CSC 346 - Cloud Computing

03 - Networking, HTTP & HTML

HTTP

Getting What We Ask For

HTTP: How Browsers & Servers Communicate

HTTP 1.1 - http://www.w3.org/Protocols/

- TCP Connection, usually over port 80 or 443
- Text Based Instructions
- Simple Verbs
  - GET, POST, PUT, DELETE, HEAD, CONNECT, OPTIONS, TRACE
- Optional Headers

HTTP

Basic GET Example

- HOST header is required for HTTP/1.1
- Two CRLF to indicate the request has finished
  - CRLF = \r\n   Although most Web Servers will accept \n

```
GET / HTTP/1.1
Host: www.example.com

```

```
"Although the line terminator for the start-line and header fields is
   the sequence CRLF, a recipient MAY recognize a single LF as a line
   terminator and ignore any preceding CR."
```

http://tools.ietf.org/html/rfc7230#section-3.5

- Verbs and HTTP versions **are** Case Sensitive

```
get / HTTP/1.1
host: example.com

HTTP/1.1 501 Not Implemented
```

```
get / http/1.1
host: example.com

HTTP/1.0 505 HTTP Version Not Supported
```

- Headers **are not** Case Sensitive

```
GET / HTTP/1.1
hoSt: exAMPle.cOm

HTTP/1.1 200 OK
```

Basic HTTP Example

Request

```
GET / HTTP/1.1
host: example.com
```

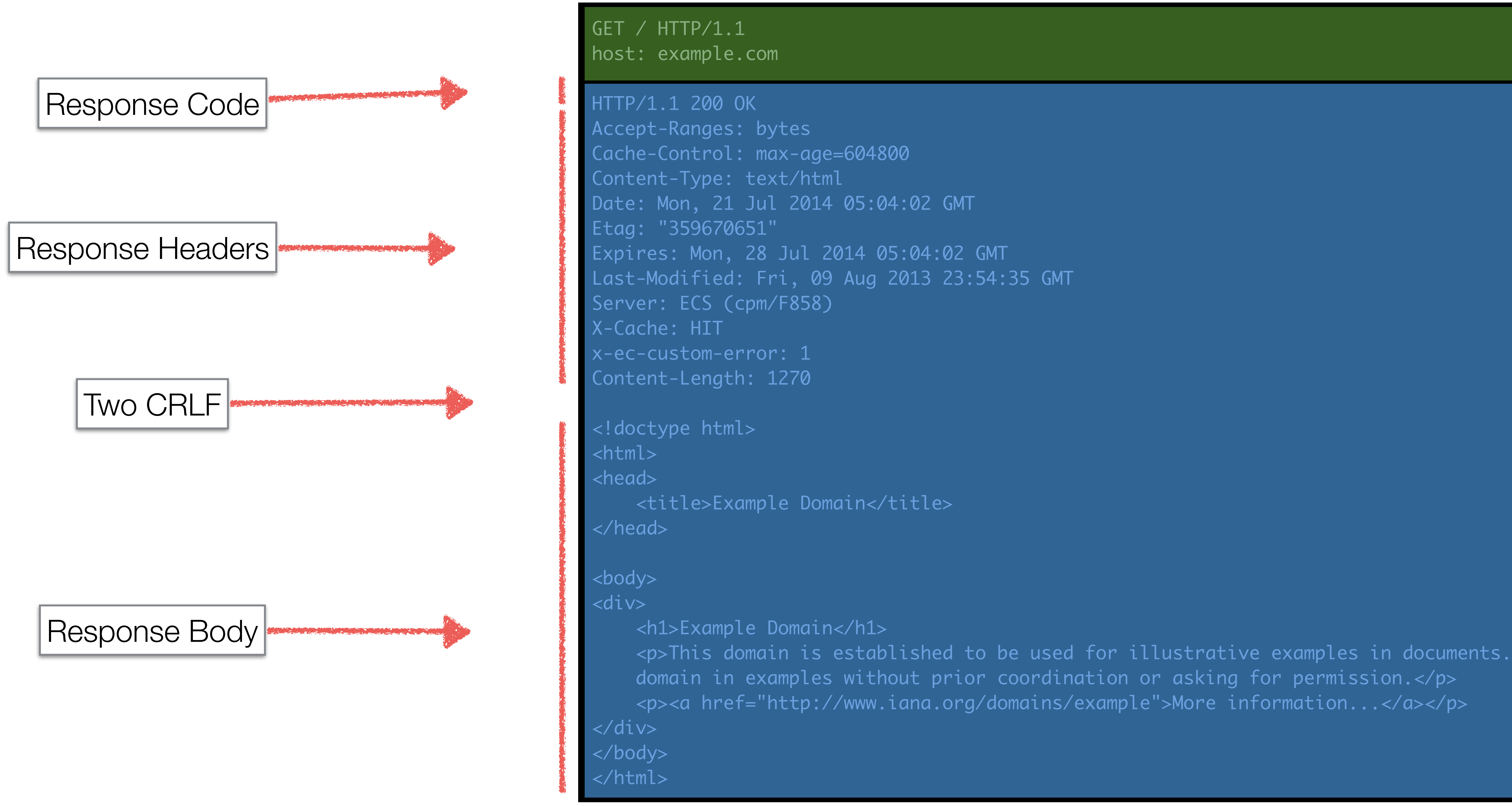
Response

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Mon, 21 Jul 2014 05:04:02 GMT
Etag: "359670651"
Expires: Mon, 28 Jul 2014 05:04:02 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (cpm/F858)
X-Cache: HIT
x-ec-custom-error: 1
Content-Length: 1270

<!doctype html>
<html>
<head>
    <title>Example Domain</title>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is established to be used for illustrative examples in documents. You may use this
    domain in examples without prior coordination or asking for permission.</p>
    <p><a href="http://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

Basic HTTP Example



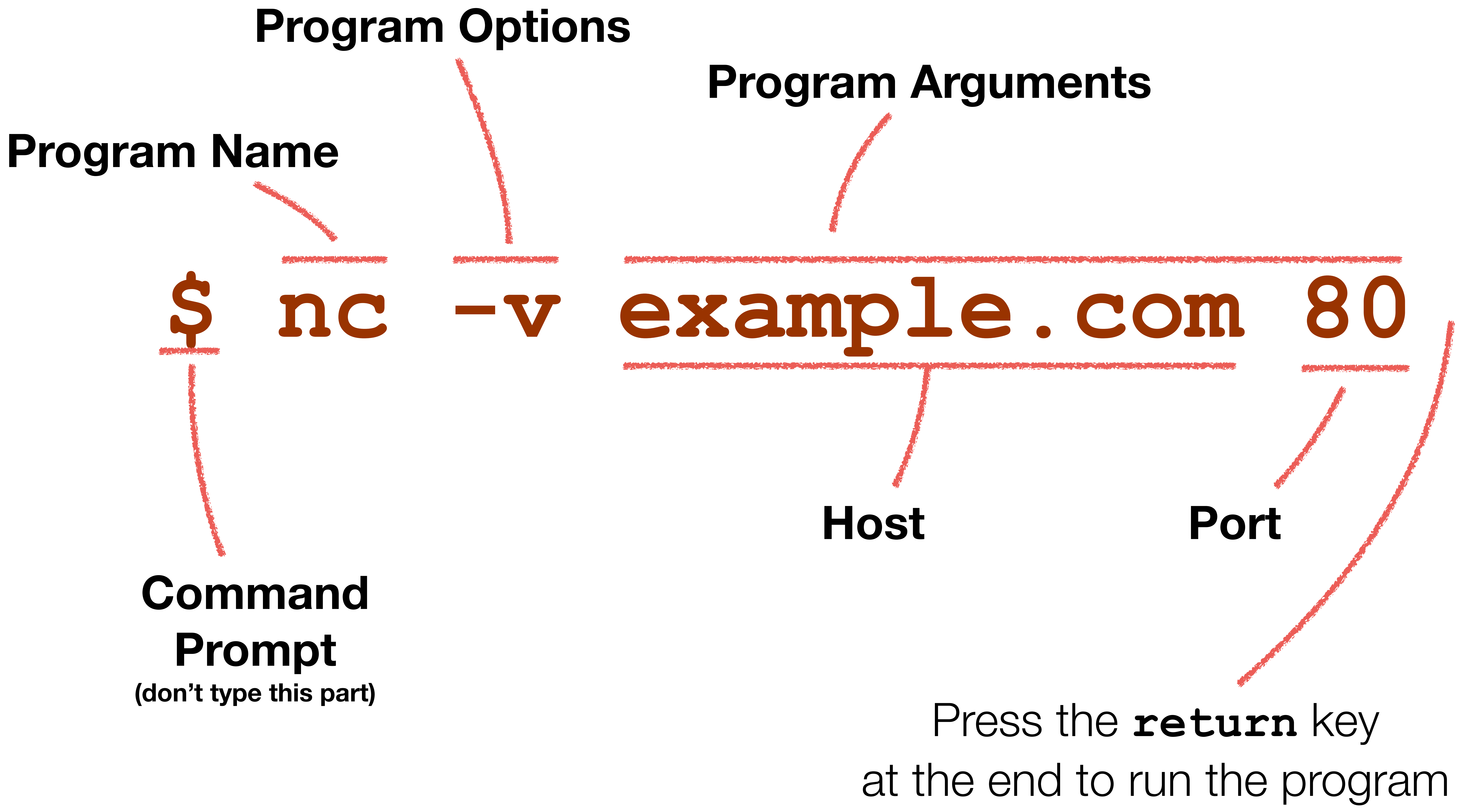
```
GET / HTTP/1.1
host: example.com
```

