API Development Cheatsheet

📡 HTTP & REST Basics

Method	Purpose	Example Endpoint
GET	Read data	`/notifications`
POST	Create data	`/notifications/send`
PATCH	Update partial	`/users/:id`
DELETE	Delete data	`/notifications/:id`

Status Codes

- `200 OK` – Successful response
- `201 Created` – New resource created
- `400 Bad Request` – Invalid input
- `401 Unauthorized` – No valid token
- `404 Not Found` – Resource doesn't exist
- `500 Server Error` – Unexpected crash

🔷 TypeScript Snippets

```
// Basic types
let id: string = "abc123";
let count: number = 5;

// Function with types
function sendSMS(phone: string): boolean {
  return true;
}

// Interface
```

```
interface Notification {
  id: string;
  message: string;
  sent: boolean;
}

// Union
type Status = "sent" | "pending" | "failed";
```

⚙️ Hono Quick API

```
import { Hono } from "hono";

const app = new Hono();

app.get("/health", (c) ⇒ c.text("OK"));
app.post("/sms", async (c) ⇒ {
  const body = await c.req.json();
  return c.json({ status: "received" });
});
```

Middleware Example

```
app.use("*", async (c, next) ⇒ {
  console.log("Request received");
  await next();
});
```

🧪 Zod Validation

```
import { z } from "zod";

const smsSchema = z.object({
```

```
  to: z.string().min(10),
  message: z.string().max(160),
});

type SMSInput = z.infer<typeof smsSchema>;
```

🧬 OpenAPI Route with Zod

```
import { createRoute } from "@hono/zod-openapi";

export const sendSMS = createRoute({
  method: "post",
  path: "/sms",
  tags: ["SMS"],
  request: {
    body: {
      content: {
        "application/json": {
          schema: smsSchema,
        },
      },
    },
  },
  responses: {
    200: {
      description: "SMS sent successfully",
      content: {
        "application/json": {
          schema: z.object({ success: z.boolean() }),
        },
      },
    },
  },
});
```

🗃️ API Module Structure

```
modules/
└── sms/
    ├── controller.ts
    ├── routes.ts
    ├── schema.ts
    ├── service.ts
```

🧮 Drizzle ORM Basics

```
// schema.ts
import { pgTable, varchar, uuid } from "drizzle-orm/pg-core";

export const notifications = pgTable("notifications", {
  id: uuid("id").defaultRandom().primaryKey(),
  message: varchar("message", { length: 160 }),
});

// db query
await db.insert(notifications).values({
  message: "Hello from SMSX!",
});
```

🔐 Auth Middleware (simplified)

```
app.use("/api/*", async (c, next) => {
  const token = c.req.header("Authorization")?.split(" ")[1];
  if (!token || !isValidJWT(token)) {
    return c.json({ error: "Unauthorized" }, 401);
  }
```

```
  await next();
});
```

📚 Debugging Tips

Tool	Usage
`pnpm dev`	Start dev server
`pnpm test`	Run tests with Vitest
`console.log`	Use in controllers
`pnpm lint:fix`	Auto-fix style issues

📦 Common Scripts

```
pnpm install       # Install deps
pnpm dev           # Dev server
pnpm build         # Compile to dist
pnpm lint:fix      # Fix lint issues
pnpm test          # Run tests
pnpm drizzle-kit push # Sync DB schema
```

🧠 Mental Model

> Each API is a contract → schema-defined, type-safe, self-documented, and testable.