runtime on Google **V8**, runs JS **outside browser**.
Single-threaded event loop + **non-blocking I/O** (V2 Q29 FALSE: not multi-threaded). **No window/DOM**; has `fs`, `http`, `path`. Top-level `this` = `module.exports` (NOT global). libuv thread pool handles some I/O underneath.

npm / package.json
npm (V2 Q19) = Node Package Manager.
`package.json`: metadata, scripts, **dependencies** (runtime) vs **devDependencies** (tests/lint).
Semver MAJOR.MINOR.PATCH: `^1.2.3`=any 1.x.x; `~1.2.3`=only 1.2.x; `1.2.3`=exact.
node_modules auto-gen, DO NOT commit. **package-lock.json** DO commit.

Modules
ESM (course): `export const add=...; import {add} from './m.js';`
CommonJS (legacy): `module.exports={add}; const {add}=require('./m');`

Express Cheat Lines
```
import express from 'express';
const app=express();
app.use(express.json());    // parse JSON body
app.use(express.static('public'));
app.get('/u/:id',(req,res)=>{
   const id=req.params.id; // :id
   const q=req.query.sort; // ?sort=
   res.status(200).json({id});});
app.post('/u',(req,res)=>{
   const u=req.body; // needs json()
   res.status(201).json(u);});
app.listen(3000);
```

Middleware
Sig: `(req,res,next)`. MUST call `next()` or send response (else hangs). Runs in registration order.
Error MW: 4 args `(err,req,res,next)`.
```
app.use((req,res,next)=>{log;next();});
const validate=(req,res,next)=>{
  if(!req.body.name) return res.status(400)
    .json({error:'name required'});
  next();};
app.post('/x',validate,handler);
```
res helpers: `res.send()`, `.json()`, `.status(c)`, `.cookie()`.
req: `params` (/u/:id) · `query` (?q=hi) · `body` (JSON, needs `json()`) · `get('X-Foo')` · `cookies`.

Testing

Pyramid (bottom up): Unit (many, 1 fn) → **Integration** (modules together) → **System/E2E** (whole app; Playwright/Cypress/Selenium) → **Acceptance** (user view).
Order: Unit → Int → Sys → Accept (V1 Q27 & V2 Q28 FALSE because they reverse it).
Levels = how much (unit/int/E2E). **Types** = what aspect (functional, UI, perf, a11y, security).

Test Doubles (VEFF convention)
Stub: replace external resource + canned data (V1 Q14: a,d).
Mock: simulate behavior + verify calls/args.
Spy: wrap real fn, observe calls, original still runs.
Course Stub = "replace external resource" AND "mimic fns". V2 Q13: **c** (simulate complex modules).

TDD / BDD / Regression
TDD (V2 Q17): Red (fail) → Green (min pass) → Refactor. Tests FIRST.
BDD: Given-When-Then.
ATDD: acceptance from user view, before code.
Regression: test written AFTER bug fixed to reproduce it; prevents return.

Jest + Supertest
```
describe('add',()=>{
  beforeEach(()=>resetData());
  it('adds',()=>{
    expect(add(2,3)).toBe(5);});});
```
Matchers: `toBe(===)`, `toEqual` (deep), `toStrictEqual`, `toMatchObject` (subset), `toContain`, `toHaveProperty('a.b',v)`, `toHaveLength`, `toThrow`, `toHaveBeenCalledWith`.
Hooks: `beforeAll/Each`, `afterAll/Each`.
`jest.fn()`=mock fn; `jest.mock(m)`=auto-mock; `jest.spyOn(o,m)`=spy.
Supertest pattern:
```
const r = await request(app)
  .post('/api/v1/tasks')
  .set('Authorization','Bearer X')
  .send({title:'x'});
expect(r.statusCode).toBe(201);
```
Pass **app**, not running server. Always `await`.
Coverage = Statements/Branches/Functions/Lines. 100% ≠ **good tests** (happy-path only).

