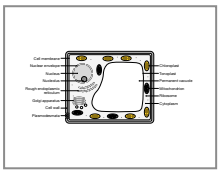
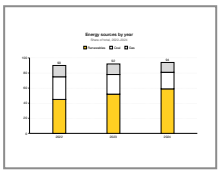
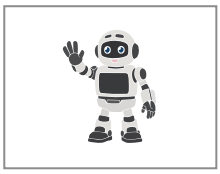
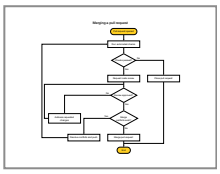
Espresso API

Send a request in words and get one picture back as an SVG: a drawing of a real thing, a textbook diagram, a chart or a map. This guide has everything you need to call it, and 61 requests that are known to work.

Connect

Setting	Value
Base URL	`https://ls-espresso-openai-yezmxbhcwq-ew.a.run.app/v1`
Endpoint	`POST /chat/completions`
Model	`espresso`
Authentication	`Authorization: Bearer <your key>`

The endpoint speaks OpenAI's Chat Completions, so any OpenAI client works once it points at the base URL. The answer is one JSON response; nothing is streamed. Your key works for 48 hours. When it stops, get a new one on the page that gave you this one, with the same email.

Call it from your server, not from a web page: browsers are not allowed to call it directly, and your key stays out of the page that way.

Quick start

Put your key in `ESPRESSO_API_KEY` first.

curl

```
curl -sS https://ls-espresso-openai-yezmxbhcwq-ew.a.run.app/v1/chat/completions \
  -H "Authorization: Bearer $ESPRESSO_API_KEY" \
  -H 'Content-Type: application/json' \
  -d '{"model": "espresso",
       "messages": [{"role": "user", "content": "a red bicycle"}]}' \
  | jq -r '.choices[0].message.content' > bicycle.svg
```

Python, with the OpenAI SDK

```
import os
from openai import OpenAI

client = OpenAI(
    base_url="https://ls-espresso-openai-yezmxbhcwq-ew.a.run.app/v1",
    api_key=os.environ["ESPRESSO_API_KEY"],
    timeout=300,
    max_retries=0,
)
completion = client.chat.completions.create(
    model="espresso",
    messages=[{"role": "user", "content": "an acoustic guitar"}],
    extra_body={"labels": True, "language": "de"},  # Espresso's own fields
)
with open("guitar.svg", "w") as file:
    file.write(completion.choices[0].message.content)
```

TypeScript, with the OpenAI SDK

```
import { writeFile } from "node:fs/promises";
import OpenAI from "openai";

const client = new OpenAI({
  baseURL: "https://ls-espresso-openai-yezmxbhcwq-ew.a.run.app/v1",
  apiKey: process.env.ESPRESSO_API_KEY,
  timeout: 300_000,
  maxRetries: 0,
});

try {
  const completion = await client.chat.completions.create({
    model: "espresso",
    messages: [{ role: "user", content: "a hot air balloon" }],
    // Espresso's own fields are not in OpenAI's types: pass them through.
    ...({ photo: true } as object),
  });
  await writeFile("balloon.svg", completion.choices[0].message.content ?? "");
  // The photo that `photo: true` asked for, when one was made.
  const { generated_image } = completion as unknown as {
    generated_image?: { image: { b64_json: string } };
  };
  if (generated_image) {
    await writeFile("balloon.png", Buffer.from(generated_image.image.b64_json, "base64"));
  }
} catch (error) {
  if (error instanceof OpenAI.APIError && error.code === "unsupported_request") {
    // 422: these words name nothing to draw. Ask for other words.
  } else throw error;
}
```

JavaScript, with no SDK

```
const response = await fetch("https://ls-espresso-openai-yezmxbhcwq-ew.a.run.app/v1/chat/completions", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.ESPRESSO_API_KEY}`,
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    model: "espresso",
    messages: [{ role: "user", content: "map of Iceland with Reykjavík marked" }],
  }),
  signal: AbortSignal.timeout(300_000),
});
const body = await response.json();
if (!response.ok) throw new Error(`${response.status} ${body.error.code}: ${body.error.message}`);
const svg = body.choices[0].message.content; // "<svg …>…</svg>"
```

What it draws

There is no field for the kind of picture. The words of the request choose it.

Kind	What you get	Say it like this
Drawing	A drawing of a real thing as it looks. This is what you get when the words ask for none of the other three.	"a red bicycle", "the Brandenburg Gate", "a golden retriever puppy, sitting"
Diagram	A diagram as a school textbook draws it, with its labels: science, maths, anatomy, geography.	"Bohr model of sodium", "a labelled plant cell", "graph of y = x² − 4"
Chart	A chart or structured diagram laid out from the figures, steps or items you give.	"bar chart of votes: Yes 412, No 288, Abstain 37", "flowchart of how a pull request gets merged"
Map	A map of a place, with what you name marked or highlighted.	"map of Peru", "map of Germany with Berlin, Hamburg and Munich marked"

Request

A JSON body, sent with `Content-Type: application/json`.

