Why PR and code reviews?

It's not just about getting code into the master branch

Chance to catch bugs & code quality issues, early

An exchange of best practices and experiences

Learning opportunity for Reviewee & Reviewer

Without pairing, review is the primary way to make sure knowledge of a feature change not siloed

Why PR and code reviews?

Finally

It's cheaper to do things right the first time

We should improve our code quality

Foster constructive exchange of ideas and knowledge

If pairing intensively on a branch, the PR may be formality because everything was already reviewed during develepment.

Reviewee's Mindset

Be humble, expect comments

It is not YOUR code, it the team's code

Leave the code better than you found it

Respect people's time

Reviewer's Mindset

Be polite

No voice - less context

Use to pass on knowledge of best practices

Don't try to fix everything, avoid nitpicking

Don't just hope for the code to work

Give sincere praise, on good practices

A good starting point would be:
Start with making sure to know what the ticket says
so you can see if code does is meant
If PR more complex
then open project check in its context
for test coverage
resharper/compiler/sonar warnings
etc.

PR size

Make your PRs

How small should be decided

PR should be handled within

Disadvantages of large PRs:

Difficult and long to review
May result in fast approvals
People don't bother reviewing properly
hesitate invest so much time

Benefits of small PRs:

Easier to review
spot bugs and issues
Faster fix
merge
Fewer merge conflicts

Prepare A good PR

Commit atomic self-contained changes

Have on-point commit messages

Unrelated changes? Separate commit

Write useful descriptions and titles

Review your own code first

Helpful questions for the self review:
Are all style commitments fullfilled?
tests runnning well?
build pipelines running
warnings analyzed and removed if possible?
Does code fit into architecture commitments?
changes fullfill requested requirements?
Is implementation best one I can provide?

Commenting On PRs

Avoid "you" => use "we", "me", "code"...

Try use Observe/Impact/Request pattern

If possible or useful add code examples

Don't try to fix everything

Use lables, e.g. "minor: xxx", "major: "hint: xxx"

Add praise to good implementation

Review quickly

Iterations Commenting On PRs

have a maximum on iterations for review e.g. 2 iterations

more iterations will increase the stress level on both sides

Found X issues, raise all at once => then: let people fix them if found more critical BUT: don't change your mind or start challenging the design

Iterations Commenting On PRs

As soon as there's something complicated that maybe needs discussion
Talk to the person directly after review is done
Instead of leaving a comment likely induce stress

Finish what you started and do not disappear from reviewing

PR finished & Post work

You might:
Fix as suggested
the way you prefer
Push back
Regardless: reply to ALL comments
"fixed" or "done"
"done x"
No response may be perceived "you ignored me"
And DON'T "resolve" comment
Let reviewer check and click resolve themselves!