3 tests for DELETE /tasks/:id (V1 Q37)
(1) **200** happy: seed, DELETE, body==deleted, gone on GET.
(2) **400** bad input: `/tasks/abc`, err 'taskId must be integer'.
(3) **404** not found: `/tasks/99999`, err 'does not exist'.

Security

OWASP Top 10 (2021): A01 Broken Access Control (#1, IDOR), A02 Cryptographic Failures (plaintext/MD5/weak TLS), **A03 Injection** (SQL/NoSQL/XSS/cmd), A04 Insecure Design, A05 Misconfiguration, A06 Vulnerable Components, A07 Auth Failures, A08 Software Integrity, A09 Logging Failures, A10 SSRF.

XSS (V1 Q26 TRUE)
Inject client scripts. **Stored** (in DB, every viewer) / **Reflected** (URL echoes) / **DOM** (JS innerHTML w/ user data).
Defenses: output encode, `textContent` NOT `innerHTML`, CSP header, HttpOnly cookies.

SQL Injection
`"SELECT * FROM u WHERE n='"+in+"'"` with input `' OR '1'='1 -` → all rows / bypass.
Fix: parameterised: `db.query('...WHERE n=$1',[n])`; ORM; validate; least privilege.

AuthN vs AuthZ (V2 Q9: b)
AuthN = WHO you are. **AuthZ** = WHAT you can do. AuthN first.
MFA factors: *know* (pw) · *have* (token/phone) · *are* (biometric). SMS weakest.

Basic Auth (V2 Q8: d)
`Authorization: Basic <base64(user:pass)>`.
base64 = encoded NOT encrypted. Insecure without HTTPS.

Sessions vs Tokens vs JWT
Session: opaque ID cookie, server stores state. Cookie flags: **HttpOnly** (JS can't read, blocks XSS cookie theft) · **Secure** (HTTPS only) · **SameSite=Lax/Strict** (blocks CSRF).
JWT: `header.payload.sig` (base64url). Stateless. SIGNED (HS256/RS256) not encrypted → **payload READABLE**. Verify: pin alg, check exp/iss/aud, never `alg:'none'`.

Passwords / Replay / HMAC
Storage: slow adaptive hash **bcrypt** (cost 10–12), scrypt, argon2. NEVER MD5/SHA256 raw.
Salt = per-user random, stored w/ hash (defeats rainbow tables). **Pepper** = server-wide secret outside DB.
Replay attack: captured signed req resent.
Defense (V2 Q27 TRUE): short-lived **timestamp + nonce** + HMAC over (method+path+body+ts).
Security by obscurity (V2 Q30 TRUE phrasing): relying only on hidden details. WEAK alone — use defense-in-depth.
CSRF defenses: SameSite cookies, CSRF token, check Origin/Referer.

HTTPS/TLS
HTTP over TLS, port 443. Provides **confidentiality, integrity, authenticity** (NOT availability). Handshake: clientHello → serverHello+cert → verify CA/domain/expiry → key exchange → symmetric session. **HSTS** forces HTTPS.

Networking

TCP (V1 Q1: c): **reliable, ordered, error-checked** delivery, stateful (3-way handshake: SYN → SYN-ACK → ACK; close FIN/ACK x2).
UDP: fast, unordered, unreliable (streaming, DNS).
IP: addressing+routing. IPv4 4×8-bit (127.0.0.1=localhost). IPv6 huge. Private: 10/8, 172.16/12, 192.168/16. CIDR /24 = 24-bit net.
DNS: host → IP. Hierarchy: root . → TLD `.com` → authoritative → record (A/AAAA/CNAME/MX/TXT). Recursive

resolver caches w/ TTL.
Ports: HTTP 80, HTTPS 443, SSH 22, DNS 53, FTP 21, SMTP 25, POP3 110, IMAP 143.
OSI 7: Physical, Data Link, **Network** (NOT three-tier!), Transport, Session, Presentation, Application.
URI: `scheme://host[:port]/path[?query][#frag]`. Query `?k=v&k2=v2` (V1 Q29 TRUE).
HTTP req: `METHOD /path HTTP/1.1` + headers + body. **Res:** status-line + headers + body.
HTTP stateless — cookies restore state via `Set-Cookie:` / `Cookie:`.

