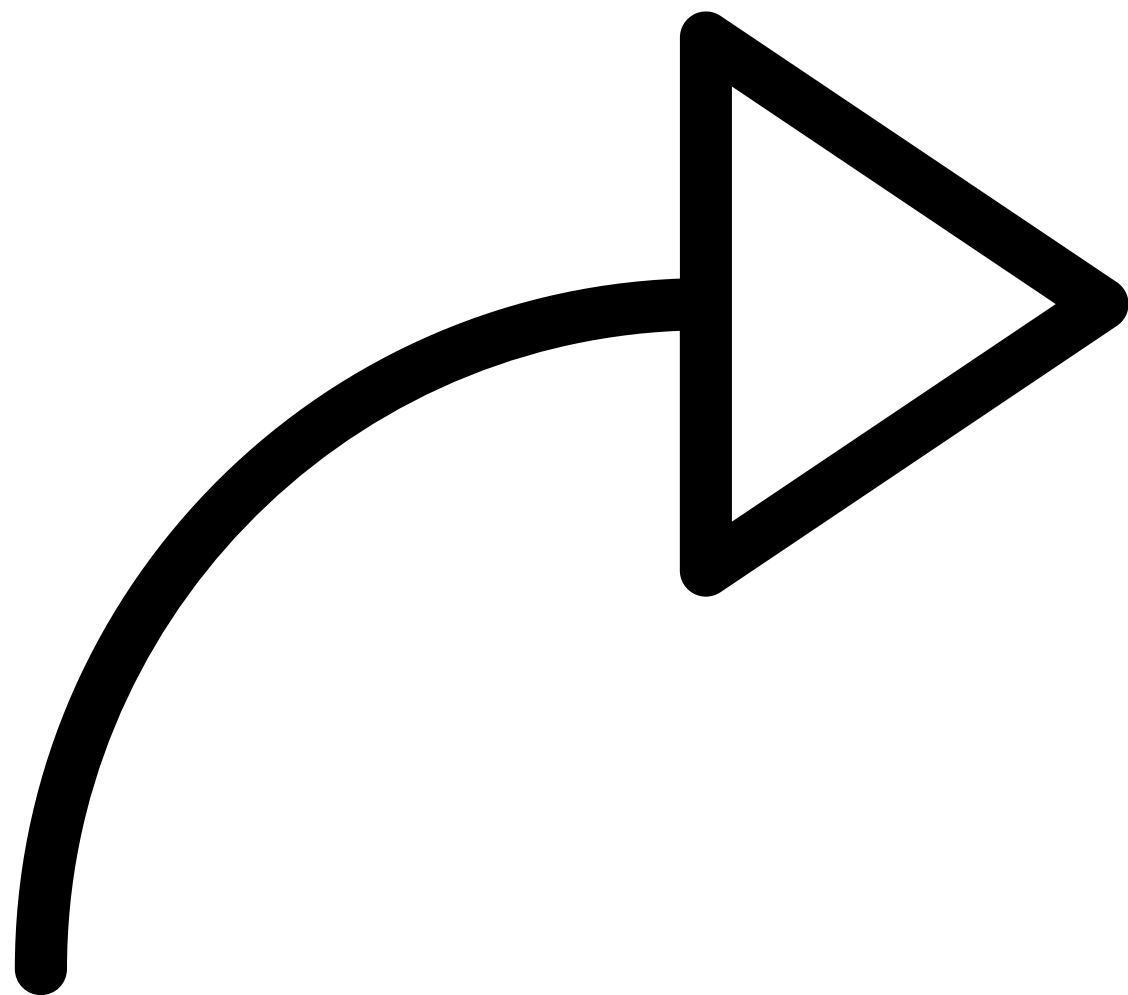


Making the Request

Client-side

Think: “What do I want from the server?”



```
xhr = new XMLHttpRequest();
xhr.onreadystatechange = handle_res;
xhr.open("GET", "/movies")

function handle_res() {
    if(this.readyState  $\neq$  4) return;
    if(this.status  $\neq$  200) {
        // error!
    }
    console.log( this.responseText )
}
```




