HATEOAS

Episode IV: A new REST

QUOIN

WE SIMPLY BUILD GREAT SOFTWARE

Hữu Đà Trần

HATEOAS?

Misspelled IHATESOA? (Service-Oriented Architecture)

New buzzword means new ways to make money!

Hypermedia **A**s **T**he **E**ngine **O**f **A**pplication **S**tate



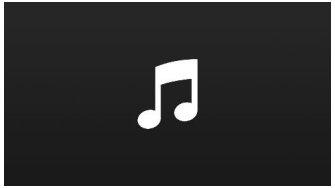

Hypermedia?

Everybody should know this one!

Web

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc in rutrum arcu. Mauris ullamcorper augue.

"Hyper"



Episode IV: A New REST

(REpresentational State Transfer)

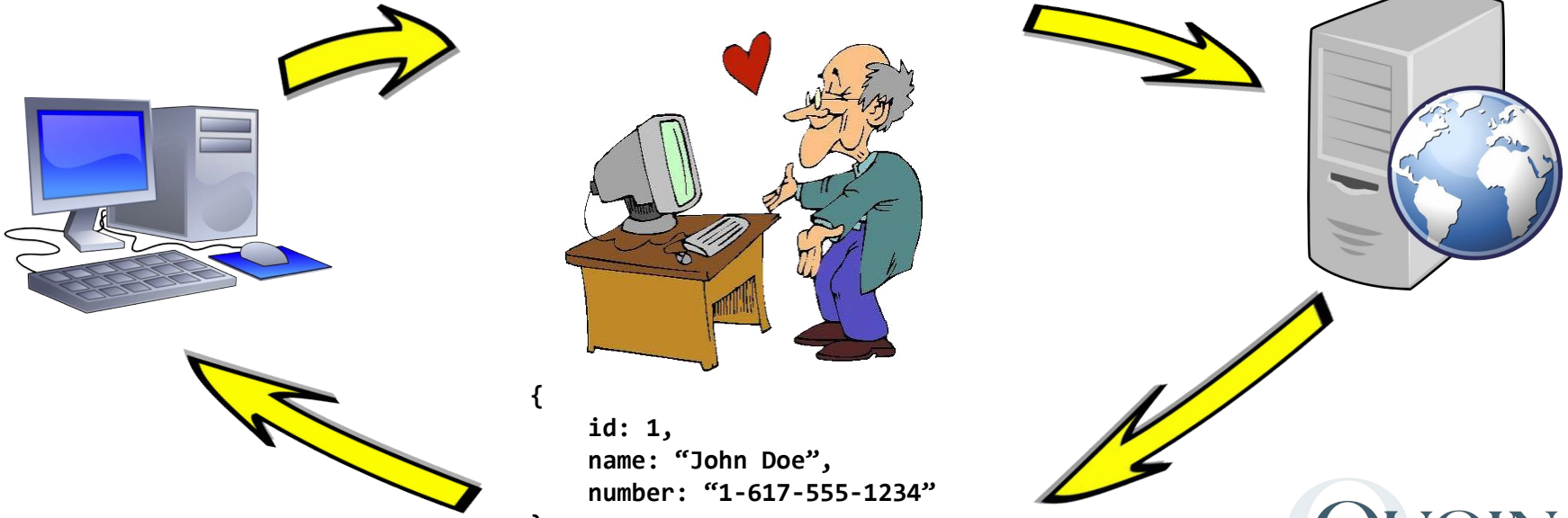
A constraint of the REST application architecture that distinguishes it from most other network application architectures.

