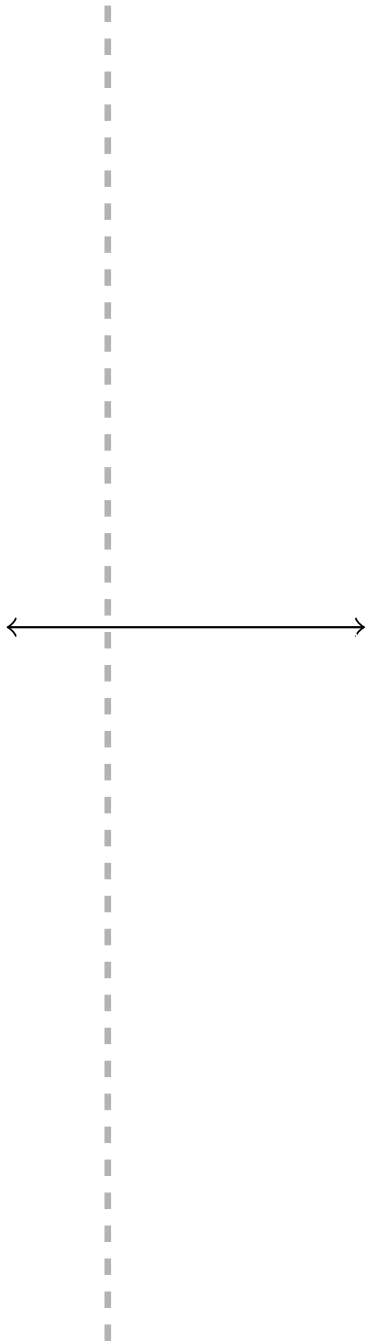
Modern web infrastructure: problems and solutions

E. Fonn, K. Johannessen

March 18, 2019

1: How to communicate over the internet?

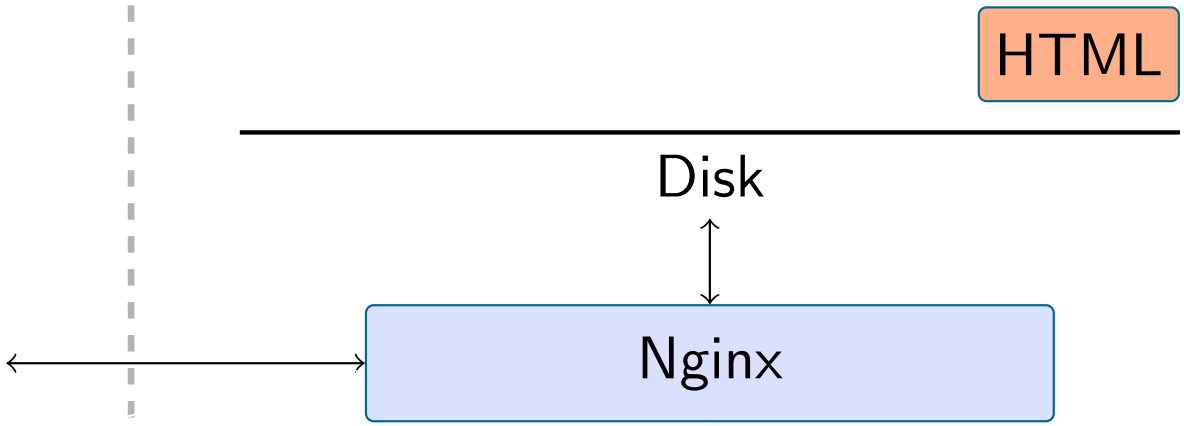
- HTTP: Hyper-Text Transfer Protocol
- Defines the "legal" form of a request and a response
- Two type of requests: GET and POST
- Many types of response (you have probably seen some of these)
  - 200: OK
  - 403: Forbidden
  - 404: Not found
  - 500: Internal server error
  - 503: Service unavailable
  - 418: I'm a teapot
  - etc.



2: How to respond with a meaningful webpage?

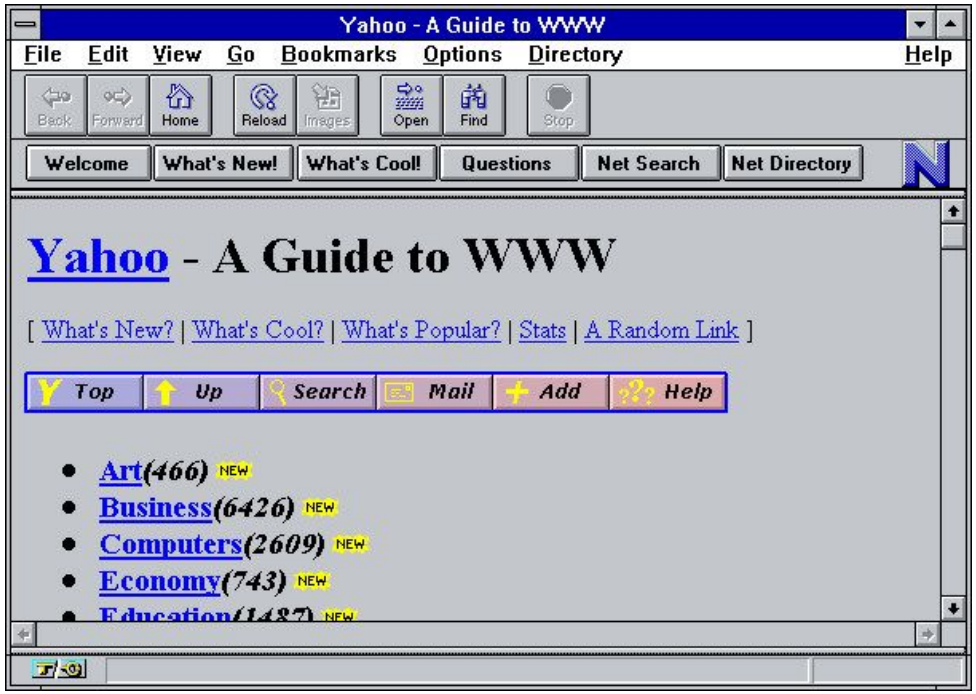
- HTML: Hyper-Text Markup Language
- Typically contains text, paragraphs, logical structure and references to other resources.
- HTML is what makes the “web” different from the “internet”.