CS 4241

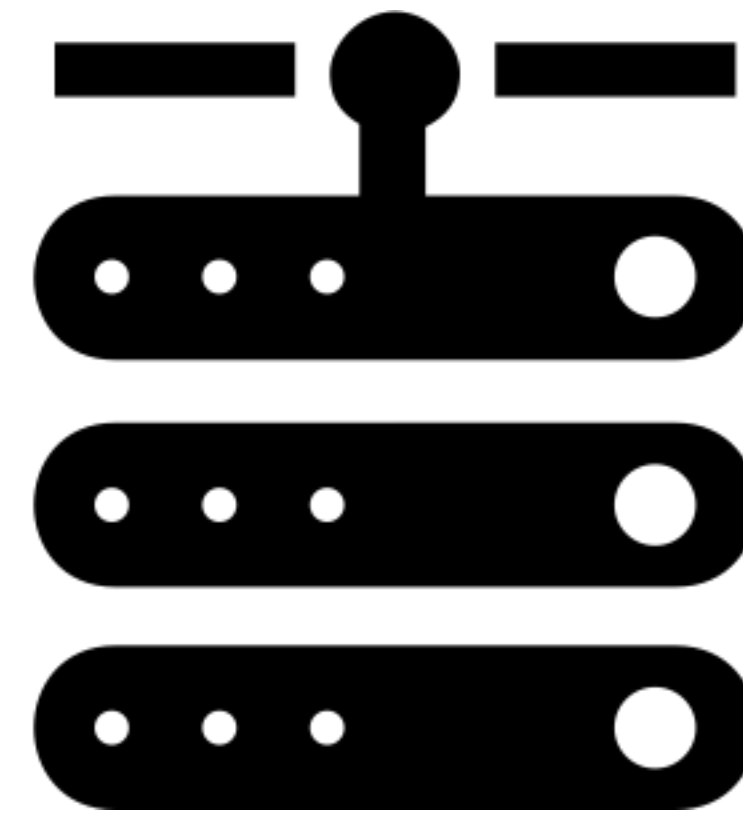
WebWare

ajax (async javascript)

