Week 14

Bring laptop with abilty to remote connect to best-linux.cs.wisc.edu to class

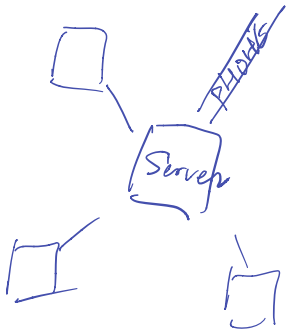
Final Exam

- **Friday, May 10th, 5:05 pm to 7:05 pm**
  - **Lecture 001: Room 1125 of Biochemistry Building**
  - **Lecture 002: Room B10 of Ingraham Hall**
  - **Lecture 004: Room B102 of Van Vleck Building**
- **UW ID required**
- **Makeup exam emails sent**
- **See posted exam information (exam topics and latest "news")**
  https://pages.cs.wisc.edu/~deppeler/cs400/exams
- **TA availability as regularly scheduled through Thursday 5/2**

Verify that your scores are correctly entered on Canvas.
Send email to instructor if there is an inconsistency.

Complete p6 before 10pm on Friday. Lowest program score will be dropped.

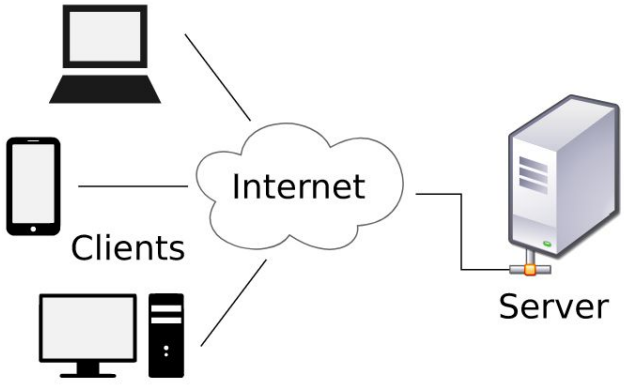
Team Project:
(A-Team 100pts) Milestone #3: due before 10pm Thursday 5/2
Follow submission instructions to create executable.jar and food.zip.
Submit team.zip (it contains source, screenshots, and executable.jar file)
See Milestone #3 rubric (linked in separate document from milestone #3)



Read: Module 14 topics

THIS WEEK:

- HTML5/CSS/JS
  - Basic HTML elements
  - HTML5 Semantic web
- Responsive Web Design (RWD)
- HTML/JS Forms
- Chrome Developer Tools
- Web Frameworks
- JavaScript (a little bit in exam)



- Course Evaluations
  https://aefis.wisc.edu
- Exam Review:
  - Complete last quiz
  - Post questions (that are not on the quiz) that we can answer and review in class

git clone http://github.com/cs400-deppeler/html-css-js-demo.git

HTML

- Hyper Text Markup Language
- content is wrapped in tags
- web Browser interpret and display html files.
- W3C - world wide web consortium

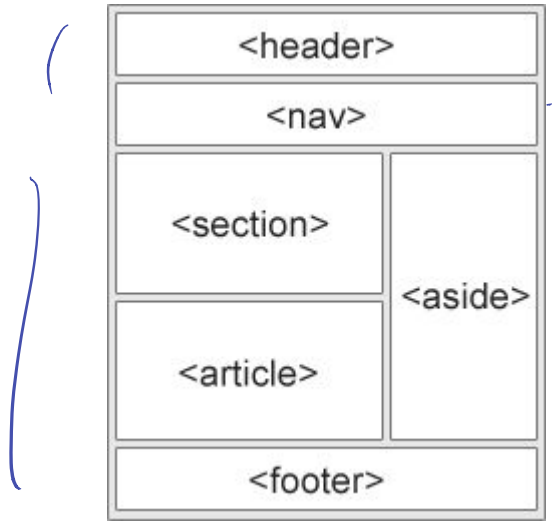
Place a paragraph's text content here.