Field	Type	
`model`	string, required	`"espresso"`.
`messages`	array, required	The last message with the role `user` is the request: a string, or an array of text parts. At most 8,000 characters. Other messages are accepted and not read.
`labels`	boolean, default `false`	`true` names each part of a drawing beside it.
`language`	string, default `"en"`	The language of a drawing's labels. One of the 55 codes under Limits.
`photo`	boolean, default `false`	`true` also returns a photo of the thing a drawing shows, as a PNG.

`labels`, `language` and `photo` shape drawings only. Sent with a request that comes back as a diagram, a chart or a map, they change nothing. Those carry their own labels, and a chart's are in the language its request is written in.

Other standard Chat Completions fields, such as `temperature` and `max_tokens`, are accepted and ignored. These are refused with a 400 that names the field: `stream: true`, `n` above 1, `tools`, a `response_format` other than text, audio, and image parts in a message.

`GET /models` lists the one model, and is a quick way to check a key.

Response

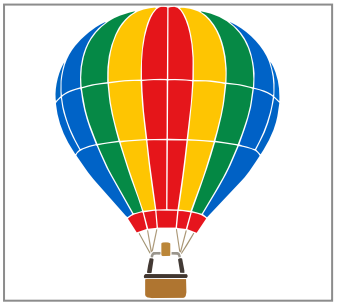
A standard chat completion. The SVG is the message's content:

```
{
  "id": "chatcmpl-…",
  "object": "chat.completion",
  "created": 1795795176,
  "model": "espresso",
  "choices": [
    {
      "index": 0,
      "message": { "role": "assistant", "content": "<svg …>…</svg>", "refusal": null },
      "logprobs": null,
      "finish_reason": "stop"
    }
  ]
}
```

A drawing asked for with `photo: true` has one more field beside `choices`:

```
"generated_image": {
  "svg": "<svg …>…</svg>",
  "image": { "format": "png", "width": 1024, "height": 1024, "b64_json": "iVBORw0KGgo…" }
}
```

`svg` is the drawing again and `image` is the photo, in the drawing's frame, so one lies exactly over the other. Labels and the photo are best effort: a drawing can come back without them, so check before you use them.





“a hot air balloon” with `"photo": true`: the drawing in `content`, and the photo that came with it in `generated_image`.

Using the SVG

- The content is one complete SVG document. It starts with `<svg xmlns="http://www.w3.org/2000/svg"` and has `width`, `height` and `viewBox`. It holds no scripts and loads no outside files.
- Save it as a `.svg` file, put it straight into a page's HTML, or use it as an image with `"data:image/svg+xml," + encodeURIComponent(svg)`.
- It is vector art: it stays sharp at any size, and you can recolour, animate or take apart its shapes.

- Words in a picture are real `<text>`, set in `Saans, 'Helvetica Neue', Helvetica, Arial, sans-serif`, so they show in the first of those fonts the viewer has.
- For a PNG, render the SVG yourself, with sharp, resvg or CairoSVG, or on a canvas in the browser.
- A picture is a few kilobytes to about a megabyte. Detailed maps are the largest.

Errors

Errors have OpenAI's shape:

```
{
  "error": {
    "message": "This request can't be drawn. Name a thing to draw, or describe a diagram, a chart with its
data, or a map.",
    "type": "invalid_request_error",
    "param": "messages",
    "code": "unsupported_request"
  }
}
```

Status	`code`	When	What to do
400	`invalid_request`	A field can't be used; `param` names it.	Fix the request.
400	`invalid_json`	The body is not JSON.	Fix the request.
401	`invalid_api_key`	The key is missing, wrong or past its 48 hours.	Get a new key.
404	`model_not_found`	The model is not `espresso`.	Fix the request.
413	`request_too_large`	The body is too large.	Send less.
415	`unsupported_media_type`	The body was not sent as JSON.	Set `Content-Type`.
422	`unsupported_request`	The words name nothing to draw, such as "hello, how are you today?".	Ask for other words.
429	`rate_limit_exceeded`	Too many requests at once or in one second.	Wait for `Retry-After` seconds, then send it again.
502	`generation_failed`	The picture could not be made.	Send it again.
504	`timeout`	The picture took too long.	Send it again.

Limits

- One picture per request. Nothing is remembered between requests: to change a picture, send the whole request again in other words.
- Text in, pictures out: no image input, no text answers, no tables.
- No streaming, tools, JSON response format or `n` above 1.
- At most 2 requests a second, counted over everyone at the hackathon together. One more is a 429: wait a second and send it again.
- Give a request up to 300 seconds and turn off your client's automatic retries. Drawings take the longest, so show that something is loading.

- `language` is one of: en, fr, de, es, it, pt-PT, pt-BR, nl, ca, eu, gl, cy, ga, mt, sv, da, nb, fi, is, pl, cs, sk, hu, sl, hr, ro, et, lv, lt, bg, sr, sr-Latn, bs, mk, sq, el, tr, uk, ru, be, zh-Hans, zh-Hant, ja, ko, id, ms, fil, vi, th, hi, bn, ar, fa, ur, he.

