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
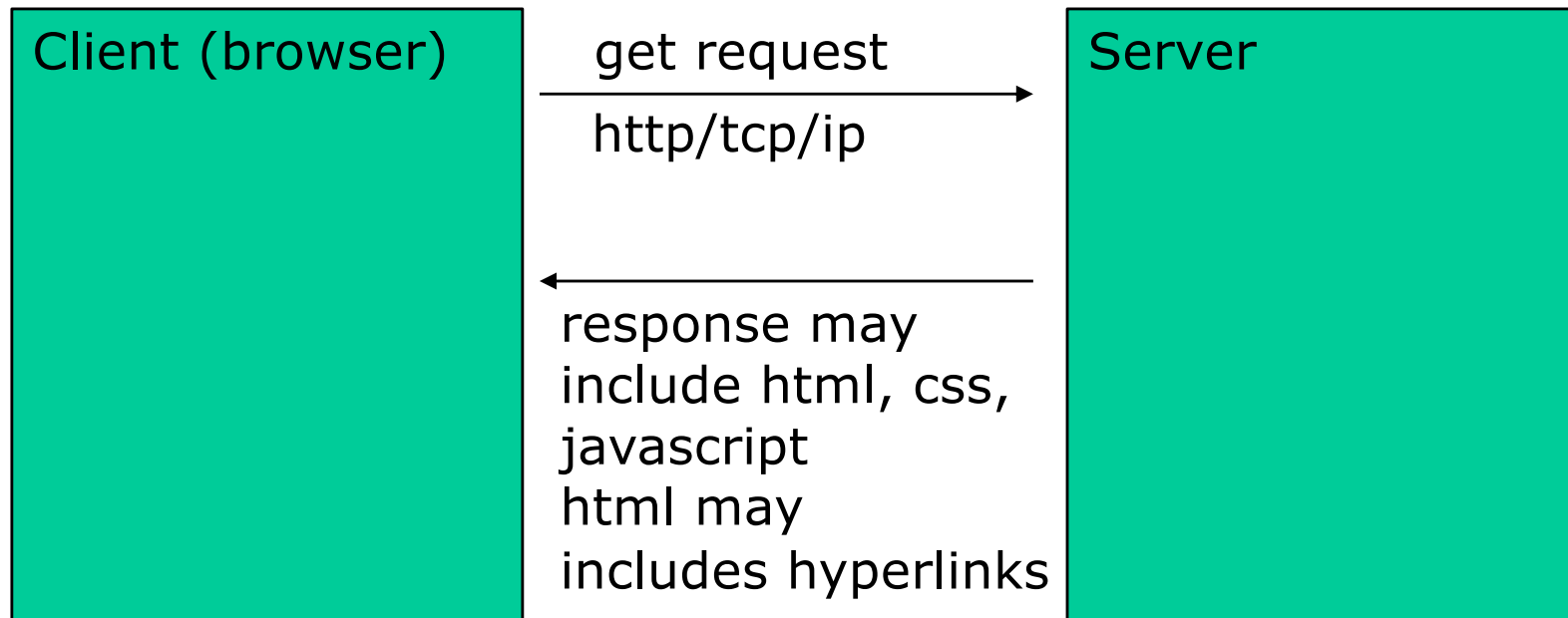
Coaxial cable

- Wireless

Optical – e.g. infrared, electromagnetic

Radio – most common (from NFC to satellite)

