Carnegie Mellon University

95-733 Internet of Things
Four interaction styles

Four Interaction Styles

- Four common client server interaction patterns using HTTP.
- Some details on the last three styles.
- Note: You should be working on Project 1. And prepare for next week's quiz 2 on "The Computer for the 21st Century ".

The Internet Protocol Suite

HTTP, Websockets, DNS, XMPP, MQTT, CoAp	Application layer
TLS, SSL	Application Layer (Encryption)
TCP, UDP	Transport Layer
IP(V4, V6), 6LowPAN	Internet Layer
Ethernet, 802.11 WiFi, 802.15.4 (used by Zigbee, Thread)	Link Layer

Important principles:

Layering, modularity, separation of concerns
Each layer focuses on a particular set of concerns and abstracts these concerns from the layer above.

In this course, we will mainly study the application layer.

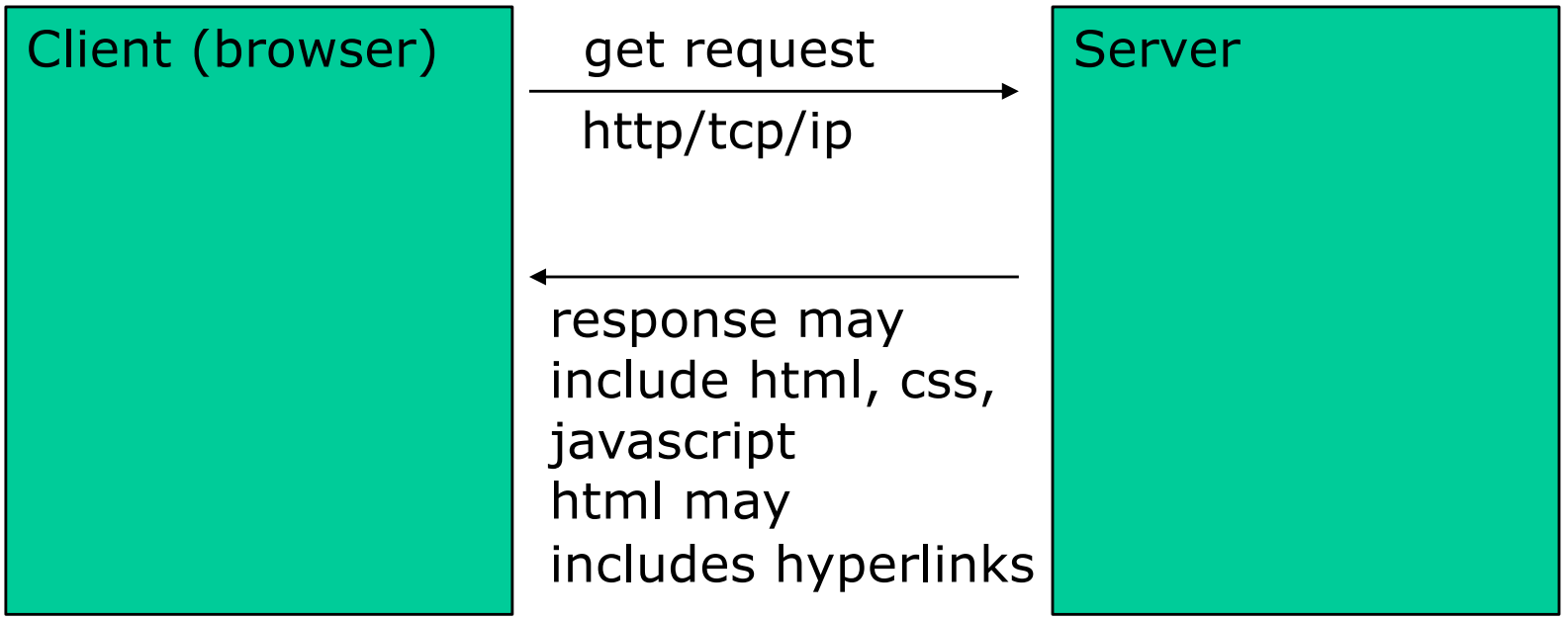
Types of Physical Communication Channels

Communications Channels (two types)

- Wireline
  - Optical (fiber) using light
  - Electrical using voltages and current
    - Twisted pair
    - Coaxial cable
    - Waveguide
- Wireless
  - Optical – e.g. infrared, electromagnetic
  - Radio – most common (from NFC to satellite)
  - Acoustic – e.g. underwater communications

Each of these is a major topic.