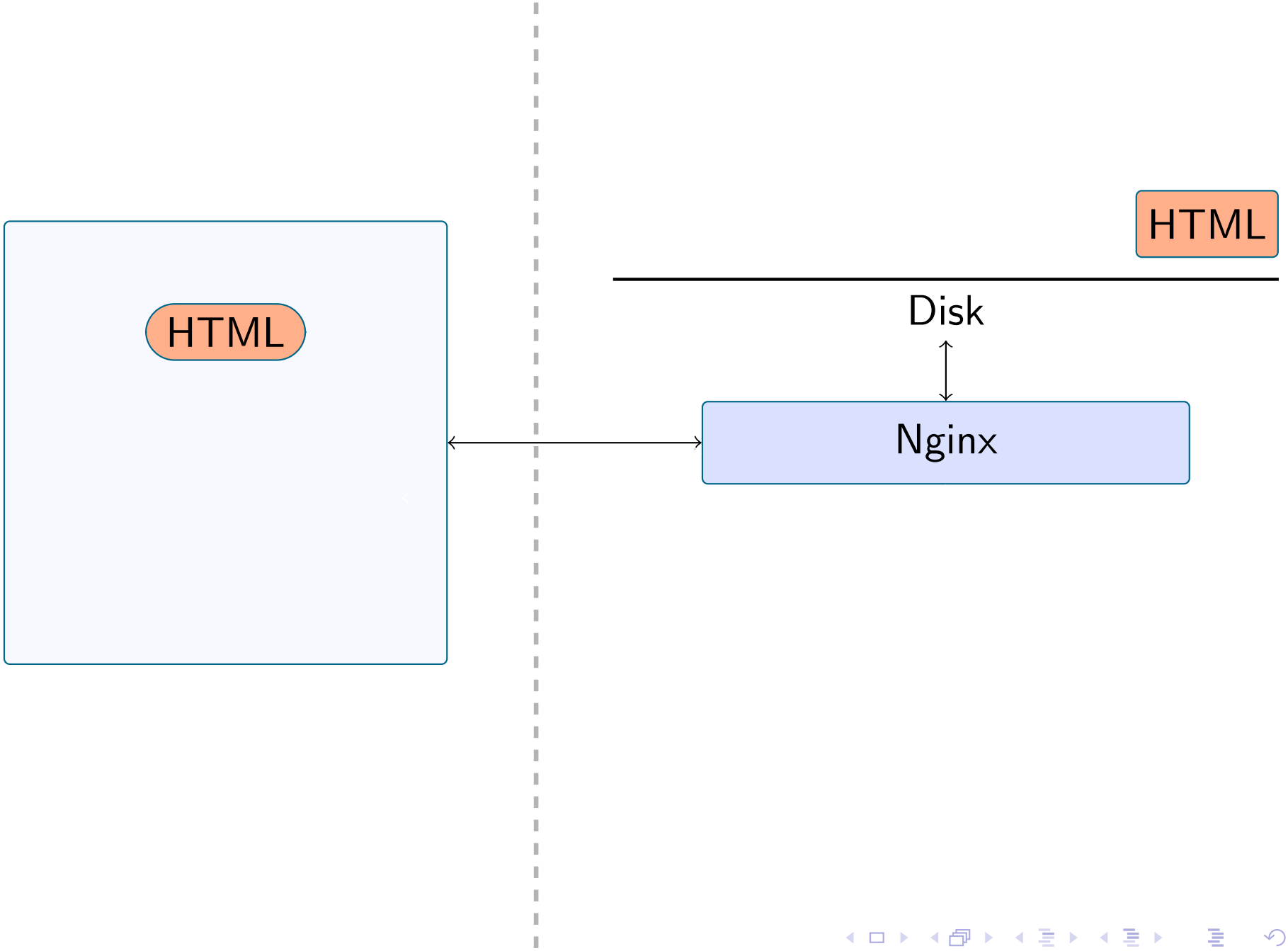
```
<h1>Welcome to my website</h1>
<p>Text in a paragraph.</p>
<a href="http://sintef.no">Good science here</a>
<strong>Bold</strong>, <em>italic</em>, etc.
```


HTML
Disk
Nginx

3: How to show a webpage?

