

# **CS6**

## **Practical System Skills**

**Fall 2019 edition**

Leonhard Spiegelberg  
[lspiegel@cs.brown.edu](mailto:lspiegel@cs.brown.edu)



## 11.99 Recap

---

**Last lecture:** Regular expressions + use in `grep/sed/awk`

- . match one arbitrary character
- \* 0 or more matches of preceding item
- + 1 or more matches of preceding item
- ? 0 or 1 match of preceding item (optional)
- (...) capturing group
- {m} match exactly m times
- {m,n} match at least m, at most n times
- [...] define character class

# Recap Quiz - let's develop a regex!

---

Regex task	Examples	should not match
match hexadecimal RGB colors	#123456 #abDE87 #001200	123456 #XYabcd #ghabcd
match non-negative integers 0-999 without leading zeros	0 1 98 999	1000 02 003

⇒ please collaborate & use e.g. [regex101.com](https://regex101.com)!

# Recap Quiz - let's develop a regex!

---

Regex task	Solution (not unique per se)
match hexadecimal RGB colors	<code># [0-9a-fA-F] {6}</code>
integers 0-999 without leading zeros	<code>^([0-9] [1-9][0-9]{0,2})\$ or ^0\$ ^[1-9][0-9]{0,2}\$</code>

⇒ please collaborate & use e.g. [regex101.com](https://regex101.com)!

⇒ BRE or ERE syntax is both fine. (Use ERE, it's easier...)

## Another regex exercise

---

Can you develop a regex to match numbers 0-255?

yes! break up into ranges.  $\Rightarrow$  Useful e.g. for IPv4 addresses

Solution:

```
^([0-9]| [1-9][0-9]| 1[0-9]{0,2}| 25[0-5]| 2[0-4][0-9])$
```

$\Rightarrow$  Good idea to have semantic logic in regex?

# Practical example: regex

---

Fetching lecture slides via grep/sed/curl:

```
curl -s https://cs.brown.edu/courses/csci0060/lectures.html | grep -Eo  
'href=".*\.pdf' | sed 's/href="\.// ' | sed  
's|^|https://cs.brown.edu/courses/csci0060|' | xargs -n1 curl -O
```

Try to fetch lecture slides from other courses:

E.g., <https://web.stanford.edu/class/cs142/>, <http://cs.brown.edu/courses/cs1951g/>,...

# 12 Websites & HTML

**CS6** Practical System Skills

Fall 2019

Leonhard Spiegelberg *[lspiegel@cs.brown.edu](mailto:lspiegel@cs.brown.edu)*

## 12.00 Resources

---

⇒ There are many resources to learn web development online, e.g.