Terrestrial & Satellite Microwave

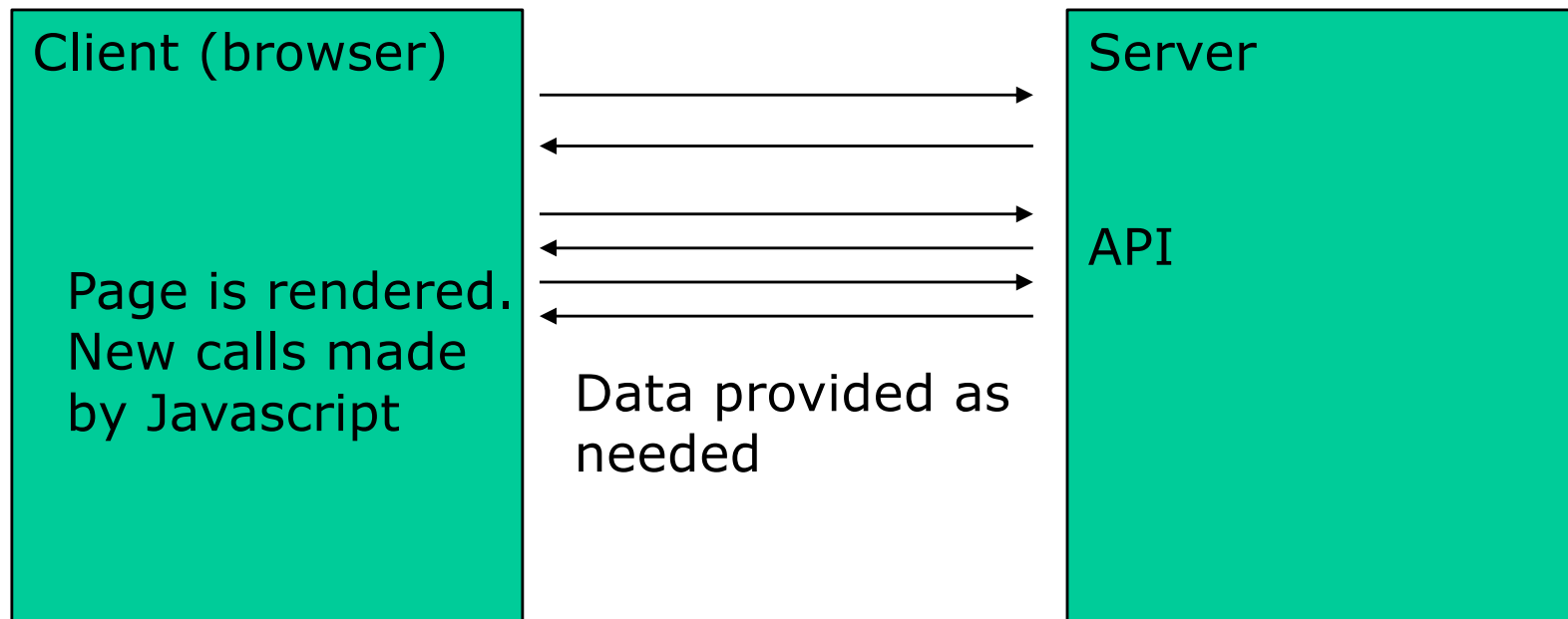Acoustic – e.g. underwater communications

Each of these is a major topic.

Four Styles of HTTP Interaction Style(1)

Client (browser)

get request

http/tcp/ip

response may include html, css, javascript html may includes hyperlinks

Server

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)

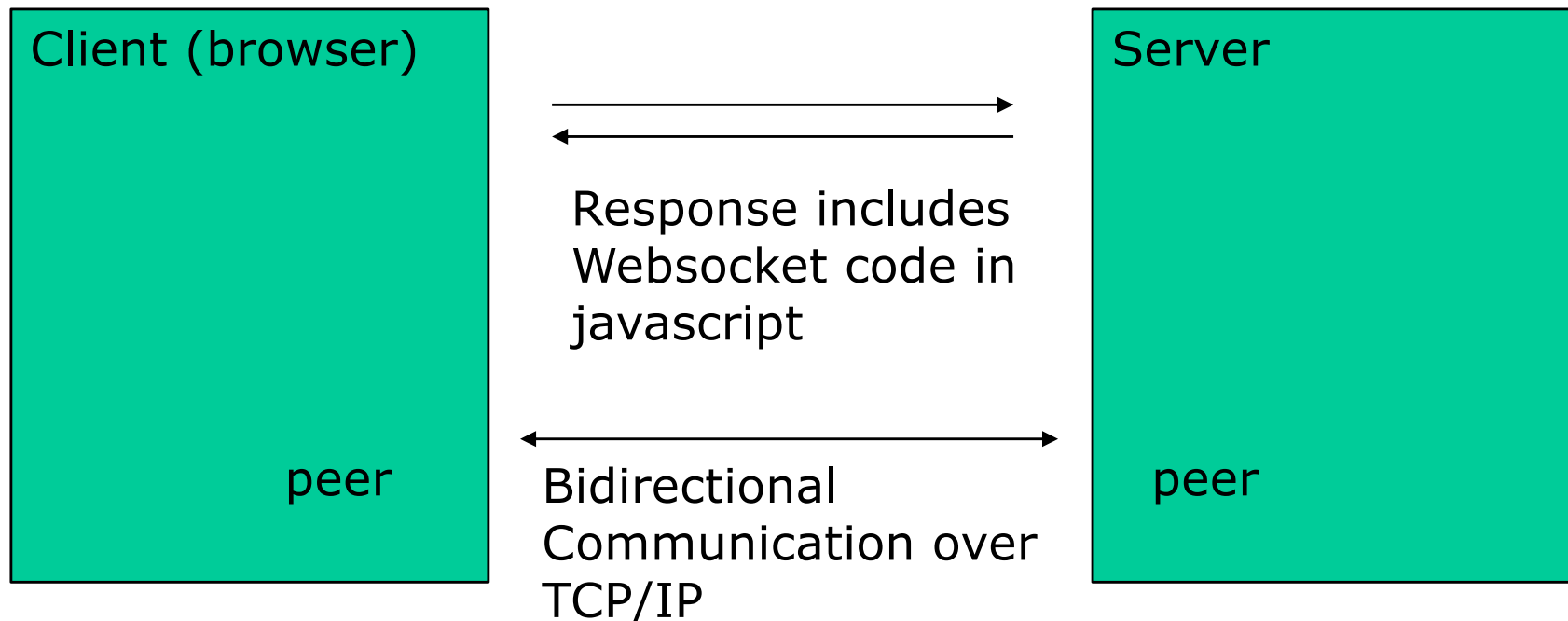| Client (browser)	Server
Page is rendered. New calls made by Javascript	API

Data provided as needed

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)

Client (browser)

Server

Response includes
code to cause additional
requests to others.
E.g., an <img> tag
is returned.
This causes
an HTTP Get to occur
elsewhere.