- Wikipedia

IHATESOA

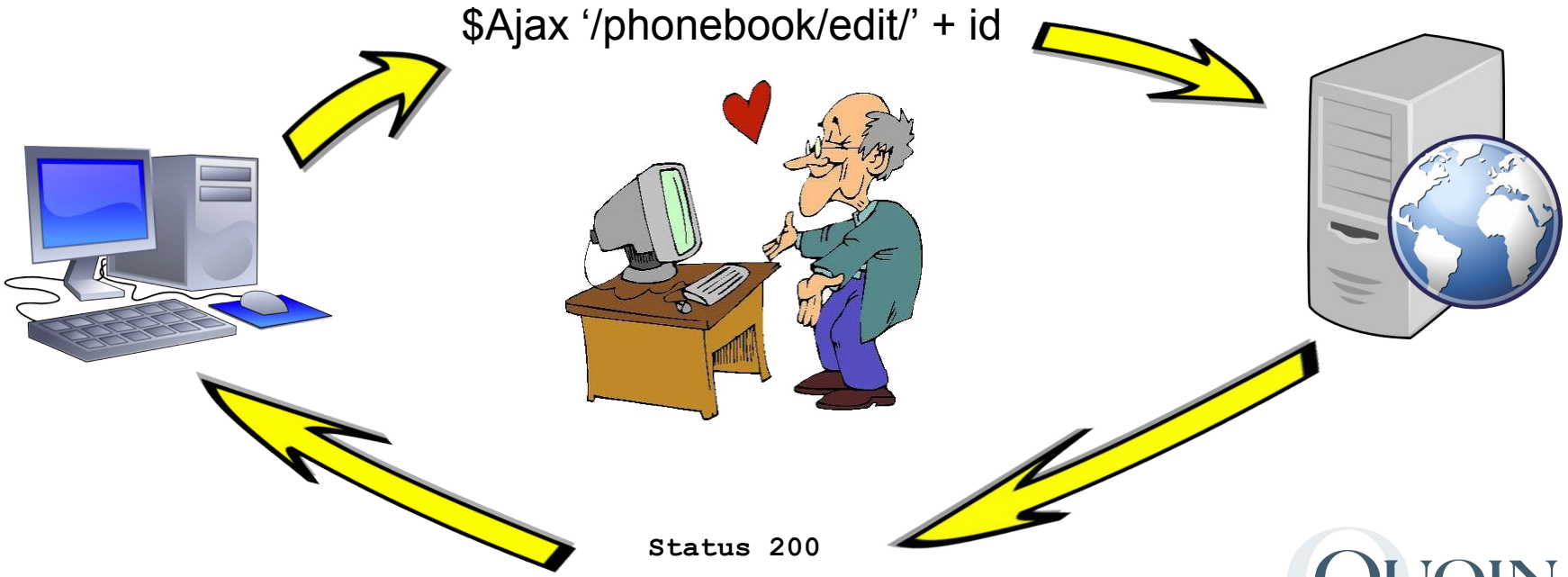
Traditional REST

```
$Ajax /phonebook/list
```



```
{
    id: 1,
    name: “John Doe”,
    number: “1-617-555-1234”
}
```

Why is it a problem?



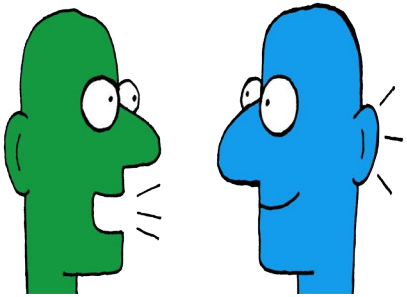
Still worked. So what?

$Ajax '/phonebook/edit/' + id

Status 404

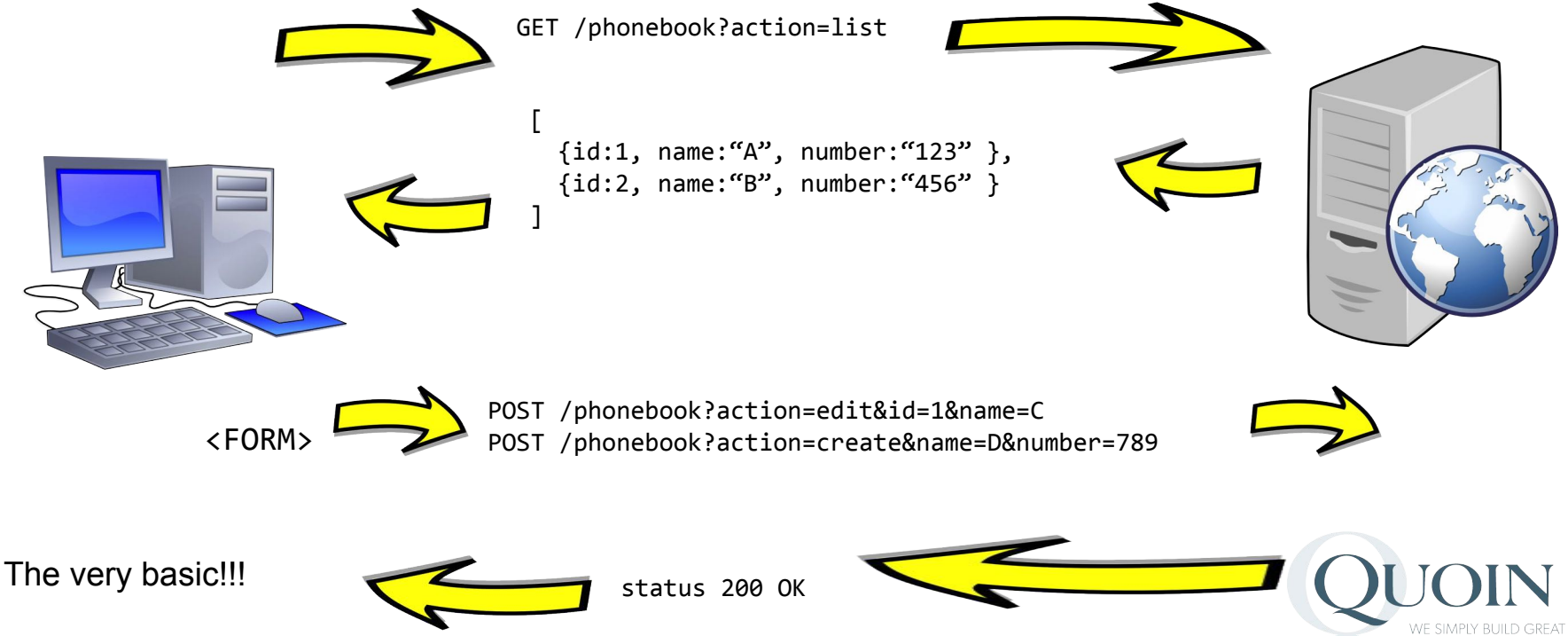
Wait? What happened?