Architecture

Three-tier (V2 Q10: a,c,d — NOT Network!): **Presentation** (HTML/CSS/client JS) → **Business Logic** (Node/Express, REST) → **Data** (DB). Only adjacent layers talk; no skipping; no upward calls. Network = OSI, NOT three-tier.
Layered: easy to exchange/add layers.
Caching CAN reduce reliability (stale).
Server-side (V1 Q31 / 6pt template)
Code runs on server (Node), not browser. Processes requests, hits DB, returns HTML/JSON.
Advantages (pick 2): (1) security — code/creds hidden, (2) access to server resources (DB,FS), (3) offloads heavy CPU, (4) SEO-friendly, (5) centralised updates, (6) enforce runtime version.
Client-side (V2 Q14: a,c)
Runs in browser. **Advantages:** visible/changeable in devtools (a), offloads server work (c). **NOT hidden** (b false), **cannot enforce JS version** (d false).

Linting (V1 Q7 / V2 Q35)

Static analysis of source code WITHOUT executing — enforces style + finds errors. **Not** minification (a) or compilation (c) or live debug (d).
ESLint rules: `no-unused-vars`, `eqeqeq`, `no-var`, `prefer-const`, `semi`, `indent`, `no-console`, `no-unreachable`, `no-dupe-keys`, `no-undef`, `curly`, `quotes`, `camelcase`, `no-empty`, `max-len`.
Config: `.eslintrc.json`. Run `npx eslint .` or `npm run lint`. Integrates editor + pre-commit + CI. **Benefits:** catch bugs pre-runtime, uniform style, reduce PR nits.

Week 1–12 Gotchas

W1 Net: Internet = hardware+TCP/IP; **WWW = info-space on it** (different). IP alone ≠ host ID (need subnet). DNS hostname→IP.
W2 HTML: `<br>`/`<img>`/`<hr>` are VOID (no close). `<b>` doesn't mean bold semantically. `<tbody>` OPTIONAL. All tags can have `id` (global).
W3 CSS: `+` = ADJ sib (not general; use `~`). `::before` double colon. Class>id is FALSE (id more specific). Cascade UA<User<Author<!Author<!User.
W4 HTML+CSS: `<label for="id">` binds. `for` attribute on label, NOT input. Form `action` URL, `method` POST/GET.
W5 JS: `7+"10"→"710"` (string coerce); `"710"+5→"7105"`; `27==="27"→FALSE`. AJAX async default; `async:false` blocks UI.
W6 Test: Levels = unit/int/E2E. Types = func/UI/perf/a11y. Perf+UI = TYPES not LEVELS. Jest CAN test async.
W7 Node: arrow at top-level Node module: `this` = `module.exports` (empty) so `this.x` `undefined`. Regular fn needed for obj methods using `this`. `http` built-in; no `window`.
W8 REST: 3-tier = Presentation/BusinessLogic/Data (NOT Network). Caching CAN reduce reliability (stale). Code-on-Demand OPTIONAL.
POST /users/42/messages creates MSG.
W9 Test: Stubs replace external resources + mimic fns, NOT live services. Supertest = HTTP-endpoint testing. 100% coverage ≠ good tests.
W10 Sec: Basic Auth = base64, insecure w/o HTTPS. Session IDs unpredictable + server-stored. Tokens scoped+expirable+stealable. HMAC defeats tampering BUT NOT replay alone (need timestamp/nonce).
W11 Sec deep: A01 Broken Access Control #1. TLS = CIA (no Availability). bcrypt cost 10–12. **Salt** per-user / **Pepper** server-wide. Cookie flags: HttpOnly/Secure/SameSite. JWT signed not encrypted.
W12 Exam: 100pt / 3h / 1×A4 (both sides). 30 MCQ + 10 FIB + 10 T/F + 50 Open. **1+ MCQ answers possible (must select ALL).** Selecting all to guess → wrong answers can zero score.

