

LING 408/508: Computational Techniques for Linguists

Lecture 15

Today's Topics


- Homework 6 graded
- New topic:
 - start with html5
 - leading to building your own webserver

A last view of bash...

JULIA EVANS
@b0rk

errors

by default, bash will continue after errors



oh, was that an error? who cares, let's keep running!!!


uh is that really the best choice?

bash programmer

set -e stops the script on errors

```
set -e  
unzip file.zip
```

typo! script stops here!



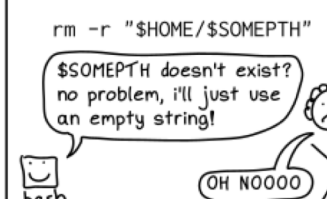
this makes your scripts WAY more predictable

bash programmer

by default, unset variables don't error

```
rm -r "$HOME/$SOMEPTH"
```

\$SOMEPTH doesn't exist? no problem, i'll just use an empty string!



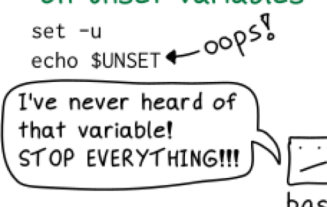
OH NOOOO

bash programmer

set -u stops the script on unset variables

```
set -u  
echo $UNSET
```

oops!

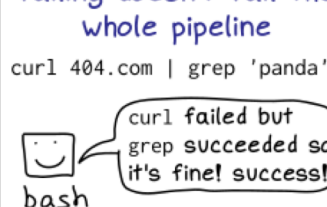


I've never heard of that variable! STOP EVERYTHING!!!

bash programmer

by default, a command failing doesn't fail the whole pipeline

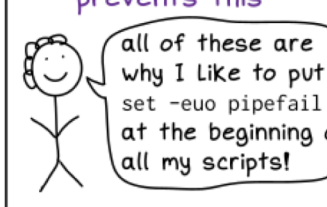
```
curl 404.com | grep 'panda'
```



curl failed but grep succeeded so it's fine! success!

bash programmer

set -o pipefail prevents this



all of these are why I like to put set -eo pipefail at the beginning of all my scripts!

bash programmer

Browser

- Nowadays browsers are very powerful in their own right (can compute locally, not just communicate with a webserver)



CSS (Cascading Style Sheets)

SVG (Scalable Vector Graphics)

- cf. HTML5 canvas

Javascript

- *programming language*

DOM (Domain Object Model)