Something changed on the server, but was not properly communicated to the client... offline.



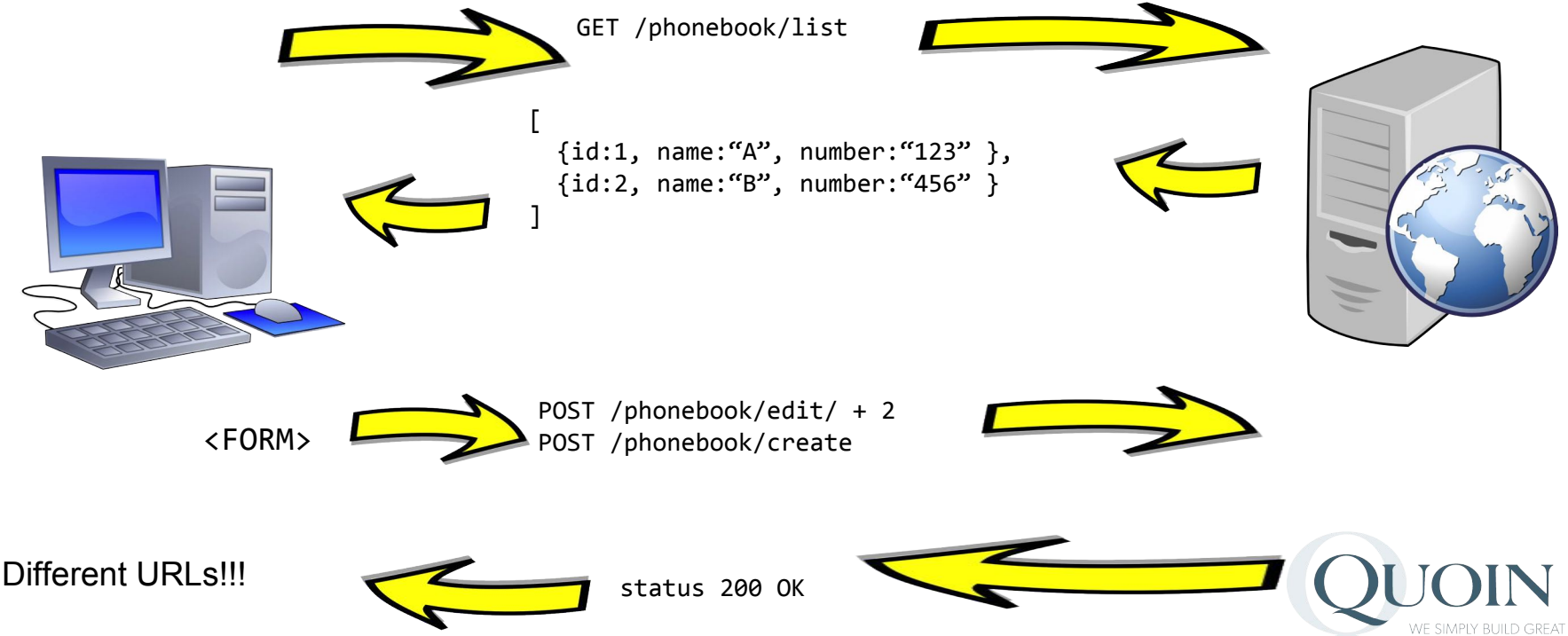
People need to talk to people when something changes.