Past Exam — MCQ Multi-Select Catalog (highest value)

V1 Q2 alt purposes (3 right): **a, b, d** — SEO + fallback + screen readers. NOT `srcset` (c).
V1 Q3 form tags: a, b — `<form>`+`<input>`. NOT `<section>` (semantic), NOT `<table>` (data).
V1 Q10 arrow vs trad: a, b — no own `this` + concise. NOT "must return" / no params.
V1 Q13 POST /users/42/messages: a, c — creates msg + does NOT create user. NOT retrieval, NOT list.
V1 Q14 Stubs: a, d — replace external resources + mimic fns. NOT make tests complex / connect live.
V1 Q15 GET /genres test: b, d — array ≥1 element + first has id+name. NOT 404, NOT only id.
V2 Q1 idempotent: a, c, d — GET, PUT, DELETE (+HEAD, OPTIONS). NOT POST.
V2 Q10 three-tier: a, c, d — Presentation, Business Logic, Data Access. **NOT Network** (that's OSI).
V2 Q11 JS facts: a, c — weakly typed, multi-paradigm. NOT strongly, NOT client-only.
V2 Q14 client-side: a, c — changeable + offloads server. NOT hidden, NOT can enforce JS version.

Past Exam — Lightning Round

V1 MCQ Q1–15 single-answer + multi
1→**c** (TCP=reliable+ordered) · 2→**a,b,d** · 3→**a,b** · 4→**b** (selector) · 5→**b** (#id) · 6→**b** (breakpoints) · 7→**b** (analyse code) · 8→**b** (false; == coerces) · 9→**a** (Unit) · 10→**a,b** · 11→**c** (404) · 12→**d** (undefined) · 13→**a,c** · 14→**a,d** · 15→**b,d**.
V2 MCQ Q1–15
1→**a,c,d** · 2→**c** (nav links) · 3→**a** (A; browser this=window) · 4→**b** (em=parent, rem=root) · 5→**b** (POST) · 6→**b** (async, no reload) · 7→**a** (executes when defined) · 8→**d** (Basic Auth) · 9→**b** (identity vs access) · 10→**a,c,d** · 11→**a,c** · 12→**a** (deletes user 39) · 13→**c** (simulate modules) · 14→**a,c** · 15→**b** (bad syntax).
V1 + V2 Fill-in (5 each)
V1: 16=**Node.js** · 17=**GET & PUT** (also DELETE/HEAD/OPTIONS) · 18=**==** · 19=**header & footer** (any semantic pair) · 20=**visibility & display**.
V2: 16=**Stateless & Client-Server** (any pair) · 17=**TDD** · 18=**Mocks & Stubs** (VEFF: stubs=replace ext.) · 19=**npm** · 20=**Callbacks**.
V1 T/F (Q21–30)
21=**F** (JS can be anywhere) · 22=**T** (403=authed, denied) · 23=**T** (`let` for mutable) · 24=**F** (`<link>` in head) · 25=**T** (innerHTML get & set) · 26=**T** (XSS injects client scripts) · 27=**F** (Integration BEFORE System) · 28=**F** (null ≠ undefined strictly) · 29=**T** (? marks query) · 30=**T** (HATEOAS optional).
V2 T/F (Q21–30)
21=**T** (IP=unique host id) · 22=**T** (regression reproduces fixed bug) · 23=**F** (500≠timeout; 504) · 24=**F** (use `let` not `var`) · 25=**T** (`<img>` void) · 26=**F** (padding doesn't shrink content with content-box) · 27=**T** (timestamp+hash blocks replay) · 28=**F** (Integration covers more than module-to-module) · 29=**F** (Node single-threaded) · 30=**T** (security by obscurity = hiding impl).