How to ask

- **Name the kind of picture** when you want a chart, a diagram or a map: "bar chart of…", "flowchart of…", "map of…". Without it you get a drawing of the thing.
- **Charts: put the data in the request.** Plain words are enough: "Yes 412, No 288, Abstain 37". These all work: bar, stacked bar, line, scatter, pie, donut, radar, Gantt, timeline, flowchart, sequence diagram, decision tree, mind map, org chart, Venn diagram, cycle and 2×2 matrix.
- **Steps and ideas can be left out.** "flowchart of how a pull request gets merged" comes back with sensible steps. Give your own to decide them.
- **Real-world figures you leave out are looked up**, as in "bar chart of the GDP of the G7 countries". Check them, and send your own figures when they matter.
- **Maps: name the place, then what to mark or highlight.** Cities and landmarks are marked, countries are highlighted, and a river can be drawn with the cities along it.
- **Drawings: name one thing**, with its colour, pose or view if you care. Add `labels: true` for the names of its parts and `language` for the language they are in.
- **Diagrams: name the subject the way a textbook does.** "the water cycle", "cross-section of a volcano", "free-body diagram of a block on a 30° incline".

Give it to your agent

One tool is enough. Have it call the endpoint with `request` as the user message and hand the SVG back:

```
{
  "name": "draw",
  "description": "Draw one picture from a request in words and get it back as SVG. Draws real things, textbook diagrams, charts from the figures or steps given, and maps. One picture per call, and nothing is remembered between calls, so put everything in the request.",
  "parameters": {
    "type": "object",
    "properties": {
      "request": {
        "type": "string",
        "description": "What to draw: 'a red bicycle', 'the water cycle', 'bar chart of votes: Yes 412, No 288', 'map of Peru'."
      },
      "labels": {
        "type": "boolean",
        "description": "Name the parts of a drawing of a real thing."
      },
      "language": {
        "type": "string",
        "description": "Language code for those labels, such as 'de'."
      }
    },
    "required": ["request"]
  }
}
```

Ideas

- **A tutor that draws.** Answer a question, then illustrate the answer: the diagram for the science, the chart for the numbers, the map for the place.
- **Picture vocabulary in 55 languages.** Any object, with its parts named in the learner's language: "a running shoe" with `labels: true` and `language: "ja"`.
- **Talk to your data.** Turn a sentence, a spreadsheet row or a query result into the chart that shows it.
- **Meetings into diagrams.** A plan becomes a Gantt chart, a process a flowchart, a debate a decision tree, a brainstorm a mind map.
- **Trips and stories on a map.** Mark the stops of a journey, or highlight the countries a story passes through.
- **Things to make.** The drawings are clean vector shapes: cut them as stickers, plot them with a pen plotter, animate them with CSS, or print them as a colouring book.
- **Drawing and photo together.** With `photo: true` the photo lies exactly under the drawing: fade from one to the other, or let people trace it.

Examples

Every request below was sent to the live endpoint on 27 November 2026 and came back as a picture. Each one is the whole text of the user message.

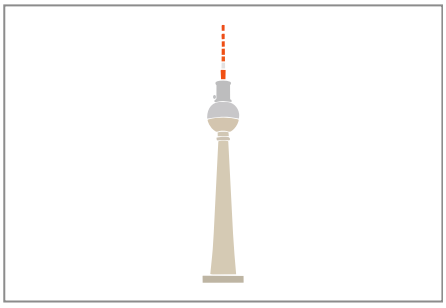
Drawings of real things (15)



