CS50 Lecture 8: Internet

Jordan Mandel

2021/11/04

The Internet

- internet is the network of computers
- routers control where the data goes between the computers; they are computers themselves
- protocols are the basic agreed algorithms for communication between computers
- packets of data are sent around the internet
- **IP** stands for internet protocol. **IP Addresses** are unique for computers connected to the inernet.
  - Routers have a table mapping IP addresses to cables each connected respectively to their own routers. The routers communicate with each other with their own protocols to route packets.
- **DNS** is provided by **Internet Service Providers** (**ISP**) to map things like `cnn.com` to actual IP addresses.
- **TCP** (translation control protocol) provides ability for a server at the same IP address to provide multiple services by appendign a **port number** to the end of the IP address. `80` is webpage. `503` is a secure web page.
  - can resend packets if it's not received. Asks for labeling/counting of packets.
  - data packets can take different routes.
  - net neutrality means that routers treat all packets equally.
  - brian asked for a cat, identifying which IP address was appropriate.
  - send multiple packets, add a sequence number.
  - can just assume internet works and build software on top.

Web Development

- The web is just one application on the internet. Zoom and email are others
- `HTTP` controls how we interact with TCP/IP packets: `HTTP` is what goes on inside the packets; `TCP` sends/receives the packet.
  - `GET` gets data, `POST` sends data.
- URL stands for uniform resource locator
- `https://www.example.com/`
  - `https` is the secure version of `http`
  - `.com` is the top level domain. for commercial. some top level domains have restrictions.
  - `www` lets us know we're on the worldwide web. but it's optional because it's mostly implied.
  - `/` at the end requests a default file, usually `index.html`.
- An HTTP request might start:

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

```
- `GET` means we are getting a file. `/` means a default file. Or more specific `GET /index.html`
- Using version 1.1 of `HTTP`.
- have to specify the website because the server might be hosting multiple.
```

- The response might be (each line is a header)

```
HTTP/1.1 200 OK
Content-Type: text/html
...
```

```
- `HTTP` version followed by status code.
- Type of content included in packet
- Then the actual packet content.
```

  - We can type in an `http` web address and it will switch over to `https`. This gets http status code `301 Moved Permanently`
  - Others include
    - `200 OK`
    - `301 Moved Permanently`
    - `304 Not Modified` lets us use the cache.
    - `307 Temporary Redirect`
    - `401 Unauthorized`
    - `403 Forbidden`
    - `404 Not Found`
    - `418 I'm a Teapot`
    - `500 Internal Server Error` might be caused by buggy code on a server
    - `503 Service Unavailable`
    - `302`, `304`, `307`.

There could be other headers.
`search?q=cats` is an example of a user input.

HTML

- Hypertext Markup Language
- An example

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>
        hello, title
        </title>
    </head>
    <body>
        hello, body
    </body>
</html>
```

```
- first line says it's html.
- then we have a tag.  the `lang="en"` is an attribute tag; specifies that the page is in english.
- there is a tree like heirarchical data structure.
- can access it by running http-server on our own computer.
```

Tag Examples

- `<p></p>` denotes paragraph breaks
- `<h1></h1>` is a first level heading

- For an unordered List

```
<ul>
<li>foo</li>
<li>bar</li>
</ul>
```

- replace with `<ol>` for an ordered list.
- For a grid table: (makes a keypad): `<tr>` are rows, `<td>` are columns.