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Mon, 21 Jul 2014 05:04:02 GMT
Etag: "359670651"
Expires: Mon, 28 Jul 2014 05:04:02 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (cpm/F858)
X-Cache: HIT
x-ec-custom-error: 1
Content-Length: 1270

<!doctype html>
<html>
<head>
    <title>Example Domain</title>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is established to be used for illustrative examples in documents.
    domain in examples without prior coordination or asking for permission.</p>
    <p><a href="http://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

Command Line Basics

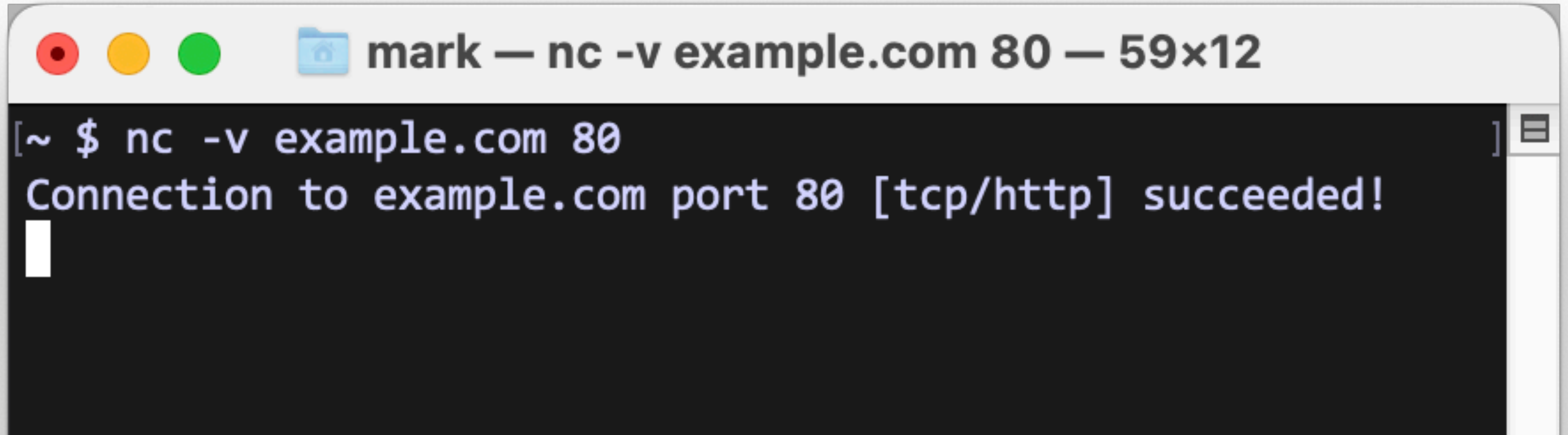


```
$ nc -v example.com 80
```

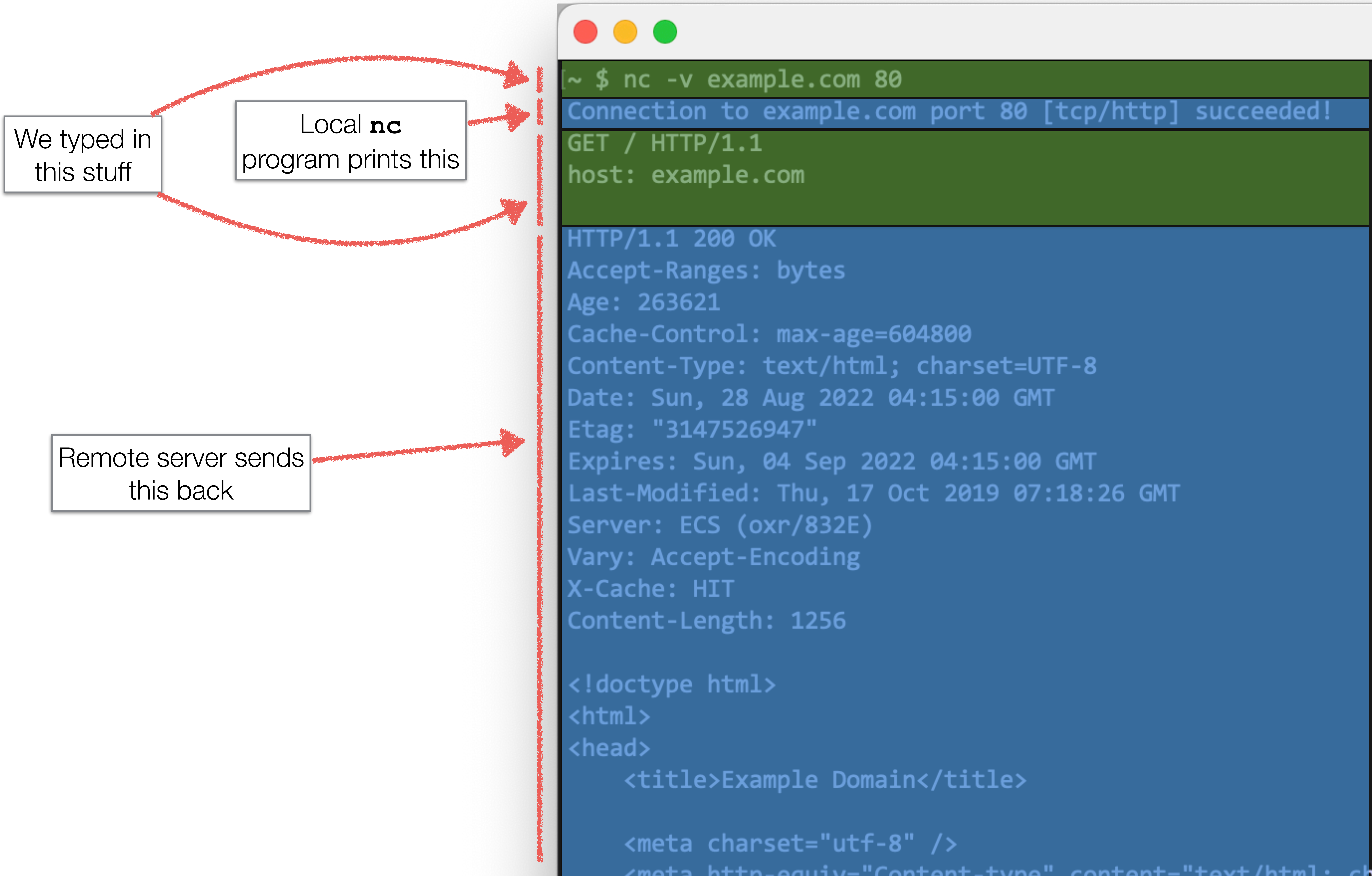
Press the **`return`** key
at the end to run the program

HTTP With NetCat - nc

- We used to do this with **`telnet`** but most environments no longer have this available by default
- Use nc (netcat) now instead
  - Opens a raw TCP socket connection to the target
- Key parts: **`host`** and **`port`**



```
~ $ nc -v example.com 80
Connection to example.com port 80 [tcp/http] succeeded!
```


We typed in this stuff
Local **nc** program prints this
Remote server sends this back

