

Assume you are expert in k6 for performance testing then all its Programmable skills and experienced in its official documentation and a good guide for beginners then Provide all the necessary topics to cover to best in coding in k6 for performance testing from beginner to expert divided by stages and what theory to learn beforehand

Great! Since you want to **master K6 for performance testing** from **beginner to expert**, including coding and theory, here's a **complete roadmap**, based on K6's official documentation, real-world experience, and best practices.

Before You Begin K6 – Prerequisite Theory

To effectively use K6, understand the following core **theoretical concepts**:

Basic Theory:

1. **What is Performance Testing?**
 - Load, stress, spike, endurance, and scalability testing
2. **Client vs Server-side performance**
3. **HTTP Protocol Basics**
 - Methods (GET, POST, PUT, DELETE)
 - Status codes (200, 404, 500, etc.)
 - Headers, cookies, authentication
4. **JavaScript Fundamentals**




- Variables, functions, loops, conditionals
 - ES6 features like arrow functions, async/await
5. **JSON basics** – for request/response payloads
 6. **REST APIs** – CRUD operations, request/response formats
 7. **Basic Terminal/CLI usage** – for running K6
-

K6 Programming & Scripting Roadmap (Beginner to Expert)

Stage 1: Beginner – K6 Basics

Goal: Understand what K6 is and how to run your first test.


Topics:

-  What is K6 and why use it?
-  Installing K6 (via brew, choco, Docker)
-  `k6 run` and `k6 init` commands

Basic script structure:

```
import http from 'k6/http';  
import { sleep } from 'k6';
```

```
export default function () {  
  http.get('https://test.k6.io');  
  sleep(1);  
}
```

-
-  `default` function – Virtual User behavior

-  Using `sleep()` for pacing

Practice:

- Write a simple GET request
 - Run from CLI using `k6 run script.js`
-

Stage 2: Core Scripting – Intermediate

Goal: Write detailed load test scripts with multiple HTTP methods and realistic flows.




Topics:

-  HTTP module methods: `get()`, `post()`, `put()`, `del()`

Checking response status:

```
import { check } from 'k6';
```

```
check(response, {  
  'status is 200': (r) => r.status === 200  
});
```

-
-  `group()` – for organizing test flow
-  Parameters and headers:
 - Query strings
 - Custom headers (e.g., `Authorization`)
-  Sending data in body (JSON, form-data)

✓ Reading external data using `open()`:

```
const data = JSON.parse(open('./data.json'));
```

-
- ✓ Tags & Thresholds
- ✓ Environment variables: `__ENV`

✓ K6 Options block:

```
export const options = {  
  vus: 10,  
  duration: '30s',  
};
```

-

📌 **Practice:**

- Simulate login + fetch dashboard
- Simulate failed login & handle errors

🔄 Stage 3: Data-driven Testing & Modularization

Goal: Create reusable, maintainable scripts using modules and external data.

📘 **Topics:**

✓ Reuse functions using modules:

```
import { login } from './utils.js';
```

-
- ✓ Data-driven tests using CSV or JSON
- ✓ Custom metrics: `Trend`, `Counter`, `Gauge`, `Rate`
- ✓ Tags for detailed metrics

✓ Setup and Teardown:

```
export function setup() { }  
export function teardown(data) { }
```

-

📌 Practice:

- Login using data from JSON file
- Record and assert response time (Trend)

Stage 4: Advanced Load Testing

Goal: Simulate real-world scenarios with complex load patterns.

📘 Topics:

✓ Stages configuration:






```
stages: [  
  { duration: '1m', target: 10 },  
  { duration: '2m', target: 50 },  
  { duration: '1m', target: 0 },  
]
```


- - ✓ Ramping VUs vs Constant Load
 - ✓ Handling cookies & sessions
 - ✓ Check payload size, headers
 - ✓ HTTP/2, TLS options
 - ✓ Rate-limiting / retry logic
 - ✓ Execution Scenarios (VUs per scenario)
-

Stage 5: Distributed, Cloud, and CI Integration

Goal: Scale and automate your performance testing.

Topics:

-  Using **k6** **c1oud** with Grafana Cloud
-  CLI flags for automation
-  GitHub Actions, GitLab CI, Jenkins integration
-  Dockerizing K6 scripts
-  Visualizing results in InfluxDB + Grafana

 Thresholds for pass/fail criteria:






```
thresholds: {  
  http_req_duration: ['p(95)<500'],  
}
```





-

Stage 6: Expert Level – Plugins, Extensions & Real-world Projects

Goal: Extend K6, integrate with complex workflows, and simulate production-like tests.

Topics:


-  Building and using K6 extensions (**xk6**)
-  Kafka, Redis, gRPC testing via xk6
-  Running browser-based tests (**k6/browser**)
-  WebSocket testing
-  Complex auth (JWT, OAuth 2.0)




-  Parameterizing tests with high-scale data
 -  Simulating spikes and load balancers
 -  Writing helper functions for:
 - Random data generation
 - Load balancing URLs
 - Test orchestration across teams
 -  Test result interpretation and tuning backend systems
-

BONUS: Key K6 Concepts to Master

Concept	Importance
<code>http.batch()</code>	Parallel requests
<code>check()</code>	Validating response
<code>fail()</code>	Force test failure
<code>group()</code>	Organize script
<code>options.thresholds</code>	Pass/fail criteria
<code>scenarios</code>	Complex execution models
<code>metrics</code>	Deep performance insight
<code>setup/teardown</code>	Test initialization/cleanup

Resources

-  Official Docs: <https://k6.io/docs/>

-  Test Examples: <https://github.com/grafana/k6>
 -  xk6 Extensions: <https://k6.io/blog/tags/xk6>
 -  Grafana K6 Cloud: <https://k6.io/cloud>
-

Final Suggestion:

Build your **own test suite** with:

- Modular login/logout
- Scenario-based scripts
- Setup with test users and cleanup
- Trend visualizations + pass/fail reports

Would you like a GitHub-style template project to get started or some real-world test case examples next?