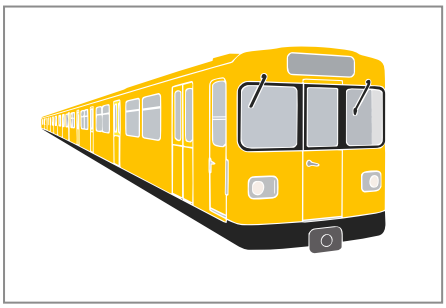
a red bicycle



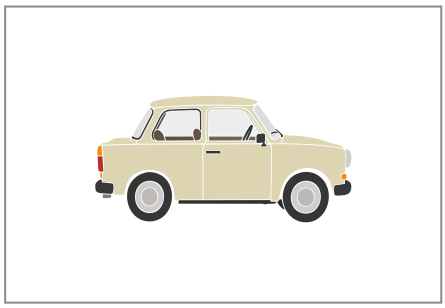
the Brandenburg Gate



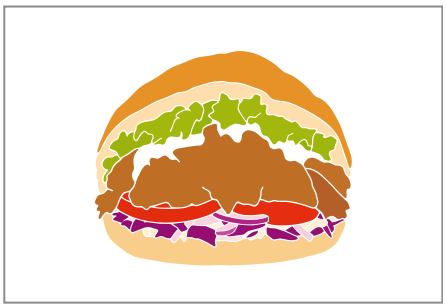
the Berlin TV Tower



a yellow Berlin U-Bahn train



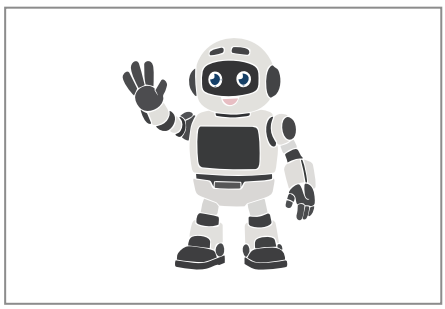
a Trabant car



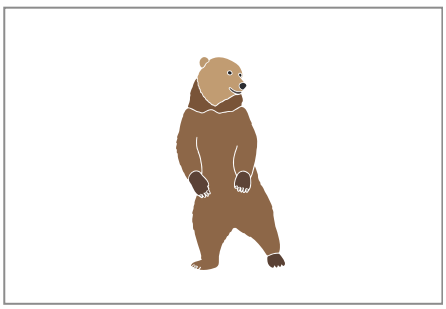
a döner kebab



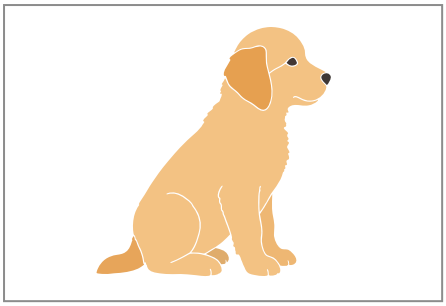
a cup of espresso on a saucer



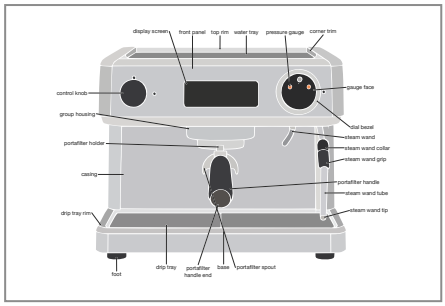
a friendly toy robot



a brown bear standing on its hind legs

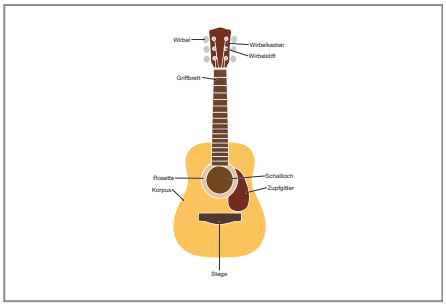


a golden retriever puppy, sitting



an espresso machine

```
"labels": true
```



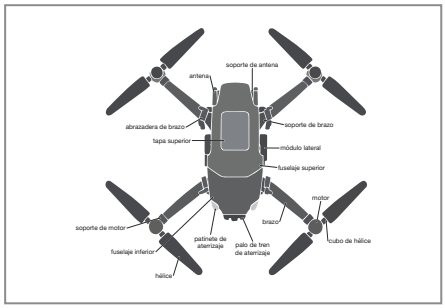
an acoustic guitar

```
"labels": true, "language": "de"
```



a running shoe

```
"labels": true, "language": "ja"
```



a quadcopter drone

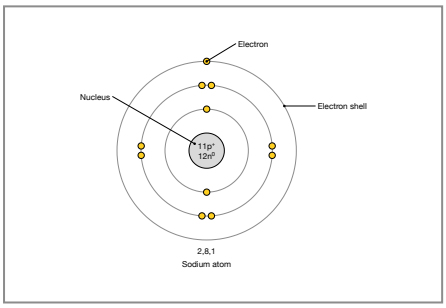
```
"labels": true, "language": "es"
```



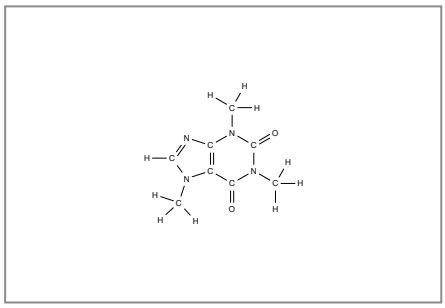
a hot air balloon

```
"photo": true
```

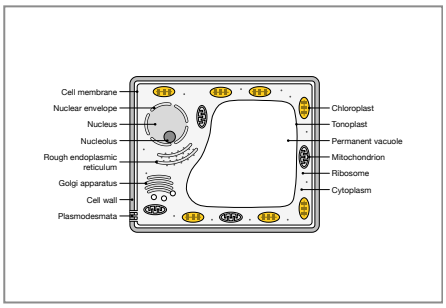
Textbook diagrams (19)