Past Exam — Open Q Templates (memorize structure)

V1 Q31 Server-side (6pt)
Code runs on server (Node), not browser; processes requests, hits DB, returns HTML/JSON. **2 advs (any 2):** (1) security (code+creds hidden), (2) DB/FS access, (3) heavy CPU offload, (4) SEO, (5) central updates, (6) enforce runtime ver.
V1 Q32 Box Model (8pt)
4 concentric layers in→out: **content**(w/h) → **padding**(inner space) → **border** → **margin**(outer space, transparent). Total W = content+2pad+2border+2margin (content-box). `border-box` → w INCLUDES pad+border. **Draw the diagram!**
V1 Q33 Minimal Form (8pt) — code template

```
<h1>Welcome to My Simple Page</h1>
<p>Basic webpage with a form.</p>
<form>
   <label for="n">Name:</label>
   <input type="text" id="n" name="n">
   <label for="e">Email:</label>
   <input type="email" id="e" name="e">
   <input type="submit" value="Submit">
</form>
```

V1 Q34 4xx vs 5xx (6pt)
4xx = CLIENT fault (fix on client): 400 bad syntax, 401 need auth, 403 authed-denied, 404 missing.
5xx = SERVER fault: 500 unhandled exc, 502 bad gateway, 503 down, 504 upstream timeout. **Give 2 examples per class.**
V1 Q35 Injection (8pt)
Untrusted input interpreted as code because no sanitise/parameterise. **SQLi ex:** `"WHERE name='"+in+"'"` with `' OR '1'='1 -` → all rows / auth bypass. **Outcomes:** data theft, auth bypass, DB drop, OS exec. **Fix:** prepared statements, ORM, validate input, least privilege.
V1 Q36 Express DELETE comments (6pt)
(1) import Express+bodyParser; (2) create app, base path, port; (3) `app.delete` route; (4) validate `:taskId` integer → 400; (5) findIndex by id; (6) <0 → 404 with msg; (7) splice removes+keeps deleted; (8) `res.status(200).json(deleted)`.
V1 Q37 = see "3 tests for DELETE"
V2 Q31 Inline vs Block (8pt)
Block (div, p, h1-6, section, ul, form): new line, full width, w/h respected, vertical+horiz margins.
Inline (span, a, em, strong, img, label): in-flow, content-width, ignores w/h, only horiz margins.
Inline CANNOT contain block. `inline-block` mixes both. **Give 3 of each.**
V2 Q32 Selector definition (6pt)
Selector defines WHICH elements a rule-set applies to. **Examples:** type `p{}`; class `.btn{}`; id `#main{}`; attribute `[href]{}`; descendant `ul li{}`; pseudo `a:hover{}`.
V2 Q33 DOM (6pt)
Tree, in-memory, language-neutral model of HTML built by browser at parse time. `DOMContentLoaded` fires when parsed. **JS interacts via:** select (getElementById, querySelector), read/write (textContent, innerHTML, get/setAttribute), structure (createElement, appendChild, remove), style (style.*, classList), events (addEventListener).
V2 Q34 Specificity = see CSS box
V2 Q35 Linting = see Linting section
V2 Q36 Resource (5pt)
Resource = URI-addressable entity (noun). Example book: `GET /api/v1/books` list · `GET /api/v1/books/42` read · `POST /api/v1/books` create · `PUT /api/v1/books/42` replace · `PATCH .../42` update · `DELETE .../42` remove. Uniform Interface: same URL, verb differs.
V2 Q37 Event Loop (8pt) = see JS Core box
V2 Q38 Endpoint flaws = see REST box

