CS50's
Understanding Technology

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://orcid.org/0000-0001-5338-2522)
(https://www.quora.com/profile/David-J-Malan)
(https://www.reddit.com/user/davidjmalan) (https://twitter.com/davidjmalan)

Web Development

by Spencer Tiberi

Internet Recap (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=0m09s)

- All the computers on the internet are interconnected that supports HTTP and TCP/IP
- The internet is an infrastructure to get data from a server to a client
    - Supports emails, video conferencing, etc.
- The web is one specific application or service that runs atop the internet
    - Assumes an internet exists to get data from point A to point B
        - Layers functionality that allows us to click, browse, etc.

Web Browser (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=1m3s)

- Web browsers are found on phones, computers, and game consoles

- They have a space to enter a URL (Uniform Resource Locator)
  - Prefixed by `http://` or `https://`
- When typing in a URL, you're sending a request from your device to some remote server
  - The server looks at your request and figures out how to respond
  - Like when we previous requested cat images!

Web Server (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=2m52s)

- A computer that has a CPU, RAM, and a hard drive
- Rack servers 
  - Sized so they can be stacked
- Odds are your company has many of these if they have a web server

HTTP (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=4m21s)

- We send requests to web servers
  - Language of these requests is HTTP (Hypertext Transfer Protocol)
- Request: `GET /cat.jpg HTTP/1.1`
  - `1.1` refers to HTTP language 1.1
- Response by the server: `HTTP/1.1 200 OK`
  - This literally means that everything was okay with the request

HTTP Status Codes (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=6m3s)

Code	Status	Meaning
200	OK	Everything is OK
301	Moved Permanently	Browser should redirect to new location

Code	Status	Meaning
302	Found	Browser should redirect to new location
304	Not Modified	Browser will cache files if things don't change to save time/bytes from requests
401	Unauthorized	Not authorized to view content; Could require login
403	Forbidden	Not able to view content
404	Not Found	The requested data could not be found because it doesn't exist on the server
500	Internal Service Error	Not your fault; The server erred

HTML (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=8m37s)

- In addition to the HTTP headers that include status codes, the bits representing an image or website will be sent to you
- The language that builds websites is HTML (Hypertext Markup Language)
    - Sent as a response for a request for a web page
- HTML isn't a programming language but rather a markup language
    - Allows you to format, but doesn't have control flow such as loops and conditions

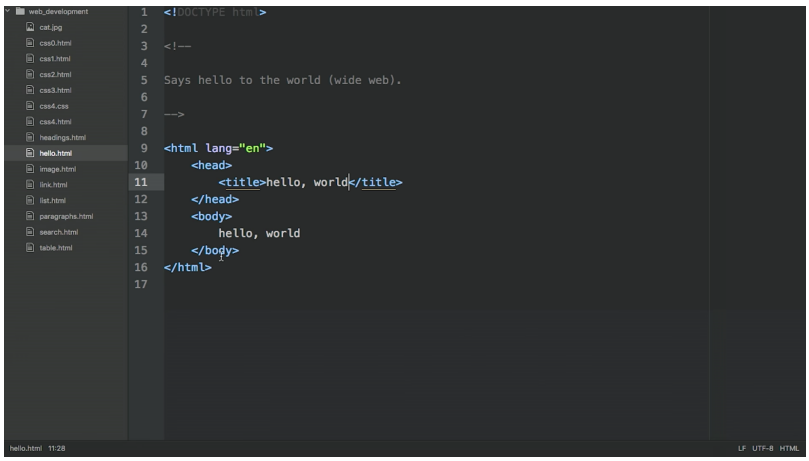
```
<!DOCTYPE html>

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

        - This is html for a webpage that says "hello, world"
- To implement webpages, you need to write HTML
    - Editors like Atom and Sublime Text exist to help write HTML
        - Ultimately, all you need is a computer, a keyboard, and some way of typing out text!
- `<!DOCTYPE html>`

- `<!DOCTYPE html>`
    - Lets the browser know the following file is written in HTML 5
- `<html lang="en">`
    - Specifies that the webpage is written in English
- `<head></head>`
    - Example of open and close tags
- First tag opened is the last tag closed
    - HTML is a tag-based markup language
        - Tags have attributes
- Standard extension for a webpage is .html
- David clicks the hello.html file and loads the page
    - In the top corner of the browser tab is the title
        - This comes from the `head`
    - The `body` contains 99% of the webpage's content
    - This page is a local document, so the address is where David saved it
        - Not on a web server, so no one else can access it
- There are web hosts to serve up websites we write
    - We can also buy our own domain name

Atom (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=19m0s)

- TextEdit is not designed for web page development
- Free alternatives made for web development exist
- Atom is an example of one of these editors



- Fun fact: these notes were indeed created on Atom!
- In Atom, you can open multiple files at once
- Colors are added for readability
    - These colors don't effect how the webpage will render

- HTML supports comments
    - To help colleagues who look at your code know your intentions of the code

Links (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=21m17s)

- Links can make our pages more dynamic by linking to other pages
    - `<a></a>` are anchor tags that can be used for links

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>link</title>
    </head>
    <body>
        Visit <a href="http://www.harvard.edu/">Harvard</a>.
    </body>
</html>
```

