CSS: Review (and a bit more) Protocols: HTTP Servers

CSS examples and images credited to Katie Seaborn

Cascading Style Sheets

- Separate structure from presentation
- “Simple” mechanism to attache style to structured documents
- Removes requirement for further formatting tags from HTML proper

A CSS file consists of one or more rules. Each rule starts with a selector, which specifies an HTML element(s) and then applies style properties to them.

```
selector {
    property: value;
    property: value;
}
p {
    font-family: sans-serif;
    color: red;
}
```

A style can select multiple elements separated by commas. Individual elements can still have their own style.

```
p, h1, h2 {
 color: green;
}

h2 {
 background-color: yellow;
}
```

An “**id**” provides a unique identifier for an element on a page; must used once (your call). A “**class**” provides a general way of accessing certain elements.

```
p.intro { font-family: Arial, sans-serif; }
p#mission { font-family: Times, serif; }
```

```
<p class=“intro”>Coding Horror! Coding Horror!</p>
<p id=“mission”>Our mission is to combine programming
and <q>human</q> factors with geekiness!</p>
```

As you saw in the lab, anchors have **states** that can be styled using **pseudo-classes** as selectors, e.g. :hover

```
a:link { color: #FF0000; } /*unvisited link */
a:visited { color: #00FF00; } /*visited link */
a:hover { color: #FF00FF; } /*mouse over link */
a:active { color: #FF00FF; } /* active link */
a:focus { color: #FF00FF; } /* focused link */
```

Where does it go?

- CSS rules can go in several different places:
- Internal styles:
    - inline

```
<p style="color: blue; font-size: 1.5em;">
```

- embedded

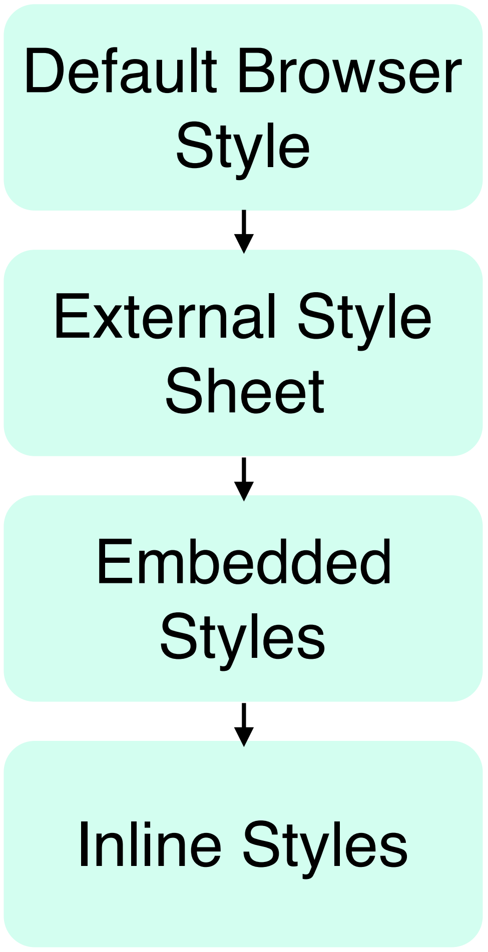
```
<head> <style type="text/css">
h1,h2,h3{
 color: green;
 font-weight: bold;
}
</style>
</head>
```