Basic HTML Elements

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Exams</title>
    <link rel="stylesheet" href="styles.css">
    <script src="myscript.js">
  </head>

  <body>
    <header><h1>Exam Information</h1></header>

    <!-- display course calendar -->
    <img src = "spring_exam_calendar.png"
         alt = "March and May calendars with 3/15 and 5/11 shaded" />
    <table>
      <tr><th>Midterm</th><td>5-7 PM</td><td>October 25th</td></tr>
      <tr><th>Final</th><td>7:45AM - 9:45AM</td><td>Dec 15th</td></tr>
    </table>

    <h2>Topics</h2>
    <ul>
      <li><b>Java 8</b>: functional interfaces, lambda expressions</li>
      <li><b>Java FX</b>: Label, Button, BorderPane, Scene, Stage</li>
    </ul>

    <dl>
        <dt>Term</dt> <dd>definition</dd>
    </dl>

    <footer>
      <p><a href="home.html">Home Page</a> <br>
      Report broken links and accessibility problems to: deppeler
    </footer>

  </body>
</html>
```

HTML5 Semantic Web

Tag	Description
	used to manage a section or block of tags as a unit
	used to manage a portion of a text tag as a unit
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like: illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time



CSS

- list of rules
- promaties across pages
- define style inline, at top of of doc in <head> ,or included doc.

```
body {
    font-family: Verdana,sans-serif;
    font-size: 0.9em;
}

header, footer {
    padding: 10px;
    color: white;
    background-color: black;
}

section {
    margin: 5px;
    padding: 10px;
    background-color: lightgrey;
}

article {
    margin: 5px;
    padding: 10px;
    background-color: white;
}

nav ul {
    padding: 0;
}

nav ul li {
    display: inline;
    margin: 5px;
}
```

1. Apply external
2. Apply top of doc
3. Apply in-line

text

Try It!

Create your own web page on the CS web server and view it from the World Wide Web

1. Remote connect to best-linux.cs.wisc.edu
2. `cd ~/public/html`
3. `mkdir demo`
4. `cd demo`
5. `vi home_page.html`
6. type i to enter ***insert*** mode in vi editor
7. type in the following HTML