<https://www.w3schools.com/>

<https://www.codecademy.com/learn/learn-html>

<https://education.github.com/pack>

⇒ Slides on HTML/CSS are based on Jon Duckett's HTML&CSS

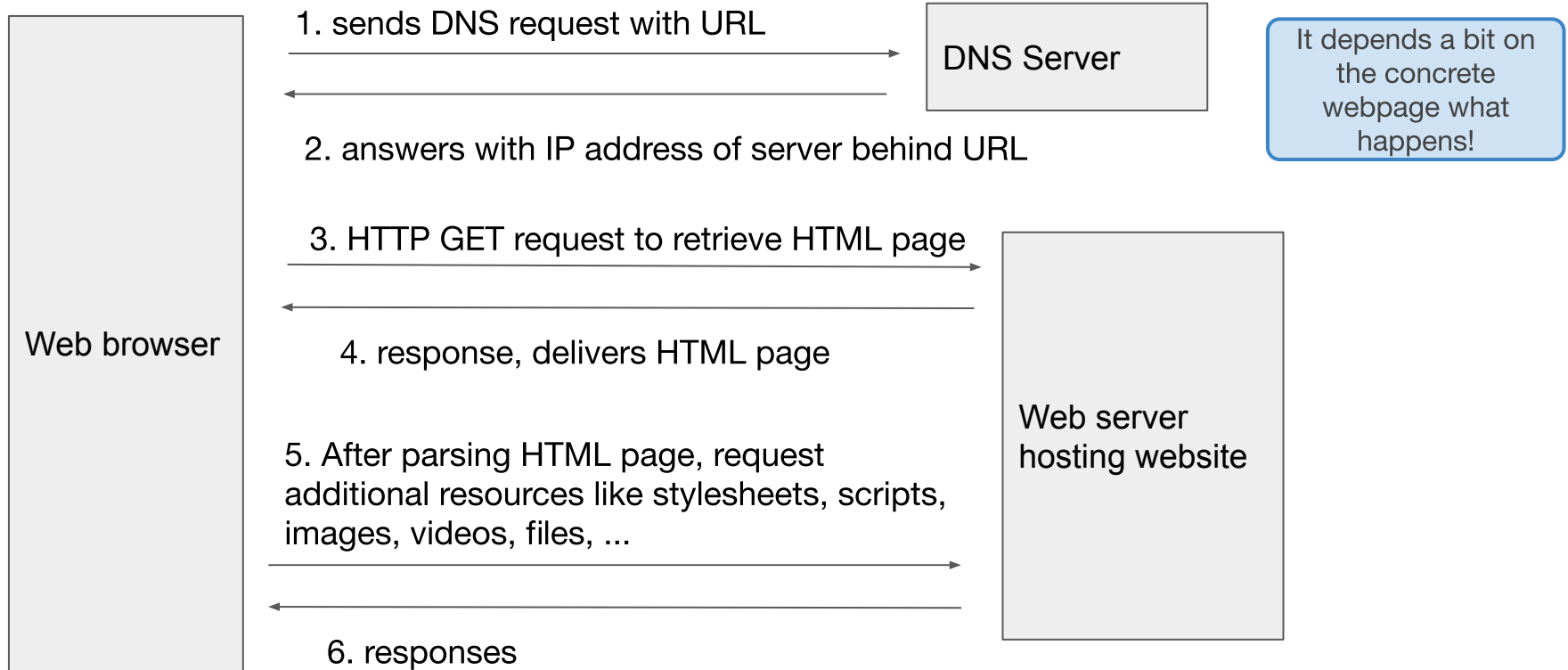
→ Chapter 1-7

→ pp. 193-194 (escape characters in HTML)

→ Chapter 10-12 and Chapter 17



# 12.01 What happens when you access a webpage?



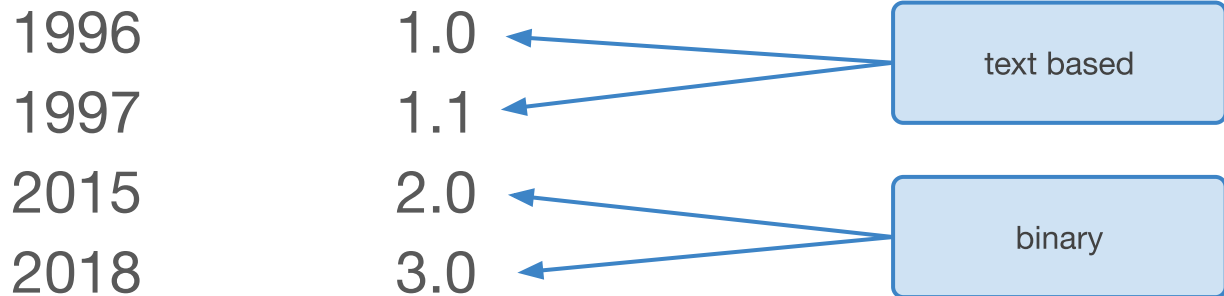
## 12.02 HTTP requests

---

⇒ base of accessing a page and resources are HTTP requests, full documentation available under <https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html>

⇒ a HTTP request is a message send via TCP over port 80.

⇒ There are multiple versions:



HTTP/3 will be coming 2019!

# 12.02 HTTP request - example

HTTP GET request for course website

GET /courses/csci0060/index.html HTTP/1.1

Host: cs.brown.edu

Connection: keep-alive

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_14\_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90

Safari/537.36

Accept: text/html

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en

first specifies type (GET),  
resource  
(/courses/csci0060/index.html)  
and protocol version

request header fields

empty line, after it an optional  
message body may be included

## 12.02 General structure of a HTTP/1.1 request

---

request message consists of

1. request line (e.g. GET / HTTP/1.1)
2. (optional) request header fields
3. empty line
4. (optional) message body

⇒ what happens after a request is sent?  
→ The server responds (hopefully)!

