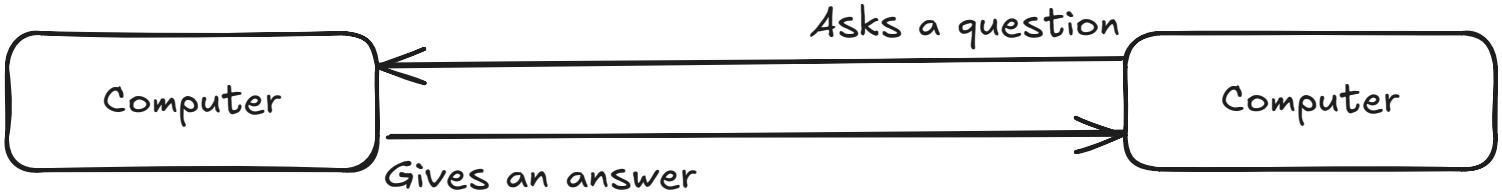
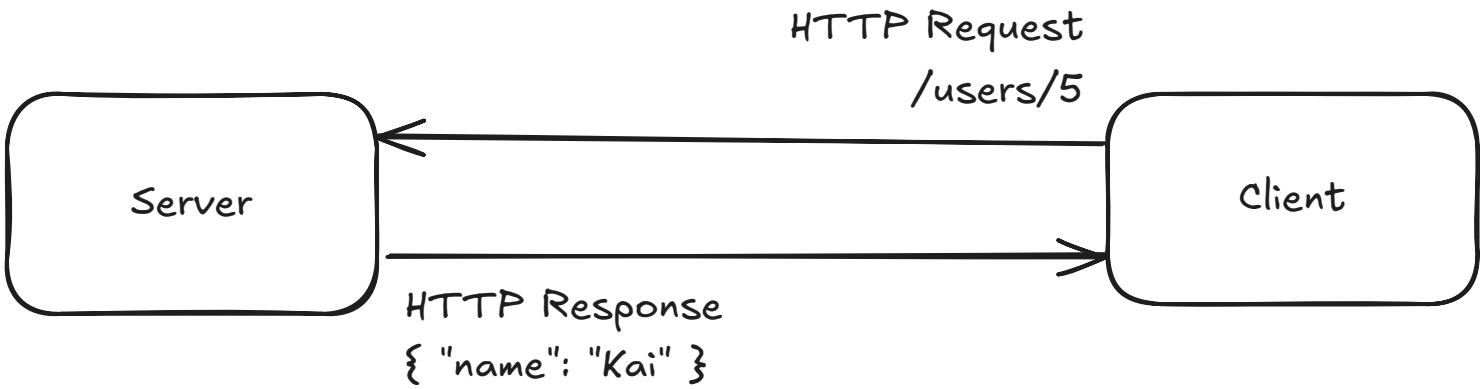
Take 2

APIs 101

Talking to other computers


Asks a question
Computer
Computer
Gives an answer


HTTP Request
/users/5
Server
Client
HTTP Response
{ "name": "Kai" }

URLs

- Uniform Resource Locator
- Way to specify an address on the web

Protocol		Host:Port	Path	Query
`https`	`://`	`www.google.com`	`/`	`?q=what+is+a+url&hl=en`
`https`	`://`	`www.take2.org`	`/our-mission`	
`https`	`://`	`stuff.co.nz`	`/`	
`http`	`://`	`localhost:3030`		
`http`	`://`	`192.168.1.254`		

URLs

When talking to servers (which are just other computers), we need to specify:

- **Protocol**: how to "speak" to the other computer
- **Host**: which name to call the other computer.
- **Port**: A server can expose up to 65,535 different ports for different services, (e.g. email server, web server, file sharing server all on the same computer)
- **Path & Query**: tells the other server which information are we looking for

HTTP

- We send a request → the other computer sends a response
- Request has: URL, Headers, Method, Body
- Response has: Headers, Method, Body

Body & Headers

- The body represents the content of the request and response, e.g. the source code for a website
- Headers are related to the content
- Example request headers:
  - `Accept`: what format would we prefer the information in
  - `Authorization`: what's our username and password
- Example response headers:
  - `Status:`: code (e.g. 200 → OK, 404 → Not Found)
  - `Content-Type`: what format is the information actually in

