



Performance Under Real Traffic Workbook

WordPress Speed & Performance

Use these sheets beside the lessons. Record what you observe, explain the change you choose and keep a way to restore the working site.

1. Test Cache Identity
2. Plan a Narrow Purge
3. Trace a Dynamic Request
4. Test After Consent
5. Check Back Navigation
6. Measure Build Cost
7. Module Review

GT Performance is entirely free. You can complete the core exercises without a paid optimization plugin.

Gaurav Tiwari



Test Cache Identity

Confirm that every reusable answer belongs to the request that receives it.

Site / page under review

Date / visitor state / device

Guided Task

1. Choose a public template and 2 inputs that may change its response.
2. Record key or bypass behavior at the origin and edge, where applicable.
3. Request the variations in both orders and check the actual visible output.

Input / Public Variation	Expected Key or Bypass	Observed Output / Evidence

Inputs that can safely share an answer and why

Mismatch, supported correction and rollback

Complete When

Variations return correct content in either request order; sensitive values stay out of the notes.



Test Cache Identity Reasoning Check

Try the questions before reading the answer guidance. This is a teaching scenario, not a measured result from your site.

Scenario

A public listing supports 2 languages and 3 currencies. A unique visitor cookie is also proposed as part of every shared cache key.

How many language-currency combinations are possible?

What must be checked before adding or discarding the visitor cookie?

Answer Guidance

1. There can be $2 \times 3 = 6$ combinations before other variations. These are hypothetical public choices, not a benchmark.
2. Determine whether it changes the response or represents private identity. Adding it may destroy shared reuse; ignoring it may be unsafe. Use bounded supported public variants or bypass.

What I will verify on my own site

Keep passwords, API tokens and private customer data out of this workbook.



Plan a Narrow Purge

Invalidate the affected dependency set without unnecessary cold work.

Site / page under review

Date / visitor state / device

Guided Task

1. Choose a harmless staging edit and list affected displays.
2. Record invalidation scope, subsequent cache decisions and background work.
3. Choose the smallest sufficient purge and define useful warming and retention.

Changed Item / Display	Expected Invalidation	Observed Work / Correctness

Purge scope and evidence for any wider action

Warming budget, stale-data boundary and recovery

Complete When

The chosen scope covers changed content and has a reasoned resource budget.



Plan a Narrow Purge Reasoning Check

Try the questions before reading the answer guidance. This is a teaching scenario, not a measured result from your site.

Scenario

Editing 1 article clears all origin and edge HTML. Popular pages begin rebuilding while an import is still running.

Does this timing alone prove a cache stampede?

What smaller invalidation set should be investigated?

Answer Guidance

1. No. Compare requests, queue state and resource timestamps. Establish whether repeated builds and contention actually occurred.
2. The article and displays that reuse changed values, such as excerpts and listings. Shared-template changes can justify a wider scope.

What I will verify on my own site

Keep passwords, API tokens and private customer data out of this workbook.



Trace a Dynamic Request

Separate initial HTML speed from the cost and rate of a later feature request.

Site / page under review

Date / visitor state / device

Guided Task

1. Record Fetch/XHR requests during 1 search, filter or harmless form action.
2. Count the requests and identify the slowest route without exporting private payloads.
3. Choose a browser or server change while retaining authorization and feature behavior.

Action / Route / Method	Count / Timing / Status	Cause / Correctness / Next Step

Request-rate change versus request-cost change

Token refresh, permissions and rollback checks

Complete When

The feature request is measured separately and its security and required behavior still work.



Trace a Dynamic Request Reasoning Check

Try the questions before reading the answer guidance. This is a teaching scenario, not a measured result from your site.

Scenario

Typing the word hosting sends 1 request per character. The document is a cache HIT, but each search request calls a remote API.

How many character-triggered requests can this create?

Which separate costs need investigation?

Answer Guidance

1. The 7 characters can create 7 requests before corrections. This is a hypothetical interface design.
2. Debouncing can reduce unnecessary request rate. Remote-call reuse and query design address remaining server cost. An HTML cache HIT does not measure either.