- *programmatic access to documents*

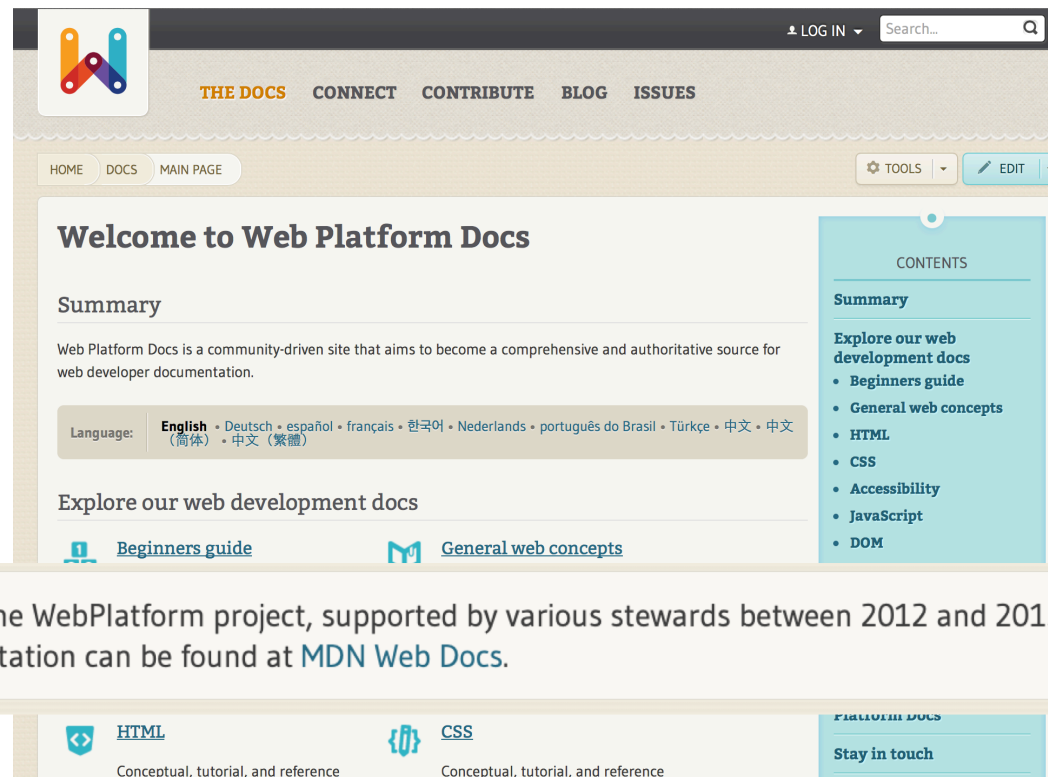
Websockets

- An API to interact with regular programs

HTML

- HTML: Hypertext Markup Language
 - Web browser: can read and render pages written in HTML
 - **currently:** HTML5
- What is "hypertext"?
 - linked content (Nelson, 1963)
 - nowadays: selected text/images/video can have arbitrary associated actions
- Hypercard for the Macintosh (1987)
- World Wide Web (WWW) (1992)
 - World Wide Web Consortium (W3C)

Reference



<https://webplatform.github.io>

Now deprecated...

Notice: The WebPlatform project, supported by various stewards between 2012 and 2015, has been **discontinued**. This site is now available on [github](https://github.com). New documentation can be found at [MDN Web Docs](https://mdn.io).

Reference

- <https://developer.mozilla.org/en-US/docs/Web/Tutorials>
-

HTML Tutorials

Introductory level

Introduction to HTML

This module sets the stage, getting you used to important concepts and syntax, looking at applying HTML to text, how to create hyperlinks, and how to use HTML to structure a webpage.

MDN HTML element reference

A comprehensive reference for HTML elements, and how the different browsers support them.

Intermediate level

Multimedia and embedding

This module explores how to use HTML to include multimedia in your web pages, including the different ways that images can be included, and how to embed video, audio, and even entire other webpages.

Advanced level

HTML forms

Forms are a very important part of the Web — these provide much of the functionality you need for interacting with websites, e.g. registering and logging in, sending feedback, buying products, and more. This module gets you started with creating the client-side parts of forms.

Creating a Simple Web Page with HTML

An HTML guide for beginners that includes explanations of common tags, including HTML5 tags. Also includes a step-by-step guide to creating a basic web page with code examples.

HTML Challenges

Use these challenges to hone your HTML skills (for example, "Should I use an `<h2>` element or a `` element?"), focusing on meaningful markup.

HTML tables

Representing tabular data on a webpage in an understandable, accessible way can be a challenge. This module covers basic table markup, along with more complex features such as implementing captions and summaries.

Tips for authoring fast-loading HTML pages

Optimize web pages to provide a more responsive site for visitors and reduce the load on your web server and Internet connection.

Reference

- <https://developer.mozilla.org/en-US/docs/Web/Tutorials>
-

CSS Tutorials

Introductory level

CSS basics

CSS (Cascading Style Sheets) is the code you use to style your webpage. *CSS Basics* takes you through what you need to get started. We'll answer questions like: How do I make my text black or red? How do I make my content show up in such-and-such a place on the screen? How do I decorate my webpage with background images and colors?

CSS first steps

CSS (Cascading Style Sheets) is used to style and lay out web pages — for example, to alter the font, color, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features. This module provides a gentle beginning to your path towards CSS mastery with the basics of how it works, what the syntax looks like, and how you can start using it to add styling to HTML.

Intermediate level

CSS layout

At this point we've already looked at CSS fundamentals, how to style text, and how to style and manipulate the boxes that your content sits inside. Now it's time to look at how to place your boxes in the right place in relation to the viewport, and one another. We have covered the necessary prerequisites so can now dive deep into CSS layout, looking at different display settings, traditional layout methods involving float and positioning, and new fangled layout tools like flexbox.

CSS building blocks

This module carries on where *CSS first steps* left off — now you've gained familiarity with the language and its syntax, and got some basic experience with using it, its time to dive a bit deeper. This module looks at the cascade and inheritance, all the selector types we have available, units, sizing, styling backgrounds and borders, debugging, and lots more.

The aim here is to provide you with a toolkit for writing competent CSS and help you understand all the essential theory, before moving on to more specific disciplines like [text styling](#) and [CSS layout](#).

Styling text

Here we look at text styling fundamentals, including setting font, boldness, and italics, line and letter spacing, and drop shadows and other text features. We round off the module by looking at applying custom fonts to your page, and styling lists and links.

Common CSS Questions

Common questions and answers for beginners.

Fluid Grids

Design layouts that fluidly resize with the browser window, while still using a typographic grid.