External Style Sheets

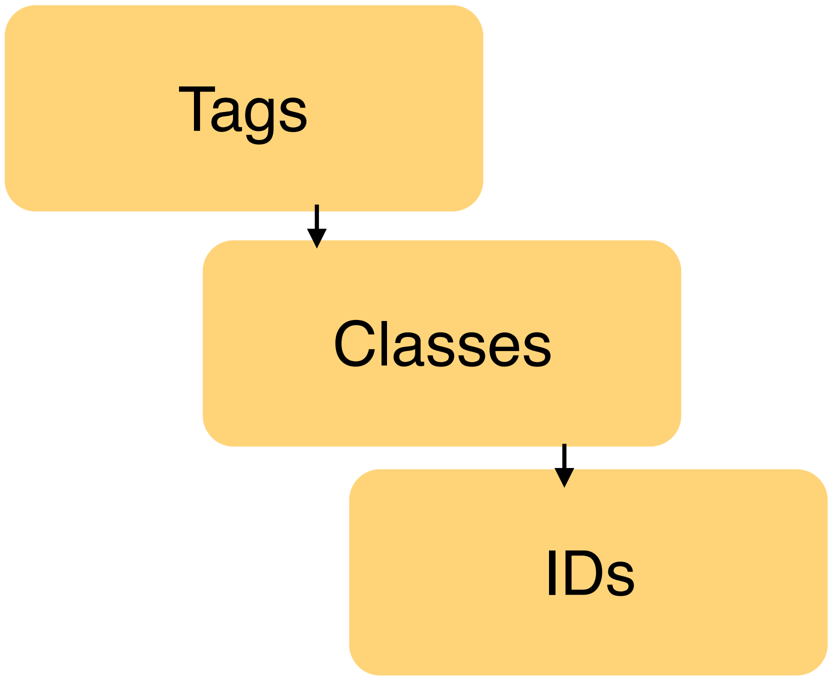
```
<head>

<link href=”styles.css” rel=“stylesheet” />

</head>
```

styles.css just contains the rules

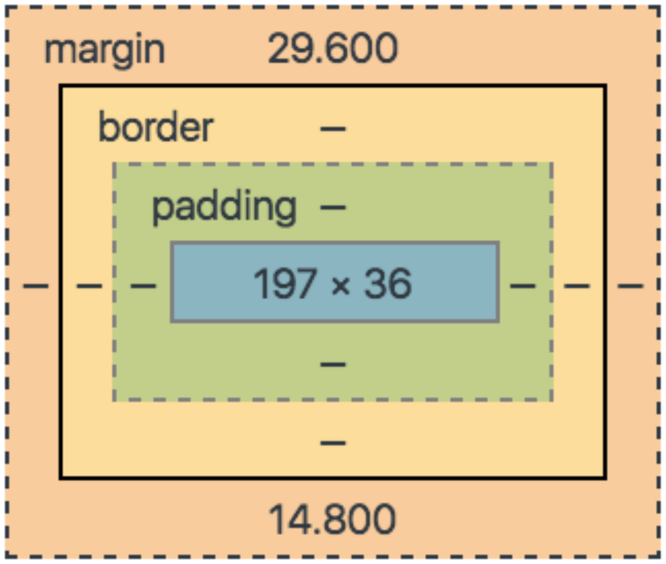
The Cascade



Rules applied from least specific to most specific.



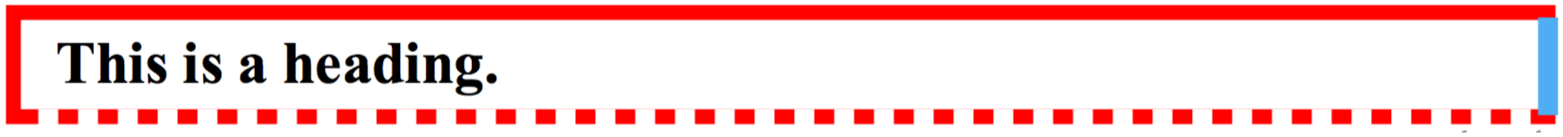
Layout



- CSS Box model
  - Each element: content, padding, border margin
  - width = content width + L/R padding + L/R border + L/R margin
    height = content height + T/B padding + T/B border + T/B margin

