Welcome

CS1101S Studio Session Week 13: *Web Development & Final Review*

Niu Yunpeng

niuyunpeng@u.nus.edu

November 13, 2018

Overview

1. *Web development (optional)*
    - Overview
    - Basic knowledge - static page
    - Advance knowledge - dynamic page
2. Information security *(optional)*
    - Web security
3. Final review
    - What we have learned
    - To prepare for final examination

Web Development

Application software

Many engineers have dedicated their career to develop

- Desktop application
- Web application
- Mobile application
- ...

Web Development

What is the trend of this industry?

- ~~Desktop first?~~
- Web first?
- Mobile first?
- ...

Web Development

Answer

- Web first? Yes.
- Mobile first? Yes.
- ...

Result...

- The web goes mobile.

Web Development

How does the Internet work?

- Magic.
- A lot of magic.

Web Development

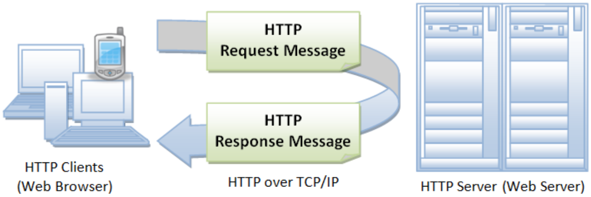
Before we go on...

- URL: Uniform Resource Locator
- HTTP: HyperText Transfer Protocol
- DNS: Domain Name System
- HTML: HyperText Markup Language
- CSS: Cascading Style Sheets
- ...

Web Development

Basic idea

- You enter an URL: request for something by its unqiue identifier.
- Translate using DNS and request reaches the server.
- Find the requested resources on the server and return.
- Render the resources in your browser.



Web Development

From the most basic - static webpage

You have three “weapons” with you as follows:

- HTML(5) for content
- CSS(3) for style
- JavaScript for action

Web Development

Your first webpage

```
<!DOCTYPE html>
<html>
<head>
    <title>My heading</title>
</head>
<body>
    <p>Hello, world!</p>
</body>
</html>
```

Instructions

- Save as ⟨*something*⟩ *.html* locally.
- Open it in the browser.

Web Development

Make it prettier - apply style on elements

Edit and save as ⟨*something*⟩ *.css* locally:

```
p {
  color: #00ffff
}
```

Use the CSS style defined just now

In ⟨*something*⟩ *.html*, modify it to become:

```
<head>
    <title>My heading</title>
    <link rel="stylesheet" type="text/css"
          href="something.css">
</head>
```

Web Development

Embed CSS style in HTML

Edit and save as ⟨*something*⟩ *.html* locally:

```
<!DOCTYPE html>
<html>
<head>
    <title>My heading</title>
</head>
<body>
    <p style="color: red;">Hello, world!</p>
</body>
</html>
```

Web Development

Embed JavaScript in HTML

Edit and save as ⟨*something*⟩ *.html* locally:

```
<!DOCTYPE html>
<html>
<head>
    <title>My heading</title>
</head>
<body>
    <p>Hello, world!</p>
    <button onclick="my_alert();">Click here</button>
    <script>
        function my_alert() {
            alert("Haha. This is interesting!");
        }
    </script>
</body>
</html>
```

Web Development

Dynamic webpage

- Our website should display customized content for every user.
- The return resources may vary even though you enter the same URL.

Solution

- Process the files before giving response to the users.
- Use server-side programming.

Web Development

Thanks to dynamic webpage ...

- The world of Internet has become so colorful.

Web Development

To make a dynamic webpage ...

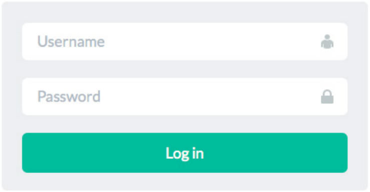
You need to learn the things below:

- Select a server-side programming language: *PHP*, *Ruby*, *Java*, ...
- Select a database engine:
  - SQL database: *MySQL*, *PostgreSQL*, *MariaDB*, ...
  - NoSQL database: *MongoDB*, *Redis*, *Firebase*, ...
- Select a server: *Apache*, *Tomcat*, *Nginx*, ...