```
<table>
    <tr>
        <td>1</td>
        <td>2</td>
        <td>3</td>
    </tr>
    <tr>
        <td>4</td>
        <td>5</td>
        <td>6</td>
    </tr>
    <tr>
        <td>7</td>
        <td>8</td>
        <td>9</td>
    </tr>
    <tr>
        <td>*</td>
        <td>0</td>
        <td>#</td>
    </tr>
</table>
```

Links

- For an image `<img alt="Harvard University" src="harvard.jpg">`, have to have the image in the directory. Alt is the hover text maybe. Describes the image.
- For a link `<a href="image.html", id="harvard"> Visit Harvard</a>` . `href` is where the link actually takes us. `a` is the anchor tag.
- Have to actually have the protocol. `<a href="https://www.harvard.edu"> Visit Harvard</a>`
- Can make an exploit by changing the URL (`href` attribute). ## CSS 1
- can specify style. It goes in another tag
- the `a:hover` is a pseudoselector; to select IDs

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>
            link
        </title>
            <style>
            #harvard
            {
            color: #ff0000
```

```
            }
            #yale
            {
            color: #00ff00
            }

            a
            {
                color: #ff0000;
                text-decoration: none;
            }

             a:hover
            {
                 text-decoration: underline;
            }
             </style>

 </head>
    <body>
        Visit <a href =="https://www.harvard.edu" id = "harvard">Harvard></a>
        or <a href="https://www.yale.edu", id = "yale"> yale </a>
    </body>
<html>
```

Search

- More complicated, but for a search entry:

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>search</title>
    </head>
<body>
    <form action="https://www.google.com/search" method="get">
        <input name="q" type="search">
        <input type="submit" value="Search">
    </form>
</body>
</html>
```

- post method also available but it is hidden deeper.

CSS 2

- properties instead of attributes
- we added stye tag in head
- or can go `<link href="styles.css" rel="styesheet">`
- Here we have the styles in the tags themeelves

```
<!DOCTYPE html>
```

```
<html lang="en">
    <head>
        <title>
        css
        </title>
    </head>



    <body>
        <header style="font-size: large; text-align: center;">
            John Harvard
        </header>
        <main style="font-size: medium; text-align: center;">
            Welcome to my home page!
        </main>
        <footer style="font-size: small; text-align: center;">
            Copyright &#169; John Harvard
        </footer>
    </body>

</html>
```

- Can move a style to a parent tag

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>
            css
        </title>
    </head>

    <body style = "text-align: center;">

        <header style="font-size: large;">
            John Harvard
        </header>

        <main style="font-size: medium;">
            Welcome to my home page!
        </main>

        <footer style="font-size: small">
            Copyright &#169; John Harvard
        </footer>
    </body>
</html>
```

- Can include all the styles in the `<head>` tag; called selectors; these are ***type selectors***; also ***class selectors***, ***id selectors***

```
<!DOCTYPE html>
```

```
<html lang="en">

    <head>
        <style>

            header
            {
                font-size: large;
                text-align: center;
            }

            main
            {
                font-size: medium;
                text-align: center;
            }

            footer
            {
                font-size: small;
                text-align: center;
            }

        </style>

        <title>
        css
        </title>
    </head>

    <body>
        <header>
            John Harvard
        </header>
        <main>
            Welcome to my home page!
        </main>
        <footer>
            Copyright &#169; John Harvard
        </footer>
    </body>
</html>
```

- Can make our own classes; the ***class selectors*** from before; recall pseudo-selectors from before

```
<!DOCTYPE html>

<html lang="en">
    <head>

        <style>
            body
            {
                background-color: #ff0000
                color: white;
```

```
            }
            .centered
            {
                text-align: center;
            }

            .large
            {
                font-size: large;
            }

            .medium
            {
                font-size: medium;
            }

            .small
            {
                font-size: small;
            }

        </style>

        <title>
        css
        </title>

    </head>

    <body>
        <header class="centered large">
            John Harvard
        </header>
        <main class="centered medium">
            Welcome to my home page!
        </main>
        <footer class="centered small">
            Copyright &#169; John Harvard
        </footer>
    </body>

</html>
```

- Cand do two things:
  - can change the centered attribute to be in the `body` tag.
  - As mentioned before: can put the css in its own file that has the contents of the `style` tag above:
- can put the `centered` in the `<body>` tag.