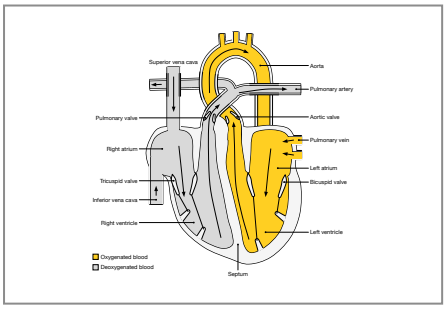
Bohr model of sodium



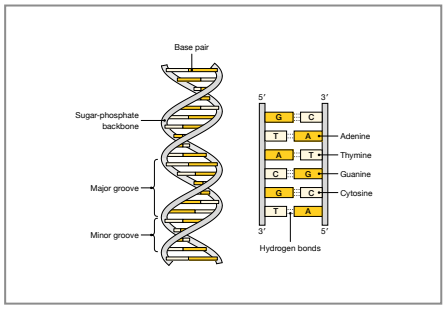
the structure of a caffeine molecule



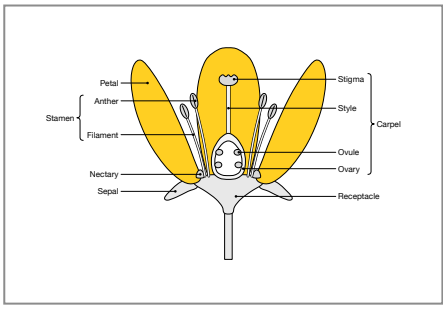
a labelled plant cell



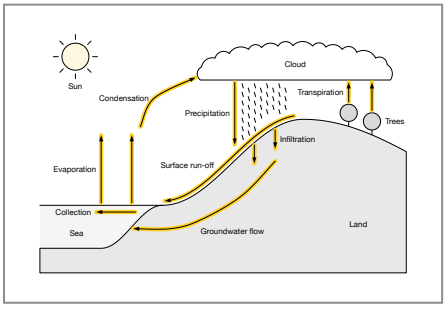
the human heart, labelled



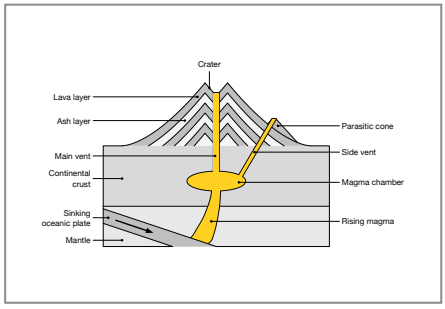
the DNA double helix, labelled



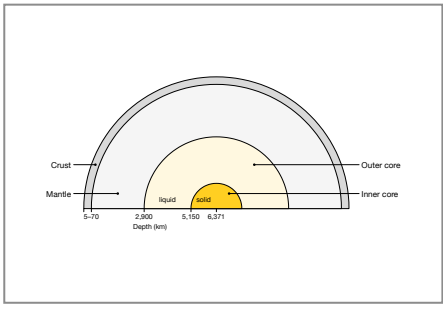
parts of a flower



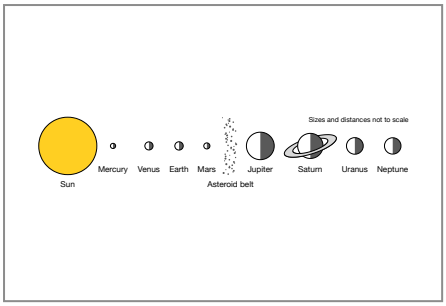
the water cycle



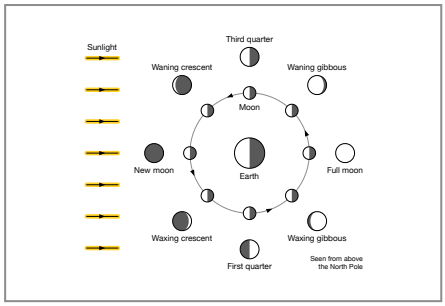
cross-section of a volcano



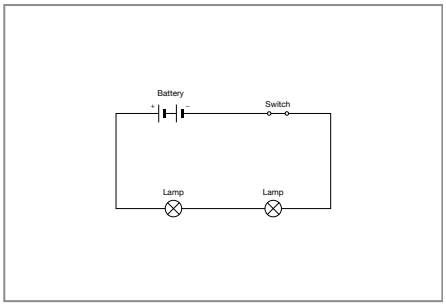
the layers of the Earth



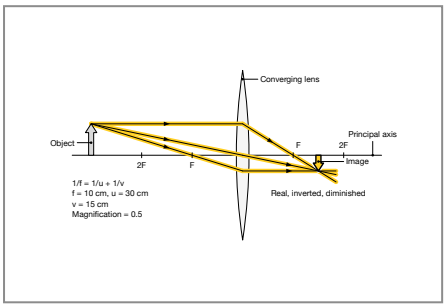
the solar system



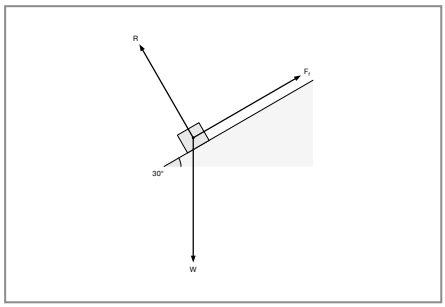
the phases of the Moon



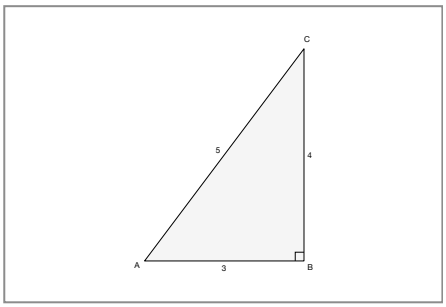
series circuit with two bulbs and a switch