browsers



servers



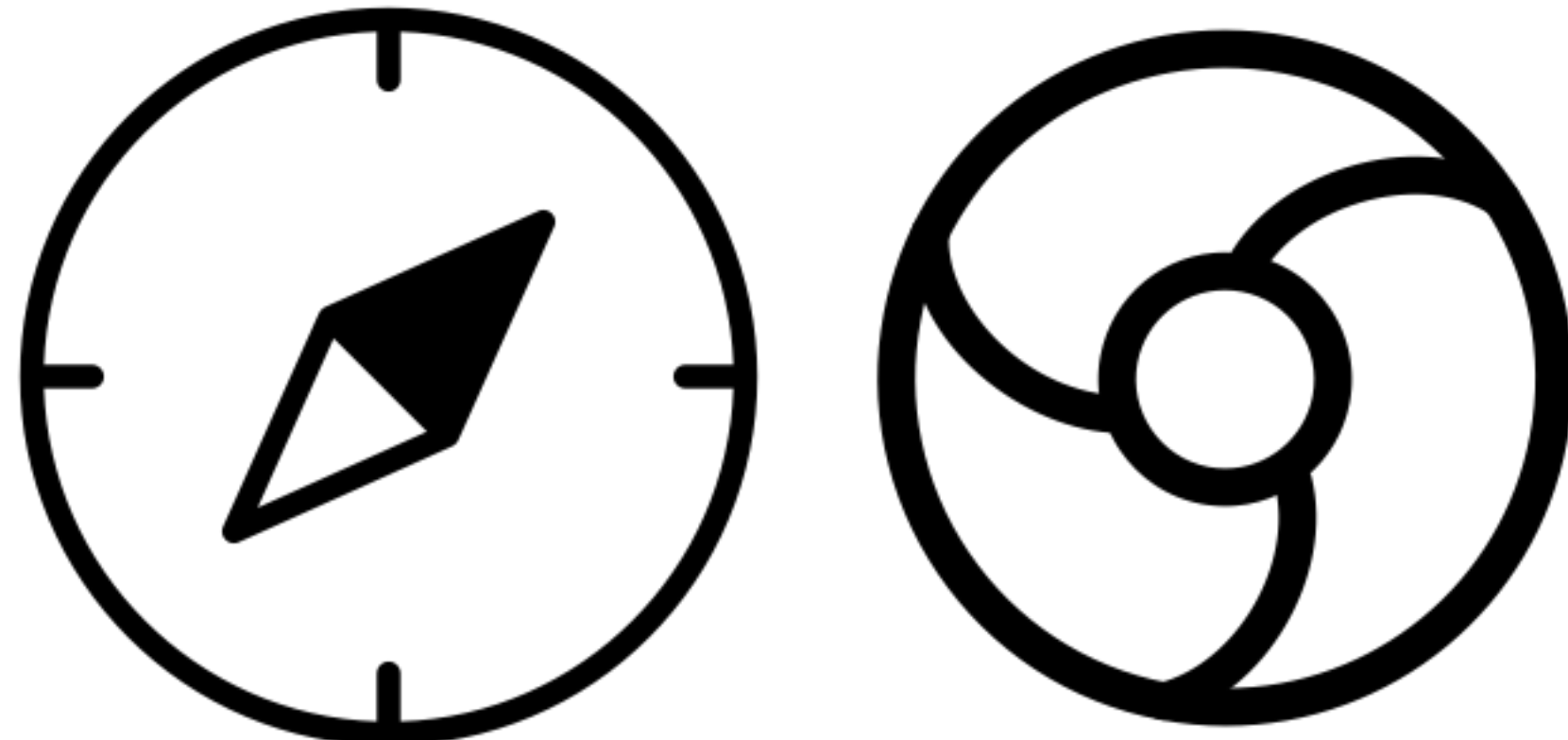