Web Development

How to distinguish different users

- Prompt users to sign in when they visit the website.
- Save some informaton in `COOKIE` and `SESSION`.
- Whenever one user enters a URL, display the correct content based on the information stored in `COOKIE` and `SESSION`.
- Clear the `COOKIE` and `SESSION` when the user tries to logout.



Web Development

Recommended modules at SoC

- CS3216 Software Product Engineering for Digital Markets
  - *Paired with CS3217 to offer.*
- CS3226 Web Programming and Applications *(discontinued)*
  - *Last-time offered in AY2016/2017 Semester 2.*

Prestigious opportunity at SoC

- Computing for Voluntary Welfare Organisations (CVWO)

Web Development

What to expect at CVWO

- Build web-based administration system for VWOs (Voluntary Welfare Organisations) in Singapore.
- Get awesome hand-on experiences even since Year 1.
- Give back to the society using your CS knowledge.
- Receive a stipend of $????/month for 3 months.
- Claim 6MCs as SIP (Student Internship Program CP3200).
- Claim CIP hours (for foreign scholars).
- Priority on-campus housing.

Web Development

Last year - CVWO 10th Anniversary

Web Development

To join CVWO in Year 1 summer

- Obtain **really very good** grades in CS1101S, CS2030, CS2040.
- Do one or two assignments during December break.
- Apply in March and wait for inteview in April.
- Show strong programming skills and adequate project experiences.

Web Development

Looks attractive, isn't it?

- Of course!

However ...

- Not easy to get in, very competitive.
- Admit only 8 - 10 top students every year.

Not that hard as well ...

- Last year, two students from my Studio got into CVWO.

Overview

Web security

Web security

Web security

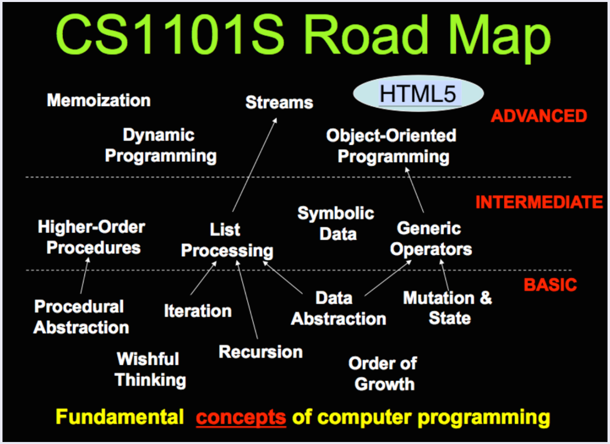
An interesting problem

- Click here `http://cs1101s.azurewebsites.net/web_sec/`

Overview

Final Review

Re-visit the CS1101S roadmap



Final Review

Things we have covered in this semester

- Components of programming language
- Wishful thinking/abstraction
- Recursion/iteration
- Higher-order programming
- Pair/list/tree processing
- Array & loop
- Data structure design
- Memoization/dynamic programming
- Literal object
- Stream
- Meta-circular evaluator
- ...

Final Review

Components of programming language

- Primitives:
  The smallest constituent unit of a programming language.
- Combination:
  Ways to put primitives together.
- Abstraction:
  The method to simplify the messy combinations.
  - To abstract data: use naming;
  - To abstract procedures: use functions.
  - Sometimes, naming and functions are combined together.

Final Review

Wishful thinking/abstraction

To make a good abstraction:

- Modularity:
  Separate multiple steps (and sub-steps).
- Readability:
  Easy for others to read and understand.
- Reusability:
  Provide a generic interface to be used commonly.
- Maintainability:
  Convenient to debug, refactor and deploy.

Final Review

Recursion

- Due to the top-down approach.
- Use substitution model to understand.
- Use pair/list/tree/stream processing.
- May give rise to recursive or iterative call (due to tail recursion).

Iteration

- Due to the buttom-up approach.
- Use environment model to understand.
- Use array and loop.

Final Review

Classical examples of recursion

Can they be solved using iteration as well?

- Factorial
- Square root
- Power function
- Fibonacci
- Greatest common divisor (GCD)
- Least common multiple (LCM)
- Hanoi tower
- Coin change
- Permutation / combination

Final Review

Higher-order programming

Why we can do higher-order programming:

- Functions are also constants or variables.
- They are not special.
- They just behave like normal constants or variables.