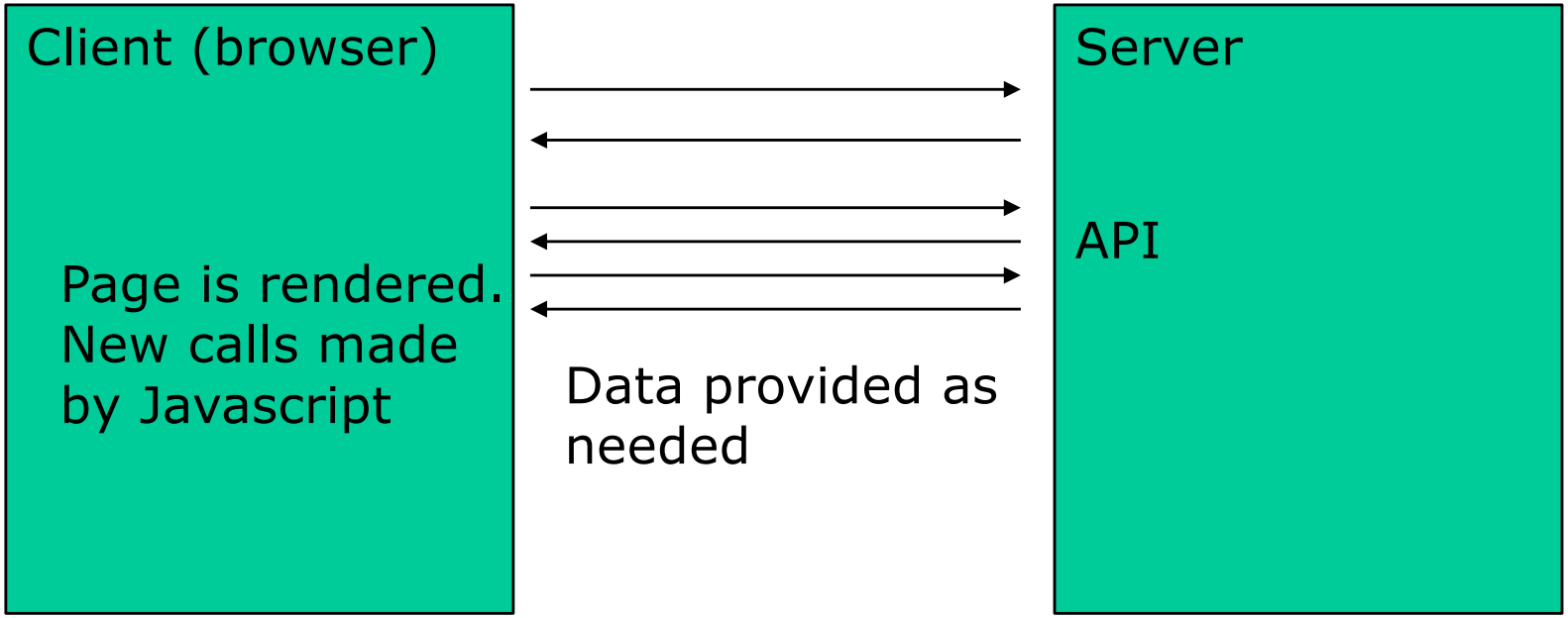
Four Styles of HTTP Interaction Style(1)