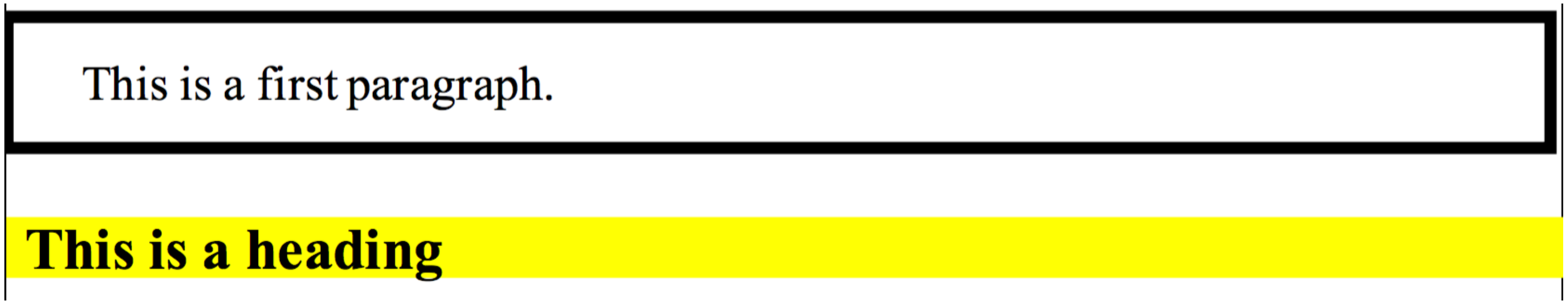
Borders help visually separate content:

```
h2 {
  border: 5px solid red;
  border-right: 8px solid blue;
  border-bottom: 5px dashed red;
}
```


This is a heading.

Padding to let element breathe

```
p { padding: 20px; border: 3px solid black; } h2
{ padding: 0px; background-color: yellow; }
```


This is a first paragraph.

This is a heading

Width: percentage or pixels

Can apply max-width or min-width

(especially for responsive reasons)

```css
img { width: 100%; max-width: 200px; }
```

CSS

100% here is 120px

output

max 200px even though container is wider

output

Float removes an element from the normal document flow; text wraps around.

```
img.headericon {
        float: right; width: 130px;
}                                                   CSS
```

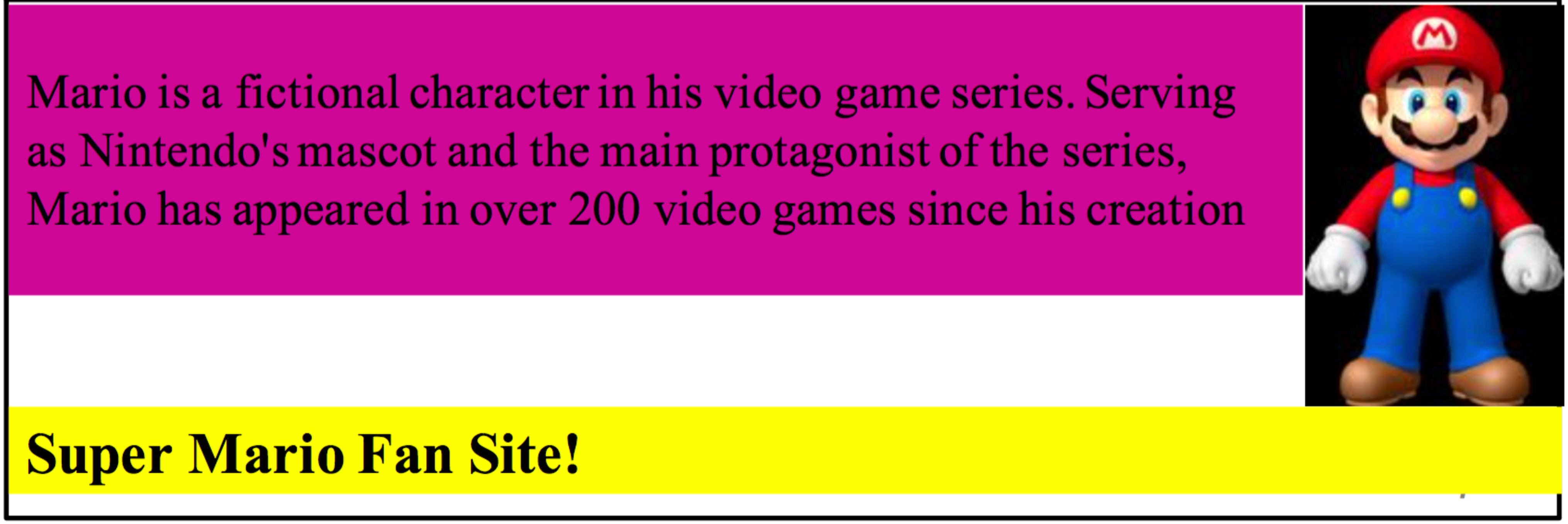
Ghostbusters is a 1984 American science fiction comedy film written by co-stars Dan Aykroyd and Harold Ramis about three eccentric New York City parapsychologists-turned-ghost capturers.