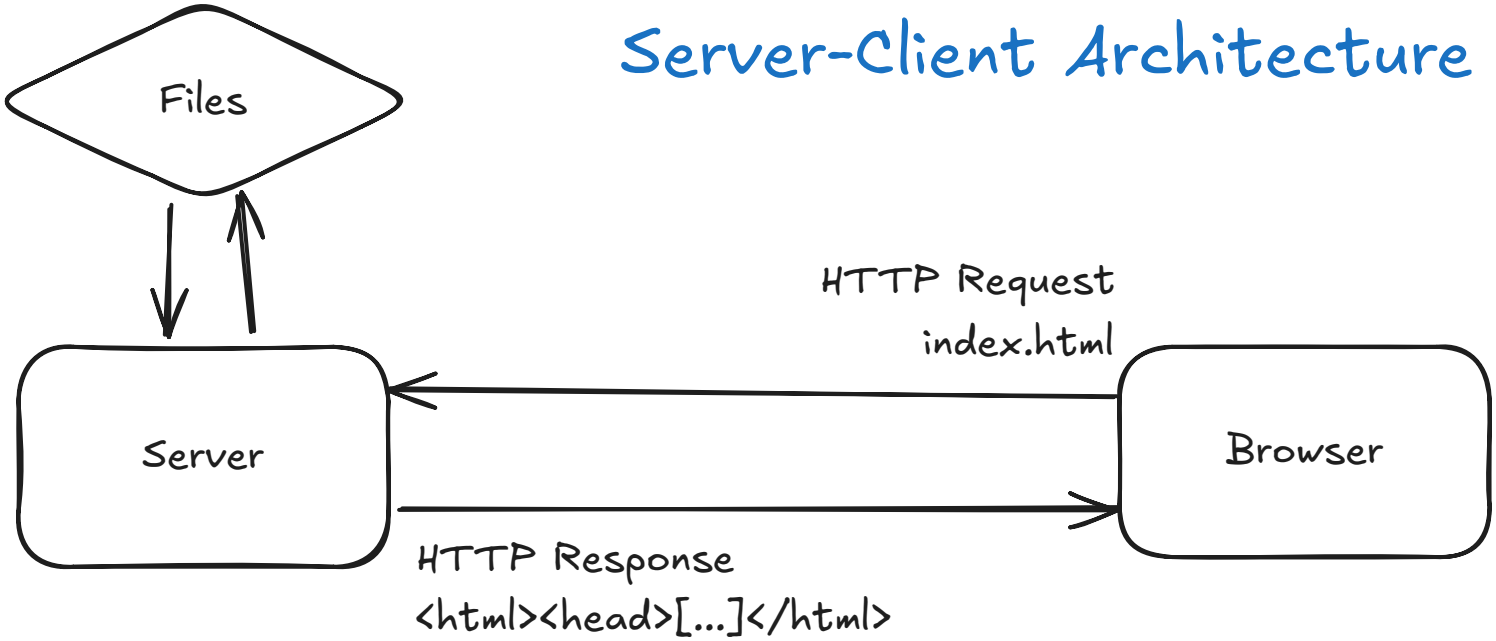
Method

- Sometimes called the HTTP verb
- `GET` : Retrieve something (like a webpage)
- `POST` : Create something (like a new item in a shop)
- `PUT` or `PATCH` : Modify something
- `DELETE` : Delete something

Can we see these requests?

Using Chrome DevTools

Requesting a website



Requesting information and rendering with React

JSON

- JavaScript Object Notation
- We get it from somewhere and use it in JavaScript
- Almost like JS objects, but all keys are quoted
- **Must** use double quotes

See the difference:

```
{ name: "John"; } // JavaScript
```

```
{ "name": "John" } // JSON
```

Promises

- Some things take time.
- They are marked with `async`:
  - `async function sleep() { ... }`
  - `async () => { ... }`
- They don't return their return value, but instead a `Promise`
- We must `await` a `Promise` to get to its value

Promises

```
function double(number) {
  return number * 2;
}
console.log(double(6)); // 12
```

```
async function double(number) {
  return number * 2;
}
console.log(double(6)); // Promise { status: ... }
console.log(await double(6)); // 12
```

'Old' syntax

```
function double() {
  return new Promise((resolve, reject) => {
    resolve(number * 2);
  });
}

const result = double(6)
  .then(function(result) {
    console.log(result);
  });
```

fetch

- Talk to other computers.
- Takes two arguments: URL (string) and options (object)

```
const response = await fetch("www.google.com");
console.log(response); // <!doctype html><head>....
```

- Let's talk to an API in React

```
1  const [forecast, setForecast] = useState();
2
3  const response = await fetch(
4    "https://api.open-meteo.com/v1/forecast" +
5      "?latitude=-36.86&longitude=174.77" +
6      "&daily=temperature_2m_max,temperature_2m_min",
7  );
8
9  if (response.ok) {
10   let data = await response.json(); // Convert JSON to JS
11   setForecast(data);
12 }
```