The response or user interaction may cause code to execute within the browser

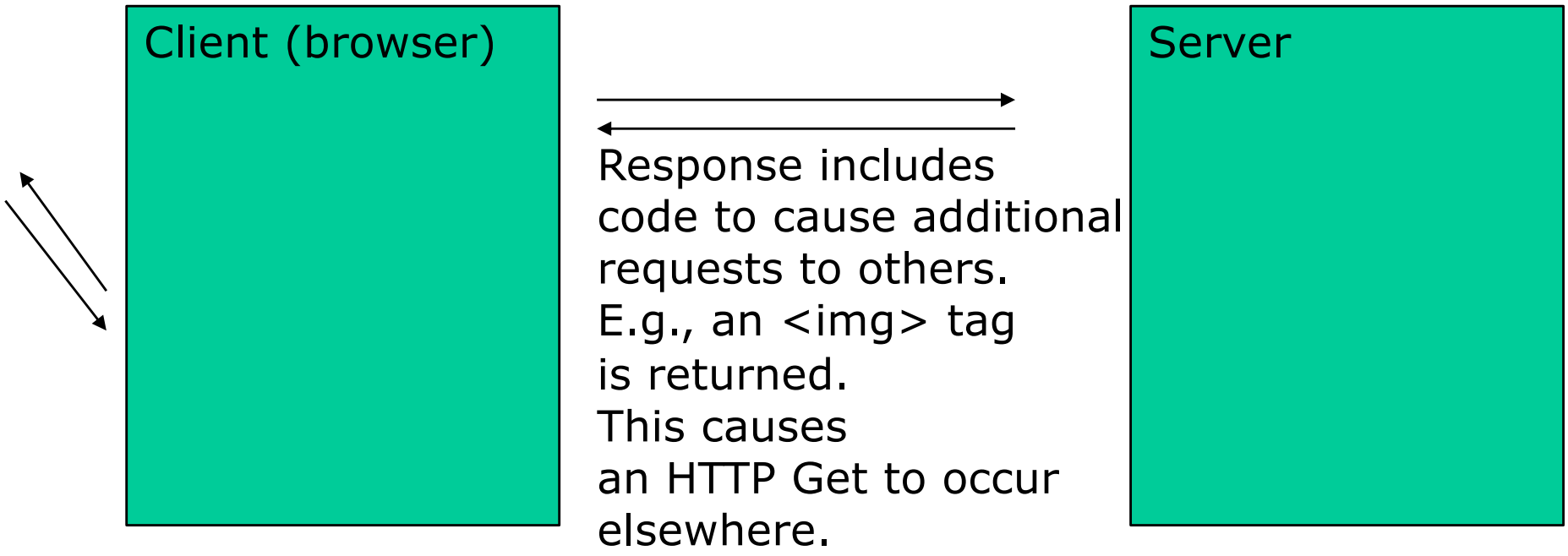
The request may cause code to execute on the server

Four Styles of HTTP Interaction Style(2)



Asynchronous (non-blocking) javascript and xml (AJAX)
No need for a full page refresh
Same origin policy enforced by browser

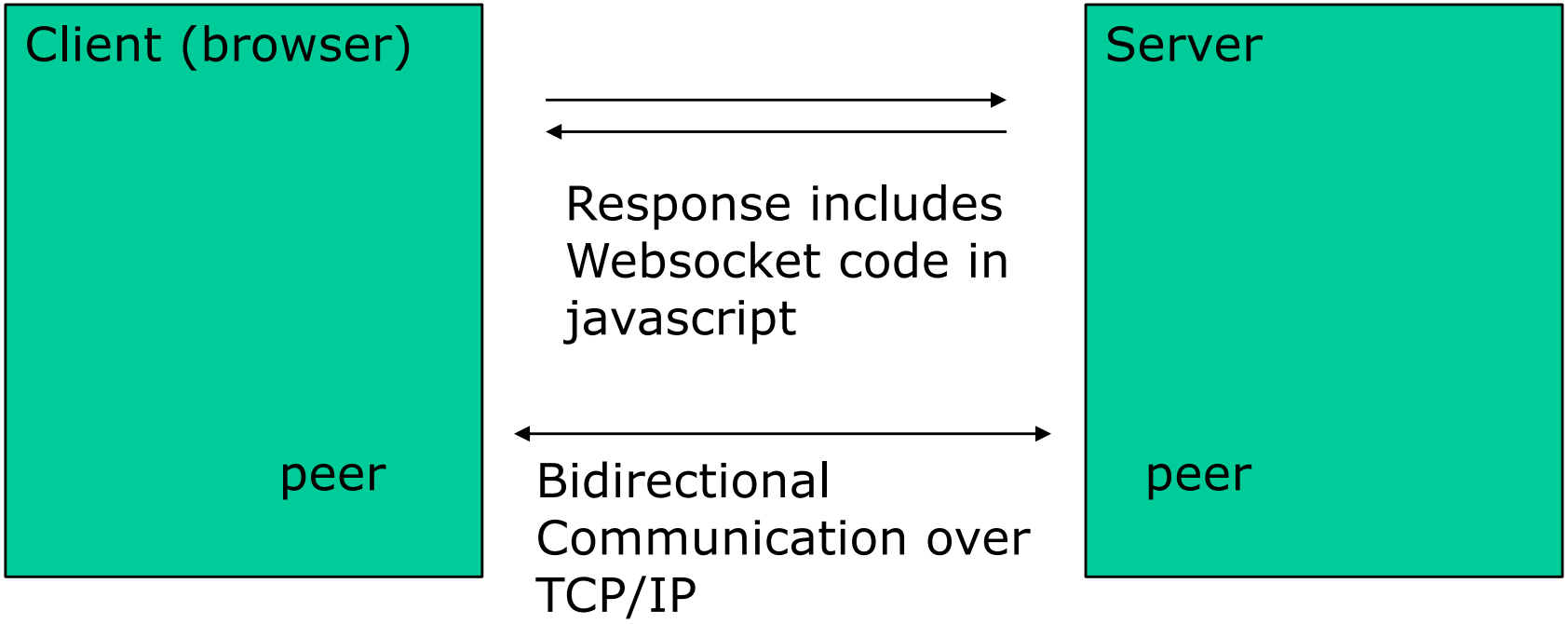
Four Styles of HTTP Interaction Style(3)