output

Need to clear elements that follow a floating element to prevent wrapping/overlapping.

```css
p { background-color: fuchsia; }
h2 { clear: right; background-color: yellow; }
```

CSS



Mario is a fictional character in his video game series. Serving as Nintendo's mascot and the main protagonist of the series, Mario has appeared in over 200 video games since his creation

Super Mario Fan Site!

HTTP

Protocols

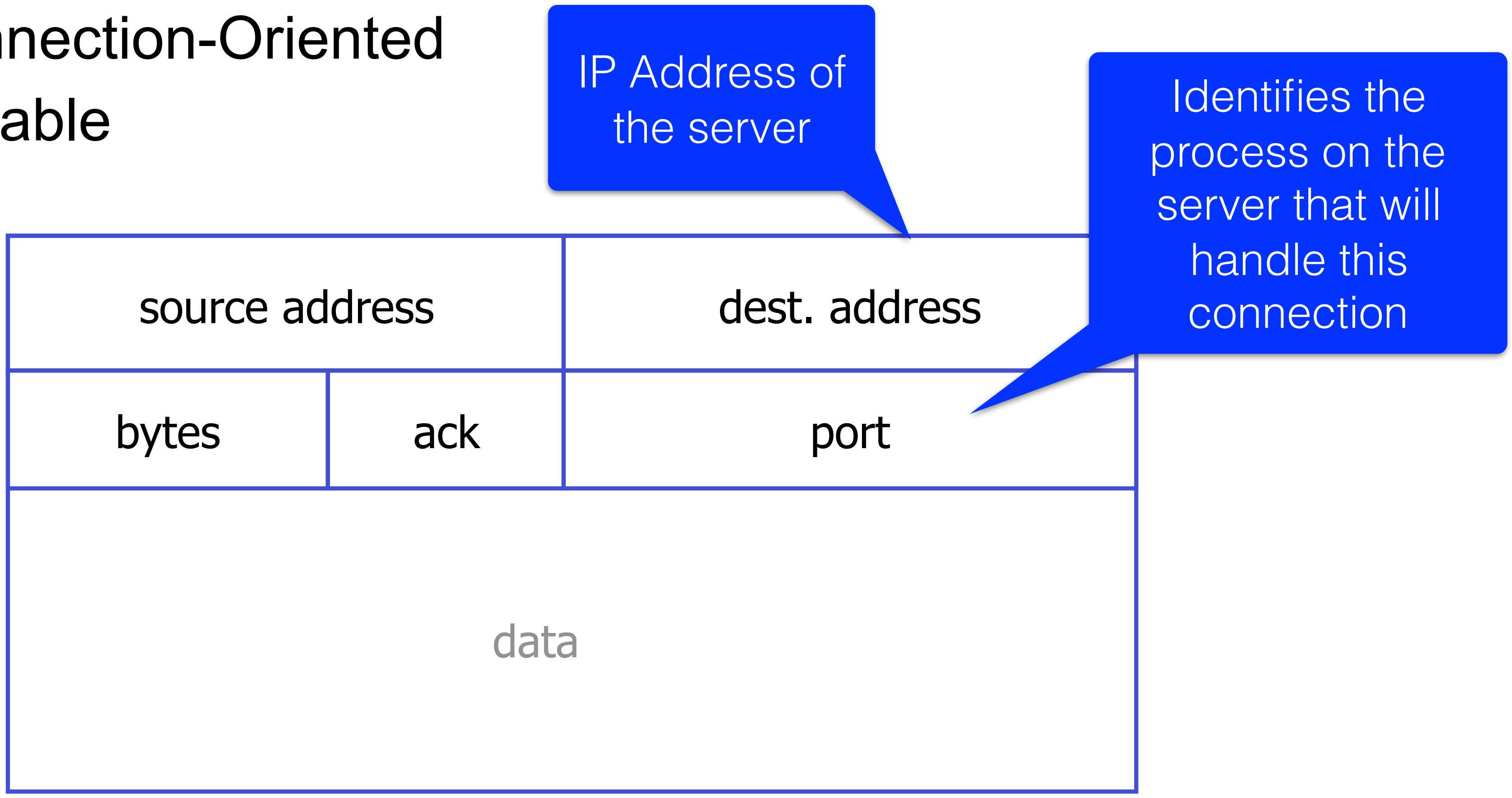
Application Layer FTP, HTTP, SSH, IMAP
Transport Layer TCP, UDP
Internet Layer IP
Link Layer Ethernet, WiFi