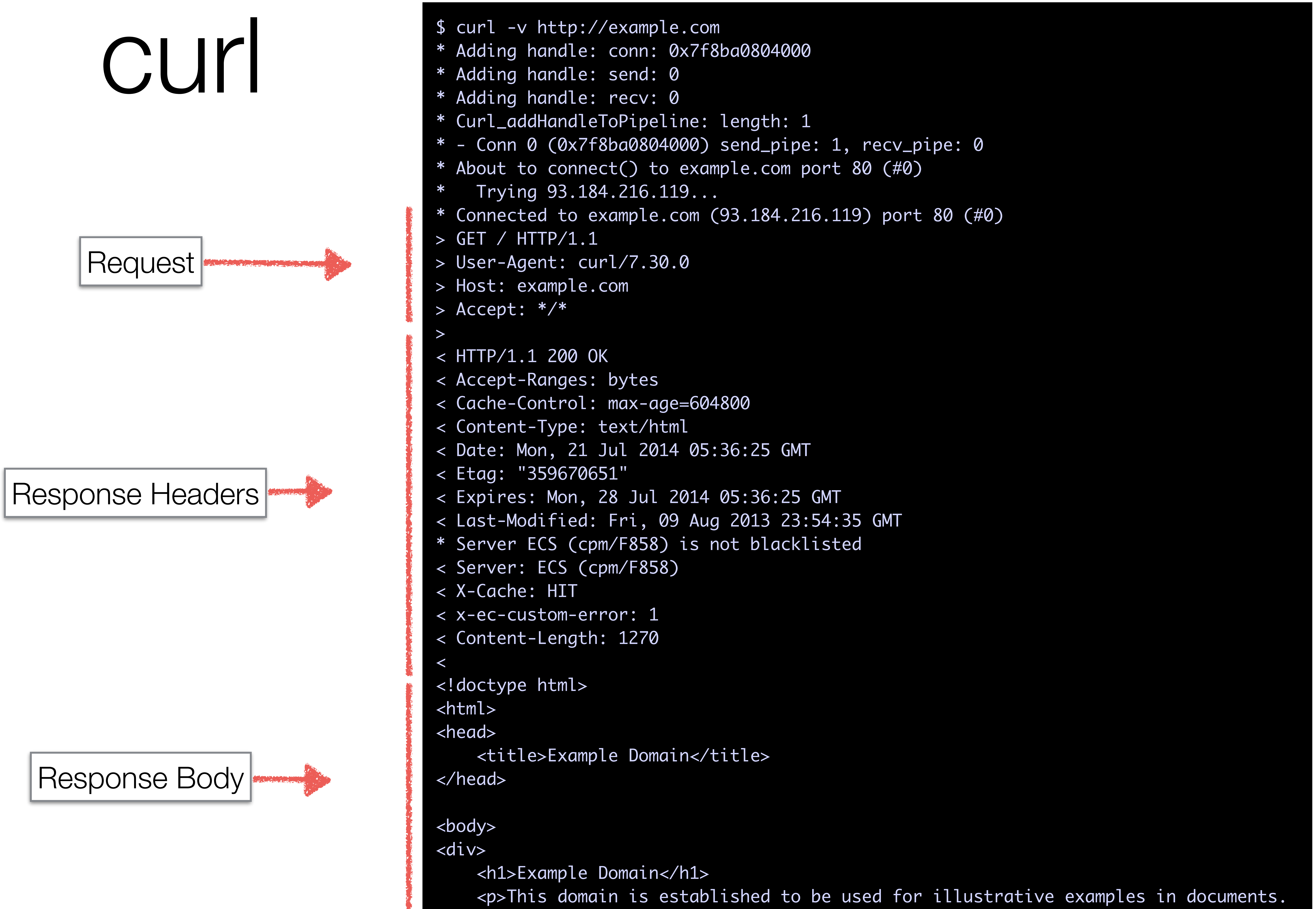
```
~ $ nc -v example.com 80
Connection to example.com port 80 [tcp/http] succeeded!
GET / HTTP/1.1
host: example.com

HTTP/1.1 200 OK
Accept-Ranges: bytes
Age: 263621
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Sun, 28 Aug 2022 04:15:00 GMT
Etag: "3147526947"
Expires: Sun, 04 Sep 2022 04:15:00 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (oxr/832E)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1256

<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; c
```

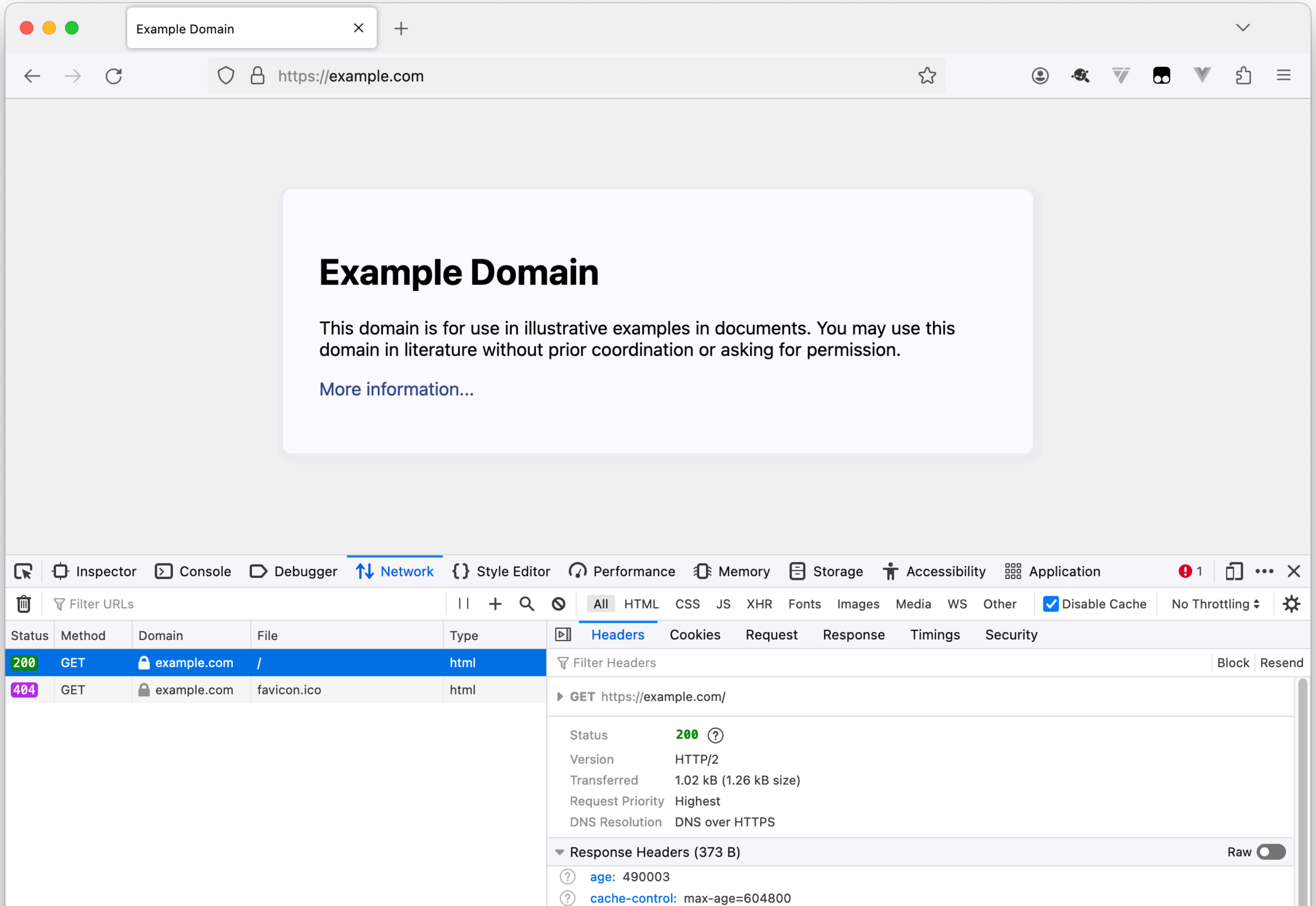
curl



```
$ curl -v http://example.com
* Adding handle: conn: 0x7f8ba0804000
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x7f8ba0804000) send_pipe: 1, recv_pipe: 0
* About to connect() to example.com port 80 (#0)
*   Trying 93.184.216.119...
* Connected to example.com (93.184.216.119) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.30.0
> Host: example.com
> Accept: */*
>
< HTTP/1.1 200 OK
< Accept-Ranges: bytes
< Cache-Control: max-age=604800
< Content-Type: text/html
< Date: Mon, 21 Jul 2014 05:36:25 GMT
< Etag: "359670651"
< Expires: Mon, 28 Jul 2014 05:36:25 GMT
< Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
* Server ECS (cpm/F858) is not blacklisted
< Server: ECS (cpm/F858)
< X-Cache: HIT
< x-ec-custom-error: 1
< Content-Length: 1270
<
<!doctype html>
<html>
<head>
    <title>Example Domain</title>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is established to be used for illustrative examples in documents.
```

Examine Requests in Firefox



Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

Status	Method	Domain	File	Type
200	GET	example.com	/	html
404	GET	example.com	favicon.ico	html

GET https://example.com/

Status	200
Version	HTTP/2
Transferred	1.02 kB (1.26 kB size)
Request Priority	Highest
DNS Resolution	DNS over HTTPS

Response Headers (373 B)

age: 490003
cache-control: max-age=604800

Response Codes

`1XX`	Informational
`2XX`	Successful
`200`	OK
`3XX`	Redirection
`301`	Moved
`4XX`	Client Error
`404`	Not Found
`5XX`	Server Error
`500`	Internal Server Error

http://tools.ietf.org/html/rfc7231#page-4

HTTP/2.0

- New binary method of allowing multiple requests through a single TCP socket
- More of a change to how the protocol is implemented on the wire than in the concepts of how the protocol works
- Advanced topic, if you're interested in more details:
    - `http://http2-explained.readthedocs.org/en/latest/src/http2protocol.html`
- Otherwise, just know its a thing