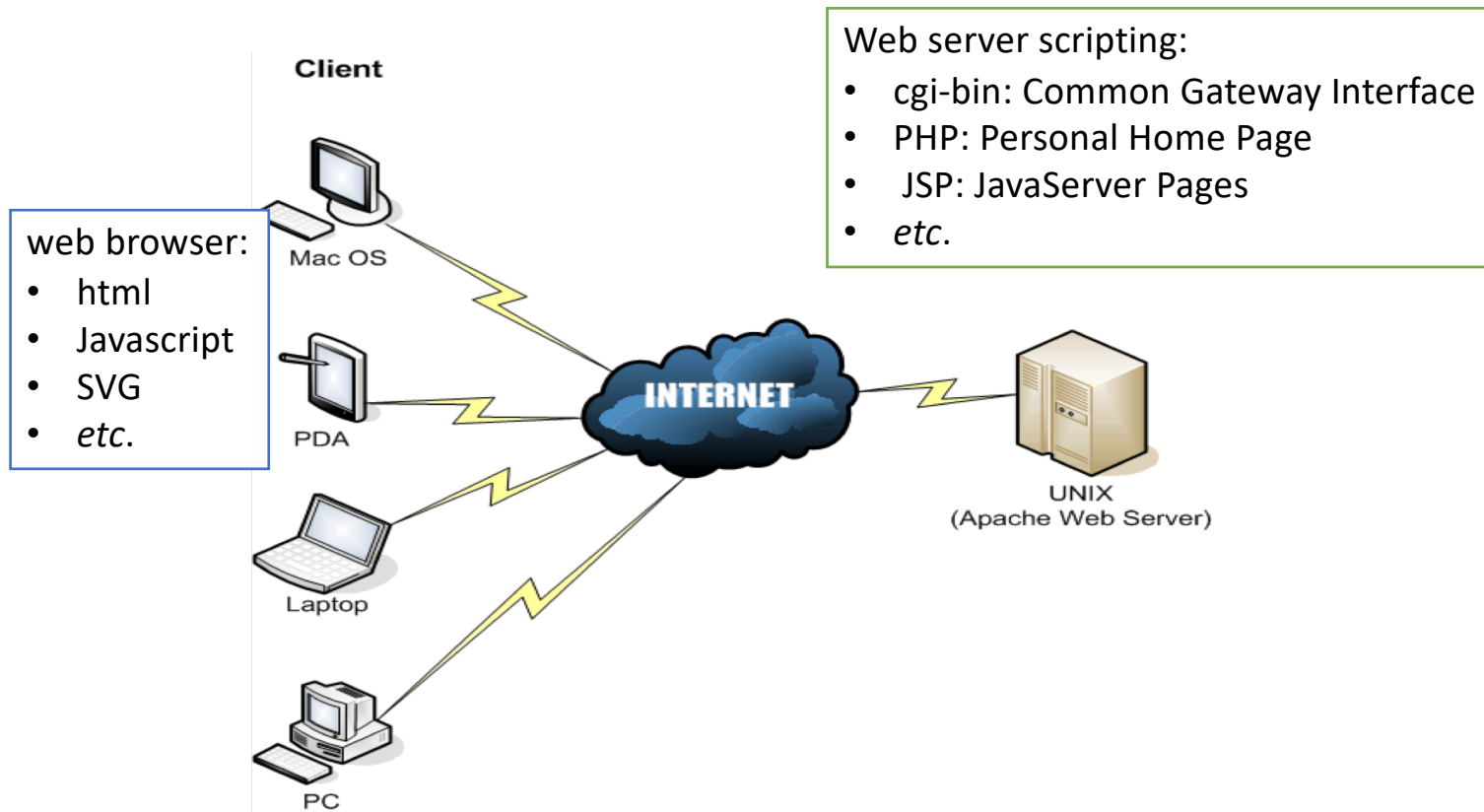
CSS Challenges

Flex your CSS skills, and see where you need more practice.