IP

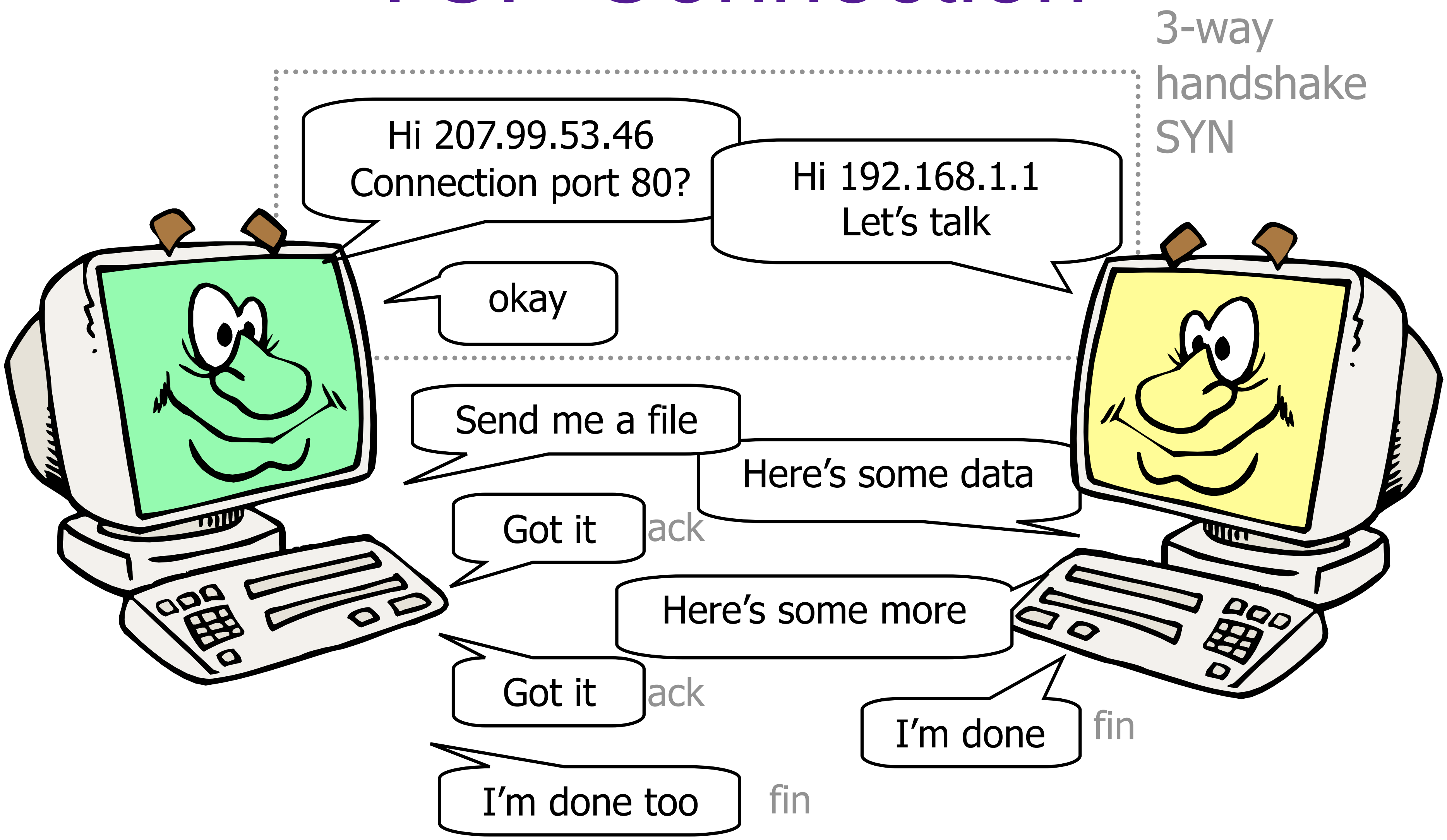
- Connectionless protocol
- Transfer packets from source address to destination
- IPv4 address: 128.100.31.200
- IPv6 address: fe80::225:90ff:fe5a:2a02/64