Code Templates (copy-paste)

Jest unit test

```
import {add} from './math.js';
describe('math',()=>{
 beforeEach(()=>db.reset());
 it('adds',()=>{
  expect(add(2,3)).toBe(5);});
 it('mock',()=>{
  const m=jest.fn()
   .mockReturnValue(7);
  expect(m(1)).toBe(7);
  expect(m).toHaveBeenCalledWith(1);
 });
 it('throws',()=>
  expect(()=>add('a',1)).toThrow());});
```

Supertest GET/POST/DELETE

```
import request from 'supertest';
import app from '../app.js';
it('GET',async()=>{
 const r=await request(app)
  .get('/api/v1/users');
 expect(r.status).toBe(200);
 expect(Array.isArray(r.body))
  .toBe(true);});
it('POST 201',async()=>{
 const r=await request(app)
  .post('/api/v1/users')
  .send({name:'Al'});
 expect(r.status).toBe(201);
 expect(r.body)
  .toHaveProperty('id');});
it('DEL 404',async()=>{
 const r=await request(app)
  .delete('/api/v1/users/9999');
 expect(r.status).toBe(404);});
```

Express app + error MW

```
const app=express();
app.use(express.json());
app.get('/u/:id',(req,res)=>{
 const id=parseInt(req.params.id);
 if(!Number.isInteger(id))
  return res.status(400)
   .json({error:'bad id'});
 res.status(200).json({id});});
app.use((err,req,res,next)=>{
 res.status(500)
  .json({error:'server'});});
app.listen(3000);
```

SQL: vulnerable vs safe

```
// VULN - string concat
db.query("...WHERE n='"+n+"'");
// input ' OR '1'='1 -
// SAFE - parameterised
db.query("...WHERE n=$1",[n]);
```

JWT / bcrypt / HMAC

```
const p=jwt.verify(tok,SECRET,
 {algorithms:['HS256']});
const h=await bcrypt.hash(pw,12);
const ok=await bcrypt.compare(pw,h);
const sig=crypto
 .createHmac('sha256',SECRET)
 .update(method+path+body+ts)
 .digest('hex'); // +timingSafeEqual
```

fetch async/await + chain

```
async function load(){ try{
 const r=await fetch('/api/x');
 if(!r.ok) throw new Error(r.status);
 return await r.json();
}catch(e){console.error(e);}}
// chain:
fetch('/api/x')
 .then(r=>r.ok?r.json()
  :Promise.reject(r.status))
 .then(d=>console.log(d))
 .catch(e=>console.error(e));
```

DOM event + closure

```
btn.addEventListener('click',e=>{
 e.preventDefault();
 e.stopPropagation();});
function counter(){let n=0;
 return()=>++n;} // ()() -> 1,2,3
```

Flex / Grid / Media

```
.center{display:flex;
 justify-content:center;
 align-items:center;gap:8px}
.grid{display:grid;gap:10px;
 grid-template-columns:1fr 2fr 1fr}
@media(max-width:600px){.grid{
 grid-template-columns:1fr}}
```

curl

```
curl -X POST host/api/users \
 -H 'Content-Type:  application/json' \
 -d '{"name":"X"}'
curl -X DELETE host/api/users/1 \
 -H 'Authorization:  Bearer TOK'
```

Flags: `-i` headers · `-v` verbose · `-o` save.