## 12.02 HTTP responses

---

⇒ The server to which the request was sent to, answers with a response. Similar to a request the response message consists of:

1. status line including *status code* and *status message*

e.g. HTTP/1.1. 200 OK

2. response header fields

3. empty line

4. optional message body

### Example response message

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 10 Oct 2019 13:04:41 GMT
Content-Type: text/html
Content-Length: 2084
Connection: keep-alive
Last-Modified: Tue, 08 Oct 2019 20:41:58 GMT
ETag: "15fa-5946c3317148d-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
```

## 12.02 HTTP requests

---

There are multiple kinds of HTTP requests:

GET	read request, i.e. retrieve specified resource. Should not have side effects.
HEAD	similar to GET, but only delivers headers back. I.e. no message body (data is not delivered)
POST	write request, i.e. post data to server to store/process...
PUT	store data under supplied URI, i.e. for file upload
DELETE	delete data under supplied URI, i.e. for file deletion

## 12.03 Diving into the web

---

⇒ You can use Chrome → Help → Developer → Developer Tools to access tools to trace how a webpage is loaded

⇒ Perform a hard refresh to see resources loaded/requests in the network header.

→ DEMO

## 12.03 Curl

---

- ⇒ unpractical to write requests by hand, thus tool to make life easier:
  - curl (or wget) to perform requests for you!
- ⇒ cURL is a widespread tool to interact with URLs.
  - <https://curl.haxx.se/>
- ⇒ can be used to download files using multiple protocols, but also to issue HTTP requests!
  - many options to set headers, message, request type, ...



## 12.03 Curl - GET request

---

curl performs per default a GET request

→ detailed documentation under <https://curl.haxx.se/docs/httpscripting.html>

→ use `-v` to understand what curl does, or `--trace-ascii logfile.txt`

There are several options to perform HEAD or POST requests:

<code>-I</code>	<code>--head</code>	performs HEAD request
<code>-d data</code>	<code>--data data</code>	performs POST request with data as message body
<code>-L</code>	<code>--location</code>	sometimes servers redirect pages (i.e. you'll receive a 3XX response), use this option to let curl follow the redirect.

## 12.03 More on cURL

---

⇒ you can also explicitly specify HTTP request details in cURL, e.g. via

```
curl --data "" --header "Accept: text/html" --request GET cs.brown.edu
```

⇒ for now: curl to retrieve web pages / resources

⇒ Later in CS6: Filling out web forms & testing REST APIs

**Note:**

Sometimes curl doesn't deliver something, because the webserver will only reveal a page to a web browser. To remedy this, add necessary headers to your request! E.g., <https://developers.whatismybrowser.com/useragents/explore/> to find a user agent.

## 12.03 Curl, saving output to files

---

⇒ curl write output per default to stdout

→ i.e., could use stdout redirection to save file.

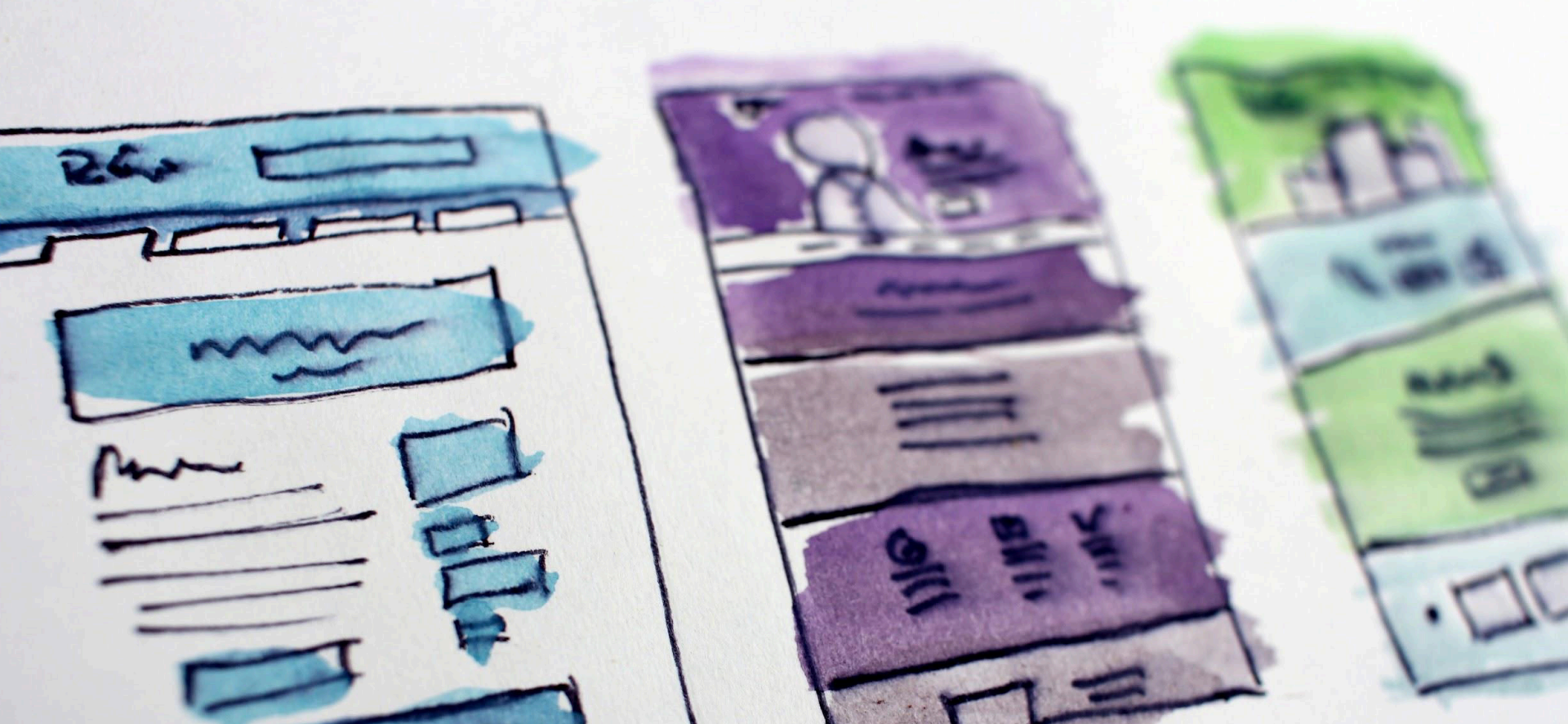
⇒ per default curl shows download progress, disable via `-s`

⇒ use `-O` to use last part of url as filename to save to.

⇒ Alternative: use `-o filename` to specify a filename to save to  
(`-O` - will write to stdout).

Example:

```
curl -L cs.brown.edu/courses/csci0060/assets/slides/slides9.pdf -o lecture09.pdf
```



Writing a webpage from scratch...

## 12.04 HTML

---

HTML = Hypertext Markup Language

hypertext: text which contains links to other hypertext documents

markup: annotations to structure content

⇒ HTML is the format to ship webpages.

⇒ HTML pages are text documents.

⇒ There are multiple versions of HTML, in this course: HTML5

## 12.04 HTML tags

---

- ⇒ HTML consists of a series of (nested) elements.
- ⇒ HTML elements tell the browser how to display the content.
- ⇒ HTML elements are represented by tags.
- ⇒ HTML tags usually come in pairs with an opening tag `<tag>` and a closing tag `</tag>`, the content is enclosed by these two tags.

### **Example:**

```
<p>Some text here...</p>
```

## 12.04 HTML tags - continued

---

Some tags don't have a closing tag, their syntax is simply `<tag>`.

⇒ In HTML5 they are known as void elements (don't try to close them with `</tag>!`), as they must not contain any content.

⇒ tags may have attributes, syntax is:

```
<tag attribute="value"></tag>
```

**Example (tag with several attributes):**

```
<div class="blue" id="main" width="100px"></div>
```

## 12.04 HTML tags

---

⇒ HTML tag names and attribute names are case-insensitive **but** attribute values (and content) are case sensitive.

→ convention: lower case tag names and attribute names

### Example:

`<p>some text</p>` is the same as `<P>some text</P>`  
but `<p>SOME text</p>` is different than `<p>some text</p>`.

`<div class="HELLO"></div>` is the same as  
`<DIV Class="HELLO"></DiV>` but different than  
`<div class="hello"></div>`




## 12.04 HTML structure & comments

---

⇒ Structure of HTML document is defined by nesting tags, e.g.

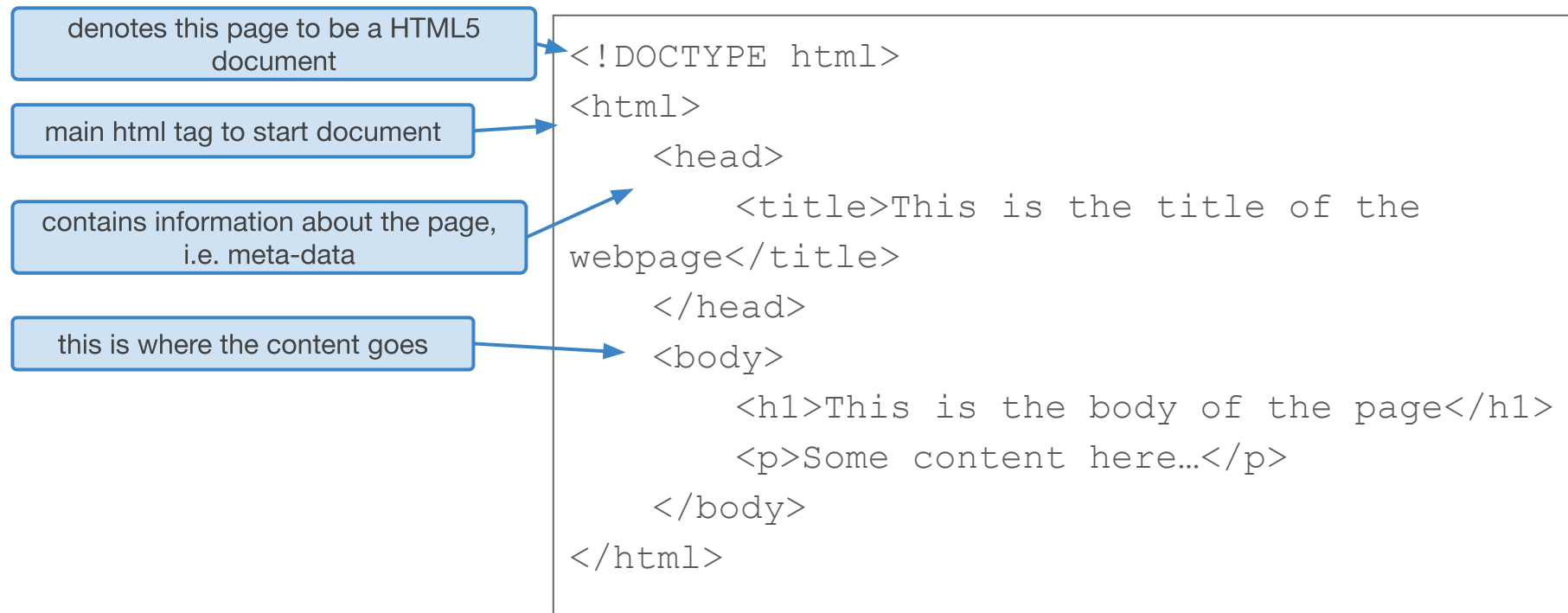
```
<outerTag>  
  <innerTag>  
    text  
  </innerTag>  
</outerTag>
```



HTML can be formatted like this,  
many editors actually provide  
autoformatting functionality for  
webpages