- The first *web browser* was Netscape Navigator, released in 1994.
- Since HTML was relatively loosely defined, browsers had to be very generous and accept a lot of variety.

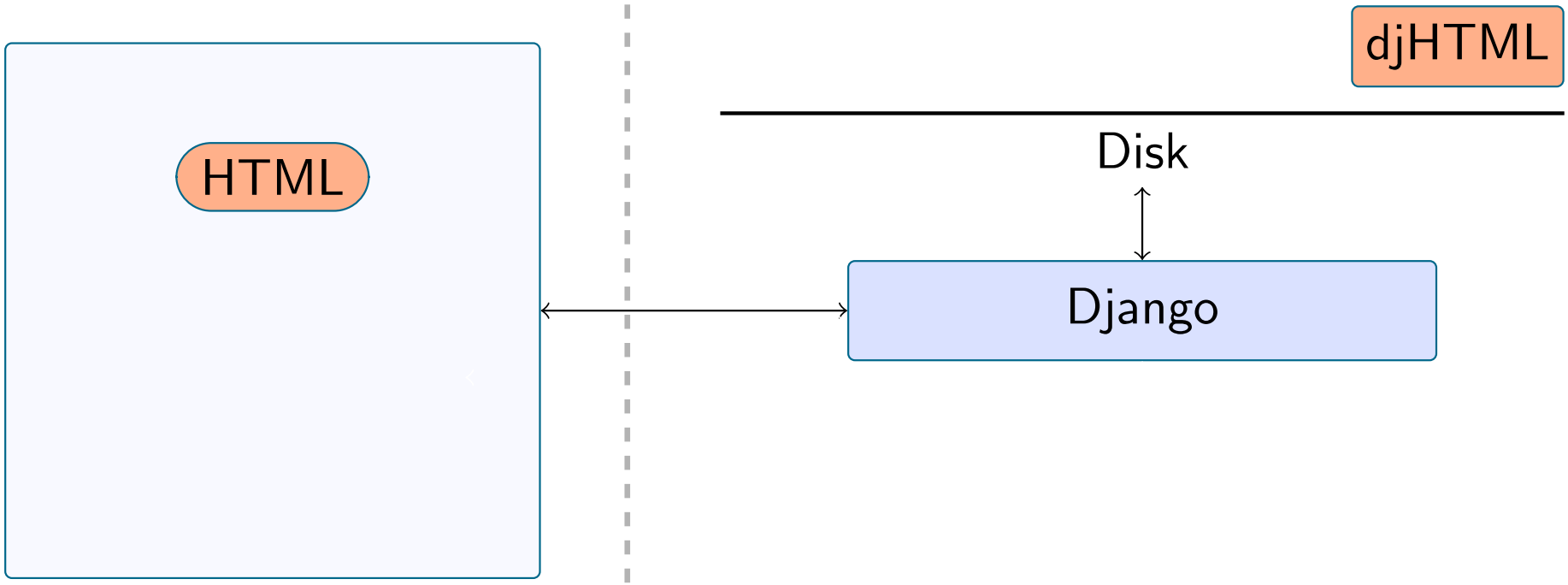



HTML
HTML
Disk
Nginx

4: How to serve dynamic content?

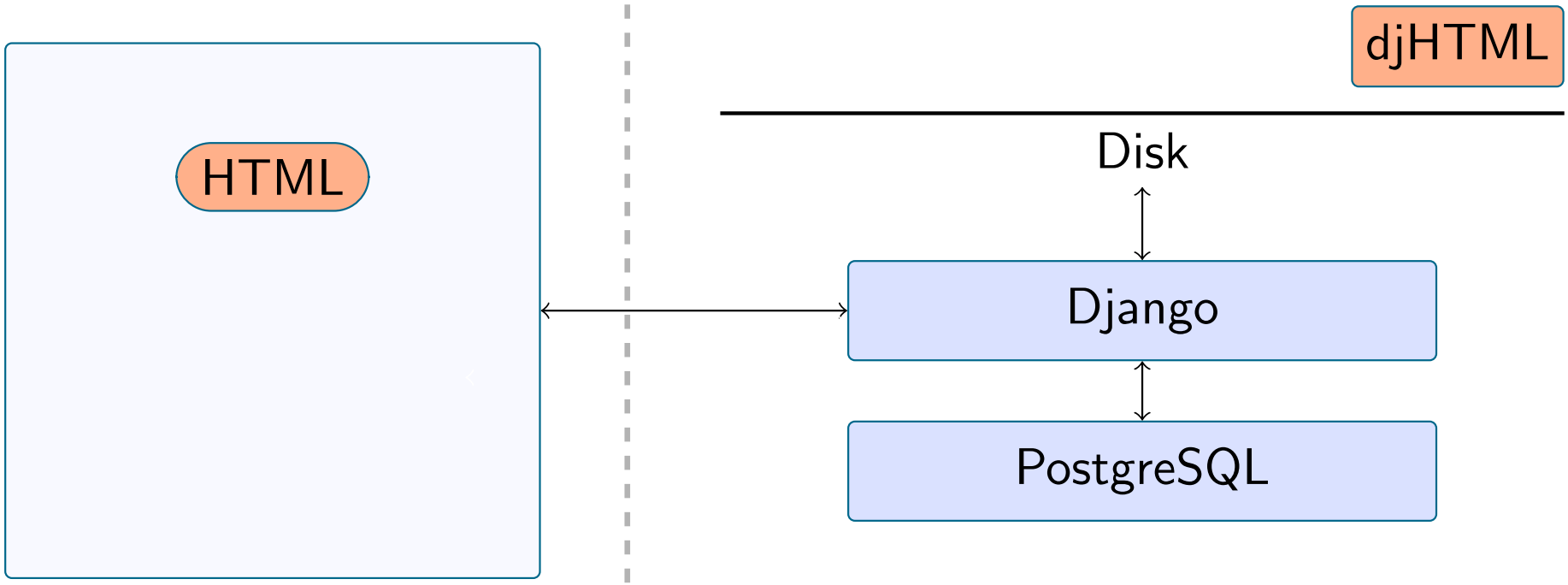
- Instead of a static server like Nginx, use a server that can generate HTML on the fly, like *Django*.
- Django is actually a framework for writing your own server, a sort of get-started kit.
- Instead of HTML, we have a *templating language* called djHTML.

```
<h1>{{ title|titlecase }}</h1>
<p>Text in a paragraph.</p>
{% if advertise %}
<a href="{{ target_address }}">Good science here</a>
{% endif %}
<strong>Bold</strong>, <em>italic</em>, etc.
```