Phase 1: World War 2

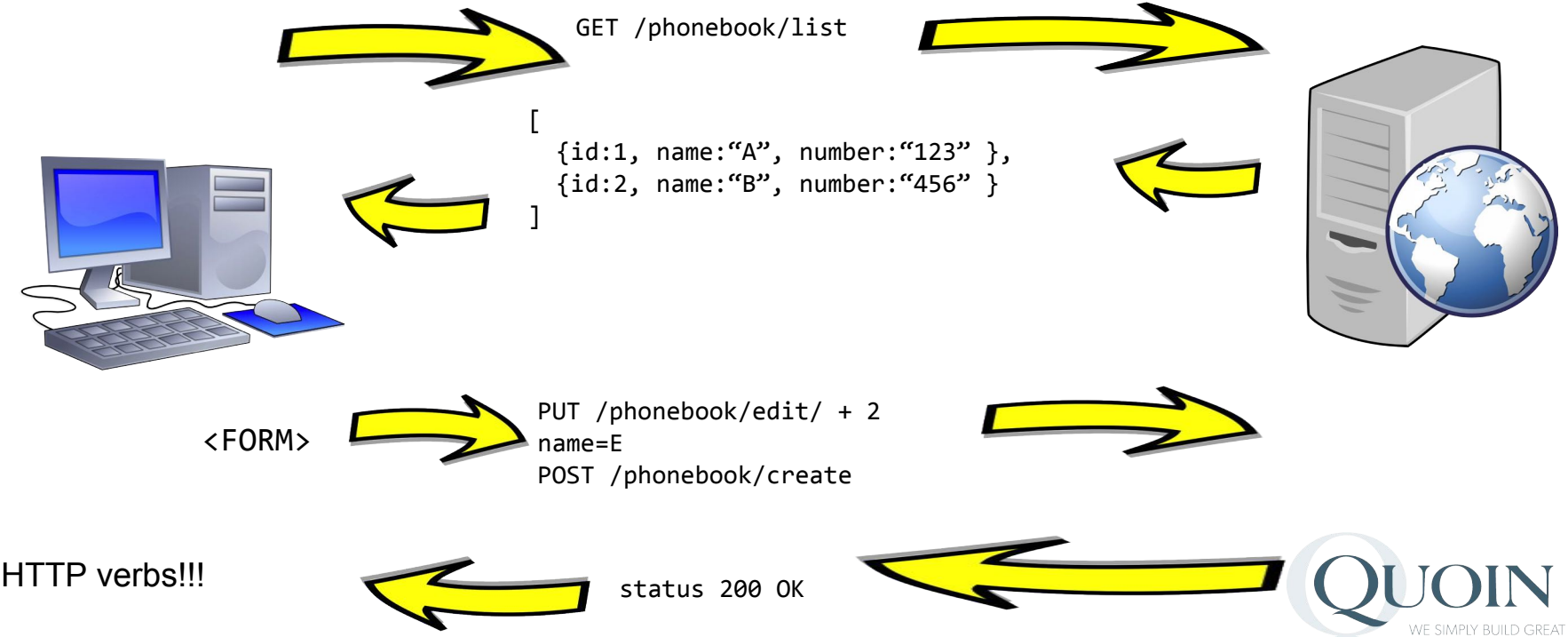


The very basic!!!

Phase 2: Baby Boomers



Phase 3: Millennials



HTTP verbs!!!

Phase 4: Yesterday

```
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!!!

Phase 4: Yesterday night

That was pretty complicate compared to what we are used to.

Phase 4: Today

Yes, that’s my boy!!!

Someone makes a change on the server!!!

Phase 4: Today night

WAIT!!! Did something just happened?!?!

I must lack some sleep!!!

Phase 4: Tomorrow

```
GET /api
```

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

```
GET res.phonebook
```

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

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

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

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

WOW!!!

Phase 4: Tomorrow night

Phase 4: Summary

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.

Questions?

This page is intentionally left blank