HATEOAS

Episode V: The Form Strikes Back

QUOIN

WE SIMPLY BUILD GREAT SOFTWARE

Hữu Đà Trần

Highlights of Episode IV

```
GET /api
```

```
200 OK
{
  home: “/api”,
  phonebook: “/api/phonebook”,
  other: “/api/others”
}
```

```
GET res.phonebook
```

```
200 OK
{
  home: “/api”,
  self: “/api/phonebook”,
  data: [
    { id:1, name:“A”, number:“123”
      url: “/api/phonebook/1”
    },
    { id:2, name:“B”, number:“456”,
      url: “/api/phonebook/2”
    }
  ]
}
```

```
PUT res.data[0].url
```
(to edit)

```
POST res.self
```
(to create a new one)

```
DELETE res.data[1].url
```
(to delete)

Hypermedia!!!

Highlights of Episode IV

One entry point to remember: `/api`

Contract to keep the attributes the same: `res.phonebook`

The server to generate the personalized URLs: `/api/address/2`

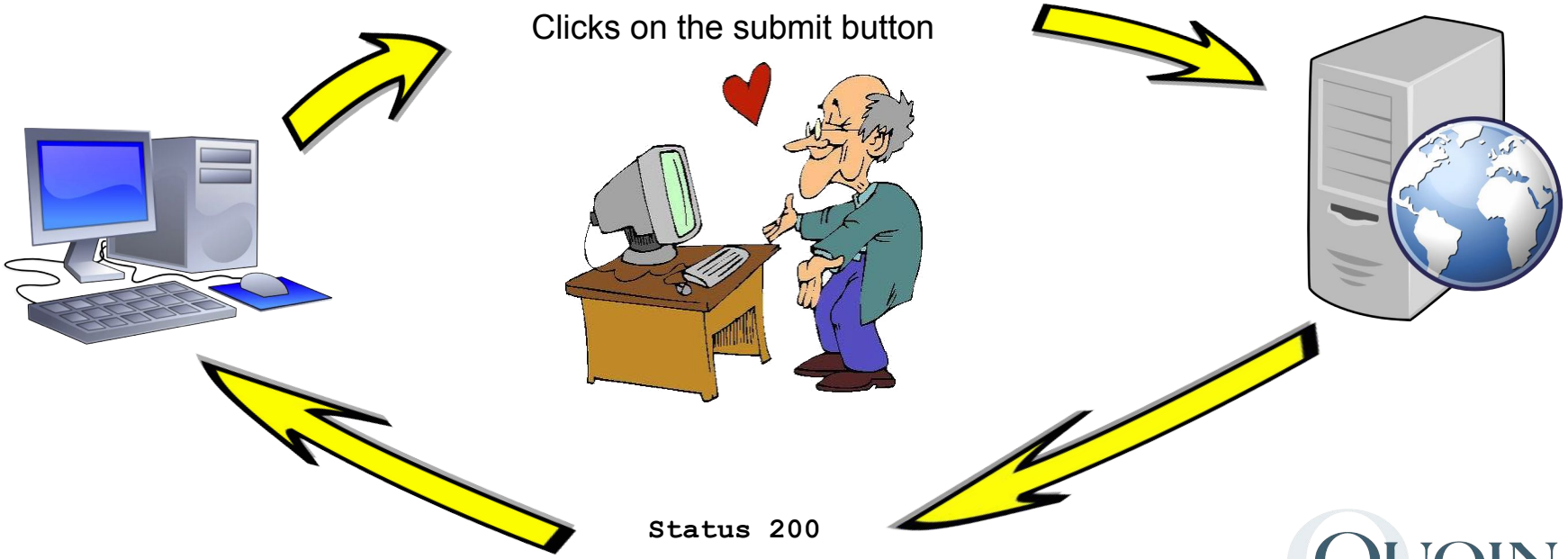
Use the verbs for actions to reduce payload: `GET, POST, PUT, DELETE`

There is still more about HATEOAS.