djHTML
HTML
Disk
Django

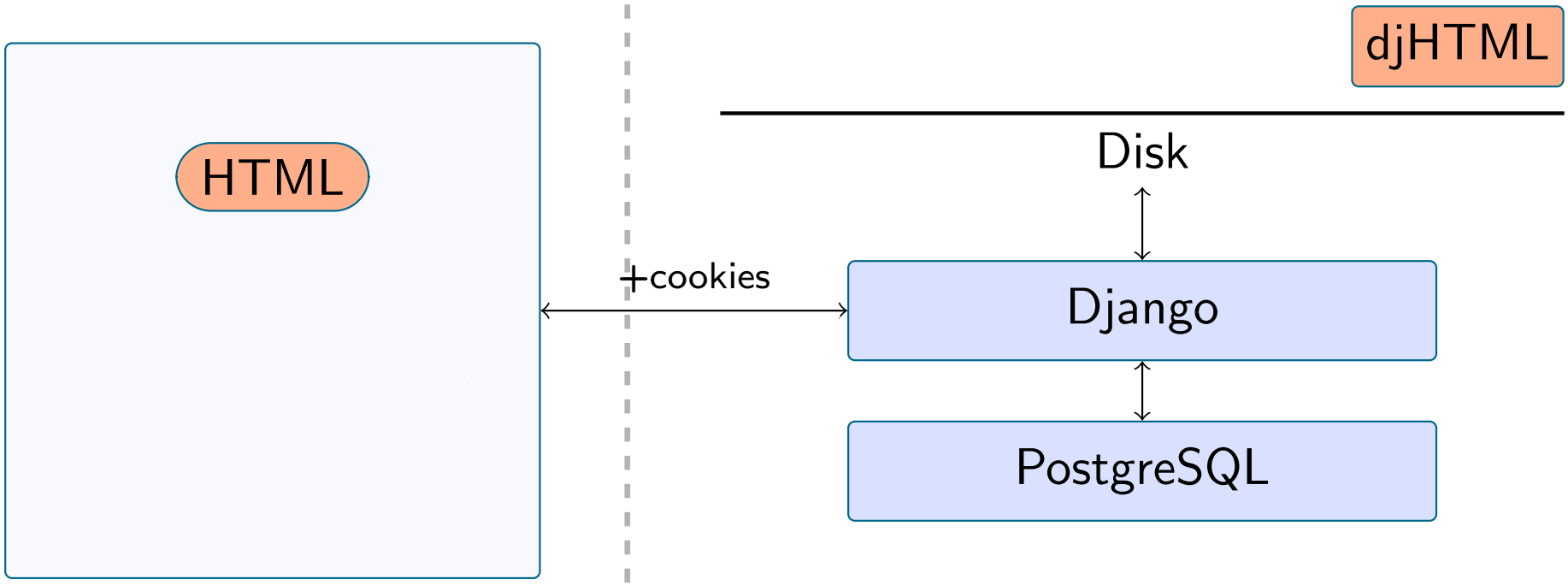
5: How to remember information?

