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

```
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 MUST recognize a single LF as a line
terminator and ignore any preceding CR.”


- Verbs and HTTP versions are Case Sensitive
  - GET / HTTP/1.1
  - Host: example.com
  - HTTP/1.1 505 HTTP Version Not Supported

- Headers are not Case Sensitive
  - GET / HTTP/1.1
  - Host: example.com
  - HTTP/1.0 501 Not Implemented

Basic HTTP Example

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

```
<!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. 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>
```

Response Code
Response Headers
Two CRLF
Response Body

Command Line Basics

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

Program Options
Program Arguments
Program Name
Command Prompt
Host
Port
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
We typed this stuff in the local nc program, it printed this:

Remote server sends this back:

```bash
$ curl -v http://example.com
```

Request:

- curl
- Request
- Request Headers
- Request Body

Response Headers:

- Response Headers
- Response Body

Examine Requests in Firefox:

- Examine
- Requests
- in Firefox
### Response Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1XX</td>
<td>Informational</td>
</tr>
<tr>
<td>2XX</td>
<td>Successful</td>
</tr>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>3XX</td>
<td>redirection</td>
</tr>
<tr>
<td>301</td>
<td>Moved</td>
</tr>
<tr>
<td>4XX</td>
<td>Client Error</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
</tr>
<tr>
<td>5XX</td>
<td>Server Error</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

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:
- Otherwise, just know it’s a thing

### Some HTML

```html
doctype html
<html>
<head>
  <title>Hello World</title>
</head>
<body>
  <p>A Basic HTML Page.</p>
</body>
</html>
```
### 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

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

```html
<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
The `<doctype ...>` preamble is not an HTML element.

- `<doctype ...>` tells the rendering engine what type of markup to expect
- HTML4.1 Transitional
- HTML5

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

The `<html>` element is the root element of our element tree

- The `<html>` element can only be preceded by whitespace characters and comments
- The `<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">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>A New Hope</title>
<meta name="description" content="Spaceships and lightsabers."
<link rel="stylesheet" href="default.css">
</head>
```

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

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

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

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

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

**Example URL:**

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

### URL

- A basic absolute URL
  ```
  http://www.arizona.edu
  ```

- A basic relative URL
  ```
  ../images/image.png
  ```
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:fischere@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

URI

[URI Diagram]

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-absolute
            / path-rootless
            / path-empty
```

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

URL Schemes

```
```

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>
  <li>With Multiple Elements</li>
</ul>
```

Tables

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

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

Headings

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

- 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.

- 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)`

- 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=""`. 

- The `img` tag is a void element, so it has no closing tag.
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
<table>
<thead>
<tr>
<th></th>
<th>GIF</th>
<th>JPEG</th>
<th>PNG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photograph</td>
<td></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Animated</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Icon or Drawing</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Transparency</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
DOM Tree

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

<p>Lorem ipsum dolor sit amet</p>
<p>consectetur adipisicing elit</p>
<ul>
<li>sed do eiusmod tempor incididunt</li>
<li>tempor incididunt</li>
</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 ≥ U+0020 (space)

---

**next up:** Networking Sockets