```
<!DOCTYPE html>
<html>
<head><title>About Me</title></head>
<body>
  <header><h1>About Me</h1></header>
  <p>write something about yourself
  fact or fiction is fine – but not too personal for the W3</p>
  <img src="myphoto.png" alt="no photo yet" width="100" height="50" />
  <footer>Created on 12/3/2018</footer>
</body>
</html>
```

8. add any or all of the following
   a. h2, list, table
   b. create a css page
   c. link the css page to your web page
9. save your file
   a. typing ESC to exit insert mode
   b. type **:w** to write your file to disk; or type **ZZ** to **save file and exit vi editor**
10. view your web page from the Internet (your web browser)
    http://pages.cs.wisc.edu/~cslogin/demo/home_page.html

Example html files available at: https://pages.cs.wisc.edu/~deppeler/cs400/html

git clone https://github.com/cs400-deppeler/html-css-js-demo.git

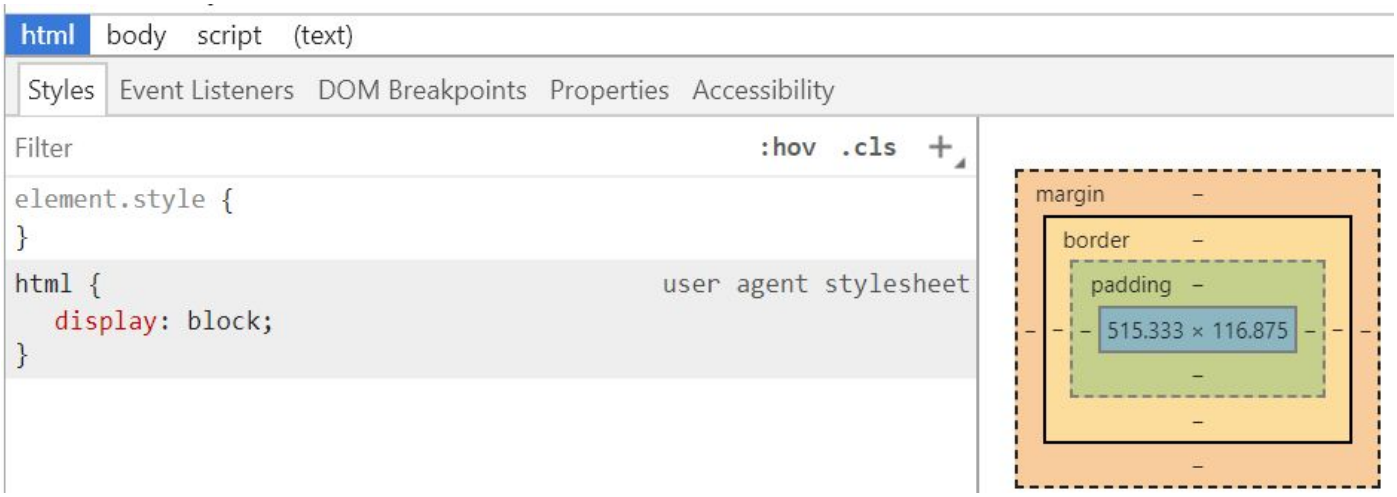
Chrome Developer Tools

https://developers.google.com/web/tools/chrome-devtools/

1. click Customize and control Google Chrome icon
2. mouse over More tools
3. click Developer Tools

Developer tools allow you to:

- view and change page styles
- debug JavaScript
- view console message
- analyze performance



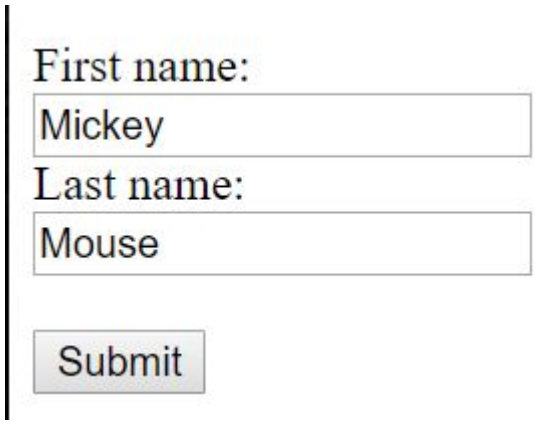
Web Forms

- user a form to get and process user input
- action attribute tells what to do

start → script, get post, where to post results.

```
<form action="/action_page.php" method="get" target="_self">
  First name:<br><input type="text" name="firstname" value="Mickey"><br>
  Last name:<br> <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

end



Submit Methods

Notes on GET method: (from w3c)

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user want to bookmark the result
- GET is better for non-secure data, like query strings in Google

Notes on POST method:

- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked
- Use POST if the form data contains sensitive or personal information.
- POST does not display info in address

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_submit_id
https://www.w3schools.com/php/default.asp // php tutorial

JavaScript

- scripting language . interpreted by web browser
- allows dynamic content
- syntax similar to java

Reference

Strings

```
"JavaScript".length
.toUpperCase()
.toLowerCase()
.trim()
```

Numbers

```
var x = 1 + 0.5;
x = 1 + 1/2;
x += 3.14 * 2 + (-7 -3);
```

Booleans

```
var lightOn = true;
lightOn = false;
```

Operators

```
+ - * / ( ) =    ==    >=    >    <    <=    !-
```

Variables

```
var x = 3 ;
var y = x + 3 ;
x = 5 ;
```

- what is the value of y => 6

Conditionals

```
if ( x > y ) {

} else {

}
```

Arrays

```
var a = [ 10, [ "hello", "good-bye" ], d ]

var e = a[0]; // a[-1]
a[ 2 ] = new String(3)+ +;

a.length
a.concat(b)
a.pop()
a.push()
a.reverse();
```