db



http



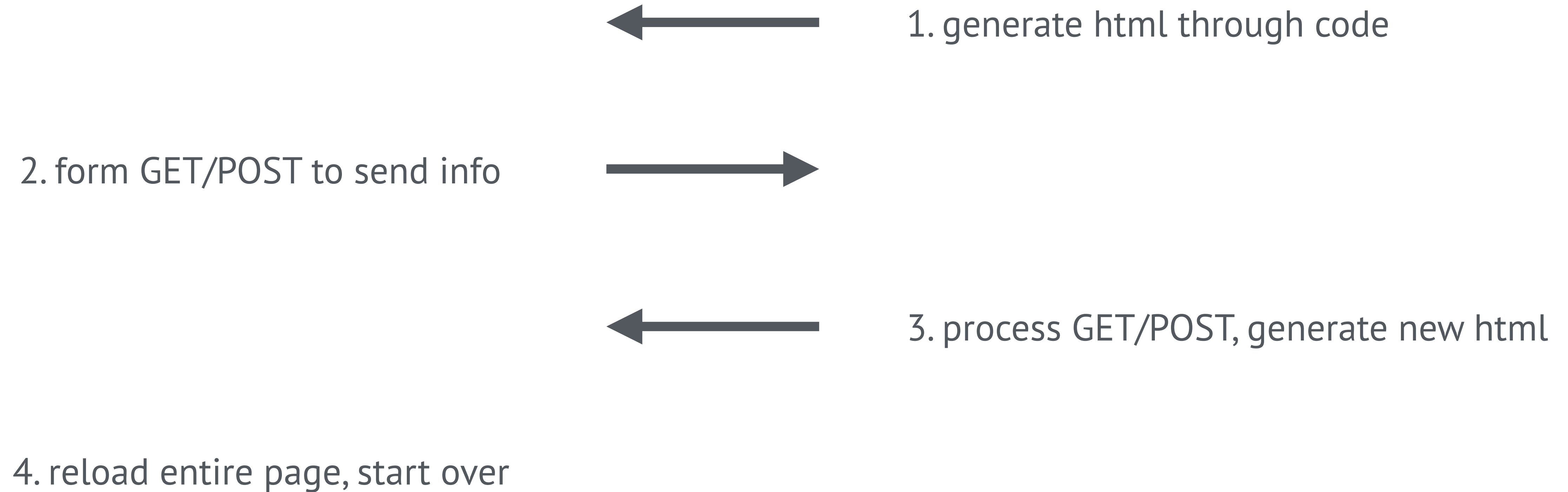
local
(net)