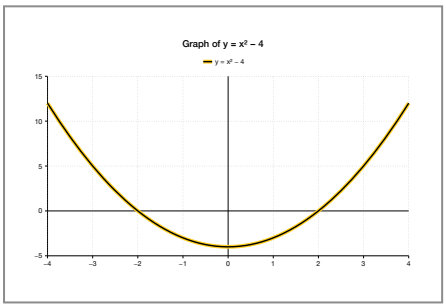
ray diagram of a convex lens forming a real image



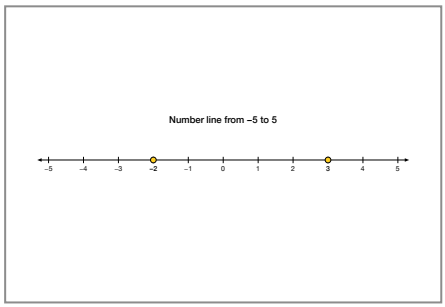
free-body diagram of a block on a 30° incline



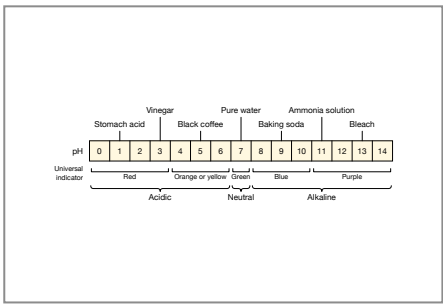
a right triangle with sides 3, 4 and 5



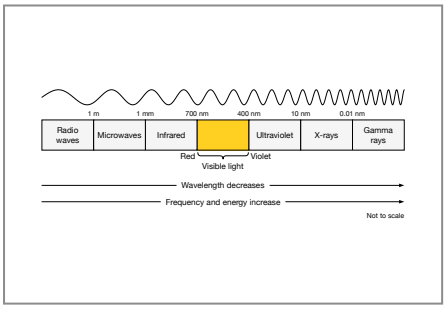
graph of y = x² − 4



a number line from −5 to 5 with −2 and 3 marked

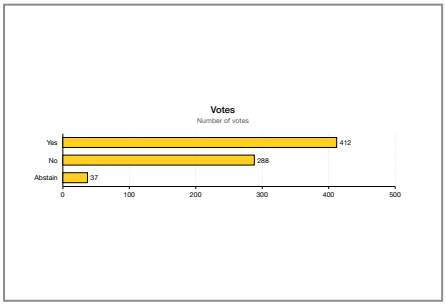


the pH scale

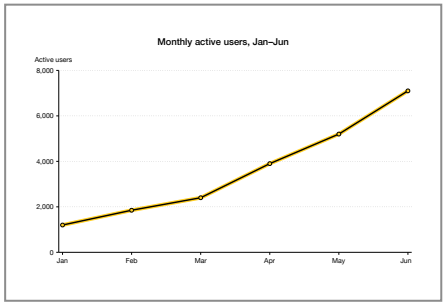


the electromagnetic spectrum

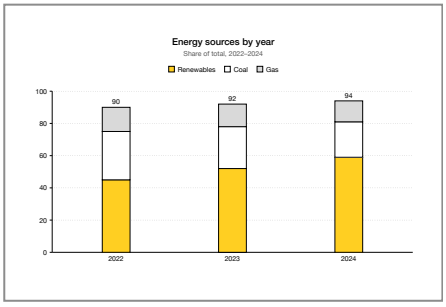
Charts and structured diagrams (18)



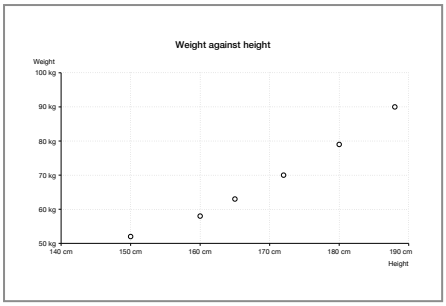
bar chart of votes: Yes 412, No 288, Abstain 37



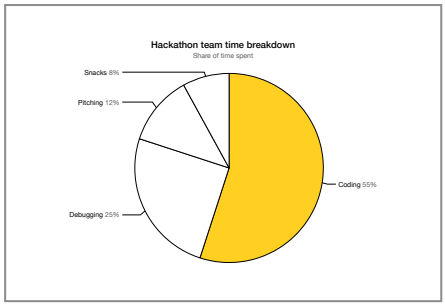
line chart of monthly active users: Jan 1200, Feb 1850, Mar 2400, Apr 3900, May 5200, Jun 7100



