— CHAPTER 03 · FOUNDATIONS · BEAT 1 — THE CELLS AND THE GRAPH

Datatypes are the cells. References are the graph.

Last chapter we cracked open a resource and named its six anatomical parts. Inside those parts lives another layer — the datatypes. By the end of this chapter you can read FHIR's five workhorse datatypes, wire resources into a graph with References, and dodge the two classic traps that quietly corrupt real systems.

What you will be able to do

- The five datatypes — the spine of FHIR
- References — the graph that connects them
- The UCUM and MRN gotchas

— CONCEPT · WHY FHIR SHIPS 50+ DATATYPES

Primitives are the atoms. Complex datatypes are built from them.

The core resources stay small precisely because they reuse a fixed set of typed building blocks everywhere instead of reinventing fields. We are not learning all fifty — just the five you meet in almost every payload.

— PRIMITIVES — THE ATOMS

- string
- boolean
- dateTime
- decimal
- uri
- code

Single, indivisible values. Every richer datatype is assembled from these.

— COMPLEX — BUILT FROM PRIMITIVES

- Quantity
- CodeableConcept
- Reference
- Period
- Identifier

The five workhorses for this chapter. Master these and you can read ninety percent of real FHIR.

— DATATYPE 1 · QUANTITY — A NUMBER DONE CAREFULLY

A number, a unit, and the code the math actually trusts.

— A REAL QUANTITY

"valueQuantity": { "value": 70, "unit": "kg", "system": "http://unitsofmeasure.org", "code": "kg" }

`value` is the number. `unit` is a human display string. `system` points at UCUM, unitsofmeasure.org. `code` is the UCUM symbol — like `kg` or `mm[Hg]`. Sub-flavors: Money carries a currency, Age and Duration fix the time dimension, SimpleQuantity drops the comparator.

- code = the math — servers compare on this
- unit = the display — for humans only

— THE SCAR · 70 KG VS 70 LBS

Display says "70 kg". The code underneath says "lb".

Servers compare on `code`, never on the display `unit`. Your screen says one thing, your math does another. Wrong display with right data is still a patient-safety bug. Always trust the `code`.

— DATATYPE 2 · CODEABLECONCEPT — CODED PLUS HUMAN

One concept, possibly many vocabularies, plus the words a human typed.

— A REAL CODEABLECONCEPT

"code": { "coding": [{ "system": "http://loinc.org", "code": "29463-7", "display": "Body weight", "userSelected": true }, { "system": "http://snomed.info/sct", "code": "27113001", "display": "Body weight" }], "text": "Body weight" }

A `coding` array holds zero or more entries; each is a `system`, a `code`, and a `display`. The free-text `text` sits beside them.

— HOW TO READ IT

One Coding when a single terminology is unambiguous. Multiple Codings when the same concept lives in LOINC and SNOMED and a local code — a translation.

`userSelected: true` marks the one the clinician actually picked; the rest are machine-added mappings.

The `text` field is the legal fallback — it wins for display when no code resolves.

Where these codes come from, and how binding strengths bind them, is chapter four.

- **coding[] — zero or more typed codes**
- **text — the human fallback, legally meaningful**

— DATATYPE 3 · REFERENCE — THE TYPED POINTER

A pointer from one resource to another. This is what makes a graph.

rel

RELATIVE

`"reference": "Patient/123"` — resolved against the same server. The everyday form.

abs

ABSOLUTE

`"reference": "https://other/fhir/Patient/123"` — a full URL pointing at another server entirely.

#

FRAGMENT

`"reference": "#prac1"` — points at a contained resource inside the same document.

— TWO SAFETY NETS

`display` is a human-readable label — an unresolvable reference can still say "Dr. Chen". And `Patient/123/_history/2` pins an immutable version, essential for audit.

Resolution order: URL, then fragment, then `identifier` logical match, then `display`.

— THE SCAR · REFERENTIAL INTEGRITY

The server does NOT check the target exists.

FHIR will happily let you reference a Patient that was never created. Dangling references are your problem to catch, not the server's.

— THE ANCHOR · ENCOUNTER + OBSERVATION LINKING

One Observation. Every datatype clicks into one graph.