- After responding to a request, the server immediately forgets all about it.
- Data is typically stored in a relational database, like PostgreSQL.
- This database is not visible from outside and can only be interacted with by the server.
- Django has special functionality for interacting with databases.


djHTML
HTML
Disk
Django
PostgreSQL

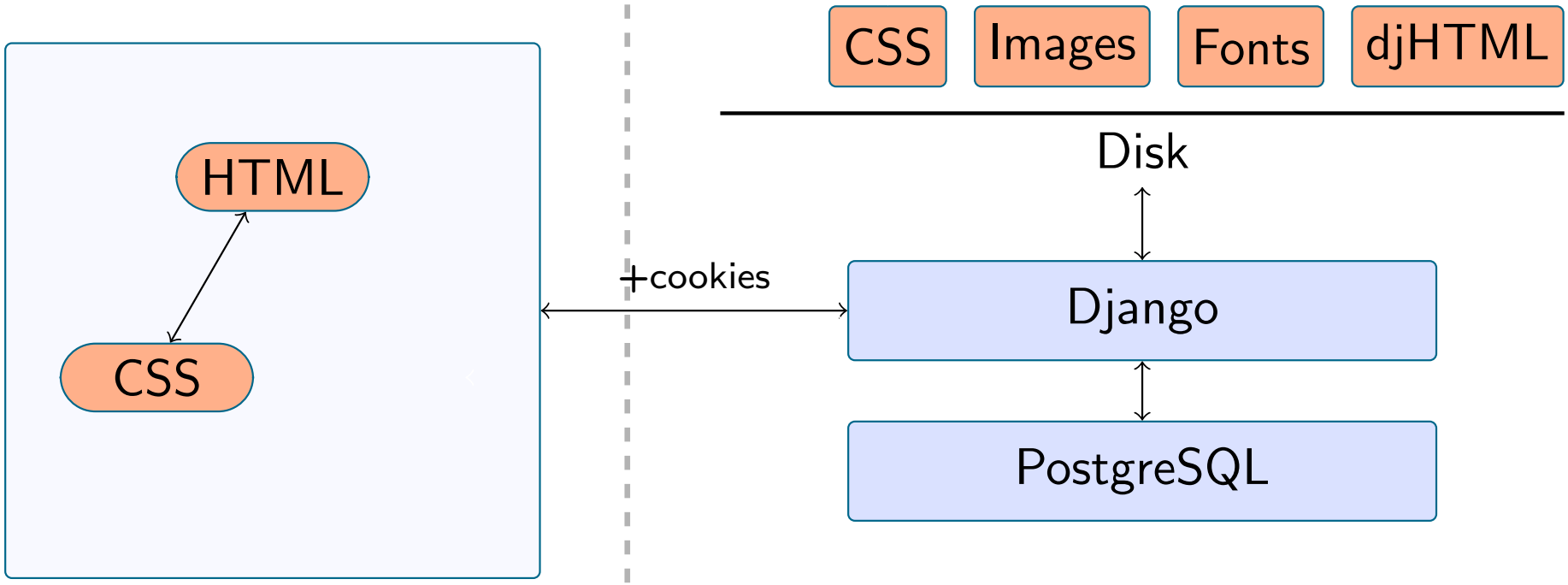
6: How to remember temporary data?

- It's (close to) impossible for the server to recognize that two consecutive requests come from the same user.
- Closely related problem: how to keep a user logged in without having to type a password on every single new request.
- *Cookies* solve this problem: a small data package added to every request and every response.
- Django automatically handles authentication via cookies.