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <link href="styles.css" rel="stylesheet">
        <title>css</title>
    </head>
```

```
    <body class = "centered">
        <header class="large">
            John Harvard
        </header>
        <main class="medium">
            Welcome to my home page!
        </main>
        <footer class="small">
            Copyright &#169; John Harvard
        </footer>
    </body>
</html>
```

Javascript

```
let counter = 0;

counter = counter + 1;
counter += 1;
counter++;

if (x < y)
{

}

if (x < y)
{

}
else
{

}

if (x < y)
{

}
else if (x > y)
{

}
else
{

}

while (true)
{

}
```

```
for (let i = 0; i < 3; i++)
{

}
```

- Can put `<script src="scripts.js"></script>`; or just `<script></script>` tag in the html.

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>
            hello, title
        </title>
    </head>
    <body>
        hello, body
        <script src="scripts.js"></script>
    </body>
</html>
```

- Before we had a form like this

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>search</title>
    </head>
<body>
    <form action="https://www.google.com/search" method="get">
        <input name="q" type="search">
        <input type="submit" value="Search">
    </form>
</body>
</html>
```

- Now we are changing it
- can write an `onsubmit` function. `-return false` means don't submit the form, just run the function

```
<html lang="en">
    <head>

    <script>
    function greet()
    {
        let name = document.querySelector('#name').value;
        alert('hello' + name')
    }
    </script>

    <title>
        hello, title
    </title>

    </head>
    <body>
```

```
        <form onsubmit="greet(); return false;">
            <input id="name"  autocomplete="off" autofocus placeholder="whats your name", type="text">
            <input type="submit", value = "Greet me">
        </form>
    </body>
</html>
```

- see it greets you!
- Trying something new (purposefully causes an error)
- Can write code that listens for events
- Pass name of function not the function itself.
- But the problem is thaat there are order problems; the `form` tag doesn't exist in the first `document.querySelector` call.

```
<html lang="en">
    <head>
        <script>
            function greet()
            {
                alert('hello, ' + name)
            }

            document.querySelector('form').addEventListener('submit', greet);
        </script>
    <title>
            hello, title
        </title>
    </head>

    <body>
        <form>
            <input id="name"  autocomplete="off" autofocus placeholder="just to hold a place", type="te
            <input type="submit", value = "Greet me">
        </form>
    </body>

</html>
```

- Fixing the error (look for it in the js console); it is an order problem.

```
<html lang="en">
    <head>
        <script>
            function greet()
            {
                alert('hello, ' + name)
            }

            function listen()
            {
            document.querySelector('form').addEventListener('submit', greet);
            }

            document.adEventListener('DOMContentLoaded', listen);
```

```
        </script>
    <title>
            hello, title
        </title>
    </head>
    <body>
        <form>
            <input id="name"  autocomplete="off" autofocus placeholder="just to hold a place", type="te
            <input type="submit", value = "Greet me">
        </form>
    </body>
</html>
```

- Can use an ***anonymous function***

```
<html lang="en">
    <head>
        <script>
            function greet()
            {
                alert('hello, ' + name)
            }


            document.adEventListener('DOMContentLoaded',
                function()
                {
                document.querySelector('form').addEventListener('submit', greet);
                }
            );

        </script>
    <title>
            hello, title
        </title>
    </head>
    <body>
        <form>
            <input id="name"  autocomplete="off" autofocus placeholder="just to hold a place", type="te
            <input type="submit", value = "Greet me">
        </form>
    </body>
</html>
```

- Can re-nest and get rid of greet function. Even though I actually strongly prefer to keep the function names.