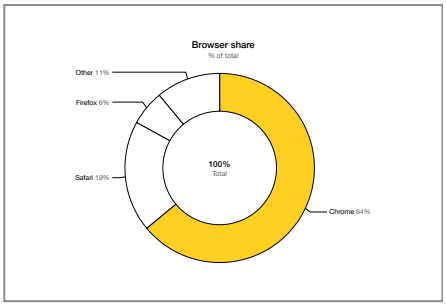
stacked bar chart of energy sources by year: 2022 coal 30, gas 15, renewables 45; 2023 coal 26, gas 14, renewables 52; 2024 coal 22, gas 13, renewables 59



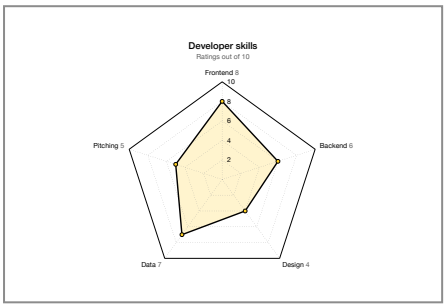
scatter plot of height in cm against weight in kg: (150, 52), (160, 58), (165, 63), (172, 70), (180, 79), (188, 90)



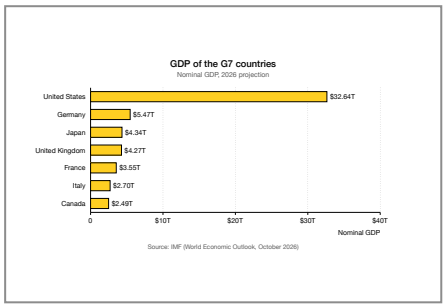
pie chart of how a hackathon team spends its time: coding 55%, debugging 25%, pitching 12%, snacks 8%



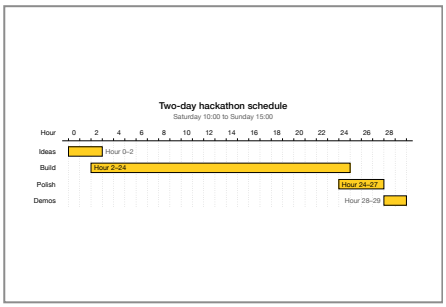
donut chart of browser share: Chrome 64%, Safari 19%, Firefox 6%, other 11%



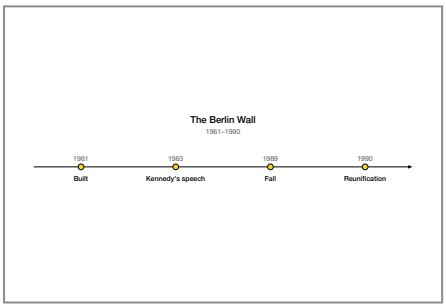
radar chart of a developer's skills out of 10: frontend 8, backend 6, design 4, data 7, pitching 5



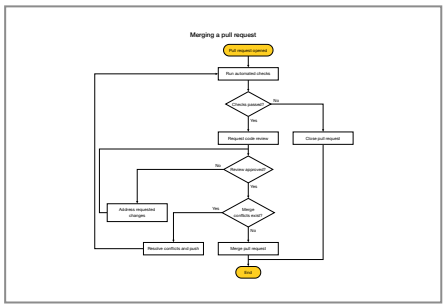
bar chart of the GDP of the G7 countries



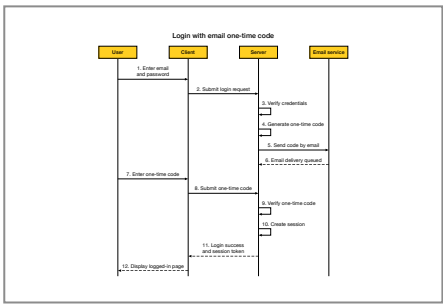
Gantt chart for a two-day hackathon: ideas Saturday 10:00 to 12:00, build Saturday 12:00 to Sunday 10:00, polish Sunday 10:00 to 13:00, demos Sunday 14:00 to 15:00



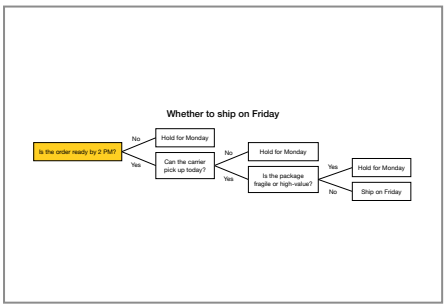
timeline of the Berlin Wall: built 1961, Kennedy's speech 1963, fall 1989, reunification 1990



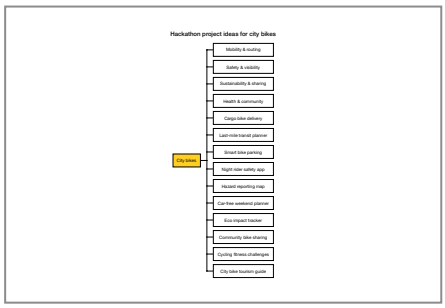
flowchart of how a pull request gets merged