djHTML
HTML
Disk
+cookies
Django
PostgreSQL

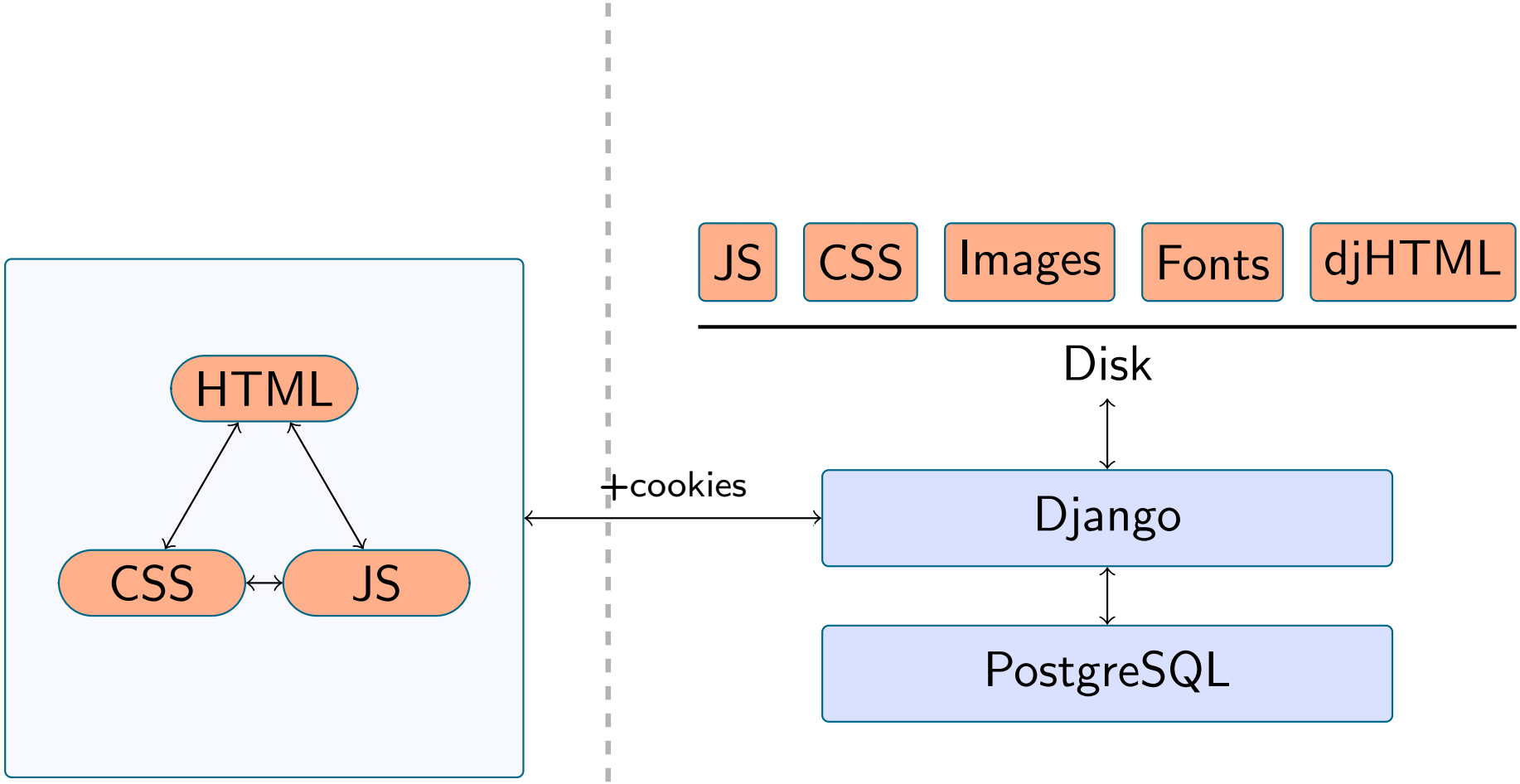
7: My website looks like it's 1994

- HTML has very limited means for style.
- CSS: Cascading Style-Sheets was invented in 1996 to solve this problem.
- A stylesheet contains a list of *rules* for how different HTML objects should look.
- It's normal for the HTML document to include a link to the CSS file, so that the browser makes a secondary request for it.
- Same goes for other artistic details like images, fonts, etc.


CSS
Images
Fonts
djHTML
Disk
HTML
CSS
+cookies
Django
PostgreSQL

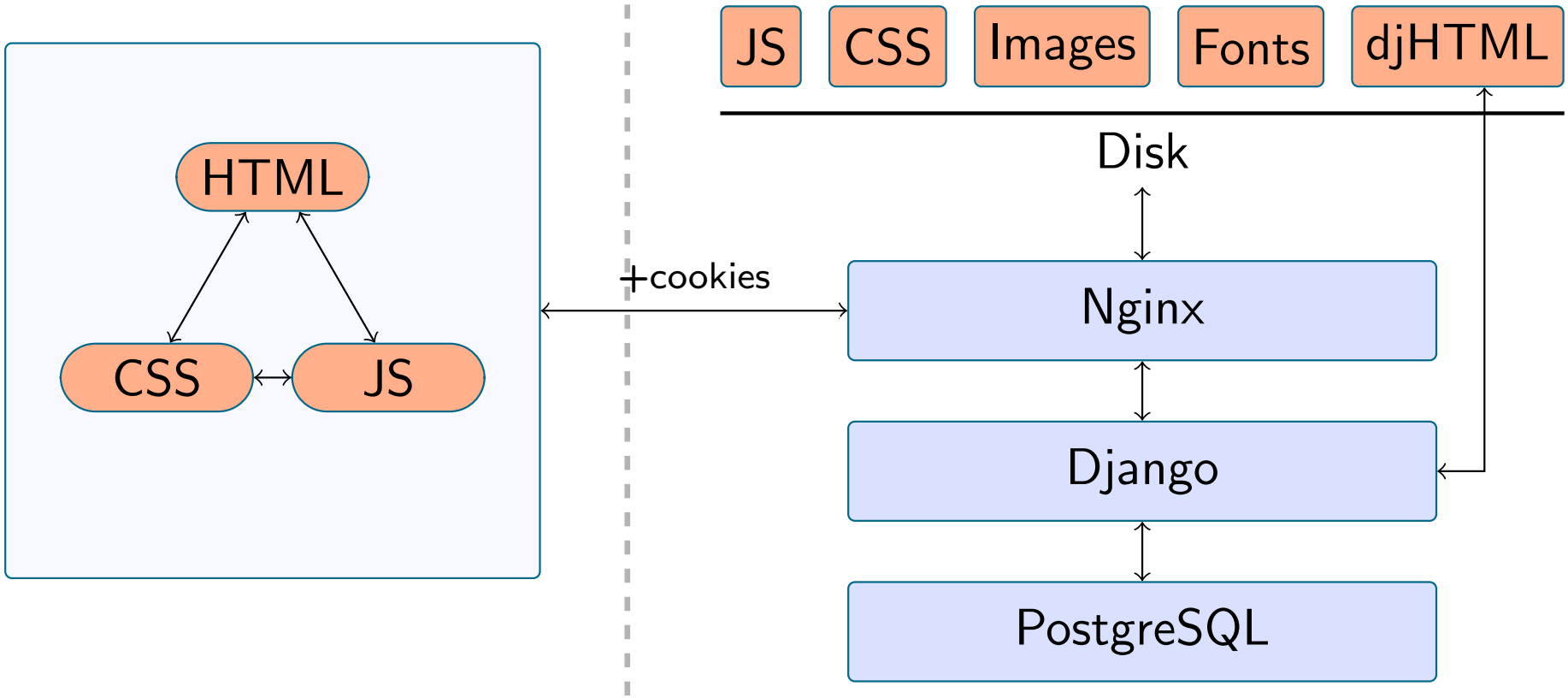
8: How to make my website *do* things?

