

A PRACTICAL LEARNING MODULE

Caching without the guesswork

A fictional teaching module about freshness, expiry and observable cache behavior.

40-minute learning session

2 practice exercises



A CLEAR FINISH LINE

What you will be able to do.

01 Explain a freshness window in plain language.

freshness

02 Trace a cache hit and an expired entry.

expiry

03 Choose a policy from an explicit requirement.

choice

Begin with a shared baseline.

Developers who can read a small function and want to reason about stale data.

Before you begin

- Functions and return values
- Comparing timestamps

Why this skill matters

A fast answer is useful only when its freshness matches the task.

LEARNING MAP

- Explain a freshness window in plain language.
- Trace a cache hit and an expired entry.
- Choose a policy from an explicit requirement.

CONCEPT 1 · FRESHNESS

Freshness is a requirement

A freshness window is the maximum acceptable age of a stored result for a stated use.

A CONCRETE EXAMPLE

A training weather widget may tolerate five minutes; a different task may not.

Watch for this misconception

A cached response is not automatically wrong. Its age must be compared with the requirement.

LEARNING MAP

- Explain a freshness window in plain language.
- Trace a cache hit and an expired entry.
- Choose a policy from an explicit requirement.

CONCEPT 2 · EXPIRY

Expiry is a decision rule

An entry is fresh while now is earlier than its expiresAt timestamp.

A CONCRETE EXAMPLE

An entry expiring at minute 15 is a hit at minute 14 and expired at minute 15.

Watch for this misconception

An expiry timestamp is not the same as the time the entry was created.

WATCH ONE WORKED EXAMPLE

Read one value, then read it again

Use fictional integer-minute timestamps. The cache holds result = 42 and expiresAt = 15.

Expected result

At minute 14: { kind: 'hit', value: 42 }.

```
function read(now, entry) {  
  if (entry && now < entry.expiresAt) {  
    return { kind: 'hit', value: entry.value };  
  }  
  return { kind: 'refresh-needed' };  
}  
  
read(14, { value: 42, expiresAt: 15 });
```

Make every step visible.

Follow the supplied example from its starting state to its result.

STEP	STATE	WHY IT CHANGES
Read the clock	now = 14	The caller supplies the comparison time.
Compare	14 < 15 is true	The entry is still inside its freshness window.
Return	kind = hit; value = 42	No refresh is required for this example.

PRACTICE 1 · EXPIRY

Find the exact boundary

✎ EXERCISE

What does `read(15, entry)` return when `entry.expiresAt` is 15? Explain the comparison.

Working constraint

Use the supplied strict less-than rule.

You are done when...

State the returned kind and explain equality.

Exercise ID: boundary · Compare your reasoning with the answer clinic.

PRACTICE 2 · CHOICE

Choose a defensible policy

 EXERCISE

A fictional dashboard accepts results at most two minutes old. Describe the expiry rule without assuming how often users visit.

Working constraint

Do not promise that expiry triggers a background refresh.

You are done when...

Separate maximum accepted age from the action on the next read.

Exercise ID: policy · Compare your reasoning with the answer clinic.

Choose. Then explain why.

At minute 14, an entry expires at minute 15. Which explanation matches the example?

☐ Return the stored value because 14 is earlier than 15.

☐ Refresh because every cached value is stale.

☐ Wait until minute 15 and refresh automatically.

Check answer

Your selection stays in this page. No learner record is sent.

Compare the reasoning.

Find the exact boundary

boundary

It returns { kind: 'refresh-needed' }.

At equality, $15 < 15$ is false. The entry is no longer fresh.

Choose a defensible policy

policy

Set the expiry to the supplied observation time plus two minutes; refresh when a later read finds it expired.

The freshness rule controls acceptance. It does not schedule work by itself.

Take the method with you.

- Define the freshness requirement before choosing a cache rule.
- Trace equality and missing-entry cases explicitly.
- Distinguish a policy from the mechanism that executes it.

REFERENCE MATERIAL

lesson

Fictional lesson worksheet · link not supplied

Original synthetic teaching material in this package; not a report of a production system.