- href (hyper reference) is set to where you want the link to go
- Blue, underlined text traditionally represents a link on a webpage
- The bottom left corner on Chrome shows the destination of a link when you hover over the text
- A link traditionally becomes purple if you've already followed that link
    - Browser remembers where you've been
        - Potential privacy concern

Images (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=26m46s)

- The web is filled with images
- `<img/>` is the image tag

```
<!DOCTYPE html>

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

```
        <img alt="Grumpy Cat" src="cat.jpg"/>
    </body>
</html>
```

- The `src` (source) attribute is set to the address of the file
- The `alt` (alternative text) attribute is what displays if the page can't be seen
- Closes itself as one tag

Paragraphs (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=30m42s)

- Even if you add spaces to format paragraphs, HTML will render without them!
- When looking at your webpage you can "view page source" on your browser to see the original HTML with your spaces, but the webpage still doesn't have these spaces
- The browser will only do what HTML tells it to do
    - The browser needs instructions in the form of HTML tags
- Paragraph tags (`<p></p>`) tell the browser to create a paragraph of text

Headings (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=34m25s)

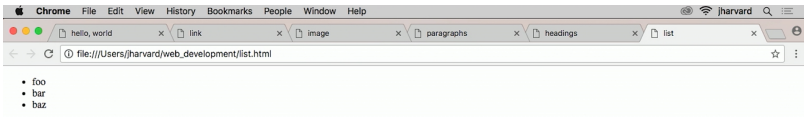
- `<h1></h1>` are the heading 1 tags
- There also exists `<h2></h2>`, `<h3></h3>`, `<h4></h4>`, `<h5></h5>`, and `<h6></h6>`
- Headings get smaller the larger the number
- These make the font larger for usage similar to marking chapters in a book

Lists (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=35m27s)

- Unordered lists
    - Use bullets (like these!)
    - `<ul></ul>`
- `<li></li>` are list item tags

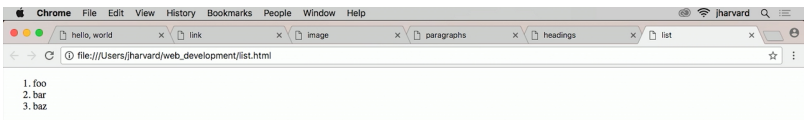
```
<!DOCTYPE html>
```

```
<html lang="en">
    <head>
        <title>unordered list</title>
    </head>
    <body>
        <ul>
            <li> foo </li>
            <li> bar </li>
            <li> baz </li>
        </ul>
    </body>
</html>
```



- Ordered lists
    - Use numbers
    - `<ol></ol>`

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>ordered list</title>
    </head>
    <body>
        <ol>
            <li>foo</li>
            <li>bar</li>
            <li>baz</li>
        </ol>
    </body>
</html>
```

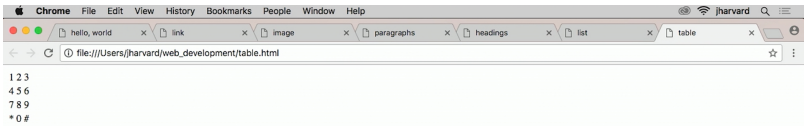


Tables (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=36m49s)

- `<table></table>` are table tags that create a table
    - `<tr></tr>` are table row tags
    - `<td></td>` are table data tag