Objects

contain other objects

"curly brace"
"key: value pair" ← dictionary

```
var s1 = { name: "Deb", pets: [ "Bo", "Kiwi" ], course: "cs400" };
s1.pets[0]     // "Bo"
s1["pets"][1]  // "Kiwi"


s1["greeting"] = "hi"
```

Console

use to display information to the console window (instead of browser page)

```
console.log( s1.pets[0] );
```

Click Me Button

```
<button id="click" onclick="clickMe()">Click Me</button><br>
<label id="tally"> </label>
<p id="count"> </p>
<script lang="JavaScrip">
  var n = 0;
  function clickMe() {
    n += 1;
    var button = document.getElementById("click");
    var tally = document.getElementById("tally");
    var count = document.getElementById("count");
    if ( button.innerHTML == "Click Me" ) {
      button.innerHTML = "Don't Click Me";
      tally.innerHTML += "1" ; <!-- up arrow "&uarr;"; -->
    } else {
      button.innerHTML = "Click Me";
      tally.innerHTML += "0" ; <!-- down arrow "&darr;"; -->
    }
    count.innerHTML = "" + n;
  }
</script>
```

Log Button

```
<button id="clear" onclick="clearLog()">Clear Log</button><br>
<input type="text" id="msg" onchange="log(this.value);" /><br>
<p id="log"></p>

<script>
  var logString = "";

  // add date -> msg to the logString
  function log(msg) {
    if (msg=="") return;
    var date = new Date();
    var logmsg = date.getMonth() + "/" + date.getDate() + "/"
               + date.getFullYear() + " -> " + msg;
    logString += logmsg + "<br>";
    document.getElementById("log").innerHTML = logString;
  }

  function clearLog() {
    logString = "";
    document.getElementById("log").innerHTML = logString;
  }
</script>
```

FYI ONLY

Web Frameworks

- Provide a library of styles and JavaScript that are standard and ready to use
- Can download the frameworks or link to the required files online

https://www.w3schools.com/bootstrap/default.asp
http://jquerymobile.com/

Responsive Web Design (RWD)

- Download (and install or extract) style sheets
- Include meta tags to read CSS information from those style sheets (don't edit these)
- Create your own customized stylesheets
- Include links to your style sheets (that override the provided styles as desired)

Popular CSS Style Packages

- Pure: https://unpkg.com/purecss@1.0.0/build/pure-min.css
- Bootstrap: https://getbootstrap.com/
- jQuery Mobile: http://jquerymobile.com/