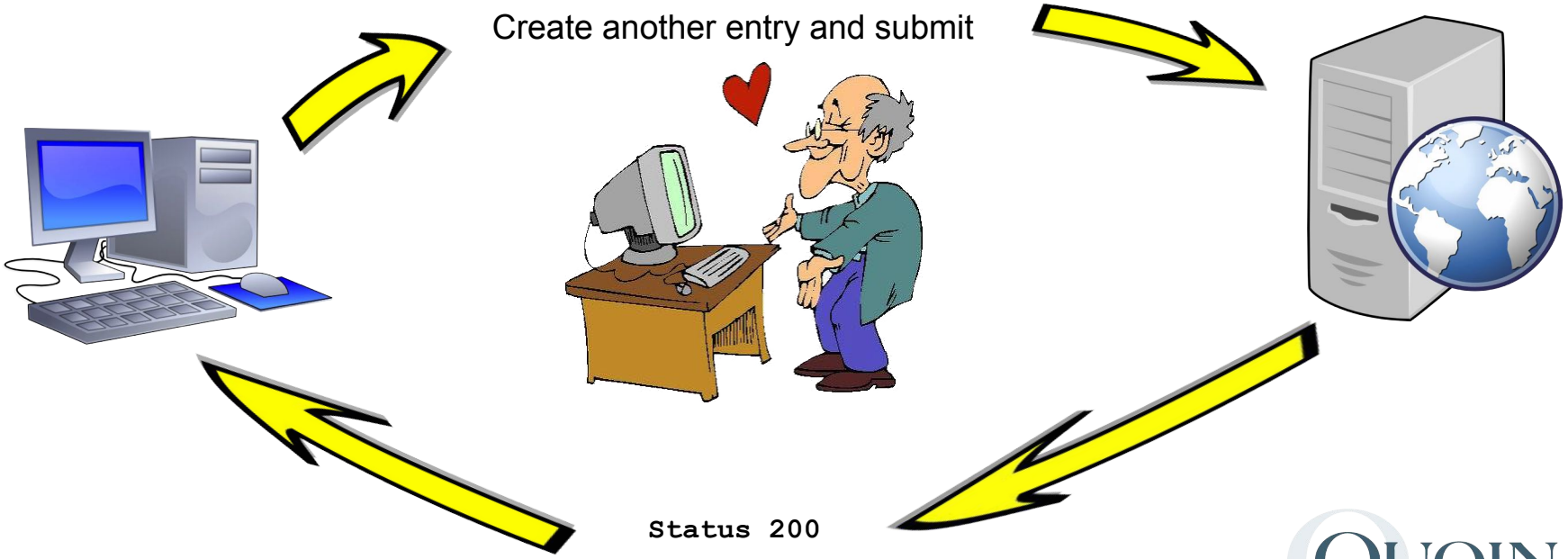
Traditional form

```
<form method="post" action="someUrl" enctype="...">
    <input type="hidden" name="name1" value="value1" />

    <label>
        <span class="lastName">Lastname:</span>
        <input type="text" name="last" maxlength="32" value="{{currentLast}}" />
    </label>

    ...

    <input type="submit" value="{{createOrUpdateLabel}}" />
</form>
```

