AI Fundamentals

A study guide. *How LLMs actually work — enough to be a calibrated user, not a magical thinker.*

01 Token

The unit of input and output.

An LLM doesn't see characters or words; it sees tokens — chunks roughly the size of a syllable. "validation" is one; "VAlidation" might be three. Pricing, speed, and context size are all measured in tokens.

In Cursor: Cmd+L responses bill in tokens; ~0.75 tokens per English word.

02 Context window

The model's working memory, in tokens.

Everything the model considers at once: your prompt, prior turns, attached files, system rules. Once full, oldest content silently drops. Modern windows: 128K (smaller) to 1M (Opus, Gemini).

In Cursor: @-mentions consume window. Max Mode raises the ceiling, not the quality.

03 Embedding

A numeric fingerprint of meaning.

Text becomes a high-dimensional vector. Similar meanings sit close together. This is how "semantic search" finds relevant code without matching keywords.

In Cursor: Cursor's @codebase indexer uses embeddings to find files related to your prompt.

04 RAG

Retrieval-Augmented Generation.

Don't put everything in the prompt; retrieve only the relevant slices. Search → pick → inject → answer. Lower cost, higher accuracy when the corpus is large.

In Cursor: @codebase, @docs, @web are all RAG. Cursor retrieves; you don't paste.

05 Tool use

The model calling code instead of just generating it.

The model decides "I need to read a file" or "run a test" and emits a function call the runtime executes. The result comes back as the next message. This is what makes Agent agentic.

In Cursor: Every Agent edit, file read, terminal command is a tool call. You see them in the chat.

06 Hallucination

Confident-sounding output that's wrong.

The model is a probability machine. When uncertain, it still outputs something coherent — invented APIs, missing imports, wrong types. It does not know it's wrong.

In Cursor: Always verify imports, file paths, and signatures before merging.

07 System prompt

Hidden instructions that shape every reply.

Sits above your chat. Tools (Cursor) compose it from rules, IDE state, attached files. You don't write it directly; you configure it.

In Cursor: .cursor/rules/ + AGENTS.md + your User Rules all flow into the system prompt.

08 Reasoning model

A model that thinks before answering.

Spends extra tokens on internal deliberation before emitting the answer. Better at planning, multi-step debugging, math. Slower and more expensive — match the use case.

In Cursor: Plan Mode is a UI for reasoning. Opus + thinking flag is the heavy version.

09 Agent

A model with tools, in a loop.

LLM + tool use + a runtime that re-invokes the model after each tool result. Reads files, edits them, runs tests, iterates against feedback. Stops on success, error, or hop limit.

In Cursor: Cursor's Agent mode. The hook system lets you control when the loop ends.

Common myths, decoded

If you hear someone say one of these, gently correct them.

"It understands the code"	It pattern-matches against text it has seen. Treat it like a very fast, well-read junior — not a domain expert.
"More context = better answers"	Up to a point. Beyond that, signal-to-noise drops. Curate context; don't just stuff it.
"Bigger model = better answer"	For nuanced reasoning, often. For mechanical edits, rarely. Try smaller first.
"It will get smarter mid-chat"	It won't. If a conversation has anchored on a wrong premise, start a fresh one.
"Tests prove correctness"	They prove the model's understanding of the spec. The spec might still be wrong.

Mental model: the calibrated user

Trust the output to the same degree you'd trust a smart junior who read the docs last week.

- Ask first, write second. Use Ask for context, Agent for execution.
- Verify the cheap thing: imports, types, file paths. The structural stuff.
- Re-read every diff. The model writes; you ship.
- When stuck, change shape — try a smaller model, a fresh chat, a different framing.
- Save the wins (rules, prompt patterns). Compounding beats one-off cleverness.