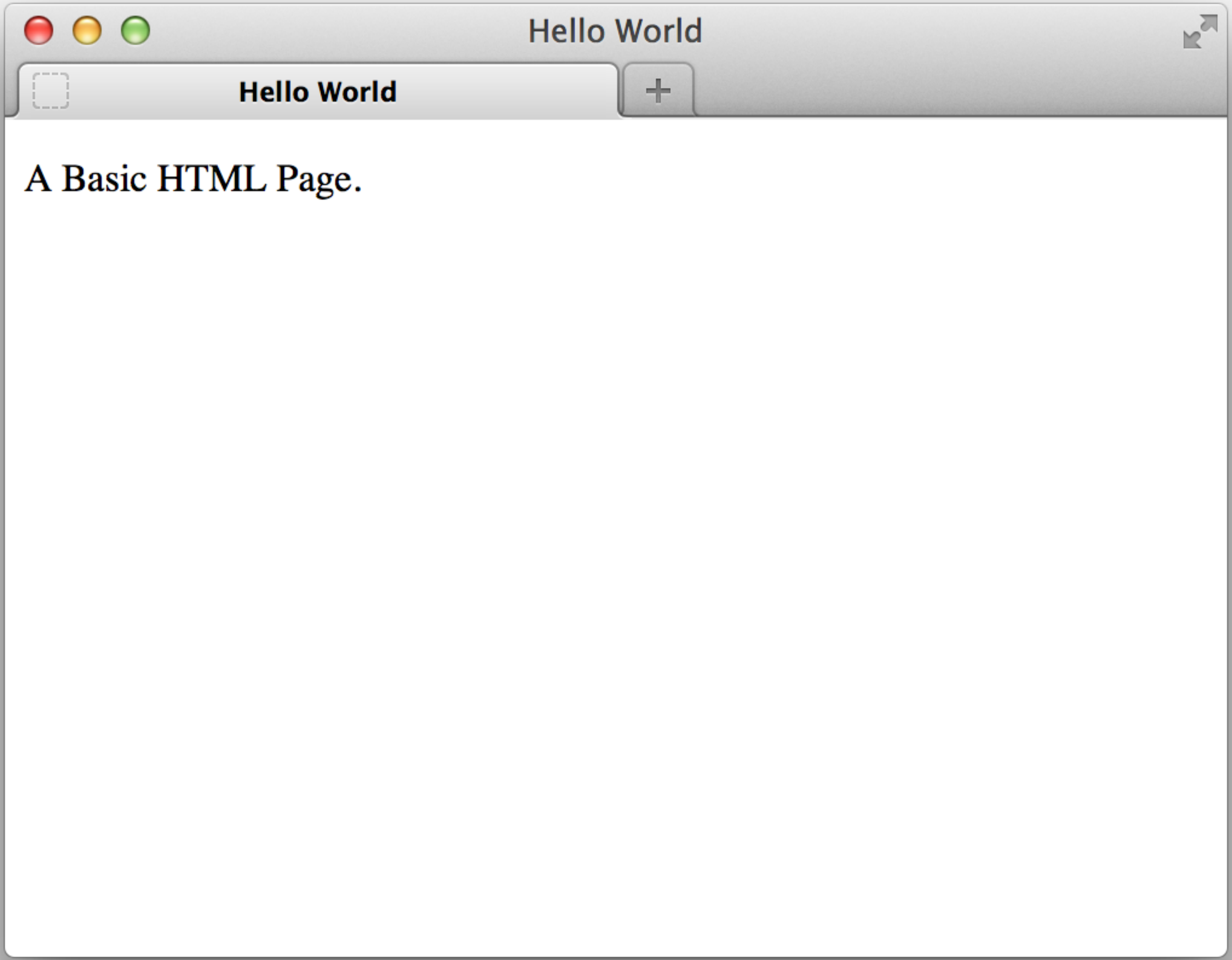
Some HTML

/some.html

```
<!doctype html>
<html>
<head>
  <title>Hello World</title>
</head>

<body>
  <p>A Basic HTML Page.</p>
</body>

</html>
```



A Basic HTML Page.

HTML Defines Content and Structure

- Content consists of Text, Images, Links, Media Assets, etc
- Structure defines the basic formatting and semantic meaning of elements
  - i.e. `<title>Hello World</title>` defines the title of the page
  - Programs can analyze the structure of a document to derive meaning
    - `h1`, `h2`, `h3` tags could be used to generate a document outline
    - Headers in a table (`<th>`) could be used by screen readers to describe data to a visually impaired individual
  - We can use the document structure to define display styles

Structure of an Element

```
<title>Hello World</title>
```

- The entire line is referred to as ***the*** `title` ***element***
- The ***name*** of this element is “title”
- `<title>` is an ***opening tag***
- `</title>` is a ***closing tag***
- `Hello World` is the ***content*** of this element

Not All Elements Need a Closing Tag

```
<body>
  <p>
   Paragraph elements can have closing tags
  </p>
  <p>or not
  <ul>
    <li>List Item elements
    <li>may also omit closing tags
  </ul>
</body>
```

http://www.w3.org/TR/html5/syntax.html#optional-tags

Not All Elements Have Content

- `<br>` the Break tag acts as a newline character for HTML
- `<hr>` the Horizontal Rule tag draws a line across a page
- `<img src="foo.gif">` the Image Tag tells the browser to go load an image in this location
- These elements are called ***void elements*** and *must not have* closing tags

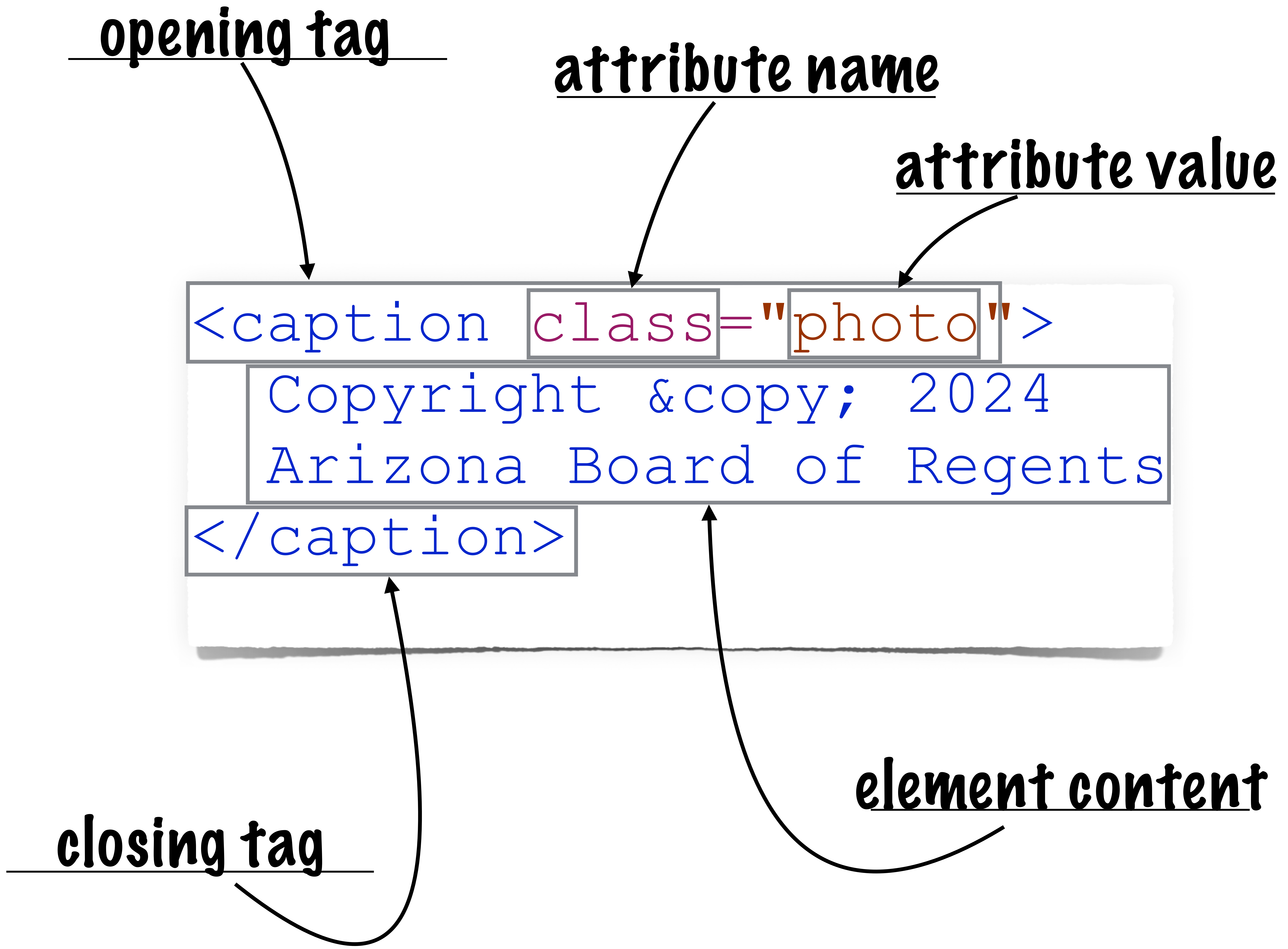
http://www.w3.org/TR/html5/syntax.html#void-elements

Attributes

```
<img src="foo.gif" class=thumbnail>
```

- Attributes for an element are defined in the element’s ***opening tag***
- Attributes always have an ***attribute name***
- Attributes may optionally have a ***value***
- Attribute values may be surrounded with either single quotes, double quotes, or nothing, depending on the content of the value