front-end

back-end

the “a3” way



front-end

back-end

a better way



1. serve/generate html

2. GET/POST ****anywhere**** to
send info



3. process GET/POST, generate only
data the client needs



4. receive response; update page
while running

front-end

back-end

ajax: asynchronous javascript

history lesson:

ajax -> created by Microsoft

ajax -> asynchronous javascript and
XML (noooooo)

fortunately, ajax is now used with
more readable formats, like json

```
<?xml version="1.0"?>
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications
with XML.</description>
  </book>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-12-16</publish_date>
    <description>A former architect battles corporate
an evil sorceress, and her own childhood to become
of the world.</description>
```

using ajax: XMLHttpRequest

```
// making your first request
```

KEY -> you are assigning a function

```
xhr = new XMLHttpRequest();  
xhr.onreadystatechange = handle_res;  
xhr.open("GET", "/myroute")
```

can specify POST/GET/DEL, etc.
route is url to send to on server

Called multiple times:

0 -> open not called yet

1 -> req has been sent

2 -> headers were received on server

3 -> response is loading

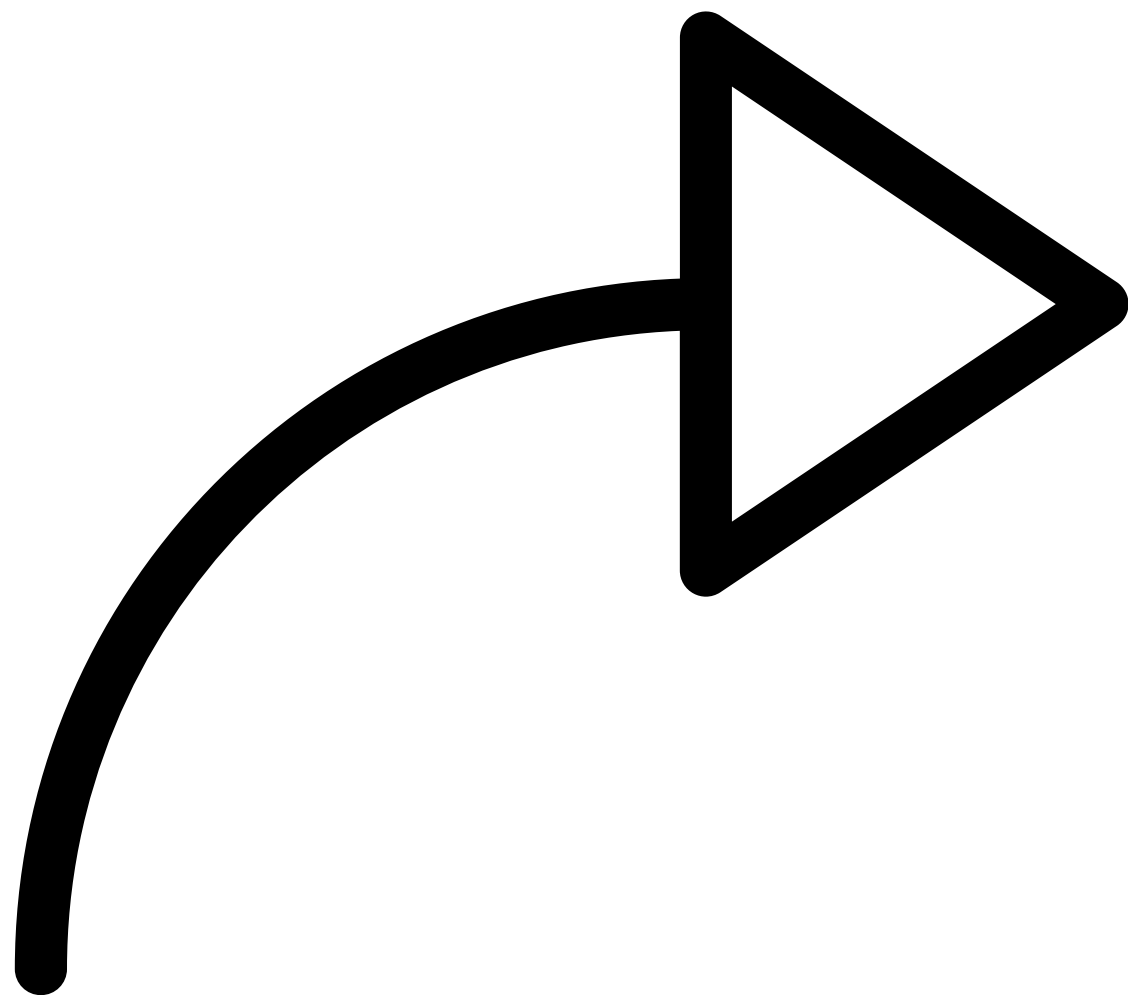
4 -> done; responseText available

```
function handle_res() {  
    if(this.readyState  $\neq$  4) return;  
    if(this.status  $\neq$  200) {  
        // error!  
    }  
    console.log( this.responseText )  
}
```