To use higher-order programming:

- Constants or variables can be functions.
- Parameters can be functions.
- Return values can be functions.

Final Review

Pair/list/tree processing

Up to now, the list library supports different kinds of functions:

- List builder: `list`, `build_list`, `enum_list`;
- List getter: `head`, `tail`, `list_ref`, `member`, `is_member`;
- List information: `is_list`, `is_empty_list`, `length`, `equal`;
- List modifier: `append`, `reverse`, `remove`, `remove_all`, `filter`, `map`, `for_each`;
- List converter: `accumulate`, `list_to_string`.

Final Review

Stream processing

Up to now, the stream library supports different kinds of functions:

- Stream builder: `stream`, `build_stream`, `enum_stream`, `integers_from`;
- Stream getter: `stream_tail`, `stream_ref`, `stream_member`;
- List information: `is_stream`, `stream_length`;
- List modifier: `stream_append`, `stream_reverse`, `stream_map`, `stream_for_each`, `stream_remove`, `stream_remove_all`, `stream_filter`;
- List converter: `list_to_stream`, `stream_to_list`, `eval_stream`.

Final Review

Data structure design

You should follow these principles:

- Understand the requirement before doing the actual design;
- Separate the interface from the implementation;
- Compare the advantage and tradeoff;
- Principle of last commitment.

Final Review

Memoization/dynamic programming

Use a table (literal object) to save all previously computed results:

```
function memoize(func) {
    const table = make_table();

    return function (x) {
        if (contains(x, table)) {
            return lookup(x, table);
        } else {
            const result = func(x);
            put(x, result, table);

            return result;
        }
    };
}
```

Final Review

Array, loop

- Array vs list
- Iteration (loop) vs recursion
  - `while` loop
  - `for` loop

Literal object

- Array vs literal object
  - Extension for table (memoization)?

Final Review

Meta-circular evaluator

- Add tags to differentiate various statements.
- Primitive data types/operators are self-evaluating.
- Use lists and literal objects to implement environment model.

- Basic evaluator?
- OOP evaluator?
- Lazy evaluator?
- Memoized evaluator?
- Tail recursion evaluator?

Final Review

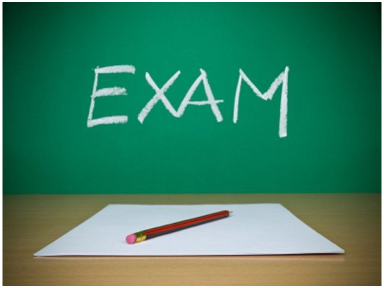
To prepare for the final examination

- Read all the materials distributed again;
- Do as many PYPs (past year papers) as possible;
- Summarize what you have learned;
- Be relaxed.

Final Review

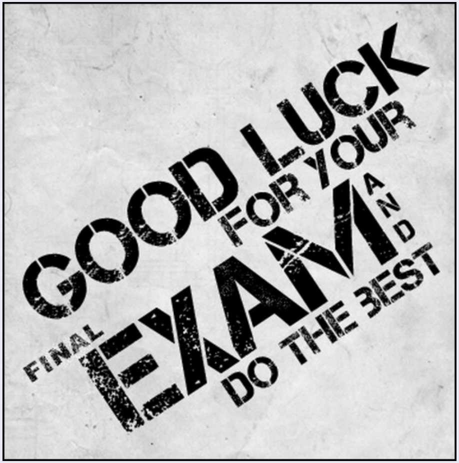
One weird thing about CS1101S final examination

- For most modules, the final will be much harder than the mid-term.
- However, CS1101S final may not be that difficult.
- ...



Good Luck

May the Source be with you!



End

The End

Copyright

Niu Yunpeng © 2017 - 2018. Under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International *License*.

Appropriate credits MUST be given when sharing, copying or redistributing this material in any medium or format. No use for commercial purposes is allowed.

This work is mostly an original by *Niu Yunpeng*. It may either directly or indirectly benefit from the previous work of Martin Henz, Cai Deshun. For illustration purposes, some pictures in the public domain are used. Upon request, detailed acknowledgments will be provided.