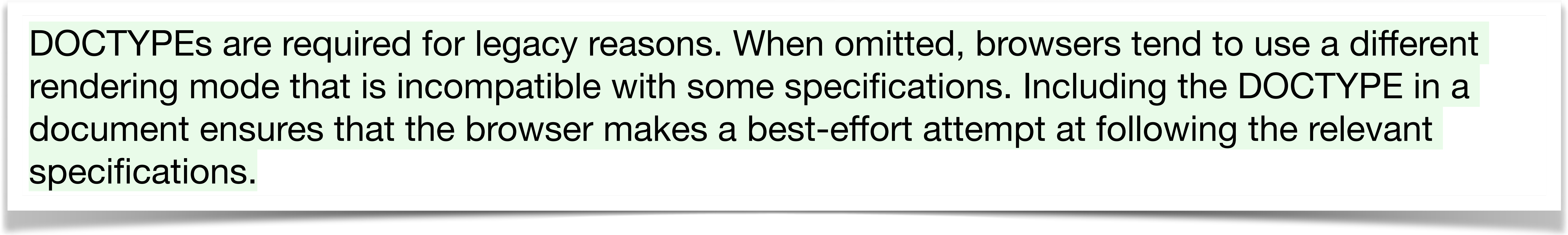
`http://www.w3.org/TR/html5/syntax.html#attributes-0`



```
<caption class="photo">
   Copyright &copy; 2024
   Arizona Board of Regents
</caption>
```

`<!doctype …>`

- The `<!doctype …>` preamble is *not* an HTML element.
- `<!doctype …>` tells the rendering engine what type of markup to expect
- HTML4.1 Transitional
  - `<!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
- HTML5
  - `<!doctype html>`



DOCTYPEs are required for legacy reasons. When omitted, browsers tend to use a different rendering mode that is incompatible with some specifications. Including the DOCTYPE in a document ensures that the browser makes a best-effort attempt at following the relevant specifications.

http://www.w3.org/TR/html5/syntax.html#the-doctype

`<html>`

- The `<html>` element is the root element of our element tree
- The HTML Element can only be preceded by whitespace characters and comments
- The HTML Element can only have two children: one `<head>` element and one `<body>` element
- From the HTML specification:
  - *An html element's start tag can be omitted if the first thing inside the html element is not a comment.*
  - *An html element's end tag can be omitted if the html element is not immediately followed by a comment.*

`<head>`

- The `<head>` element represents a collection of metadata for the Document.
- A `<title>` tag is the only required child element

```html
<head>
  <meta charset="utf-8">
  <base href="http://www.example.com/">
  <title>A New Hope</title>
  <link rel="stylesheet" href="default.css">
  <script src="example.js"></script>
</head>
```

`<body>`

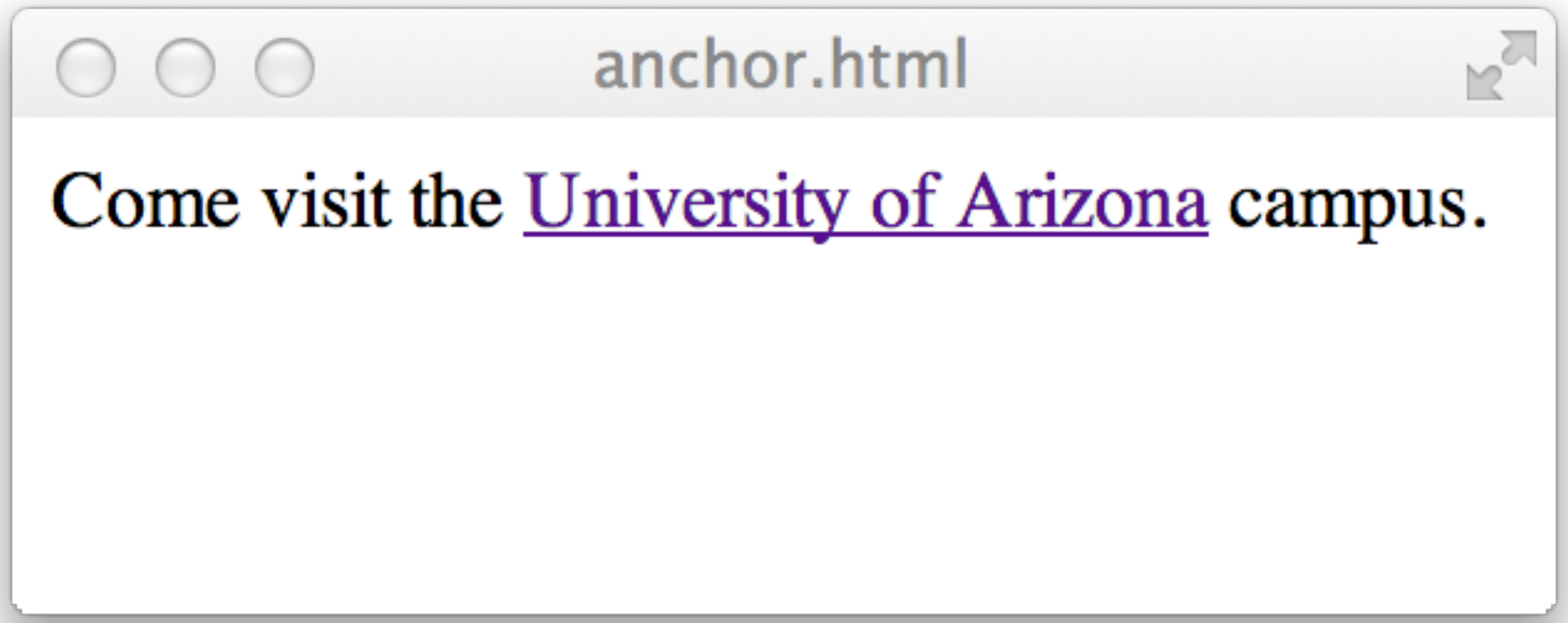
- The `<body>` element represents the content of the Document.
- Basically this holds everything you see.

```
<body>
  <img src="http://goo.gl/5LEYnp" alt="regrets">
  <form action="search.php" method="post">
    <input type="text" name="search">
    <input type="submit" value="Find Droids">
  </form>
</body>
```

Links

```
Come visit the <a href="http://www.arizona.edu">University of Arizona</a> campus.
```

- `<a>` Anchor tag
- Used to define a link to another document, or location in the same document.



Come visit the University of Arizona campus.

Links

```
<a href="http://www.arizona.edu">University of Arizona</a>
```

- `href` attribute defines what to link do.
    - This is the *Hyper* in HyperText
- Must contain a valid URL
    - Universal Resource Locator

URL

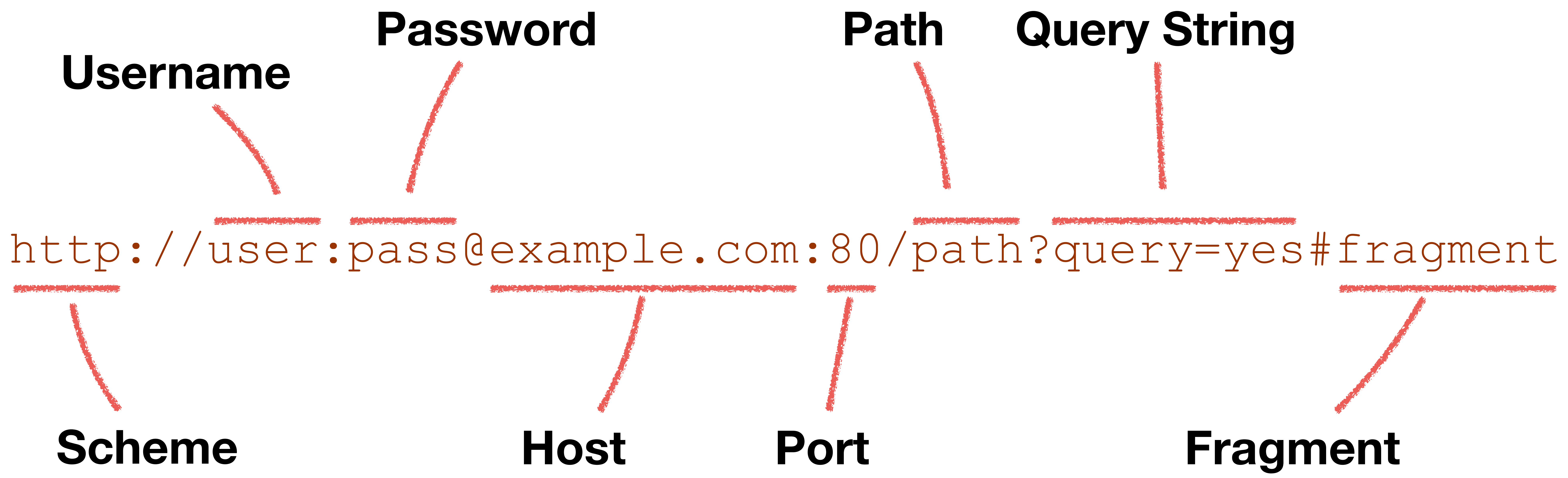
- A basic absolute URL

```
http://www.arizona.edu
```

- A basic relative URL

```
../images/image.png
```

URL


Username
Password
Path
Query String
http://user:pass@example.com:80/path?query=yes#fragment
Scheme
Host
Port
Fragment