Media Queries

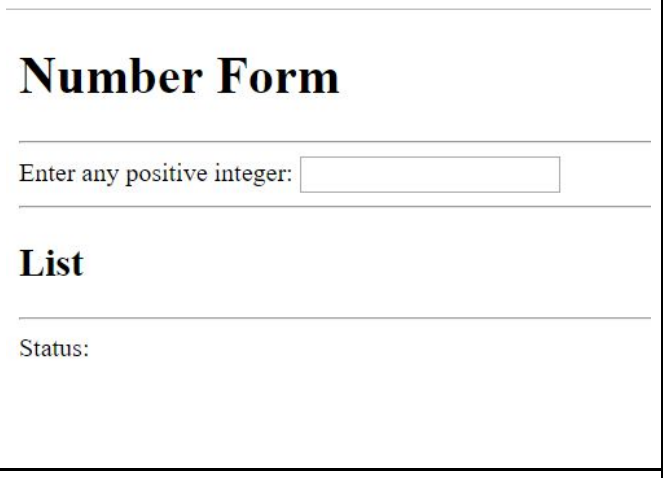
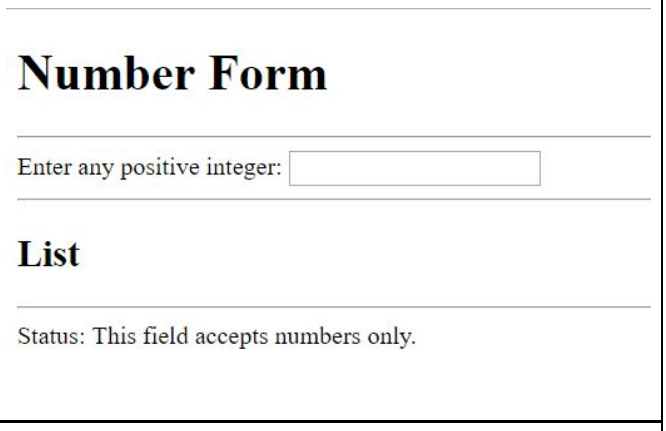
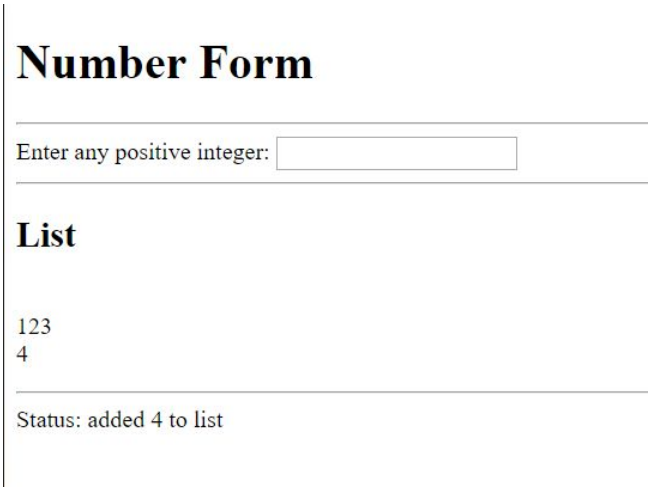
https://www.w3schools.com/css/css_rwd_mediaqueries.asp

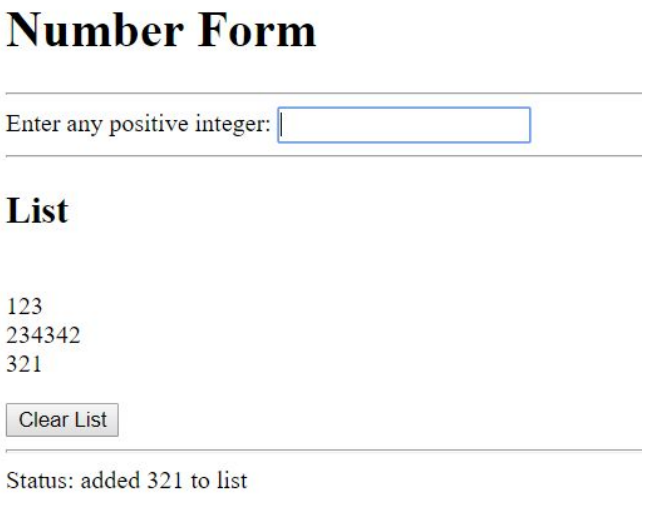
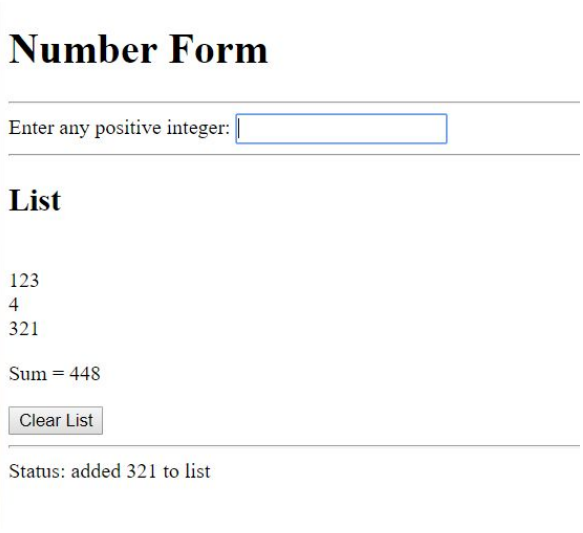
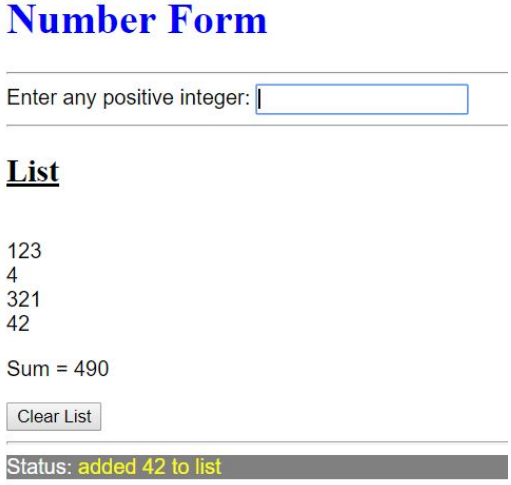
1. Create multiple sets of styles
2. Add code to query what type of device
3. Select the style set accordingly