- Like columns or cells

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>table</title>
    </head>
    <body>
        <table>
            <tr>
                <td>7</td>
                <td>8</td>
                <td>9</td>
            </tr>
            <tr>
                <td>4</td>
                <td>5</td>
                <td>6</td>
            </tr>
            <tr>
                <td>1</td>
                <td>2</td>
                <td>3</td>
            </tr>
            <tr>
                <td>*</td>
                <td>0</td>
                <td>#</td>
            </tr>
        </table>
    </body>
</html>
```



Implementing Google (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=38m52s)

- When you type google.com your browser adds “https://www.” to the beginning of the URL
    - As is needed to surf the web
- `curl` is a command ran in the terminal that behaves much like a browser
    - It sends a request like a browser and shows what html is returned

```
$ curl google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
$
```

- Capital letter tags are a bit dated
- Shows google.com is located at http://www.google.com

```
$ curl -I google.com
HTTP/1.1 301 Moved Permanently
Location: http://www.google.com/
Content-Type: text/html; charset=UTF-8
Date: Tue, 18 Apr 2017 16:05:22 GMT
Expires: Thu, 18 May 2017 16:05:22 GMT
Cache-Control: public, max-age=2592000
Server: gws
Content-Length: 219
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
```

- The `-I` flag tells `curl` to return HTML headers
  - Includes status codes and other info humans normally don't see
- Google's server has been configured to redirect users to http://www.google.com
- UTF-8 is unicode characters

- `curl http://www.google.com` returns a webpage that includes HTML and JavaScript

```
$ curl -I http://www.google.com/
HTTP/1.1 200 OK
Date: Tue, 18 Apr 2017 16:08:20 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See https://www.google.com/support/accounts/answer/151657?hl=en for more info."
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Server: gws
Set-Cookie: NID=101=qOX4BHmRkSpTusUZ58vuyvkycuxCkO32QSIhV7Uf2_EZRZJhNaeVDR5b77qM_-NrR785chL0hLH6Gio3-hrwvKB2emNiSB8w0pSXNYQEilofFbdOSc3VlgyA3JtKFp9ME4xX6j9Q5sH3vHtn; expires=Wed, 18-Oct-2017 16:08:20 GMT; path=/; domain=.google.com; HttpOnly
Transfer-Encoding: chunked
Accept-Ranges: none
Vary: Accept-Encoding
```

- Searching for cats changes the URL to https://www.google.com followed by a large sequence of characters
  - Distilling this URL to https://www.google.com/search?q=cats leads to the same results
    - We can "create" a search engine using this info!

Forms (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=48m2s)

- `<form></form>` are form tags that take attributes for an action and a method

```
<!DOCTYPE html>

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

- `action="https://www.google.com/search" method="get"` means "get me https://www.google.com/search"
- Inside the form, we can have `<input/>` tags
    - These can have name, type, value, and text attributes
- This implementation punts the searching to Google
- The browser uses the HTML form to assemble a URL
    - `https://www.google.com/search?q=cats`
- `?` in the URL means "Hey Server! Here comes my HTTP parameters!"
    - A URL may have multiple parameters separated by `&`

css0.html (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=51m47s)

- Let's make our webpages more pretty
- CSS (Cascading Style Sheets) allows us to style our webpages
    - In contrast, HTML allows us to structure our webpages

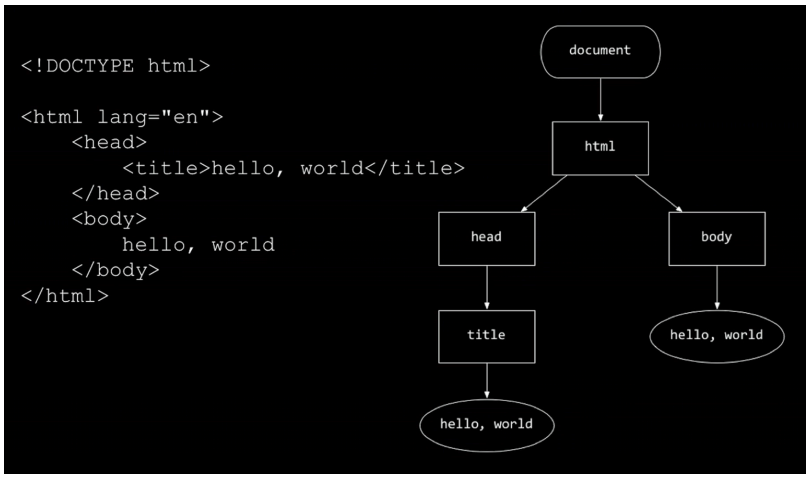
```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>css0</title>
    </head>
    <body>
        <header style="font-size: large; text-align: center;">
            John Harvard
        </header>
```

```
        </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>
```