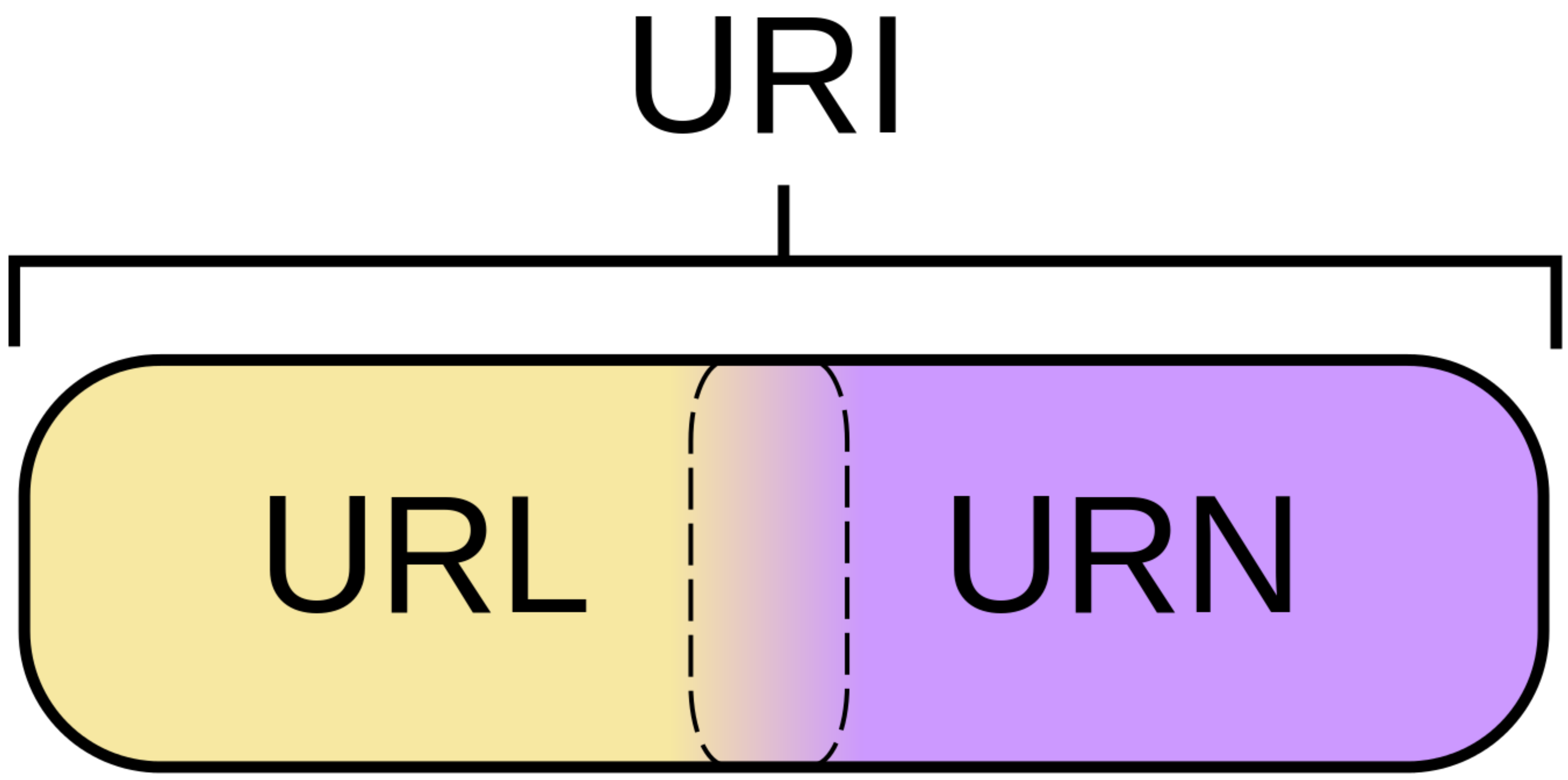
URL

- Most of these parts are null most of the time
- The following are all valid URLs

```
https://example.com
/path/to/something.html
mailto:fischerm@email.arizona.edu
foo
//ajax.googleapis.com/libs/jquery.min.js
../somepage.php?key=123
anotherpage.html#figure1
#droids
```

URI, URL, URN

- URI - Universal Resource Identifier
- URL - Universal Resource Locator
- URN - Universal Resource Name
- These are NOT interchangeable. Each has a different meaning, although there can be significant overlap
- We’re almost always going to use URLs unless otherwise explicitly mentioned



http://en.wikipedia.org/wiki/File:URI_Euler_Diagram_no_lone_URIs.svg

URI

```
The generic URI syntax consists of a hierarchical sequence of
components referred to as the scheme, authority, path, query, and
fragment.

   URI         = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

   hier-part   = "//" authority path-abempty
               / path-absolute
               / path-rootless
               / path-empty
```

http://tools.ietf.org/html/rfc3305

http://tools.ietf.org/html/std66

URL Schemes

`http://user:pass@example.c`

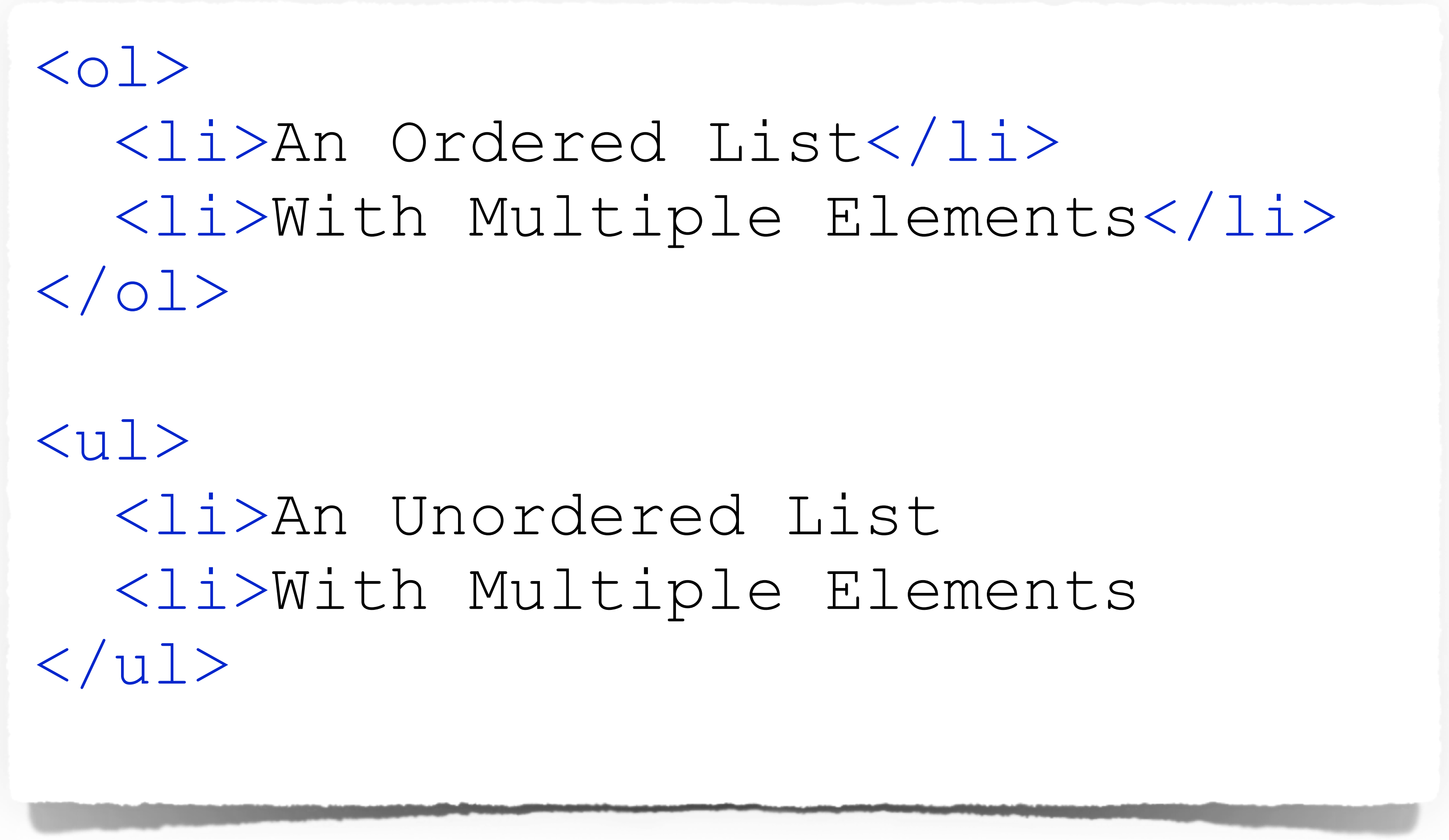
Scheme

- The Scheme tells the client how to access the resource.
- `file:///` loads the file directly from the local filesystem
- `http://` initiates an HTTP connection over TPC/IP
- `https://` establishes a secure connection over SSL, then communicates via HTTP
- `email:` hands off control to an email client
- `tel:` hands off control to a phone client
- `myapp:` Mobile platforms let you register a URL Scheme for your app