- HTML is fully static. The only way to make "things happen" is to make a new request to the server.
- Many forms of "dynamism" are better handled using Javascript, a programming language that can run *in the browser* without making any new requests.
- Like CSS, JS is sent to the browser via a separate request.


JS
CSS
Images
Fonts
djHTML
Disk
HTML
+cookies
Django
CSS
JS
PostgreSQL

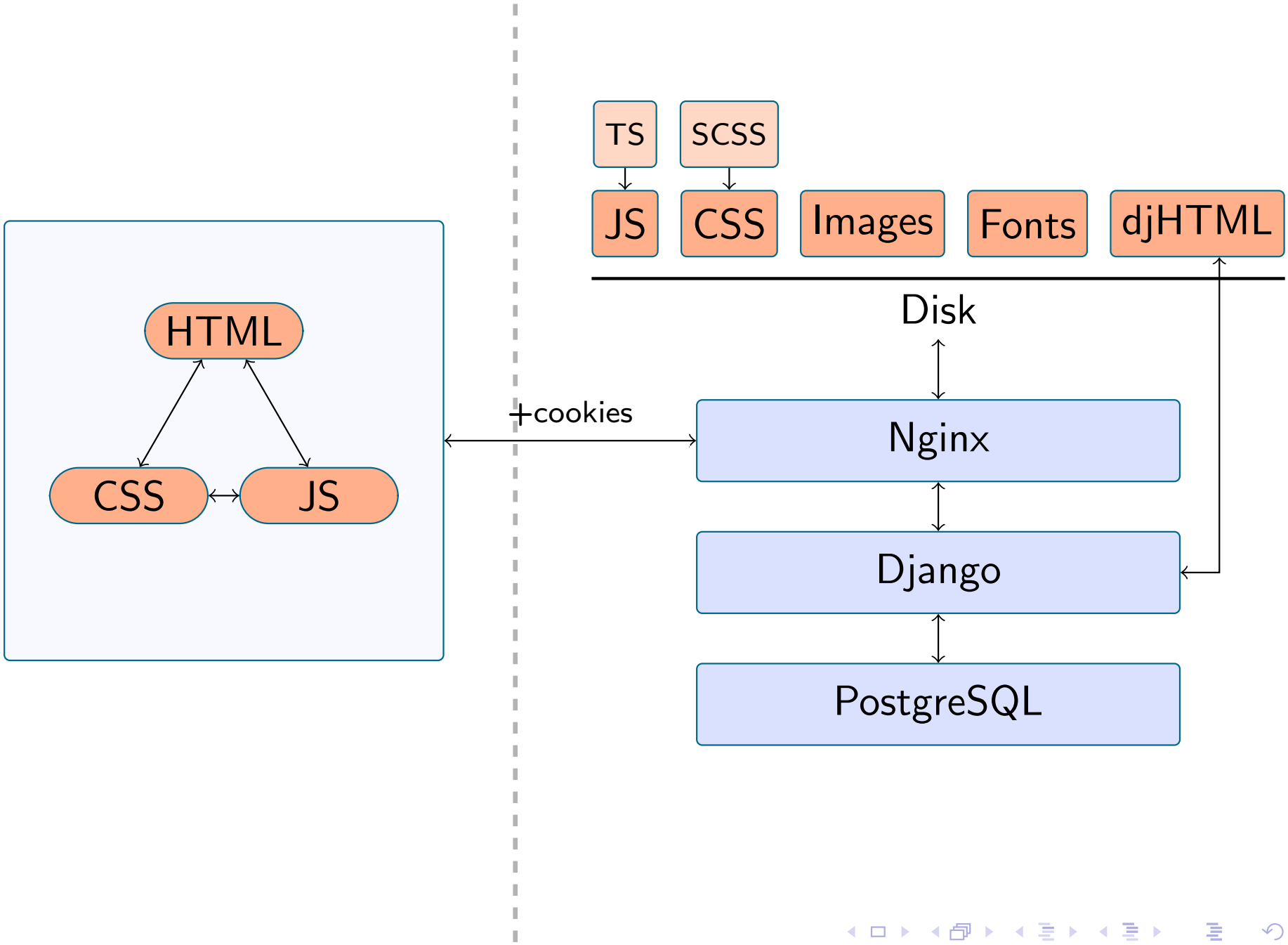
9: Now my website is really slow

- Displaying a typical webpage now means at least one request for the HTML and several more for the JS, CSS, images and fonts to go with it.
- Web frameworks like Django are designed for features and not speed. Serving static content slows them down.
- Common: a multi-layered webserver design with a fast front-facing server for static content, which proxies the requests for dynamic content to a slower server in the back.