Making the Request

Client-side

Think: “What do I want from the server?”



```
xhr = new XMLHttpRequest();
xhr.onreadystatechange = handle_res;
xhr.open("GET", "/movies")

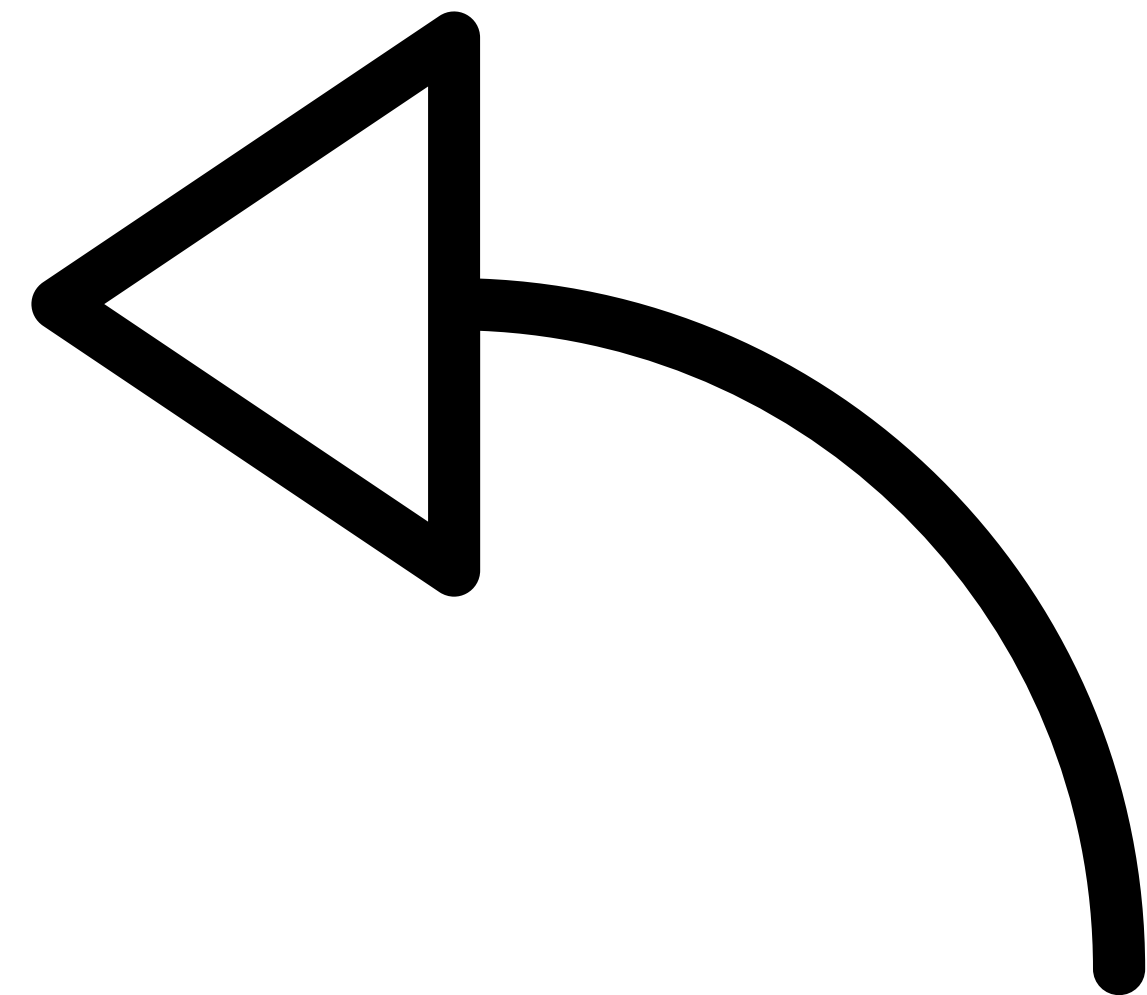
function handle_res() {
    if(this.readyState  $\neq$  4) return;
    if(this.status  $\neq$  200) {
        // error!
    }
    console.log( this.responseText )
}
```


Processing the Request

```
if(req.method == "GET")  
...  
case: 'movies':  
    send_movie_dataset(res, movies) //?
```

Server-side

Think: “What did the client ask for?”