⇒ comments can be added via `<!-- comment goes here -->`

## 12.04 Basic HTML webpage structure



⇒ save as file with `html` or `htm` extension and can open it via a web browser!

## 12.04 Tools for HTML development

---

⇒ a text editor is sufficient, however to be more productive you ideally want one with live reload functionality (i.e. displays changes without manual refresh).

⇒ There are many editors out there:

E.g. `brackets.io`, Sublime Text, Atom.io, WebStorm,  
Adobe Dreamweaver, ...

⇒ Also many webtools provide live development with a live preview:

`codepen.io`, `playcode.io`, `scratchpad.io`, ...



Welcome to the  
world of HTML  
tags...

# 1. Text manipulation tags

## 12.05 Text tags

---

⇒ Use `h1`, ..., `h6` to define headings, e.g. `<h1>Heading</h1>`

⇒ `<p></p>` defines a paragraph, i.e. to put words into a paragraph enclose them with a `p` tag.

⇒ For changing the text format, the following tags can be used

`<b>bold text</b>`

`<!-- bold text -->`

`<i>italic text</i>`

`<!-- italic text -->`

`E=MC<sup>2</sup>`

`<!-- superscript -->`

`H<sub>2</sub><sub>O`

`<!-- subscript -->`

## 12.05 line breaks & rules

---

⇒ use `<br>` to add a line break

⇒ `<hr>` will create a horizontal rule

⇒ both of these elements are void elements, as they do not contain any content and thus must not be closed!



note the whitespace

**Note:** You'll often see `<br />` or `<hr />` too, the reason for this is because that this syntax is XHTML (an older HTML version) compatible. If you're only targeting HTML5, use `<br>` and `<hr>`.

## 12.05 Semantic text tags

---

⇒ `strong` is used to strongly emphasize content, `em` to emphasize content (by default displayed as bold or italic text).

⇒ `blockquote` can be used for a long quote, `q` for an inline quote (`blockquote` is per default displayed with an indent, `q` surrounds content with "").

⇒ `abbr` can be used to introduce an abbreviation, its title attribute is displayed when hovering over it.

### Example:

```
<abbr title="National Aeronautics and Space Administration">NASA</abbr>
```

⇒ There are many more semantic tags in HTML5 available.



## 2. Lists

## 12.06 lists

---

⇒ general structure to define a list to have one enclosing tag which specifies the list type, i.e.

- ol      ordered list
- ul      unordered list

and then items enclosed by a li (list item) tag.

Example:

```
<ul>  
  <li>Item 1</li>  
  <li>Item 2</li>  
</ul>
```

## 3. Links

## 12.07 Anchor tags


---

⇒ Links are created via anchor tags `<a></a>`

⇒ can be used to link to another anchor tag within the same page or to another page

Basic syntax:

```
<a href="http://cs.brown.edu">CS Department@Brown</a>
```



hyper text reference, i.e. the link to follow when clicking on the displayed text



text that is displayed

## 12.07 Anchor tags - linking to other pages

---

⇒ instead of fully qualified URLs, one can use absolute or relative URLs

⇒ Absolute URL: similar to absolute paths starts with /

→ / is the site root, not the root dir of the host!

⇒ relative URL: work like relative paths, e.g.

```
<a href=" ../music/top50.html>Billboard Top50</a>
```

links to page in `../music` folder.

## 12.07 Anchor tags - linking to the same page

---

⇒ Any element in HTML can have an id attribute assigned to

### **Example:**

```
<h1 id="first-main-heading">Heading</h1>
```

⇒ to link to a specific part of the page, reference the id with #, i.e.

```
<a href="#first-main-heading">go to first heading</a>
```

## 12.07 Anchor tags - continued

---

⇒ `<a href="mailto:lspiegel@cs.brown.edu:">Email Leonhard</a>`  
starts user's email program

⇒ with target attribute, the destination where links are opened  
is specified, i.e.

<code>_blank</code>	open link in new window
<code>_self</code>	open link in same frame it was clicked (default)
<code>_parent</code>	open link in the parent frame

## 4. Images



## 12.08 including images

---

⇒ images can be include using `<img>` tag (void element! no closing tag!)

⇒ typically, images are stored in a separete folder `img/` or `images/` at the site root.

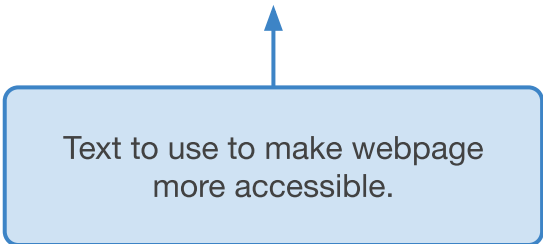
⇒ General syntax:

```

```



URL to image source



Text to use to make webpage  
more accessible.

## 12.08 images continued

---

⇒ attributes height and width can be used to specify image display dimensions

→ you can use different units to specify the value, e.g.

`width="100"` or `width="100px"` or `width="20%"`

⇒ images can be aligned using the align attribute, e.g. when placing an image within a paragraph tag (`<p></p>`) with align attribute set, text flows around it.

→ use `align="left"` or `align="right"` for horizontal alignment

→ use `align="top"`, `align="middle"` or `align="bottom"` for vertical.

## 5. Tables

## 12.09 HTML tables

---

⇒ use `<table></table>` to define a new table

⇒ a row is defined via `<tr></tr>`

→ to define individual table cells, use `<td></td>`

⇒ to define a header row, use instead of `td`, `th` tag.

⇒ with `rowspan` or `colspan` attribute, a table cell can span multiple rows or columns respectively

## 12.09 HTML tables

---

- ⇒ header row is usually enclosed by a thead tag
- ⇒ body using a tbody tag.
- ⇒ Many browser automatically add these tags
- ⇒ optionally, you may add a tfoot tag for a footer row.
- ⇒ There are tools to generate tables fast, e.g.

<https://www.rapidtables.com/web/tools/html-table-generator.html>

## 12.09 HTML table - example

---