TCP/IP

- Transmission Control Protocol.
  - Connection-Oriented
  - Reliable


IP Address of the server
Identifies the process on the server that will handle this connection
source address
dest. address
bytes
ack
port
data

TCP Connection



Application Layer Protocols

Application: communicating distributed processes

- Running on network hosts in user space
- Exchange messages to implement app
- E.g., email, file transfer, bit torrent, web

Application-layer protocols:

- One piece of an app
- Defines messages exchanged by apps
- Uses services provided by lower layer protocols

Application Layer Protocols

API: application programming interface

- Defines interface between application and transport layer

socket: Internet API

- send, receive

HTTP

- Sits on top of TCP - data payload
- Goal: transfer objects between client (browser) and server (web application)
- Separate from other Web concepts:
    - HTML: page layout
    - URLs : object naming

http in operation

Suppose user enters: http://www.tkf.toronto.on.ca

http client initiates TCP connection to http server at www.tkf.toronto.on.ca on port 80

http server at host www.tkf.toronto.on.ca accepts the connection notifying client

http client sends http request message into TCP socket

http server receives request message, forms response message and sends it into socket

HTTP is stateless

- Server does not maintain status information across client requests
- No way to correlate multiple request from some user
- Protocols that maintain “state” are complex
- past history must be maintained
- if server or client crashes, their views of “state” may be inconsistent and must be reconciled.