HTML
CSS
JS
+cookies
JS
CSS
Images
Fonts
djHTML
Disk
Nginx
Django
PostgreSQL

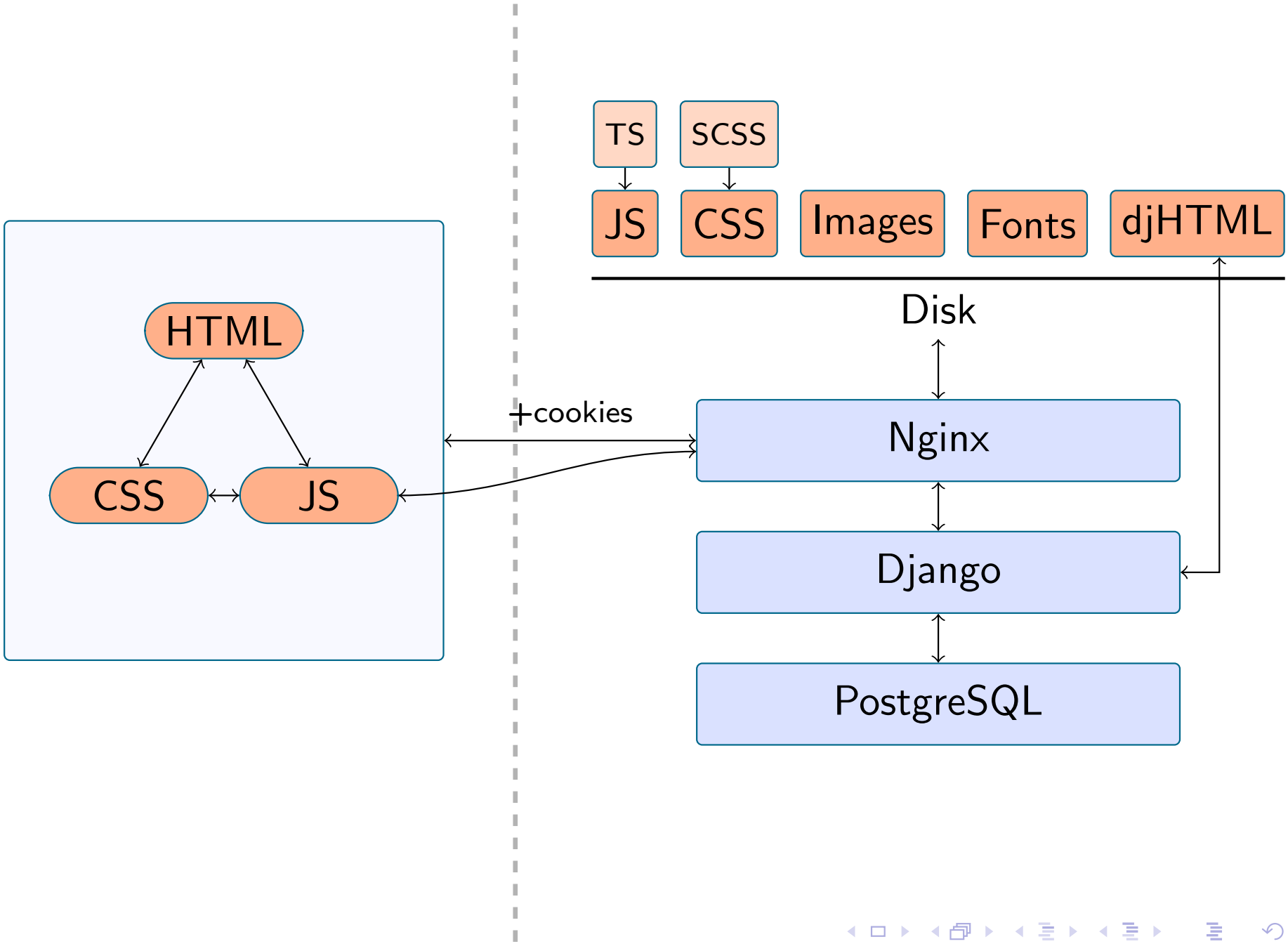
10: HTML, CSS and JS are dinosaurs

- These languages suffer from 20 years of baggage, having been originally designed for much more limited things than they are used for today.
- The languages are now standardized, but the wide array of user software still in use means that we can't rely too much on these standards.
- CSS and JS can be *generated* by compiling other, more sensible languages: SASS (Syntactically Awesome Style-Sheets) generates CSS and TS (Typescript) generates JS.


TS
SCSS
JS
CSS
Images
Fonts
djHTML
Disk
HTML
CSS
JS
+cookies
Nginx
Django
PostgreSQL

11: Even more dynamic JS

- Between the two extremes of dynamic content (local JS and full server requests) comes AJAX (Asynchronous JS and XML).
- JS programs running in a browser can make their own requests to the server, the response to which can be used on the current page.
- These requests are often small, lightweight and cheaper than a complete new request.


TS
SCSS
JS
CSS
Images
Fonts
djHTML
Disk
HTML
+cookies
Nginx
CSS
JS
Django
PostgreSQL

Everything in concert

- A typical website feature requires changes to *most* of these components in a synchronized manner.
- JS, CSS, djHTML, Django and PostgreSQL layers change almost all the time.
- Errors in CSS and JS are especially difficult to track down, because most browsers hide them.

Containerization

- Nginx, Django and PostgreSQL (and the optimization backend) all run in Docker *containers*.
- Containers provide a fixed and reproducible software environment where components can only affect their own designated "living space", interacting only with the other components in limited, predefined ways.