— OBSERVATION — A BODY-WEIGHT READING

{ "resourceType": "Observation", "status": "final", "code": { "coding": [{ "system": "http://loinc.org", "code": "29463-7", "display": "Body weight" }] }, [4] "subject": { "reference": "Patient/123" }, [1] "encounter": { "reference": "Encounter/456" }, [2] "performer": [{ "reference": "Practitioner/789", "display": "Dr. Chen" }], [3] "effectivePeriod": { "start": "2026-05-24T09:00:00Z", "end": "2026-05-24T09:05:00Z" }, [6] "valueQuantity": { "value": 70, "unit": "kg", "system": "http://unitsofmeasure.org", "code": "kg" } } [5]

- 1 · subject — Reference(Patient)
- 2 · encounter — Reference(Encounter)
- 3 · performer — Reference(Practitioner)
- 4 · code — CodeableConcept (LOINC)
- 5 · valueQuantity — Quantity (UCUM)
- 6 · effectivePeriod — Period

Three References stitch four resources into a graph — Observation, Patient, Encounter, Practitioner. The datatypes carry the clinical payload. References are the graph; datatypes are the cells. Read this one document and you have read FHIR.

— COMPARISON · PERIOD VS RANGE VS RATIO

Three shapes that look alike and mean very different things.

Period

AN INTERVAL IN TIME

`start` + `end`. An Encounter's duration, a coverage window.

Range

A BAND OF MEASUREMENT

`low` + `high`. A target blood-pressure window.

Ratio

A RELATIONSHIP

`numerator` / `denominator`. A titer of one to sixty-four, or a dose per volume.

— THE PERIOD GOTCHA

A start with no end does not mean missing data. It means ongoing.

The encounter is still open; the medication is still active. Never auto-close an open period — you would be inventing an end that never happened.

— DATATYPE 5 · IDENTIFIER — BEYOND SYSTEM AND VALUE

The facets that make a business key production-grade.

— THREE IDENTIFIERS · TWO DISTINCT SYSTEMS

"identifier": [{ "use": "official", "type": { "coding": [{ "code": "MR" }] }, "system": "urn:oid:2.16.840.1.113883.3.42", "value": "MRN-44721", "assigner": { "reference": "Organization/hosp-a" } }, { "use": "old", "system": "urn:oid:2.16.840.1.113883.3.42", "value": "MRN-10093", "period": { "end": "2024-01-01" } }, { "use": "usual", "system": "https://hosp-b.example.org/mrn", "value": "B-55120", "assigner": { "reference": "Organization/hosp-b" } }]

- **use — official, usual, temp, secondary, old**
- **type — a CodeableConcept (MR, SSN, DL)**
- **period — a validity window**
- **assigner — Reference to the issuing Organization**

— THE ANTI-PATTERN · MRN PER FACILITY

Three MRNs across two hospitals = three Identifiers, two distinct systems.

Each issuing facility needs its own `system` namespace — an OID like `urn:oid:2.16.840...` or a URI. Collapsing them under one system merges namespaces that should never touch. When an MRN changes, never delete the old one — demote it to `use: old` with a `period.end`, and add the new one as official. Identity has history.

— SYNTHESIS · OBSERVATION + THE ENCOUNTER IT REFERENCES

Every datatype from the chapter, living in two real documents.

— OBSERVATION

{ "resourceType": "Observation", "encounter": { "reference": "Encounter/456" }, [1] "code": { "coding": [{ "system": "http://loinc.org", "code": "29463-7" }, { "system": "http://snomed.info/sct", "code": "27113001" }] }, [2] "performer": [{ "reference": "Practitioner/789", "display": "Dr. Chen" }], [6] "valueQuantity": { "value": 70, "code": "kg", "system": "http://unitsofmeasure.org" } } [3]

— ENCOUNTER/456

{ "resourceType": "Encounter", "id": "456", "status": "finished", "subject": { "reference": "Patient/123" }, "period": { "start": "2026-05-24T08:30:00Z", "end": "2026-05-24T09:30:00Z" } } [4] // Patient/123 link "identifier": [{ "use": "official", "system": "urn:oid:2.16.840...", "value": "MRN-44721" }] [5]

- 1 · Reference relative form
- 2 · CodeableConcept — two Codings
- 3 · Quantity — UCUM code
- 4 · Period — start + end
- 5 · Identifier — use + system
- 6 · display fallback on a Reference

— TRY IT YOURSELF · 60 SECONDS

Pull a live Observation. Read its concept and its value.

Run this against the public HAPI sandbox. Pull one Observation and slice out the CodeableConcept code with its system, plus the Quantity.

— RUN THIS IN YOUR TERMINAL

curl https://hapi.fhir.org/baseR4/Observation?_count=1 | jq '.entry[0].resource | {code: .code.coding[0].code, system: .code.coding[0].system, value: .valueQuantity}'

That gives you the CodeableConcept's first code and its system, plus the Quantity. Then go hunt — find an Observation whose `code` carries two Codings, and name both systems.

— NEXT CHAPTER

Chapter 04 — Terminology.

We read the codes today; next we learn where they come from, and how binding strengths decide which ones are even allowed.

hapi.fhir.org · public R4 sandbox

Next · Ch 04 · Terminology