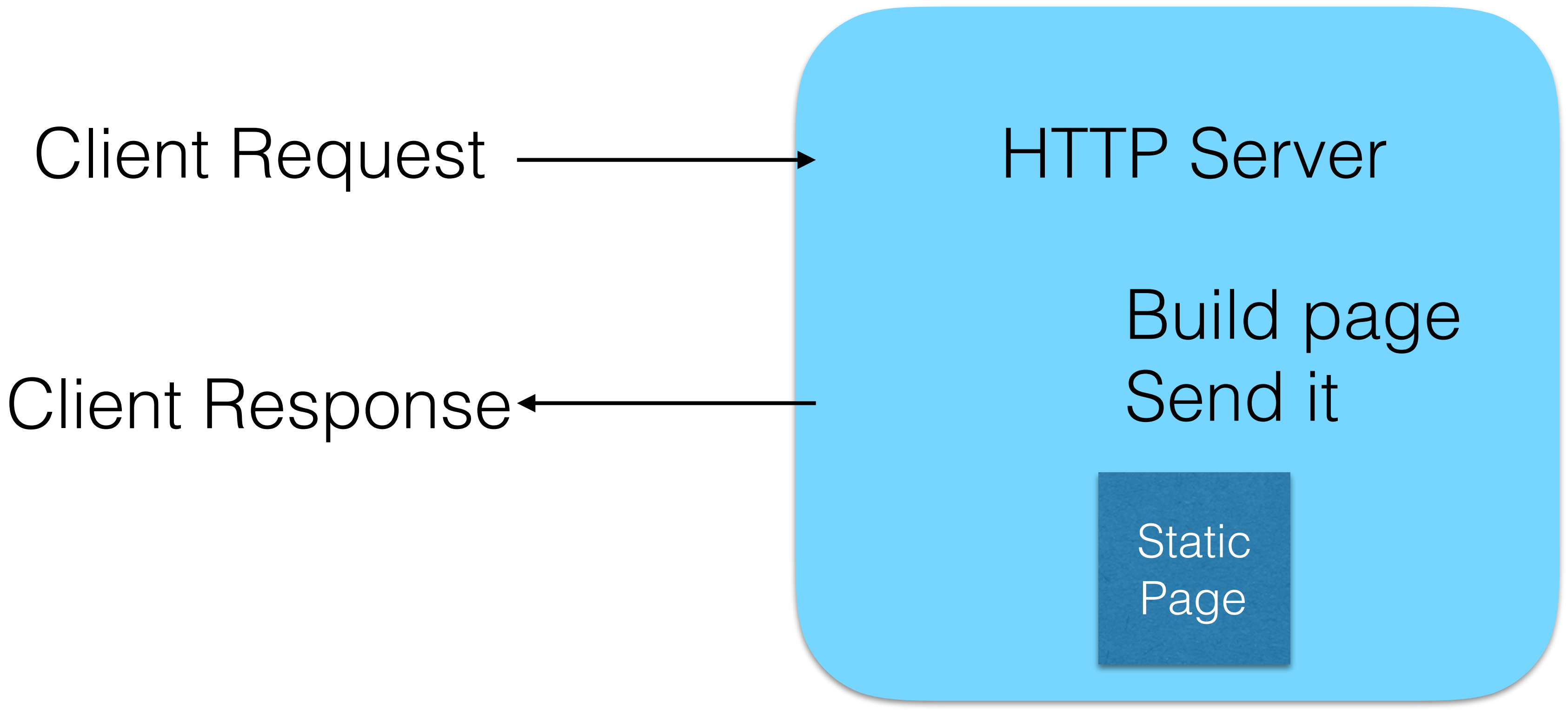
GET

```
GET /~cs209hf/cgi-bin/remark-submit.cgi?
course=csc209h&first_name=Karen&last_name=Re
id&cdf_account=reid&student_number=111222333
&email_address=reid
%40cdf.toronto.edu&assignment=a1a&request=Th
e+TA+ought+to+be+shot+for+doing+such+a
+terrible+job. HTTP/1.1
User-Agent: curl/7.18.2 (i486-pc-linux-gnu)
libcurl/7.18.2 OpenSSL/0.9.8g zlib/1.2.3.3
libidn/1.8 libssh2/0.18
Host: wwwcgi.cdf.toronto.edu
Accept: */*
```

POST

```
POST /~cs209hw/cgi-bin/processa1a.cgi HTTP/1.1
User-Agent: curl/7.18.2 (i486-pc-linux-gnu)
libcurl/7.18.2 OpenSSL/0.9.8g zlib/1.2.3.3
libidn/1.8 libssh2/0.18
Host: wwwcgi.cdf.toronto.edu
Accept: */*
Content-Length: 293
Content-Type: multipart/form-data;
boundary=----------------------------46916b928ffe

cdf_account=reid
data=1,1,412,Success
```

Simplest Server



Node examples

- Server 1: simplest possible hello world
- Server 2: serve an html file using synchronous read (illustrate problem with blocking operations)
- Server 3: attempt to deal with blocking operation.
- Server 4: use callback with expensive operation to illustrate that blocking isn't a problem.
- Server 5: the real html file version (still not perfect)