```
<table>
<thead>
  <tr>
    <th>country</th>
    <th>capital</th>
  </tr>
</thead>
<tbody>
  <tr>
    <td>U.S.A</td><td>Washington, D.C.</td>
  </tr>
  <tr>
    <td>Canada</td><td>Ottawa</td>
  </tr>
</tbody>
</table>
```



country	capital
U.S.A	Washington, D.C.
Canada	Ottawa

## 6. Structural tags

## 12.10 Structural tags

---

⇒ HTML5 introduced many new tags to structure the content of a website better (cf. Chapter 17, HTML&CSS)

⇒ use the `nav` tag together with a `ul` tag to define a navigation

⇒ parts of the website can be structured using the `section` tag

⇒ for a header/footer HTML5 provides a `header` and `footer` tag



CSS = Cascading Stylesheets

## 12.11 CSS

---

⇒ Cascading stylesheets(CSS) allow to change the appearance of a website, they're written in their own language.

⇒ Three options to use CSS:

1. write CSS code enclosed by `<style type="text/css"></style>` tag in head section
2. write CSS code in separate file and link it via a link tag within the webpage  
`<link href="css/style.css" type="text/css" rel="stylesheet">`
3. write CSS rules into style attribute, e.g.  
`<p style="color: #ccc; background-color: #000;">...</p>`

## 12.11 Basic CSS syntax

---

CSS selector

→ `p {  
 font-family: Arial;  
 color: blue;  
}`

Declarations, ; terminated.

⇒ declarations are of the form `property: value`

⇒ the selector defines to which elements the declarations should be applied.

## 12.11 Basic CSS selectors

---

⇒ each element in HTML can have a class attribute.

→ used to apply a "style" (i.e. a CSS rule) to multiple elements

→ id attribute is used to uniquely identify an element

Full list of CSS selectors e.g. under [https://www.w3schools.com/cssref/css\\_selectors.asp](https://www.w3schools.com/cssref/css_selectors.asp).

`tag { ... }`

apply rules to all `<tag>` elements

`#name { ... }`

apply rules to all `<tag id="name">` elements

`.name { ... }`

apply rules to all `<tag class="name">` elements

## 12.11 Basic CSS selectors

---

⇒ Selectors can be combined, e.g.

`h1, h2, h3 { ... }`

apply rule to h1, h2 and h3 tags

`p a { ... }`

apply rule to all `<a>` which are enclosed by `<p>`  
(can be multiple levels up)

`p>a { ... }`

apply rule to all `<a>` which are children of `<p>`,  
i.e. `<p><a></a><p>` but not  
`<p><div><a></a></div></p>`

`p.note`

target only `p` tags which have class attribute `note`

## 12.11 Basic CSS selectors

---

`p#name` targets `p` tag which has id attribute with value `name`.

⇒ typically only one identifier is assigned to the id field, but one can assign multiple classes to the class attribute!

### **Example:**

```
<p class="text text-justify">...</p>
```

## 12.11 CSS properties

---

⇒ there are many properties (depending on tag type) that can be set.

⇒ A complete list can be found e.g. here <https://www.w3schools.com/cssref/>

## 12.11 CSS

---

⇒ Learning CSS requires some time

⇒ Many resources online available

**Next lecture:** More on CSS & practical web dev tricks!



# 13 Midterm

**CS6** Practical System Skills

Fall 2019

Leonhard Spiegelberg *[lspiegel@cs.brown.edu](mailto:lspiegel@cs.brown.edu)*

## 13.01 Logistics

---

⇒ **10/22 in CIT477, 4pm** → **counts 15%**

⇒ You have **90min**, however the exam will be designed for 60-70min.

⇒ paper based, if you need special accommodations please reach out!

⇒ you're allowed to bring one cheatsheet, i.e. one piece of US-letter with your own personal notes

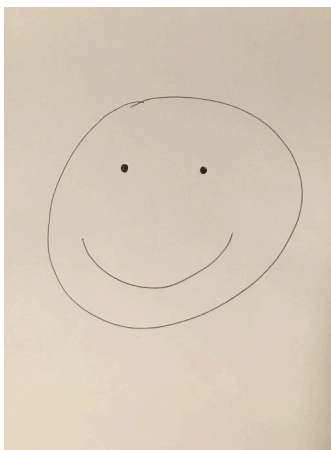
⇒ questions are a mix of multiple choice, and part answers

→ you might be required to write some small scripts,  
however solutions should not exceed 10 lines. Often there's a  
one-liner.

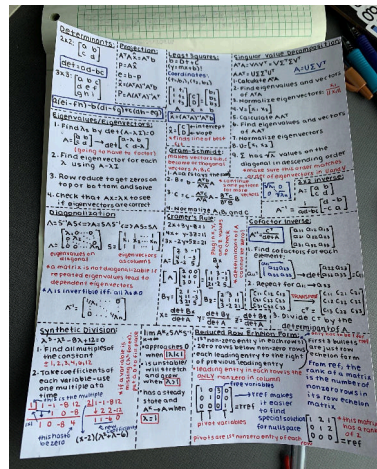
# 13.02 Cheatsheets

Each cheatsheet must contain your **own** personal, handwritten notes.

It is ok to collaborate on the content of a cheatsheet,  
i.e. preparing together for a midterm.



optimistic  
approach



defensive approach

# Content

---

⇒ Lectures 1-10 (today).

⇒ Focus on HOW to use commands, use your cheatheets for notes on options/commands.

⇒ Specific topics of interest:

**file paths, file permissions, streams, pipes, brace expansion, parameter expansion, arithmetic expansion, quoting, SSH + key setup, processes(especially signals), regular expressions&string processing, Basic HTML & everything else in homeworks + slides.**

# How to prepare?

---

- ⇒ Make sure you did all homeworks
- ⇒ make sure you can solve quizzes in recap sections.
- ⇒ write your cheatsheet
- ⇒ practice on permissions, regular expressions and piping commands
- ⇒ go to office hours & get help from your TAs!

# Example question I

---

4. (2 points) You are in `/home/ec2-user/data` and running `ls` gives you