- Here, inside `body`, we have three tags: `<header></header>`, `<main></main>`, and `<footer></footer>`
    - They include style attributes written in CSS
        - These are written as key-value pairs
        - In CSS, there is a property called `font-size`
            - CSS supports `small`, `medium`, `large`, and exact sizes such as `16px`
        - `text-align: center;` centers the text
- This example has some redundancy

DOM (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=55m52s)

- CSS supports the notion of a hierarchy



    - Rectangles here represent HTML tags or elements
    - Ovals represent text values
    - This is called a tree in Computer Science, much like a family tree
- When a browser receives a webpage, it builds a tree-like data structure in your computer's RAM
- Thus, in this case `header`, `main`, and `footer` are all child nodes of of the parent node `body`

css1.html (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=58m48s)

- We can put the `text-align: center;` attribute on the `body` element so it will pass it on to its children (`header`, `main`, and `footer`)

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>css1</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>
```

- This is better design as we can change all the text alignment at once

css2.html (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=1h1m11s)

- Combining HTML and CSS is generally frowned upon
- Makes it hard to collaborate
    - One person can work on content (HTML)
    - The other on style (CSS)

```
<!DOCTYPE html>

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

            .centered
            {
```

```
                text-align: center;
            }

            .large
            {
                font-size: large;
            }

            .medium
            {
                font-size: medium;
            }

            .small
            {
                font-size: small;
            }

        </style>
        <title>css2</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>
```

- `<style>` can be a tag as well as an attribute
- `.centered` defines a class named `centered`
    - Anything with this class with have the style attribute `text-align: center;`

css3.html (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=1h3m7s)

- We can even get rid of class attributes to further separate style from content

```
<!DOCTYPE html>
```

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

            body
            {
                text-align: center;
            }

            header
            {
                font-size: large;
            }

            main
            {
                font-size: medium;
            }

            footer
            {
                font-size: small;
            }

        </style>
        <title>css3</title>
    </head>
    <body>
        <header>
            John Harvard
        </header>
        <main>
            Welcome to my home page!
        </main>
        <footer>
            Copyright &%169; John Harvard
        </footer>
    </body>
</html>
```

- We can also give the tags CSS directly
- Will look identical, but better design

css4.html (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=1h5m5s)

- What if we remove the style altogether and store it elsewhere?

- What if we remove the style altogether and store it elsewhere?

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <link href="css4.css" rel="stylesheet"/>
        <title>css4</title>
    </head>
    <body>
        <header>
            John Harvard
        </header>
        <main>
            Welcome to my home page!
        </main>
        <footer>
            Copyright &%169; John Harvard
        </footer>
    </body>
</html>
```

- We have boiled the html down to its essence
    - No usage of style tags
- Note the `<link/>` tag with a `href` attribute of `css4.css` and a `rel` (relationship) attribute of `stylesheet`
    - This says “Hey Browser! Please link my stylesheet css4.css to this page!”
- In the same directory, we will have this stylesheet

```
body
{
    text-align: center;
}

header
{
    font-size: large;
}

main
{
    font-size: medium;
}

footer
{
    font-size: small;
}
```

- We have factored out the style to its own file
    - Easier for collaboration and sharing
    - Can use on multiple html pages
    - Can create different themes

Closing Thoughts (https://video.cs50.net/cscie1a/2017/fall/lectures/web_development?t=1h7m24s)

- Web development is about writing code
    - Specifically, HTML, which builds the structure of a webpage
    - CSS allows us to fine tune the webpage's aesthetics
- You can use these building blocks to further learn about web development on your own!
- The underlying concepts are more important than details

    - A webpage is nothing more than a text file written in HTML, CSS, and maybe some JavaScript
        - This file can be uploaded to a server to put on the internet
            - You can sign up for a web host with data centers
            - All your files will go in a folder on the server so that the webpage can be accessed on the internet
        - You can also buy a domain name and configure it to point to the web host
- These building blocks are what allow you to put your content on the internet!