```
<html lang="en">
    <head>
        <script>
            document.adEventListener('DOMContentLoaded',
                function(){
                document.querySelector('form').addEventListener('submit',   function(){
                alert('hello, ' + name)
                });
```

```
        });
        </script>

    <title>
            hello, title
        </title>
    </head>
    <body>
        <form>
            <input id="name"  autocomplete="off" autofocus placeholder="just to hold a place", type="te
            <input type="submit", value = "Greet me">
        </form>
    </body>
</html>
```

- List of other events includes `blur`, `change`, `click`, `drag`, `focus`, `keyup`, `load`, `mousedown`, `mouseover`, `mouseup`, `submit`, `touchmove`, `unload`, among many others.

Can make it respond to keyup events, any time a key is released.

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <script>

            document.addEventListener('DOMContentLoaded', function() {
                let input = document.querySelector('input');
                input.addEventListener('keyup', function(event) {
                    let name = document.querySelector('#name');
                    if (input.value) {
                        name.innerHTML = `hello, ${input.value}`;
                    }
                    else {
                        name.innerHTML = 'hello, whoever you are';
                    }
                });
            });

        </script>
        <title>hello</title>
    </head>
    <body>
        <form>
            <input autocomplete="off" autofocus placeholder="Name" type="text">
        </form>
        <p id="name"></p>
    </body>
</html>
```

Unnesting the above didn't work for me.

- There is string substitution with backticks and dollar sign bracket

Changing the background.

```
<!DOCTYPE html>
```

```
<html lang="en">
    <head>
        <title>background</title>
    </head>
    <body>
        <button id="red">R</button>
        <button id="green">G</button>
        <button id="blue">B</button>
        <script>

            let body = document.querySelector('body');
            document.querySelector('#red').onclick = function() {
                body.style.backgroundColor = 'red';
            };
            document.querySelector('#green').onclick = function() {
                body.style.backgroundColor = 'green';
            };
            document.querySelector('#blue').onclick = function() {
                body.style.backgroundColor = 'blue';
            };

        </script>
    </body>
</html>
```

- Change size

```
<!DOCTYPE html>

<!-- Demonstrates programmatic changes to size -->

<html lang="en">
    <head>
        <title>size</title>
    </head>
    <body>
        <p>
            Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam in tincidunt augue. Duis im
        </p>
        <select>
            <option value="xx-large">xx-large</option>
            <option value="x-large">x-large</option>
            <option value="large">large</option>
            <option selected value="initial">initial</option>
            <option value="small">small</option>
            <option value="x-small">x-small</option>
            <option value="xx-small">xx-small</option>
        </select>
        <script>

            document.querySelector('select').addEventListener('change', function(event) {
                document.querySelector('body').style.fontSize = this.value;
            });

        </script>
```

```
    </body>
</html>
```

- Make it blink

```
<html lang="en">
    <head>
        <script>
            // Toggles visibility of greeting

            function blink()
            {
                let body = document.querySelector('body');
                if (body.style.visibility == 'hidden')
                {
                    body.style.visibility = 'visible';
                }
                else
                {
                    body.style.visibility = 'hidden';
                }
            }
            // Blink every 500ms
            window.setInterval(blink, 500);
        </script>
        <title>blink</title>
    </head>
    <body>
        hello, world
    </body>
</html>
```

- get location

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>geolocation</title>
    </head>
    <body>
        <script>
            navigator.geolocation.getCurrentPosition(function(position) {
                document.write(position.coords.latitude + ", " + position.coords.longitude);
            });
        </script>
    </body>
</html>
```

- autocomplete

```
<!DOCTYPE html>

<html lang="en">

    <head>
        <meta name="viewport" content="initial-scale=1, width=device-width">
        <title>autocomplete</title>
```

```
    </head>
    <body>
        <input autocomplete="off" autofocus placeholder="Query" type="text">
        <ul></ul>
        <script src="large.js"></script>
        <script>
            let input = document.querySelector('input');
            input.addEventListener('keyup', function(event) {
                let html = '';
                if (input.value) {
                    for (word of WORDS) {
                        if (word.startsWith(input.value)) {
                            html += `<li>${word}</li>`;
                        }
                    }
                }
                document.querySelector('ul').innerHTML = html;
            });