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. The first server says
the third party is OK.

Four Styles of HTTP Interaction Style(4)

Client (browser)		Server
	→	
	←	
	Response includes Websocket code in javascript	
peer	←→	peer
	Bidirectional Communication over TCP/IP	

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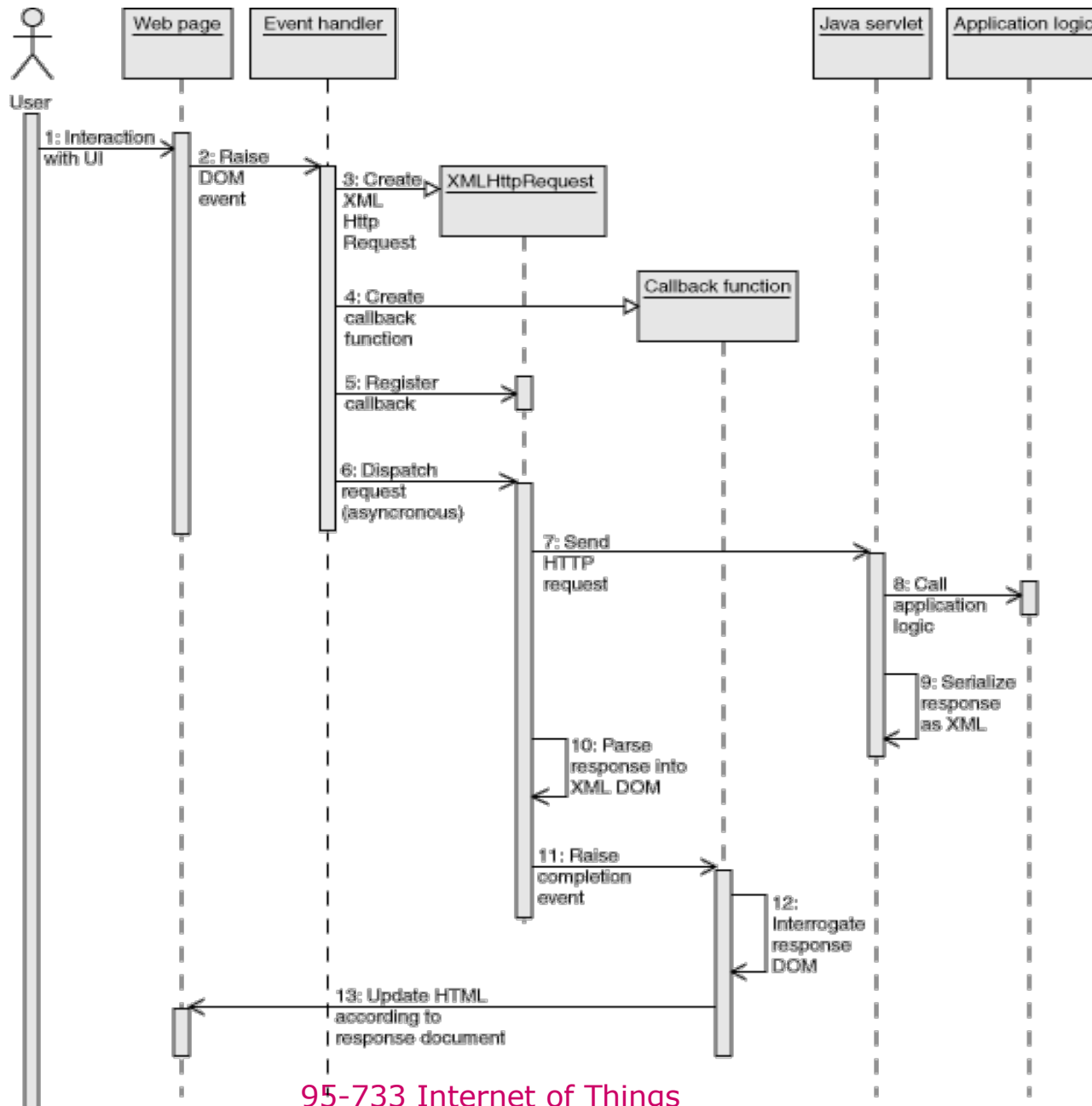   IoT - very fast binary or text

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

	Web page	Event handler		Java servlet	Application logic

User

1: Interaction with UI

2: Raise DOM event

3: Create XML Http Request → XMLHttpRequest

4: Create callback function → Callback function

5: Register callback

6: Dispatch request (asyncronous)

7: Send HTTP request

8: Call application logic

9: Serialize response as XML

10: Parse response into XML DOM

11: Raise completion event

12: Interrogate response DOM

13: Update HTML according to response document

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

post

JavaScript function

XML

XMLHTTPRequest

handler

HTML Button

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