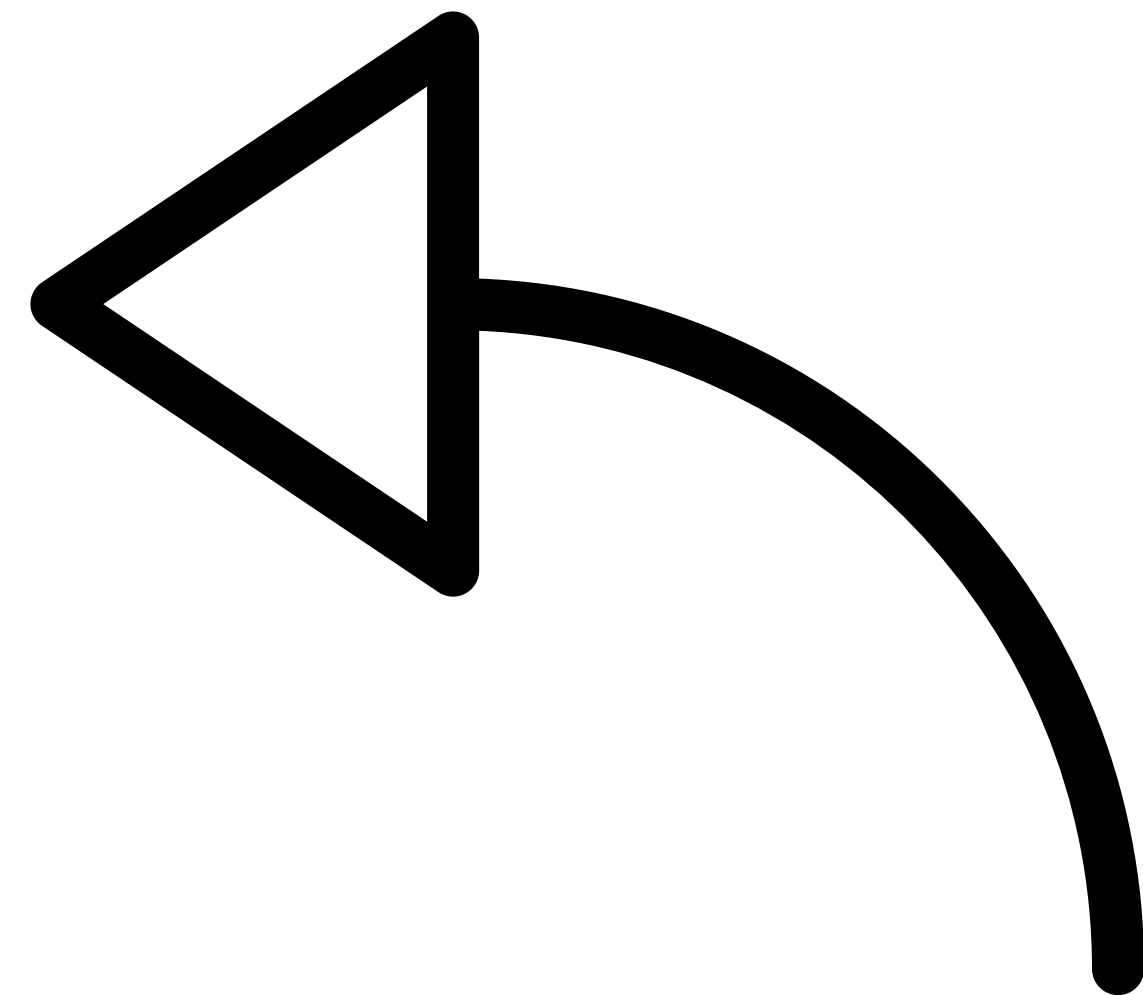


Making the Response

```
//movies = [ {title: "Sharknado", rating: ...  
function send_movie_dataset(res, movies) {  
  res.end(movies) //wrong  
  res.writeHead(  
    200,  
    {'Content-type': 'application/json'}  
  )  
  res.end( JSON.stringify(movies) )  
}
```