More Elements

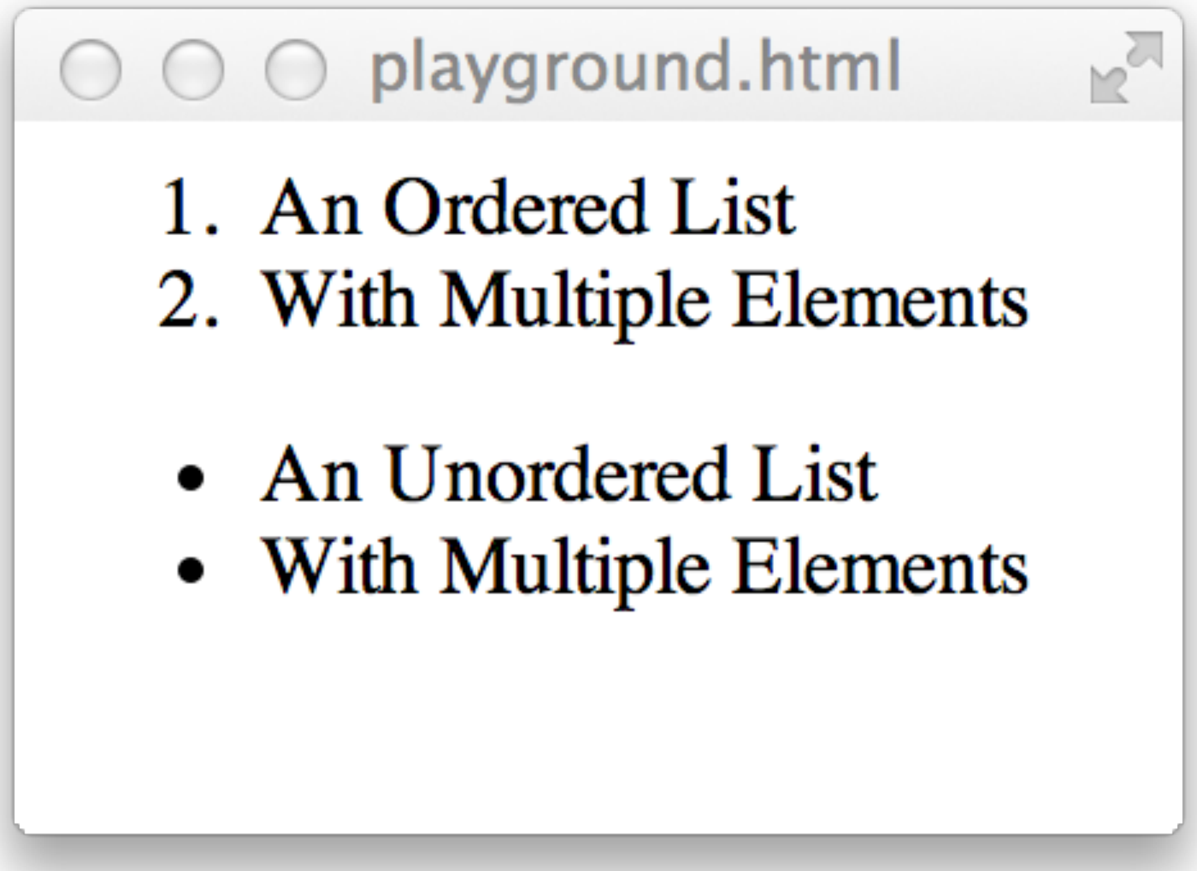
Ordered and Unordered Lists

- `<ol>` Ordered List
- `<ul>` Unordered List
- `<li>` List Element - Used for both types of lists
- Closing Tag for `<li>` may be omitted



```
<ol>
  <li>An Ordered List</li>
  <li>With Multiple Elements</li>
</ol>

<ul>
  <li>An Unordered List
  <li>With Multiple Elements
</ul>
```

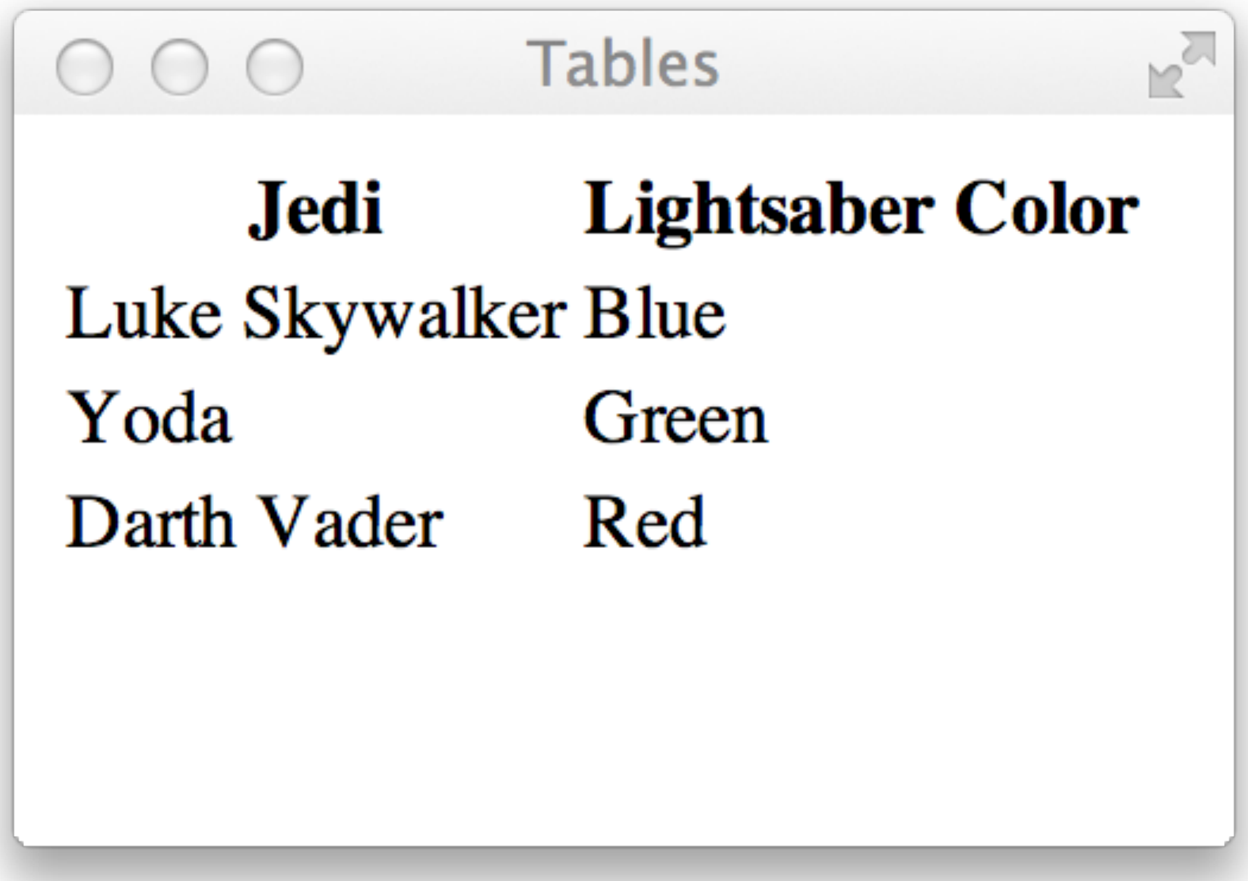


1. An Ordered List
2. With Multiple Elements

- An Unordered List
- With Multiple Elements

Tables

- `<table>` begins a table
- `<th>` table header
- `<tr>` table row
- `<td>` table data



Jedi	Lightsaber Color
Luke Skywalker	Blue
Yoda	Green
Darth Vader	Red

```
<table>
  <tr>
    <th>Jedi</th>
    <th>Lightsaber Color</th>
  </tr>
  <tr>
    <td>Luke Skywalker</td>
    <td>Blue</td>
  </tr>
  <tr>
    <td>Yoda</td>
    <td>Green</td>
  </tr>
  <tr>
    <td>Darth Vader</td>
    <td>Red</td>
  </tr>
</table>
```