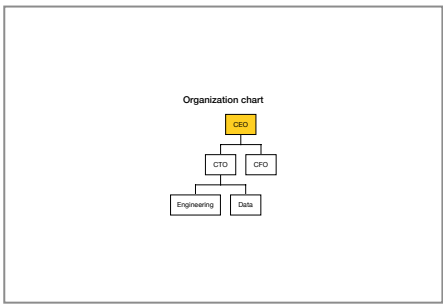
sequence diagram of a user logging in with a one-time code sent by email



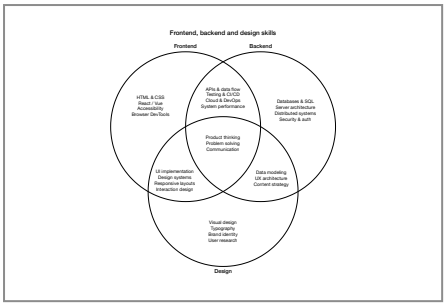
decision tree for whether to ship on Friday



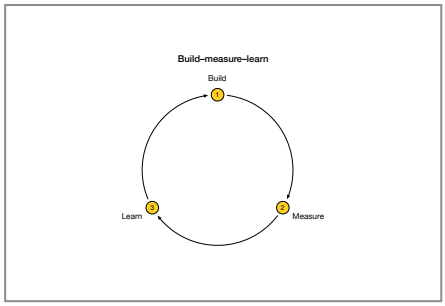
mind map of ideas for a hackathon project about city bikes



org chart: a CEO; under her a CTO and a CFO; under the CTO, Engineering and Data



Venn diagram of frontend, backend and design skills

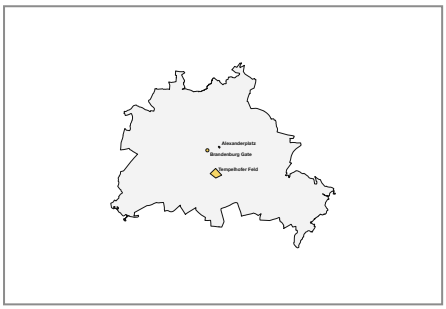


cycle diagram: build, measure, learn

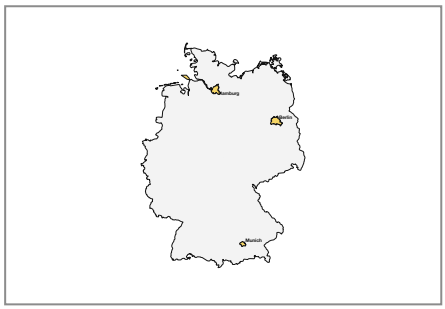


2×2 matrix of effort against impact with four features: dark mode (low effort, low impact), search (low effort, high impact), offline sync (high effort, high impact), custom themes (high effort, low impact)

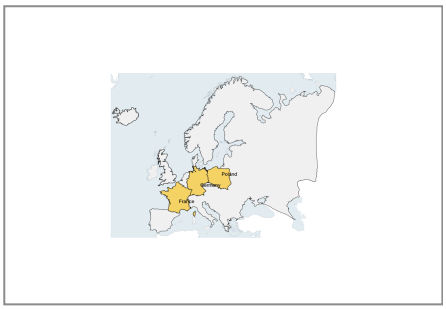
Maps (9)



map of Berlin with the Brandenburg Gate, Alexanderplatz and Tempelhofer Feld marked



map of Germany with Berlin, Hamburg and Munich marked



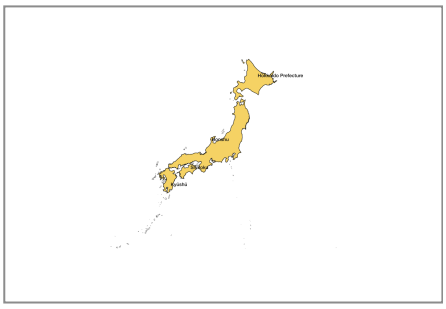
map of Europe with Germany, France and Poland highlighted



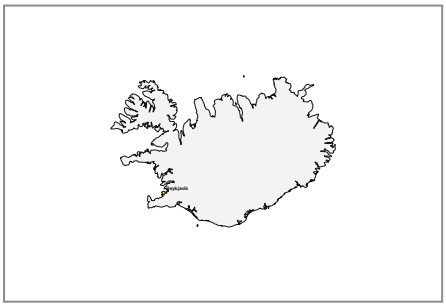
map of South America with Brazil and Argentina highlighted



world map with Brazil, India and Japan highlighted



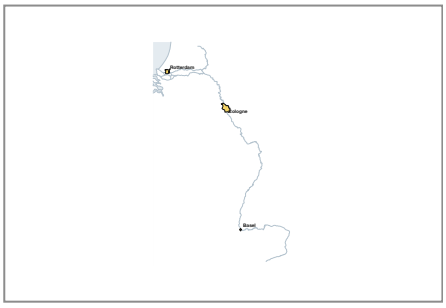
map of Japan with its four main islands labelled



map of Iceland with Reykjavík marked



map of Peru



the Rhine with Basel, Cologne and Rotterdam