What I will verify on my own site

Keep passwords, API tokens and private customer data out of this workbook.



Test After Consent

Find performance cost introduced by consent-dependent or opened features.

Site / page under review

Date / visitor state / device

Guided Task

1. List relevant before-choice, declined, accepted and returning-visitor states.
2. Record the transition and the next required interaction.
3. Inspect request chains and execution; test 1 justified change without weakening consent.

Visitor / Activation State	First Action / Observed Work	Owner / Required Behavior

Duplicate or obsolete work, or a measured initialization problem

Accepted configuration, correctness tests and rollback

Complete When

Relevant states pass, the first action remains usable and the configured consent policy is preserved.



Test After Consent Reasoning Check

Try the questions before reading the answer guidance. This is a teaching scenario, not a measured result from your site.

Scenario

Accepting consent starts several tags and a chat widget. The visitor opens the menu immediately and it responds late.

What sequence should the trace include?

Does delaying the initial tag file establish lower total cost?

Answer Guidance

1. Record before the choice, the activation and the next menu interaction. Inspect the overlapping script work and request initiators.
2. No. The tag can load further resources and expensive initialization. A later download changes timing; measure the resulting interaction and retained features.

What I will verify on my own site

Keep passwords, API tokens and private customer data out of this workbook.



Check Back Navigation

Distinguish document restoration from a new load and verify current state after return.

Site / page under review

Date / visitor state / device

Guided Task

1. Compare ordinary load, reload and Back on a representative page.
2. Run the browser restoration diagnostic and record eligibility or the reported blocker.
3. Check restored forms, cart or account state with controlled staging data.

Navigation / Browser	Restored? / Reported Reason	State / Feature Result

Owner of the reported blocker and a minimal staging correction

Required state refresh and authorization checks

Complete When

Navigation behavior is identified and required visitor state remains correct after returning.



Check Back Navigation Reasoning Check

Try the questions before reading the answer guidance. This is a teaching scenario, not a measured result from your site.

Scenario

A product page restores quickly when the shopper returns with Back, but its cart indicator contains an earlier value.

Which result is successful and which still needs work?

Would increasing the origin HTML lifetime enable bfcache?

Answer Guidance

1. Document restoration may be fast while correctness still fails. Reconcile the required current state and retain server authorization.
2. No. Origin response caching and browser document restoration are different systems. Investigate eligibility and lifecycle handling.

What I will verify on my own site

Keep passwords, API tokens and private customer data out of this workbook.



Measure Build Cost

Evaluate browser savings alongside preparation, background work and artifact reuse.

Site / page under review

Date / visitor state / device

Guided Task

1. Choose a small staging set and record baseline public and uncached tasks.
2. Enable 1 transformation and observe cold, processing and ready states.
3. Compare delivered results, server evidence and repeated invalidation before keeping it.

Page / Build State	Browser Result / Server Evidence	Uncached Task / Decision

What invalidates or prevents reuse of the artifact

Keep, limit or restore decision with evidence

Complete When

Cold and reused results are separated; the retained change has a defensible operating cost.



Measure Build Cost Reasoning Check

Try the questions before reading the answer guidance. This is a teaching scenario, not a measured result from your site.

Scenario

Ready used-CSS results reduce delivery size, but logged-in tasks slow during a large preparation queue.

What evidence would establish resource contention?

Can analyzed CSS savings be treated as compressed browser transfer savings?

Answer Guidance

1. Compare queue and request timestamps with CPU, memory or I/O evidence and matched uncached tasks. Determine whether the cost is temporary or repeatedly renewed.
2. No. They describe different measurements. Record analyzed bytes, browser transfer and operational cost separately.

What I will verify on my own site

Keep passwords, API tokens and private customer data out of this workbook.



Performance Under Real Traffic

Module Review

Choose the next action from your evidence. An unknown value is useful when it tells you what to investigate next.

The most useful evidence I collected

The change I kept, or the reason I deferred it

Required features and private-route checks

Exact rollback action and who owns it

Next investigation and the evidence it needs

Module Sign-Off

I can explain the result, reproduce the comparison and restore the previous working state.