https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mediaquery_breakpoints

- Design for smallest device first.
- Choose breaks in the content to be on their own "page" based on content.
- Identify where content hits a point where it could adapt to take advantage of a wider width.
- Write media queries' widths in ems, not pixels so layout will adapt to font size changes in addition to screen widths.
- To calculate the width in ems, divide the target width in pixels by default font size in pixels. For example: phone width of (320px) divided by 16px (the default font size) = (20em).
- Use min-width breakpoints that build on top of the mobile styles.
- The first breakpoint applies layout adjustments on top of the standard mobile styles so these can be fairly lightweight.
- Additional min-width breakpoints can be added for even wider screens that each build on the previous breakpoint styles.
- To override framework styles only for smaller screens, use a max-width breakpoint instead. This allows you to constrain your style overrides to only apply below a certain screen width. Above this width, all the normal styles will apply so this is good for certain types of overrides.

NumberForm example

1. create a number form page with an input field and a list paragraph and a status paragraph in the footer https://pages.cs.wisc.edu/~deppeler/cs400/html/listForm/NumberForm_1starterForm.html	
2. add javascript to ignore typed characters that are not positive integers, 0-9 Update status if any such characters are typed in the integer field. https://pages.cs.wisc.edu/~deppeler/cs400/html/listForm/NumberForm_2setStatus.html	
3. add positive integers to the list of values. It should clear the input field when it adds value to the list. https://pages.cs.wisc.edu/~deppeler/cs400/html/listForm/NumberForm_3addToList.html	

4. add button and code to clear the list https://pages.cs.wisc.edu/~deppeler/cs400/html/listForm/NumberForm_4addClearForm..html	
5. add a sum label and code. Clear sum if list is cleared. https://pages.cs.wisc.edu/~deppeler/cs400/html/listForm/NumberForm_5addSum..html	
6. add document css styles in the head section of the document https://pages.cs.wisc.edu/~deppeler/cs400/html/listForm/NumberForm_6addStyles..html	

Course Overview

Working in Teams

member roles: leader, facilitator, scribe, time-keeper,
stages of team development: forming, storming, norming, performing, adjourning
conflict resolution strategies: vote, build consensus, task force, ???
ice-breakers: get to know each other, brainstorm solution idea

Debugging (with and without debugger)

print statements
breakpoints
step-over, step-in, step-out of methods
monitor variable values (must know expected values at each execution step)

Testing

expected vs actual
unit testing vs End To End testing (integration testing), "code coverage"
black box vs white box testing
assert the pre-conditions
test driven development (TDD): write test first, then write code to pass the test
JUnit5 tests: fail(msg), testing exception handling, insert, lookup