Client-side web development

- **HTML:**
 - structure of content
- **CSS (cascading style sheets):**
 - presentation
- **Javascript**
 - scripting language
- **DOM (document object model):**
 - hierarchical representation of webpage
- **SVG (scalable vector graphics):**
 - 2D graphical objects and methods

Client/server model

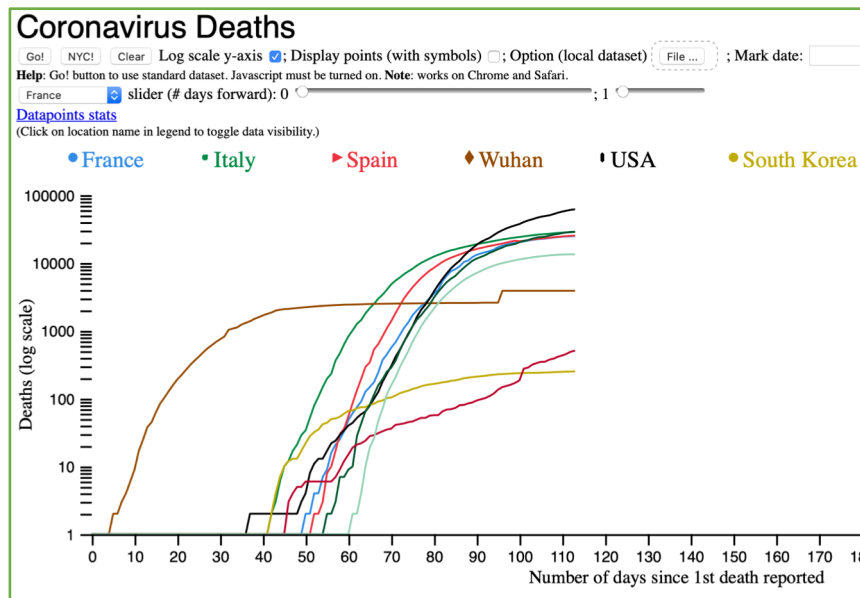


<http://www.visualbuilder.com/jsp/tutorial/introduction-to-jsp/>

Client-Server division of labor

- Example:

- <http://elmo.sbs.arizona.edu/sandiway/lockdown/>
- play with buttons and sliders ... (nothing to do with *elmo* or a webserver)



HTML

boilerplate inserted by my emacs editor:

```
1 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
2 <html> <head>
3 <title></title>
4 </head>
5
6 <body>
7 <h1></h1>
8
9
10
11 <hr>
12 <address></address>
13 <!-- hhmts start --><!-- hhmts end -->
14 </body> </html>
```

- First line:
 - <!DOCTYPE HTML>
 - signifies HTML5
- Tags:
 - <tag> ... </tag>
 - html
 - head: title, style, javascript definitions etc.
 - body: body of the document
 - h1: heading level 1 (1-6)
 - address: contact information
- "self-closing" tags:
 - hr: horizontal rule
 - br: line break optional:

- optionally paired:
 - <p> .. </p>: paragraph
- comment:
 - <!-- ... -->

HTML

- hypertext (link):
 - `text` (text presented in blue)
- URL: uniform resource locator
 - Examples:
 - <http://elmo.sbs.arizona.edu/sandiway/index.html>
 - <http://nlp.stanford.edu:8080/parser/>
 - <https://netid.arizona.edu/newid.php> (PHP)
 - <http://localhost/perl/test.pl> (mod_perl program)
 - Format:
 - protocol://host(:port)/path
 - protocol://host(:port)/path?query
 - protocol = http (hypertext transfer protocol)
 - port = TCP/IP port number

HTML

- Images:

- ``

- attribute: src

(required)

- value: URL (or filename etc.)

(jpg, gif, png supported, see note below)

- attribute: alt

(supposed to be required)

- value: text

- attribute: height

- value: pixels

- attribute: width

- value: pixels

- attribute: align

(not in HTML5)

- value: top | bottom | middle | left | right

- Can embed:

- ``

- **Note:** http://en.wikipedia.org/wiki/Comparison_of_web_browsers#Image_format_support

HTML

- Images can be embedded inside the file via base64 encoding:

```

```

- On Ubuntu and MacOS:

- `openssl base64 -in imagefile -out encoded.txt`

HTML

- Lists:
 - list item: ` ... `
 - ordered lists: ` ... `
 - unordered lists: ` ... `
 - **note**: can be nested arbitrarily deep

HTML

- inline styling applied to text elements:
- e.g. (inline text) vs. <div> (block text)
 - <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span>
 - ` ... `
 - font-size: Xpx
 - font-family: name, name ...
 - color: name (or hex RGB) e.g. #00CC00
 - background-color: name (or RGB)
 - text-align: left|right|center
 - **note:** serif, sans-serif, monospace are generic font families
 - **note:** X11 color names are okay,
http://en.wikipedia.org/wiki/X11_color_names

HTML

https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements

HTML (**Hypertext Markup Language**) elements historically were categorized as either "block-level" elements or "inline-level" elements. Since this is a presentational characteristic it is nowadays specified by CSS in the [Flow Layout](#). Inline elements are those which only occupy the space bounded by the tags defining the element, instead of breaking the flow of the content. In this article, we'll examine HTML inline-level elements and how they differ from [block-level elements](#).