        </script>

    </body>
</html>
```

- attach to light bulb

```
import os
import requests

USERNAME = os.getenv("USERNAME")
IP = os.getenv("IP")

URL = f"http://{IP}/api/{USERNAME}/lights/1/state"

requests.put(URL, json={"on": False})
```

- make it blink

```
import os
import requests
import time

USERNAME = os.getenv("USERNAME")
IP = os.getenv("IP")
URL = f"http://{IP}/api/{USERNAME}/lights/1/state"

while True:
        requests.put(URL, json={"bri": 254, "on": True})
            time.sleep(1)
                requests.put(URL, json={"on": False})
                    time.sleep(1)
```

Internet Primer

- IP addresses are four clusters of 8 bits using decimal numbers

- All between 0 and 255
- but there are too many devices. To solve this there are private and public IP addresses.
- Now we have `IPv6`, with 128 bit values, `s:t:u:v:w:x:y:z`, represented by 1 to 4 hexadecimal digits.
- `::` omits zeros
- there's also `DHCP` assigns IP addresses.
- dns is the domain name for an IP address; like the yellow pages
- not every domain nameis a website we can go to
- really like a local DNS; they are agreggators
- heirarchical DNS; the top ones are like a library with many yellow pages
- Router has 1 IP address, is a single access point, deals with multiple devices. YOu have private IP addresses from the router. A *router* is really a router, switch, access point, modem all bundled up.
- `WANS` split these components up to help them scale.

The internet is just all these networks.

IP

- Instead of a complete graph, things are connected together by routers.
- Stored in IP tables in the routers.

- The data are split into packets.
- might send packets more than one time
- called connectionless because there is no direct connection.
- no guarantee of delivery if something is dropped, have to use `TCP` for that.

TCP

- get the info to the right program on the machine, guarantee delivery. tcp uses a particular port number.
- Guarantee deliversy what packets we should expect to get, and in what order.
- TCP uses ports: `21` for FTP, `SMTP` (email) uses port 25, `DNS` uses port 53, `HTTP` uses port 80. `HTTPS` uses port 443. `DDS` is data distribution service, `RDP` is remote desktop protocol. `XMPP` is extensible message and presence protocol chat.
- IP layer, TCP layer, then data. TCP can send a message back to sender.

HTTP

- application layer protocol
- these are the ones that use ports

Example is

```
GET/ HTTP/1.1
Host: cats.com
```

response

```
HTTP/1.1 200 OK
Content-type: text.html
```

or

```
HTTP/1.1 404 Not Found
Content-type: text/html
```

A request line will usually be:

`method http-version`
`request-target`

`200` is OK. `301` is moved permanently , get redirected `302` found, page is temporarily at a new location. `307` internal redirect, often after 302 `401` unauthorized: need credentials. `403` forbidden; never indented to be accessed `404` not found `500` server error: the server had its own problem `504` the server was taking too long; gateway timeout

HTML

There has to be an `<html></html>` tag. `<head>` tag stuff not in browser window but helps us. Title tag is what shows up in a tab.

Indentation doesn't matter in HTML. There are over 100 html tags. `<b>`, `<i>`, `<u>` tags are respectively bold, underline, italic. `<p>`, `<hx>` are paragraphs and headers respectively. `<ul>`, `<ol>`, `<li>` are respectively unordered, ordered, and list items. Can take a start attribute.

`<table>`, `<tr>`, `<td>`

`<form>` an html form `<div>` an arbitrary html division. `<input name=x type=y />`. They go inside forms. Note that this is a self closing tah `<a href=X><\a>` `<img src=x \>` `<!DOCTYPE html>` for html 5

Difference between internal links an external links. Make sure it is well organized.

CSS

HTML organizes the content CSS changes how it looks.

For example:

```
body
{
    background-color: blue;
}
```

Some examples include `border: style color width`, `background-color: [keyword | #<6-digit-hex>]`, `color: [keyword | #<6-digit hex>]` color is for text, often.

`font-size: [absolute size | relative size]`

`font-family: [font name | generic name]`

`font-align: [left | right | center | justify]`

ID Selectors: for example with `#unique` id

```
#unique
{
    border: 4px dotted blue;
    text-align: right;
}
```

There are also ***class selectors***

```
.students
{
    background-color: yellow;
    opacity: 0.7;
}
```

Can use a `<style>` tags. But preffered to write a `<link>` tag.

For example improved a table by going; this is a tag selector.

```
<style>
    table
    {
        border: 1px solid blue;
    }
</style>
```

But this just puts a border around it.

Next we try `<link href="tables.css" rel="stylesheet" />` . `rel` what is this relative to our document?

The file is:

```
table
{
border: 5px solid red;
}

tr
{
height: 50px
}

td
{
/* Losts being applied here! */
background-color:black;
color: white;
font-size: 22pt;
font-family: georgia;
}
```

Javascript

-include in `<script>` tags. but better to use `src` attribute. There can be anonymous function.

Before we had

```
struct car
{
    int year
    char model[10]
}
```

Now we have

```
struct car herbie;
herbie.year = 1963;
herbie.model = "Beetle"
```

JS objects also have methods.

We can go

```
var herbie ={year: 1963, model: 'Beetle'};
```

In python we had

```
for key in list:
    blah blah
```

Now we have:

```
for (var key in obect)
{
 object[key]
}
```

also:

```
for (var key of object)
{
    // use key here directly
}
```

```
var wkArray = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
for (var day in wkArray)
{
// prints the indices
    console.log(day);
}
for (var day of wkArray)
{
// prints the actual days
    console.log(day);
}
```

Parseint can be a useful function.

```
console.log(wkArray[day] + ' is day number ' + (parseInt(day) + 1) + ' of the week!');
```

Examples of functions for arrays are `array.size()`, `array.pop()`, `array.push(x)`, `array.shift()`.

Using `map` we can apply a function to arrays.

```
var nums = [1, 2, 3, 4, 5];

nums = nums.map(function(num){
return num * 2;
});
```

There are events and event handlers.

For example:

```
<html>
    <head>
        <title>Event Handlers</title>
    </head>
    <body>
        <button onclick="alertName(event)">Button 1</button>
        <button onclick="alertName(event)">Button 2</button>
    </body>
</html>
```

And then somewhere we can have

```
function alertName(event)
{
    var trigger = event.srcElement;//what element was used to activate this
    alert('You clicked on ' + trigger.innerHTML); // the inner contents of of this
}
```

DOM

The document object

```
<html>
    <head>
        <title>Hello, world</title>
    </head>
    <body>
        <h2>Here's my page</h2>
        <p>World, hello</p>
        <a href="test.html">Link</a>
    </body>
</html>
```

Then we go

`console.dir(document)`

From there we get the document object.

DOM properties

DOM Properties include: - `inerHTML`: the HTML inside of HTML tags. - `nodeName`: the name of an HTML element or element's attribute. - `id`: the `id` attribute of an `HTML` element. - `parentNode`: a reference to a node one level up in the DOM. - `childNodes`: all the nodes one level down. - `attributes`: the html attributes - `style`: the object having the CSS/HTML of an object.

DOM methods

- `getElemtnById(id)`: get the element with this name below in the dom
- `getElementsByTagName(tag)`: Gets all the elements with the given tag name below
- `appendChild(node)`: Add the given node to the DOM below this point.
- `removeChild(node)`: remove the specified child note from the DOM

jQuery

- In raw JS we have `document.getElementById('colorDiv').style.backgroundColor = 'green'`
- in jQuery we have `$('#colorDiv').css('background-color','green');`
- `https://api.jquery.com`