Command-Line development (Linux commands and utilities)

pwd, ls, ls -al, cd, cp, mv, rm, mkdir, rmdir, diff, grep, echo
javac, java, jar
pico, nano, vi, emacs

Build Utilities: Make (W11)

how to use make
define basic Makefile rules
~~other options: ant, maven, gradle~~

Version Control: Git/GitHub

init, status, log, clone, pull, add, commit, push,

Object-Oriented Design and Diagrams (W11)

UML Class diagrams (and class summary "table"), inheritance, interfaces, classes
object model diagram, instance diagrams, use-case diagram
memory model (heap vs stack memory management)
scene diagram

Abstract Data Types: ADTs

position-oriented: lists, stacks, queues,
value-oriented: priority queues (heaps), search trees, hash tables,
relation-oriented: graphs, sets,

Trees

terminology: height, root, leaf, interior node, ascendant, descendent, sibling, full
binary search trees: ordering constraint, insert, lookup,
delete from a BST: using inorder-predecessor or inorder-successor
search trees: full, complete, height-balanced, balanced,
AVL: height-balanced, balance factor, rotations
RBT: balanced, root property, red property, black property, tri-node restructure, re-color
B-Tree: full (all leaves at same level), 2-node, 3-node, 4-node, 2-3 tree, keys interior node
B+Tree: full, m-branching factor, linked leaf nodes, good for range searches
maintain balance on insert: detect, fix,
traversals: level-order, pre-order, in-order, post-order

Hash tables

terminology: hash table, table size, load factor, key, value, Java hashCode, hash index
hash functions: int, String, double, other types
collision resolution: linear probe sequence, quadratic probe sequence, double hashing,
buckets: array, "chained", tree
resizing and rehashing
Map types: (key:value pairs) HashMap vs TreeMap

Graphs: GraphNode (W9)

terminology: vertex (node), edge, size(#edges), order(#vertices),
undirected, directed, weighted, connected,
graph vs graphnode, degree of a node
implementations: adjacency matrix, adjacency list, edge list (vertex pairs)
complexity analysis of space and operation time
operations: insert/remove vertices and edges
traversals: depth-first, breadth-first, cycle-detection
spanning trees: simple, depth-first, breadth-first,
minimum spanning trees (MST): Prim's, Kruskals
Dijkstra's Shortest path algorithm: uses priority queue
Topological ordering: requires DAG

Sets (W9/W10)

notation: is a member of, union, intersection
operations: complement, difference, symmetric difference
complexity analysis of various implementation options: array, list, hash table, tree

Sorting (W10)

radix sort: N, radix - range of unique digits, length of sequence - iterations, stable sort
flash sort: classification, permutation, insertion
A, n, Amax, Amin, m, Ai, K(a[i]), create and use boundary array (L)

Functional Programming with Java 8 (W11,W12,W13)

enumerations: define and use, .ordinal()
interfaces: abstract methods, constants, static methods, default methods
functional interfaces: at most one abstract method
java.util.Comparator
inner classes
anonymous inner classes
lambda expressions
Streams: .filter, .map, .collect, .distinct

JavaFX (W9,W10)

graphic user interface (design and layout)
controls: Label, TextField, Button, ScrollBar, ScrollPane, ObservableList,
layout containers: BorderPane, HBox, VBox, GridPane, getChildren().addAll()
Scene
Stage setScene(scene) .show()
Events: setOnAction, setOnMouseEntered, setOnMouseExited

HTML/CSS/JS (W14)

create simple web page
html tags: html, head, title, body, header, footer, p, br, img, ol, ul, dl, table
~~Chrome Developer Tools: Elements, Console, Device view~~
~~css: use of a stylesheet for body, h1, class vs id style rules~~
~~js: var, function, if, else, for, while, return, console.log~~

p3a, p3b, p4, p5, p6 Team Project
~~CourcePlanner: json, graph,~~
QuizGenerator: parse json, JavaFX GUI