An inline element does not start on a new line and only takes up as much width as necessary.

HTML

Conceptual differences

In brief, here are the basic conceptual differences between inline and block-level elements:

Content model

Generally, inline elements may contain only data and other inline elements. You can't put block elements inside inline elements.

Formatting

By default, inline elements do not force a new line to begin in the document flow. Block elements, on the other hand, typically cause a line break to occur (although, as usual, this can be changed using CSS).

- Block:
 - lists
 - headings:
 - `<h1> ... <h4>`
 - `<p>`
 - `<div>`
 - general container

HTML

- Other text element style tags (inline; semantic):
 - ` ... ` *italics*
 - ` .. ` **bold**
 - `<tt> ... </tt>` monospaced
 - `<code> ... </code>`
- Older style tags (specifies "look" or presentation):
 - ` ... ` **bold**
 - `<i> ... </i>` *italics*
- block-level:
 - `<pre> ... </pre>` preformatted

The **HTML** `<pre>` element represents preformatted text which is to be presented exactly as written in the HTML file. The text is typically rendered using a non-proportional ("**monospace**") font. Whitespace inside this element is displayed as written.

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/pre>

HTML

HTML

CSS

```
1 <pre>
2   L           TE
3   A           A
4   C     V
5   R A
6   DOU
7   LOU
8   REUSE
9   QUE TU
10  PORTES
11  ET QUI T'
12  ORNE O CI
13  VILISÉ
14  OTE- TU VEUX
```

Output

```
      L           TE
      A           A
      C     V
      R A
      DOU
      LOU
      REUSE
      QUE TU
      PORTES
      ET QUI T'
      ORNE O CI
      VILISÉ
      OTE- TU VEUX
      LA     BIEN
      SI     RESPI
            RER
```

- Apollinaire

HTML

- Tables:

- `<table> ... </table>`
- `<tr> ... </tr>`
- `<th> ... </th>`
- `<td> .. </td>`

- Attributes:

- `border="size"`
- `colspan="number"`
- `style="...;..."`
 - `border: width style color`
 - `border-width: top right bottom left`
 - `border-style: top right bottom left`
 - `border-color: top right bottom left`
 - `border-collapse: collapse | separate`
 - `padding: size`
 - `text-align: left | center | right`
 - `vertical-align: top | middle | bottom`
 - `width, height`

- Newer stuff:

- `<thead> ... </thead>`
- `<tbody> .. </tbody>`

table row

table heading element

table data (one cell)

e.g. 1px

e.g. 2 (span next two columns)

also border-left, border-top, border-right, etc.

e.g. type=solid | dotted | dashed | double | none

e.g. 100px or 100%

HTML

- General "chunks" of html:
 - `<div style="..."> ... </div>`
 - a division (block-level)
 - ` ... `
 - small chunks (inline)
- (optional) **unique identifier** `id=`
 - used to refer to the "chunk" in CSS or DOM
 - `<div style="..." id="..."> ... </div>` a division
 - ` ... ` small chunks of inline text

Sample webpage

A webpage

1 2 3
4 5 6
7 8 9

First paragraph

Some text. Some more text.

A lot more text is needed to see word-wrap. A [link](#).

Google

A SECOND PARAGRAPH.

1. First things first.
 - o One
 - o Two
2. Second things second.
3. Third things last.

Last modified: Thu Sep 18 17:28:57 MST 2014

Sample webpage

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD
HTML//EN">
<html> <head>
<title>A webpage</title>
</head>
<body>
  <h1>A webpage</h1>

  <table style="border: 1px solid blue">
    <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>
</table>
<tt><h2>First paragraph</h2></tt>
<p>
Some text. Some more text. <br> A lot more text is
needed to see
word-wrap. A <a href="http://google.com">link</a>.
<br>
<a href="http:google.com"></a>
</p>
```

```
<p style="font-family: bank gothic;font-size: 60;
color:white;background-color:black">

A second paragraph.
<ol>
  <li>First things first.
    <ul>
      <li>One</li><li>Two</li>
    </ul>
  </li>
  <li>Second things second.</li>
  <li>Third things last.</li>
</ol>
</p>

<hr>
<address></address>
<!-- hhmts start -->Last modified: Thu Sep 18
17:28:57 MST 2014 <!-- hhmts end -->
</body> </html>
```

Ungraded Homework Exercise

- Build your own html file.
- Play with the tags
- Embed a picture of yourself encoded in base64