Absolutely! Here's an **expert-level roadmap** for mastering **K6 performance testing using JavaScript**, broken down **stage by stage** — from **beginner to advanced**, covering **every critical concept, method, and function** you need to know to become a **K6 expert**.

✅ Stage 1: Basics of Performance Testing & K6 Setup

🎯 Goals:

- Understand performance testing
- Install and run K6
- Write your first test

📚 Topics to Learn:

1. **What is Performance Testing?**
    - Load Testing
    - Stress Testing
    - Soak Testing
    - Spike Testing
2. **K6 Introduction**
    - What is K6 and why use it?
    - K6 vs JMeter vs Gatling
3. **Installation & Setup**
    - Install K6 (Windows/Linux/Mac)

    - Install via Docker or Homebrew

4. **First Test**
    - Writing a simple script
    - Running the test using `k6 run script.js`

Basic Script Structure

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

export default function () {
  let res = http.get('https://test.k6.io');
  check(res, {
    'status is 200': (r) => r.status === 200,
  });
  sleep(1);
}
```

5.

✅ Stage 2: Core Concepts and Modules

🎯 Goals:

- Understand VUs, iterations, ramping, and thresholds
- Learn core K6 modules

📚 Topics to Learn:

1. **Execution Concepts**
    - Virtual Users (VUs)
    - Iterations

- Duration
- Ramp-up/Ramp-down
- Scenarios (`options.scenarios`)

2. **K6 Modules**
   - `http` — For HTTP requests
   - `check` — For validation
   - `group` — Logical groupings
   - `sleep` — Pauses between actions
   - `options` — Config for stages, thresholds, etc.

Options Configuration

```
export let options = {
  vus: 10,
  duration: '30s',
  thresholds: {
    http_req_duration: ['p(95)<200'], // 95% of requests < 200ms
  },
};
```

3.

Groups and Tags

```
import { group } from 'k6';
group('Login Flow', () => {
  // test logic
});
```

4.

✅ Stage 3: Advanced HTTP and Validations

🎯 Goals:

- Master HTTP request types
- Validate responses
- Capture performance metrics

📚 Topics to Learn:

1. **HTTP Request Types**
    - `http.get(url, params)`
    - `http.post(url, payload, params)`
    - `http.put()`, `http.patch()`, `http.del()`, etc.
    - Batch Requests with `http.batch()`
2. **HTTP Parameters**
    - Headers
    - Query Params
    - Cookies
    - Request body (JSON, form data)

Validations & Checks

```
 check(res, {
  'is status 200': (r) => r.status === 200,
  'body size is > 1000 bytes': (r) => r.body.length > 1000,
});
```

3.

Handling JSON

```
let data = JSON.parse(res.body);
console.log(data.message);
```

4.

✅ Stage 4: Thresholds, Custom Metrics, and Tags

🎯 Goals:

- Implement advanced result monitoring
- Track custom metrics
- Use tags to filter test results

📚 Topics to Learn:

Thresholds

```
thresholds: {
  http_req_duration: ['p(95)<500'], // 95% requests under 500ms
}
```

1.
2. **Tags**
    - Add custom tags to requests
    - Filter test output

```
http.get(url, { tags: { name: 'Homepage' } });
```

3.

Custom Metrics

```
import { Trend, Counter, Gauge, Rate } from 'k6/metrics';

let myCounter = new Counter('my_counter');
myCounter.add(1);

let myRate = new Rate('success_rate');
myRate.add(res.status === 200);
```

4.

Custom Summary

```
export function handleSummary(data) {
  return {
    'summary.json': JSON.stringify(data),
  };
}
```

5.

✅ Stage 5: Parameterization and Data-Driven Testing

🎯 Goals:

- Run tests with multiple sets of data
- Dynamically update request payloads

📚 Topics to Learn:

Using `__ENV` Variables

```
k6 run script.js -e ENV=dev
```

1.
2. **CSV Data Injection**

- Use `papaparse` or `open()` to load CSV/JSON

```
import { SharedArray } from 'k6/data';
const data = new SharedArray('users', () =>
  JSON.parse(open('./data.json'))
);
```

3.

Looping Through Data

```
let user = data[__VU % data.length];
```

4.

✅ Stage 6: Scenarios and Advanced Execution Control

🎯 Goals:

- Use different scenarios
- Emulate complex test flows

📚 Topics to Learn:

1. **Execution Scenarios**
    - `constant-vus`
    - `ramping-vus`
    - `constant-arrival-rate`
    - `ramping-arrival-rate`

- `externally-controlled`

```
export let options = {
  scenarios: {
    spike_test: {
      executor: 'ramping-vus',
      stages: [
        { duration: '2m', target: 100 },
        { duration: '5m', target: 100 },
        { duration: '2m', target: 0 },
      ],
    },
  },
};
```

2.
3. **Execution Control APIs**
   - `__VU` – current virtual user ID
   - `__ITER` – current iteration number

✅ Stage 7: Scripting Complex Flows

🎯 Goals:

- Simulate real-world user behavior
- Manage authentication flows

📚 Topics to Learn:

1. **Chained Requests**
   - Login → Get Token → Access APIs

2. **Cookie and Header Management**
    - Extract tokens and pass headers

```
let token = JSON.parse(res.body).token;
let headers = { Authorization: `Bearer ${token}` };
```

3.
4. **Error Handling**
    - Try-catch logic in tests
    - Conditional flows

✅ Stage 8: Reporting & CI/CD Integration

🎯 Goals:

- Export results
- Integrate with pipelines

📚 Topics to Learn:

1. **Output Formats**
    - CLI, JSON, HTML (via tools like `xk6-summary`, `k6-reporter`)
    - InfluxDB + Grafana for live monitoring

HandleSummary for Reports

```
export function handleSummary(data) {
 return {
   stdout: textSummary(data, { indent: ' ', enableColors: true }),
   'results.json': JSON.stringify(data),
 };
```

}

2.
3. **CI/CD Integration**
    - GitHub Actions
    - GitLab CI/CD
    - Jenkins
    - CircleCI
4. **Docker with K6**
    - Running K6 tests inside Docker containers

✅ Stage 9: Extending K6 with xk6 & Plugins

🎯 Goals:

- Extend K6 functionalities

📚 Topics to Learn:

1. **What is xk6?**
    - Build custom extensions
2. **Popular xk6 Extensions**
    - `xk6-browser` – browser-level testing
    - `xk6-kafka` – Kafka performance testing

    - `xk6-sql` – SQL DB load testing
    - `xk6-websockets` – for websocket testing
3. **Building xk6 Extensions**
    - Use Go to write plugins

Compile with:

xk6 build --with github.com/k6io/xk6-sql

    -

✅ Stage 10: Best Practices & Real-World Projects

🎯 Goals:

- Apply your skills in real projects
- Follow performance testing best practices

📚 Topics to Learn:

1. **Best Practices**
    - Use ramp-up & ramp-down
    - Validate endpoints before load
    - Monitor backend metrics
    - Don’t ignore logs
2. **Common Mistakes**

    - Forgetting thresholds
    - Not simulating think-time
    - Single user load != real-world load
3. **Project Ideas**
    - Load test an eCommerce flow
    - Test REST API vs GraphQL API
    - Spike test a login endpoint

🧠 BONUS: Useful K6 Resources

- 📘 Official Docs
- 🧪 K6 GitHub
- 📺 K6 YouTube Channel
- 🔧 K6 Extensions

If you'd like, I can prepare a **K6 learning checklist**, GitHub project template, or a **practice problem set** based on this roadmap.

Would you like that too?