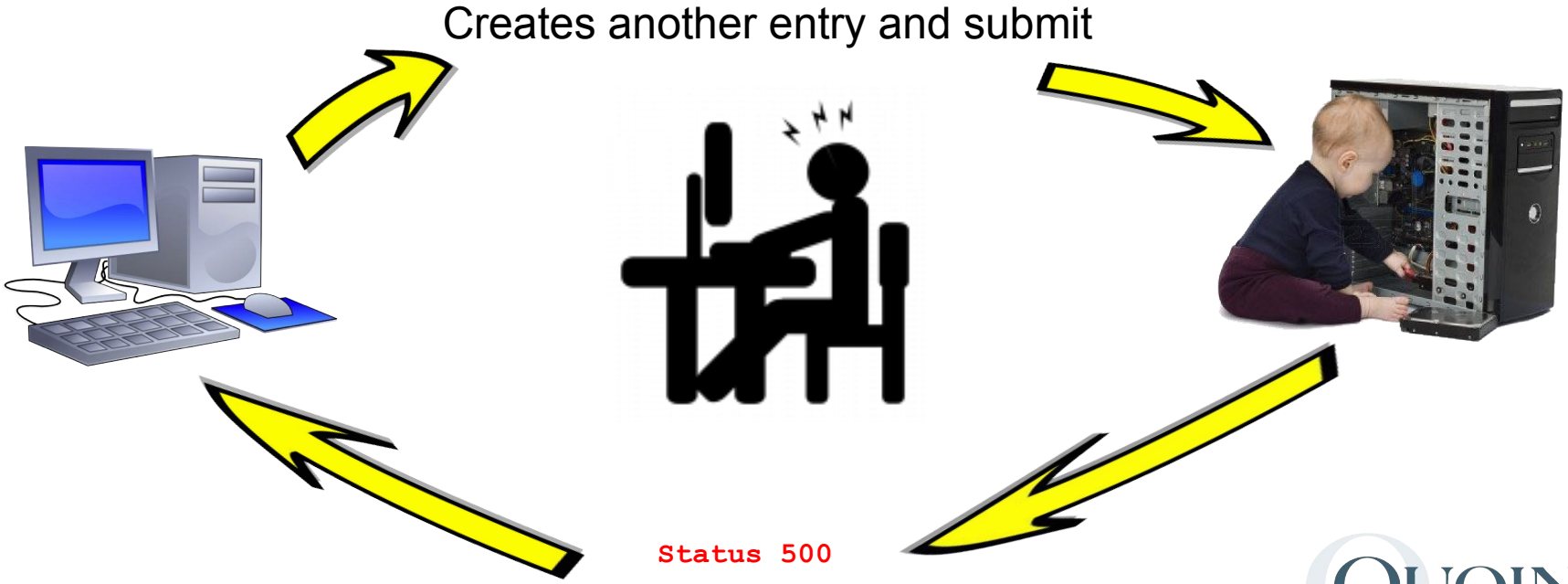
Traditional form



Why is it a problem?



Still worked. So what?



Wait? What happened?

Just like in the last episode, someone did some change on the server, because all decisions are great and should not affect users because we all know better about what the user wants!

ASSUME

Just because
I like recycling.

How can to avoid it?

Let the server tell us what the form should be.

No, not let the server generate the form.

???

Server-side definition

Something like JSON Schema.

```
{
    “form”: {
        “title”: “Phonebook”,
        “type”: “object”,
        “properties”: {
            “name”: {
                “type”: “string”,
                “required”: true,
                “maxlength”: 32
            },
            “number”: {
                “type”: “string”,
                “required”: true,
                “maxlength”: 50
            }
        }
    }
}
```

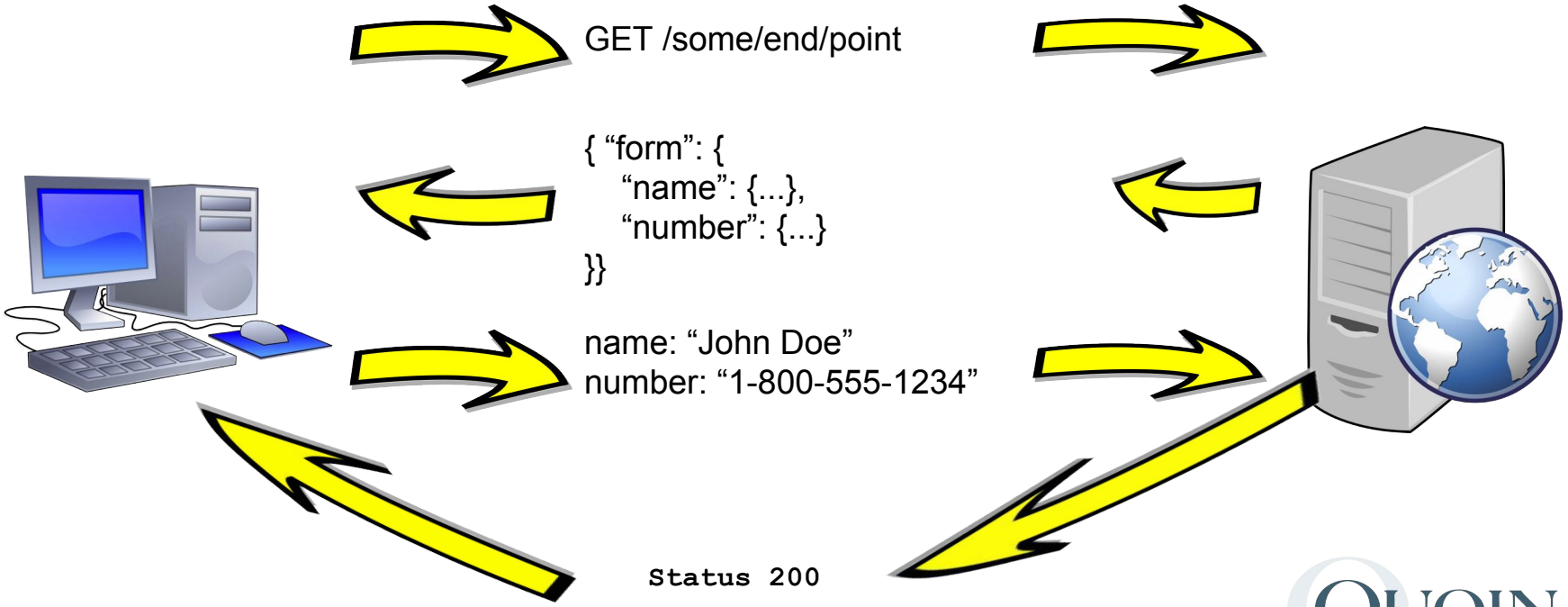
Cengage has adopted WADL for SOA (Web Application Description Language)

Client-side rendering

Yes, more code here on client-side for dynamic form generation.