```
1      2      3      4      5      6      7      8      9     10     11    123 results
```

Each entry might refer to a file or a directory. You want to put all these entries (except `results`) into the folder `results`. Write a single command to achieve this (2P): If you don't know how to achieve this with a single command, you may use up to 3 commands but you will only get 1P at most.

Solution: Can be done in multiple ways, e.g. `mv {?, ??, ???} results/`

# Example question II

---

**Problem:** You're given the following output (stdout) from `curl cs.brown.edu/cs1951g`  
Write a script (or command) to download all pdf files!

```
<!DOCTYPE html> <html >
  <body>
    <div id="maincontent">
      <h2>Slides</h2>
      
      <p>The slides are posted after the material has been covered in class.</p>
      <ul>
        <li>(1/26) <a href="slides/00-Intro.pdf">00-Intro</a></li>
        <li>(2/2) <a href="slides/01-LPTheory.pdf">01-LPTheory</a> (complete)</li>
        <li>(2/9) <a href="slides/03-IPTheory.pdf">03-IPTheory</a> (complete)</li>
        <li>(2/23) <a href="slides/04-IndexFunds.pdf">04-IndexFunds</a></li>
        <li>(3/2) <a href="slides/05-ConvexTheory.pdf">05-ConvexTheory</a></li>
        <li>(3/2) <a href="slides/05b-DualityKKT.pdf">05b-DualityKKT</a></li>
        <li>(3/9) <a href="slides/06-Unconstrained.pdf">06-Unconstrained</a></li>
        <li>(3/16) <a href="slides/07-PortfolioOptimization.pdf">07-PortfolioOptimization</a></li>
      </ul>
    </div> <!-- end of maincontent →
  </body>
</html>
```

# Example question II - solution

---

## solution:

```
curl cs.brown.edu/cs1951g | grep -Eo "slides/.*\.pdf" | sed
's|^|cs.brown.edu/cs1951g/|' | xargs -n1 curl -O
```

```
<!DOCTYPE html> <html >
  <body>
    <div id="maincontent">
      <h2>Slides</h2>
      
      <p>The slides are posted after the material has been covered in class.</p>
      <ul>
        <li>(1/26) <a href="slides/00-Intro.pdf">00-Intro</a></li>
        <li>(2/2) <a href="slides/01-LPTheory.pdf">01-LPTheory</a> (complete)</li>
        <li>(2/9) <a href="slides/03-IPTheory.pdf">03-IPTheory</a> (complete)</li>
        <li>(2/23) <a href="slides/04-IndexFunds.pdf">04-IndexFunds</a></li>
        <li>(3/2) <a href="slides/05-ConvexTheory.pdf">05-ConvexTheory</a></li>
        <li>(3/2) <a href="slides/05b-DualityKKT.pdf">05b-DualityKKT</a></li>
        <li>(3/9) <a href="slides/06-Unconstrained.pdf">06-Unconstrained</a></li>
        <li>(3/16) <a href="slides/07-PortfolioOptimization.pdf">07-PortfolioOptimization</a></li>
      </ul>
    </div> <!-- end of maincontent →
  </body>
</html>
```



# End of lecture.

Next class: Tue, 4pm-5:20pm @ CIT 477