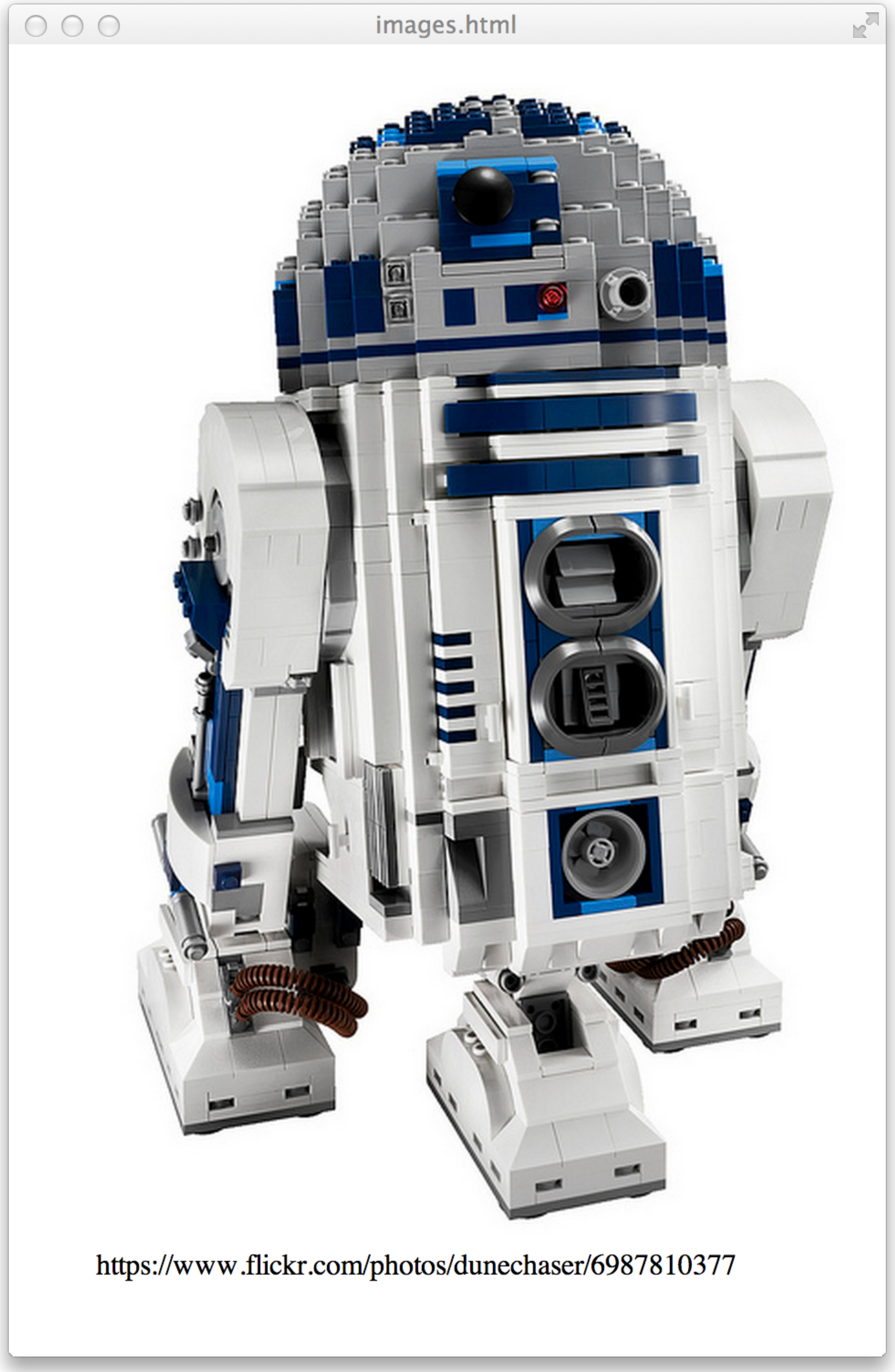
Headings

- `<h1>` 1st level heading - Biggest
- `<h6>` 6th level heading - Smallest
- `<h1> <h2> <h3>`
  `<h4> <h5> <h6>`

Images

```html
<figure>
  <img src="img/droid.jpg" alt="A Droid">
  <figcaption>
    https://www.flickr.com/photos/dunechaser/6987810377
  </figcaption>
</figure>
```

- Something other than text!
- The img tag is a void element, so it has no closing tag
- By default images are displayed at their native pixel size



https://www.flickr.com/photos/dunechaser/6987810377

Images

- Images can be resized with CSS, or with `width` and `height` attributes.
- Resized images are not resampled. The full image is sent to the browser no matter what size the image is ultimately displayed at.
- Assigning just `width` or `height` will scale the image and preserve the aspect ratio. (`width:height`)

Images

- The `alt` attribute should always be present, and should describe the image as best you can.

- Accessibility should be thought about from the very start of an HTML project, and not at the very end.

- If an image provides no useful information (a spacer image, or background gradient) an empty `alt` attribute should be used: `alt=""`

Images

- Three widely supported Image formats
  - GIF - Graphics Interchange Format
  - JPEG - Joint Photographic Experts Group
  - PNG - Portable Network Graphics
- HTML Specification does not mandate support for any particular format

GIF

- 256 distinct colors. Each GIF can have its own color pallet.
- One color can be designated as transparent.
- Can contain multiple frames for animation.
- Lossless compression, but limited format.

JPEG

- Millions of colors
- Lossy compression
    - Higher quality, less compression, larger file size
    - Smaller file size, higher compression, less quality
- Designed to be good at compressing photographs.
- No transparency

Photo © 2014 Angela Jennings

PNG

- Lossless compression
- No animation
- Several bit depth variants
  - PNG-8: 256 colors
  - PNG-24: 16 Million colors (3 8-bit channels)
  - PNG-32: 16 Million colors + 8-bit transparency
    - Allows for smooth anti-aliased transparency

WebP

- Lossless *or* lossy compression
- Animation
- Wide variety of bit-depths
- Supports Transparency (alpha channel)
- Good support for recent browsers (2020 on)

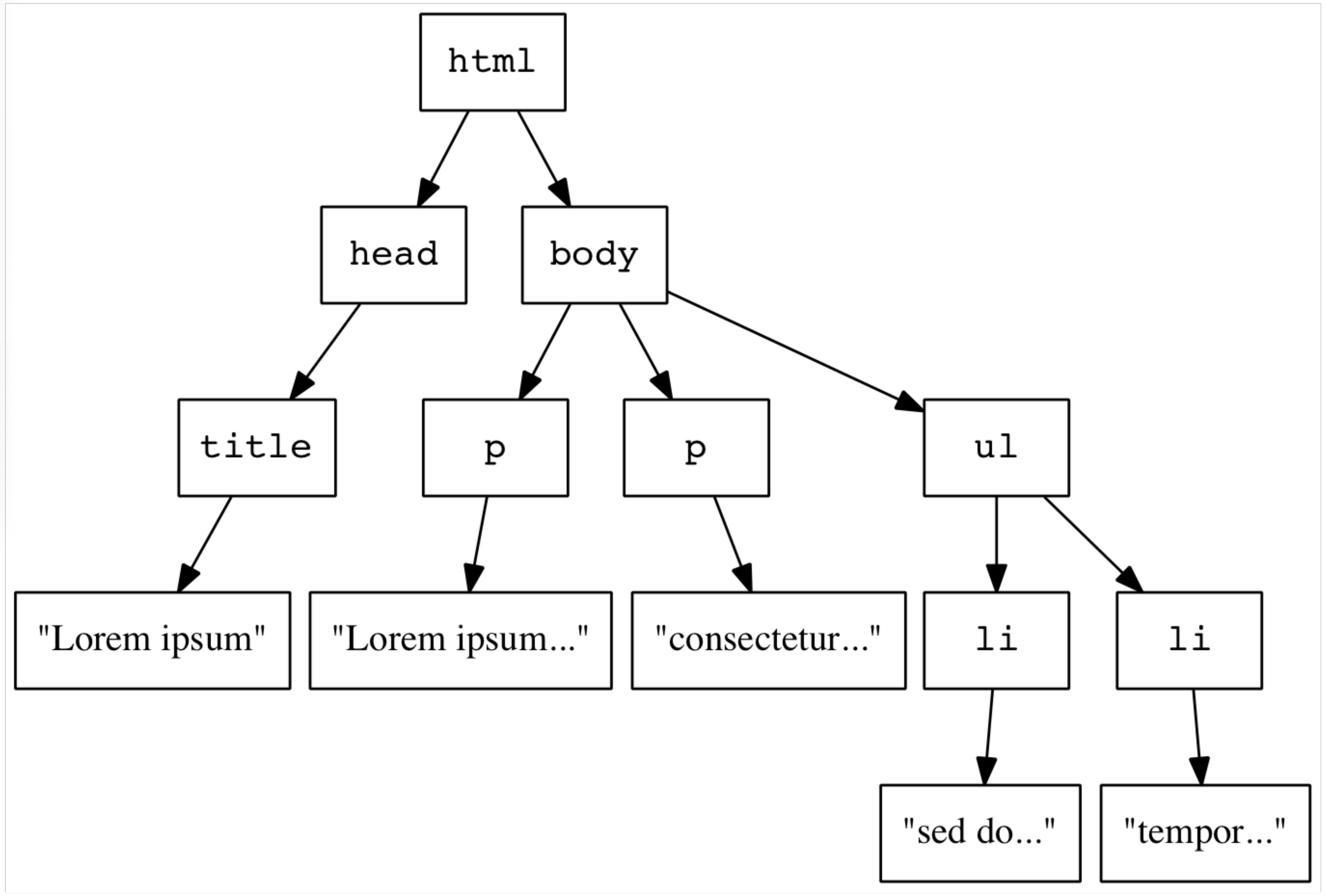
Images

	GIF	**JPEG**	**PNG**
Photograph		✔	
Animated	✔		
Icon or Drawing	✔		✔
Transparency	✔		✔

DOM Tree

```html
<!doctype html>
<head>
  <title>Lorem Ipsum</title>
</head>

<body>
  <p>
   Lorem ipsum dolor sit amet
  </p>
  <p>consectetur adipisicing elit
  <ul>
    <li>sed do eiusmod tempor incididunt
    <li>tempor incididunt
  </ul>
</body>
</html>
```



Misc Details

- HTML Tags and attribute names are **not case sensitive**
- Comments: `<!-- ••• -->`
  - Cannot nest comments. No inline comments
- Whitespace is mostly ignored. Multiple whitespace characters are condensed to a single space when rendered
- Text nodes and attribute values must be a tab, newline, form-feed, carriage-return or unicode characters ≥ than U+0020 (space)

next up: Networking Sockets