Javascript object notation with padding (jsonp) is a popular hack.
No longer restricted to the the same origin.
Upon return, a callback is executed.

Newer systems use CORS - Cross Origin Resource Sharing.
The third party server has a white list of acceptable domains.

Four Styles of HTTP Interaction Style(4)



Once the socket is established, the server may initiate the interaction.
Uses: Web - collaboration apps without polling
IoT - very fast binary or text

AJAX (Style 2)

- The traditional web employed thin clients.
- With AJAX, downloaded code interacts with the server asynchronously. The client does not block.
- This makes for a more responsive user experience.
- Single page applications are possible. This is widely used.

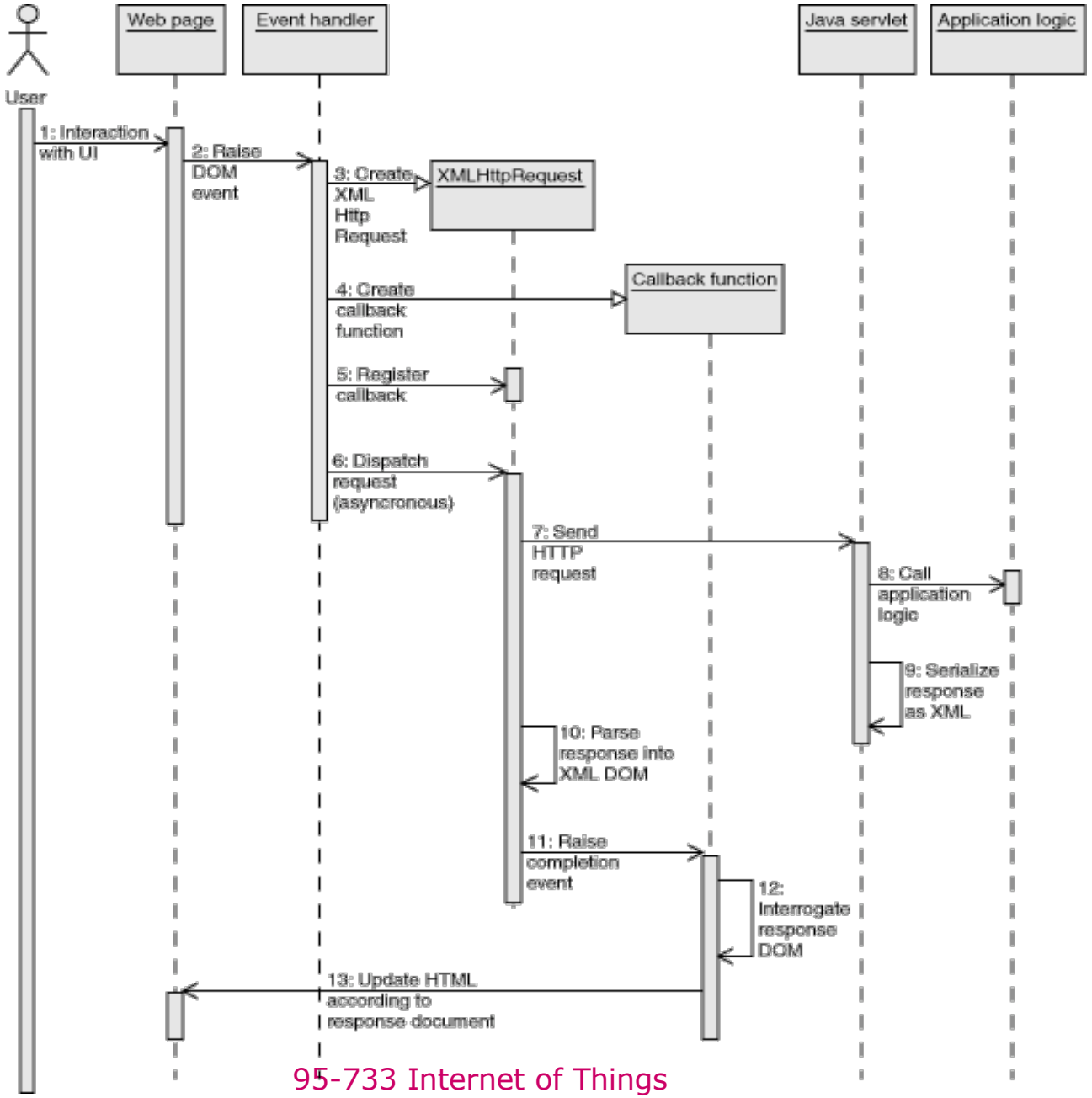
Client and server side Javascript

- Node.js executes the code that runs on the server when an HTTP request arrives.
- A file may be returned to the client but so might Javascript.
- JavaScript is not Java and runs in the browser.
- To experiment with Javascript:

  In Firefox

  tools/web developer/web console/console

Typical AJAX interaction pattern



This is a UML sequence diagram.

This shows a typical AJAX round trip.

A solid arrowhead represents a synchronous call.

A stick arrowhead represents an asynchronous signal.

Javascript Example 1 (W3C)

Some simple JavaScript code to do something with data from an XML document or JSON string fetched over the network:

```
function test(data) {
 // taking care of data
}
```

```
function handler() {
 if(this.readyState == 4 &&
    this.status == 200) {
    // so far so good
    if(this.responseXML != null &&
       this.responseXML.getElementById('test').firstChild.data)
   // success!
   test(this.responseXML.getElementById('test').firstChild.data);
  else
   test(null);
 } else if (this.readyState == 4 && this.status != 200) {
  // fetched the wrong page or network error...
  test(null);
 }
}
var client = new XMLHttpRequest();
client.onreadystatechange = handler;
client.open("GET", "test.xml");
client.send();
```

Javascript Example 2 (W3C)

If you just want to ping the server with a message you could do something like:

```
function ping(message) {
    var client = new XMLHttpRequest();
    client.open("POST", "/ping");
    client.send(message);
}
```

We are not establishing a callback handler. We are not interested in a response.

Javascript Example 3 (W3C)

Or, if you want to check the status of a document on the server, you can make a head request.

```
function fetchStatus(address) {
    var client = new XMLHttpRequest();
    client.onreadystatechange = function() {
     if(this.readyState == 4)
        returnStatus(this.status);
    }
    client.open("HEAD", address);
    client.send();
}
```

State and State Change

The state of the object. The readyState attribute must be one of the following values:

0 Uninitialized The initial value.

1 Open The `open()` method has been successfully called.

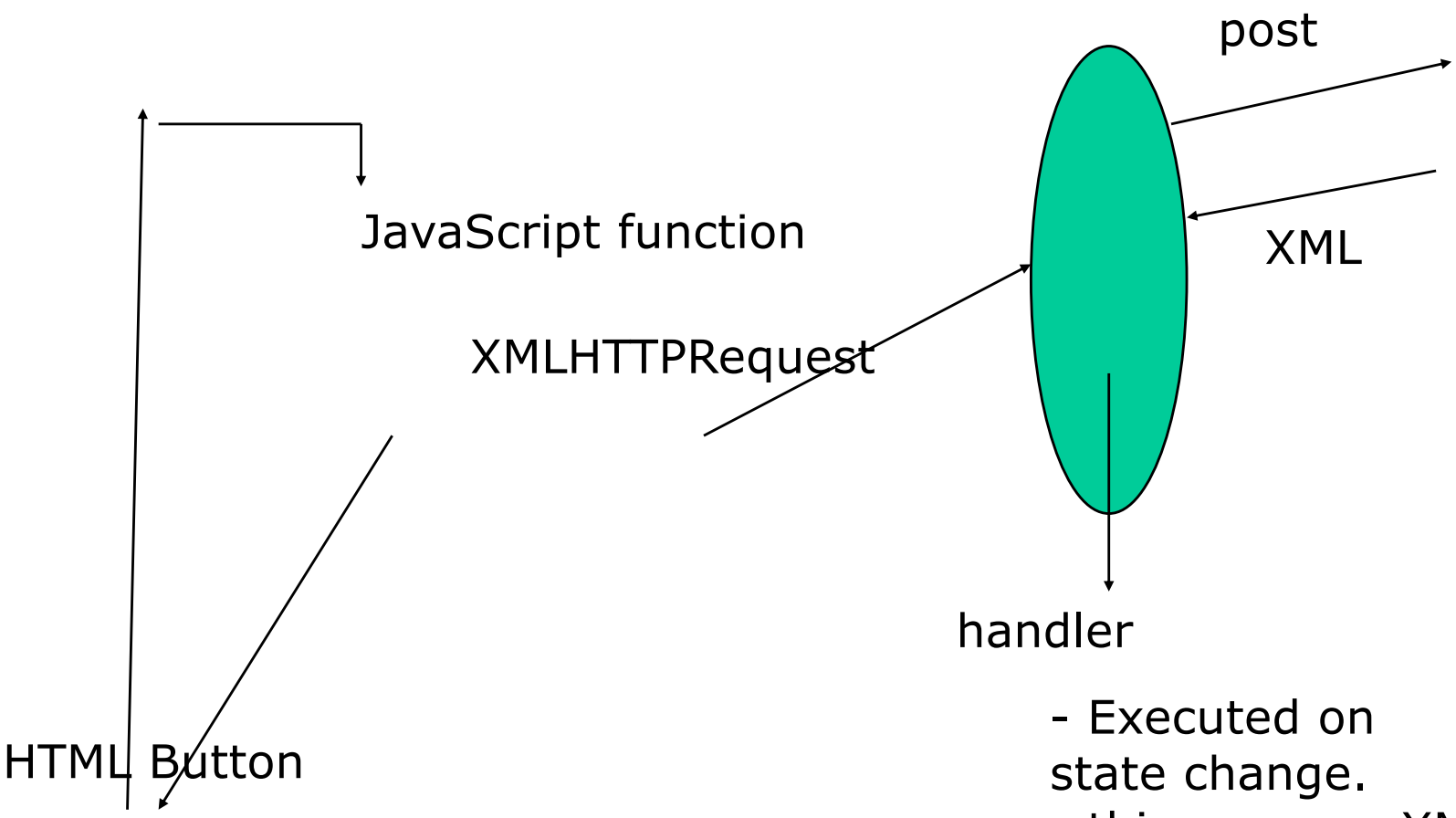
2 Sent The user agent successfully acknowledged the request.

3 Receiving Immediately before receiving the message body (if any). All HTTP headers have been received.

4 Loaded The data transfer has been completed.

When readyState ***changes value a*** readystatechange ***event is to be dispatched on the*** XMLHttpRequest ***object.***

AJAX Typical Interaction



- Executed on state change.
- this.responseXML points to the root of the XML
-write HTML to HTML DOM