In practice, we did not have a chance to implement the rendering, but we used that info to display, and to know what to send back to the server.

The day before D-day



The night before D-Day

name(32)

firstName(20)

lastName(20)

The night before D-Day

Because we all document that, right? TDD...

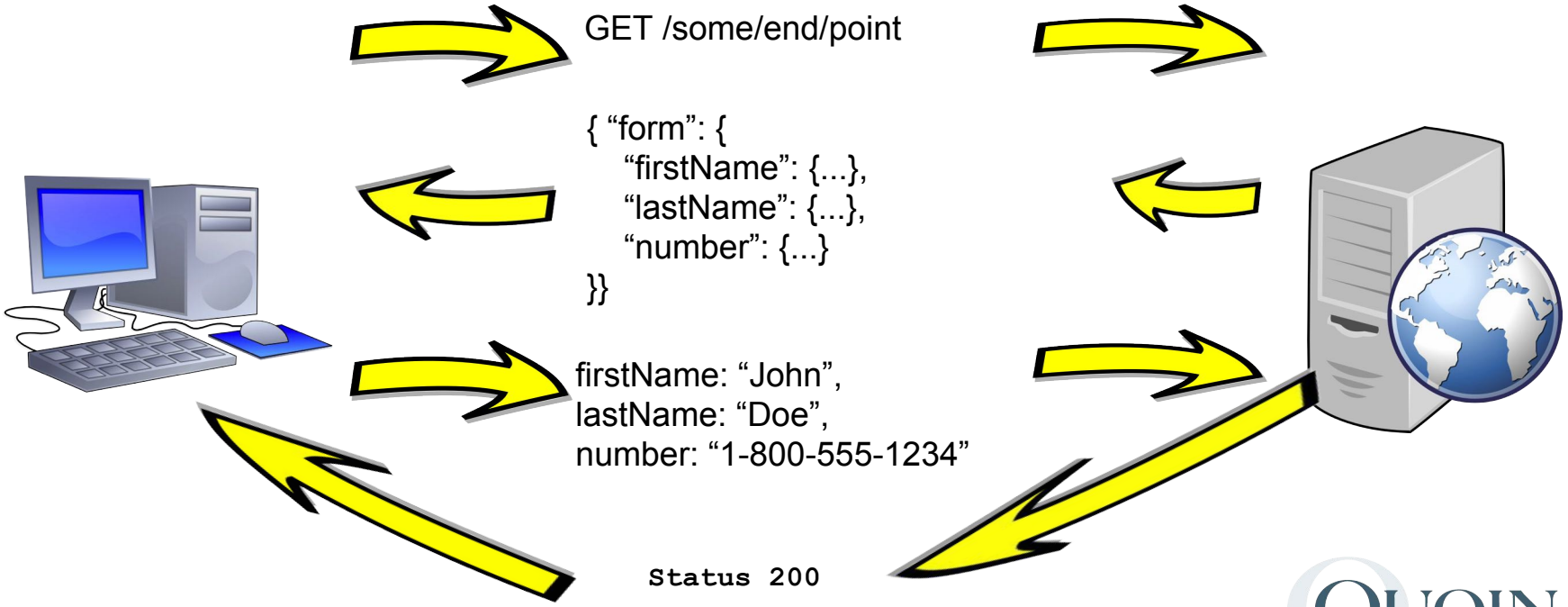
23:45 [Engineer] Hey guys, I thought we can split the name into two fields instead.

23:53 [Sales] That would be great, I know a client that would love that.

23:58 [Engineer] Great, I put it in prod, let me know if that works well.

00:02 [Sales] Thanks so much. I'll try it tomorrow and let you know.

D-Day



D-Day

Because we all document that, right? TDD...

09:47 [Sales] That worked great. I was waiting for this for so long.

09:51 [QA] Wait, what? What card was that?

Summary

Form definition is driven by the server: field name, required, maxlength, min-max value, options.

No change needed on client-side if server changes.

Questions?

This page is intentionally left blank