Server-side

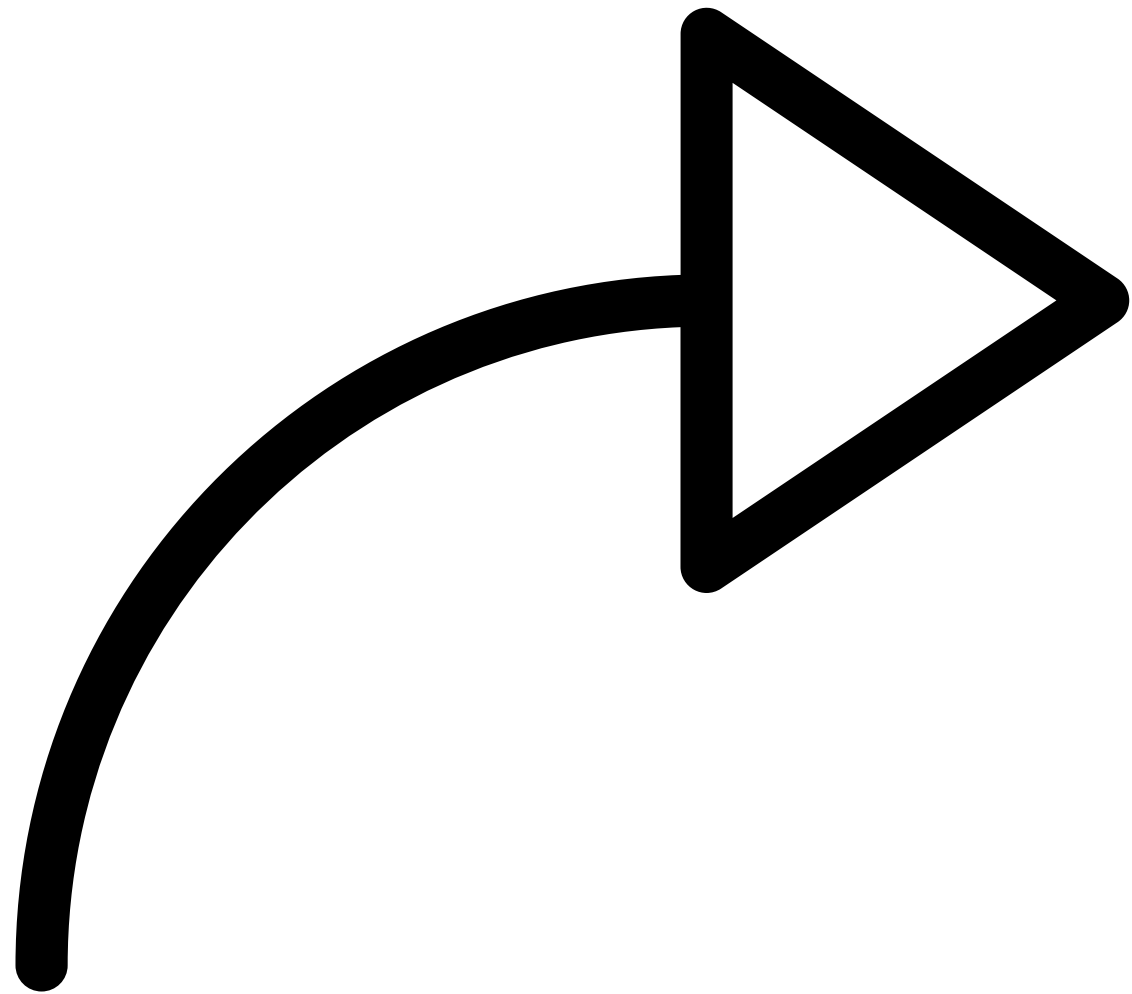
Think: “How can I get it to them
in a format the client can work
with?”



Processing the Response

Client-side:

Think: “How do I act on this new data?”



...

```
function handle_res() {  
    if(this.readyState  $\neq$  4) return;  
    if(this.status  $\neq$  200) {  
        // error!  
    }  
    var movies = JSON.parse(this.responseText)  
    update_movies_div(movies)  
}
```

Some ways to use XMLHttpRequest

...

```
function handle_res() {  
    if(this.readyState  $\neq$  4) return;  
    if(this.status  $\neq$  200) {  
        // error!  
    }  
    var movies = JSON.parse(this.responseText)  
    update_movies_div(movies)  
}
```

...

```
xhr.open("POST", "/movies")  
xhr.setRequestHeader('Content-type',  
'application/json')  
xhr.send(JSON.stringify({  
    "title": "Star Wars",  
    "rating": "5/7"  
}))
```

Getting JSON from server

Also:

sending form data
uploading images
and much, much more

Sending JSON to server

(note, this is hijacking the form action)

Some ways to use XMLHttpRequest

```
d3.json('movies.json', function(err, d) {  
  console.log(d)  
})
```

```
$.get('movies.json', function(d) {  
  console.log(d) //may require JSON.parse  
})
```

using other libs...

...

```
xhr.open("POST", "http://  
someotherserver.com/movies")  
xhr.setRequestHeader('Content-type',  
'application/json')  
xhr.send({  
  "title": "Star Wars",  
  "rating": "5/7"  
})
```

other sites..?