Per-Week Pitfalls (graded traps)

W1 Net: POST is NOT idempotent. Basic Auth is base64 ENCODED not encrypted. `/N` prefix counts bits not bytes (/24 = 3 octets).
W2 HTML: don't style with `<b>`/`<font>`/`<center>` (use CSS). Tables = tabular data only, not layout. `id` must be UNIQUE (use class for groups). Invalid nesting `<p><b>x</p></b>` unreliable. Don't use `<br>` for spacing.
W3 CSS: `+` = next sib only, `~` = all later sibs. id>class>element. User `!important` beats author `!important`. `em` compounds when nested; `rem` doesn't. `position:absolute` with no positioned ancestor anchors to body.
W4 Layout: px widths break responsive (use %). `<button>`/`<img>` are inline. `:disabled` is a pseudo-class tied to the `disabled` attr (a class won't disable). Flexbox is 1D — use Grid for 2D. Need `<600px` breakpoint. Add a NEW specific rule, don't amend old ones.
W5 JS: missing `let`/`const` = implicit global. `var` leaks out of if/for blocks. `0 == " "` is true. plain-fn callback loses this (use arrow/bind). `innerHTML` + user data = XSS. Forgot `e.preventDefault()` → form reloads.
W6 Test: set up DOM before DOM test or `querySelector` = null. Test behavior not private vars. `<script>` in head w/o `defer` runs before DOM. Test edge cases (0, neg, null, empty, wrong type). Don't write many slow E2E. `await` async tests.
W7 Node: top-level `this` = module (not global). Regular-fn callback loses `this`. Can't `new` an arrow / use it as method needing `this`. ESM only (no mixing `require`). `express.json()` BEFORE POST/PUT or `req.body` undefined. Call `next()` or hang.
W8 REST: no verbs in URI (/`getUsers` bad). Plural collections. Never 200 on error. PUT replaces whole (idempotent), PATCH updates fields.
W9 Test: `toBe`=reference (use `toEqual` for objects). `await` Supertest or test passes silently. Reset shared state in `beforeEach`. Mock external services. `spy.mockRestore()`.
W10 Deploy: test validator + not-found cases, not just happy path. Assert body not just status. Auth header on protected routes (else 401). Wrong HMAC = 'wrong hash' not 401. `export default app` for Supertest. Call /api not /.netlify/functions/api. Push to GitHub before deploy. `netlify dev` first.
W11 Sec: don't set `ACAO:*` with credentials. Reject JWT `alg:'none'`, pin alg. Don't log passwords/tokens/bodies. No default creds / debug / stack traces in prod.
W12 Exam: `<script>` need not be in head; `<link>` belongs in head. Int BEFORE System. Validate types (`parseInt`+`Number.isInteger`). try/catch or unhandled →500. `.cls`=class `#id`=id. Selector (not Property/Value) defines what a rule targets. **Don't tick-all on MCQ to guess.**

Practice Quiz — Multi-Select Drills (Wk 1–10)

Correct options (abbreviated) — the multi-select skill that costs most. Tick ALL.

W1 Networking Basics
• Which are correct regarding the TCP/IP protocol stack? ⇒ **A request is routed to a target host thro...** · **Host names are translated to IP addresses...** · **The IP protocol uses IP addresses to iden...**
• Which are true regarding HTTP? ⇒ **HTTP is stateless** · **Both the request and response of an HTTP...** · **Everything sent via HTTP is in clear text** · ...
• Which are true regarding HTTP response codes? ⇒ **The 200 status code means a request was s...** · **Codes starting with 4 indicate client errors**
• Cookies are used to add stateful behaviour to web sites. What is true regarding coo... ⇒ **Cookies are just clear-text fields saved...** · **Whenever a client sends an HTTP request t...**
W10 Security
• Which are true regarding the following HTTP request header: Authorization: "Basic Z... ⇒ **It is a security risk to send this header** · **The header is used for authentication** · **The HTTP basic auth method is used** · ...
• Which are true regarding tokens? ⇒ **Tokens can define scopes or permissions** · **Tokens can easily be stolen if not protected** · **Tokens are used instead of login credentials** · ...
• Which are true for request signing? ⇒ **If a request changes** · **The server also knows the secret used for...** · **Parts of the request are hashed using a s...**
• Which are true regarding session-based authentication? ⇒ **The browser sends a session ID with each...** · **The server stores information about the u...** · **Session IDs must be unpredictable**
• Why can vulnerable dependencies be a security risk? ⇒ **Dependencies can introduce vulnerabilitie...** · **Third-party libraries may contain known v...**
W2 HTML
• Which of the following statements are true regarding accessibility ? ⇒ **Good accessibility also helps search engi...** · **Meta tags can be used to give background...**
W3 CSS
• Which of the following statements is/are true? ⇒ **id is more specific than tag** · **class is more specific than tag**
• Which of the following statements is/are true for inline elements? ⇒ **The element takes as much width as needed** · **The width/height properties are ignored**
• Responsive Web Design ensures that web sites adapt to different... ⇒ **Input methods** · **Screen sizes** · **Screen orientation** · ...
• Which are true with respect to devices with virtual viewports ? ⇒ **Mobile phones often have a virtual viewport** · **The html meta tag with viewport attribute...** · **Media queries for small screen sizes migh...** · ...
W5 JavaScript
• Which of the following statements is/are true for server-side execution? ⇒ **Server-side resources can be accessed** · **Specific language versions can be enforce...**
• When assigning variables, which of the following is true? ⇒ **const is best when assigning a value that...** · **using let and const should be preferred o...** · **let is best when assigning a variable tha...**
W6 Testing and Debugging
• Which are correct regarding integration testing? ⇒ **Integration tests often involve more comp...** · **In integration testing**
• Which of the following are typical characteristics of unit tests? ⇒ **They test small** · **They are usually fast to run** · **They are easy to automate**
• Which of the following are examples of testing types rather than testing levels? ⇒ **Performance testing** · **UI testing**
• Which tools are commonly used to debug JavaScript directly in the browser? ⇒ **Chrome DevTools** · **Firefox DevTools**
• Which statement about Jest is correct? ⇒ **Jest can test DOM interactions using a si...** · **Jest can be used for unit testing JavaScr...** · **Jest supports testing asynchronous code**
W7 Server-side JS
• Which are true about the Node.js runtime? ⇒ **window and document are undefined** · **Node.js includes built-in modules like http**
• What happens when you run: npm install express ⇒ **Dependencies can later be reinstalled wit...** · **package.json is updated automatically** · **node_modules is created**
W8 REST
• Which are true for a system following the layered architecture style? ⇒ **It is easy to exchange layers** · **It is comparably easy to add new layers**
• Which are true regarding REST? ⇒ **HATEOAS allows exploration of a RESTful A...** · **There is no direct interaction with data** · **RESTful APIs have to use the correct HTTP...**
• Which are true regarding the following endpoint (assuming that it is a RESTful API ... ⇒ **An HTTP GET request to the given URL retu...** · **An HTTP GET request to the given URL does...** · **The given request is idempotent**
• When creating APIs, a good way to get started is to ⇒ **Discuss possible solutions with collegues** · **Write down the purpose of each endpoint** · **Write down all the required endpoints**

Exam Strategy

MCQ (30pt, 30 min): read EVERY option; multiple may be right; only tick what you can defend. Watch "only/always/exclusively/never" — usually FALSE.
FIB (10pt, 8 min): 1–2 word terms (Node.js, TDD, npm, ==, semantic tags, visibility/display). Spell exactly.
T/F (10pt, 12 min): negations & absolutes red-flag FALSE. Trust pattern memory.
Open (50pt, 75 min): ALWAYS include a concrete example (SQLi snippet, box-model diagram, HTML form code, curl/req). **Examples = half the points.**
Code-reading: draw call stack; mark `setTimeout(...,0)` as LAST in event-loop order; remember this rule (method=obj, plain=global, arrow=lexical).
Endpoint-analysis: method? URL noun? hierarchy? status? Call out: GET-for-mutation, verb-in-URL, singular noun, no /:id.
Order: do FAST ones first (T/F → MCQ singles → FIB → MCQ multis → Open). Save 15 min review.