1: Introduction and First Lecture

CS1101S: Programming Methodology

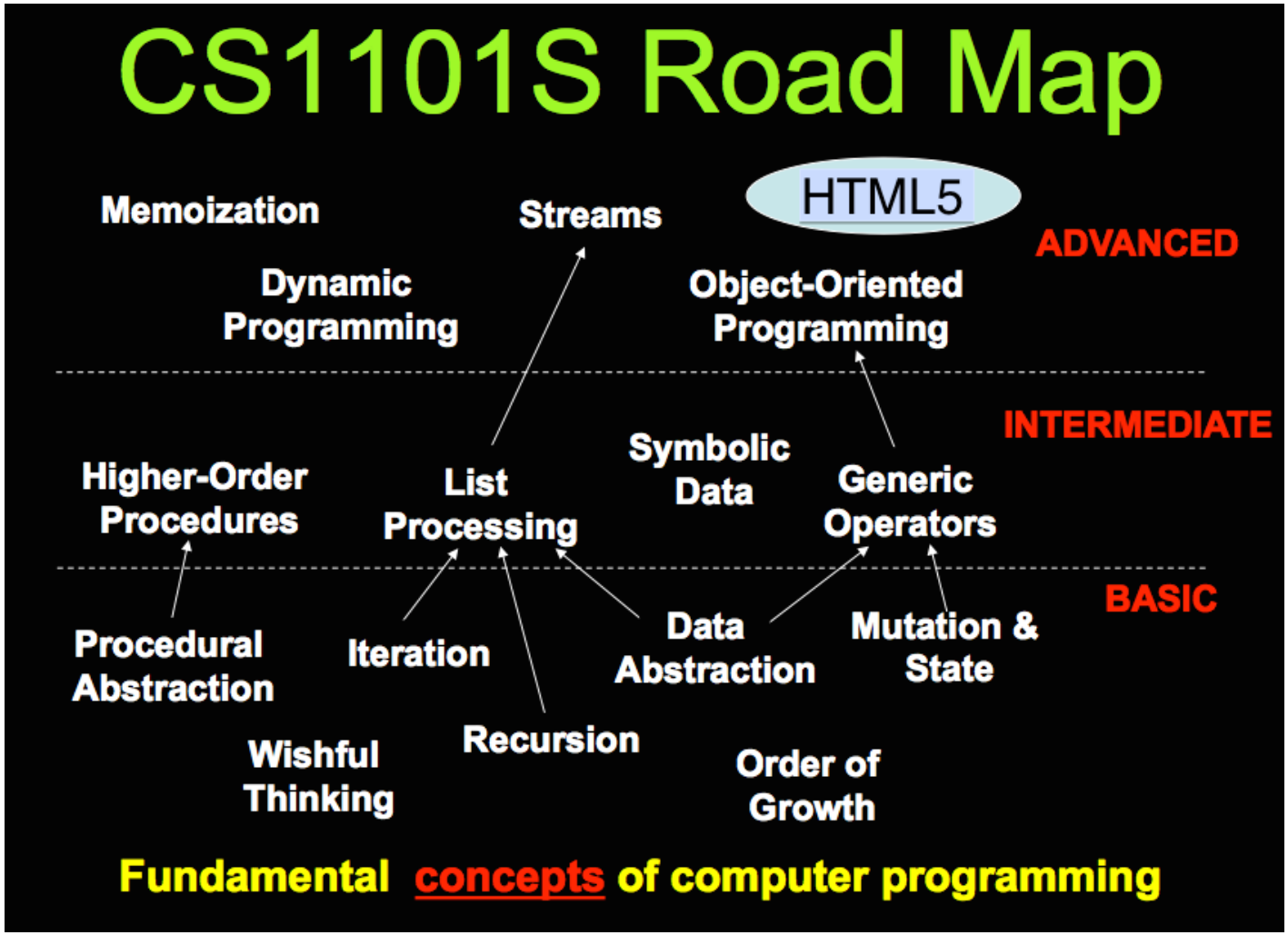
Martin Henz and Low Kok Lim

August 10, 2016

The goals of this module

- Learn *programming methodology*
- Learn how to program
- Get glimpses into the subject areas of computer science
- Get a feeling for how computer science ticks
- Fall in love with computer science

Roadmap for CS1101S



Module origins

Module concept developed at MIT in the 1980s, by Hal Abelson and Jerry Sussman

1101S Legacy: Getting started at NUS

Introduced in DISCS in Sem 1 1997/1998 as “IC1101S”

Lecturers: Jacob Katzenelson and Leong Tze Yun

Other key staff: Recitations / Honorary TA / TA

Leong Hon Wai / **Terence Sim** / **Razvan Voicu**

Following year

Jacob taught in Sem 1 1998/1999, renamed to “CS1101S”.

1101S Legacy: Developing

Sem 1 1999/2000: Leong Tze Yun

1101S Legacy: Growing

Sem 1 2001/2002: Razvan Voicu

and also in...

Sem 1 2002/2003 and Sem 1 2003/2004

1101S Legacy: Expanding

Sem 1 2004/2005: Terence Sim

and also in...

Sem 1 2005/2006

1101S Legacy: Maturing

Sem 1 2006/2007: Razvan Voicu and **Ben Leong**

1101S Legacy: Rejuvenating

Ben Leong

Ben took over the module Sem 1 2007/2008 and taught it until Sem 1 2011/2012 (including co-teaching with Razvan, 6 years)

1101S Legacy: Modernizing

Martin Henz

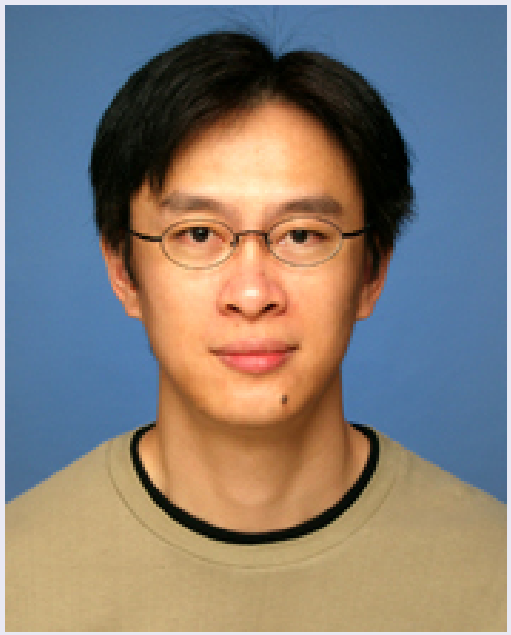
Took over the module in Sem 1 2012/2013, for the first time world-wide using JavaScript instead of Scheme

and also in...

Sem 1 2013/2014

1101S Legacy: Strengthening the team

Sem 1 2014/2015: **Martin Henz** and **Low Kok Lim**



Expanding

the material to be accessible to all CS freshmen

1101S Legacy: Pedagogical excellence

Sem 1 2015/2016: **Martin Henz** and **Low Kok Lim**

Aiming for the highest standards in education

1101S this year: Experience The Source

Sem 1 2016/2017: **Martin Henz** and **Low Kok Lim**

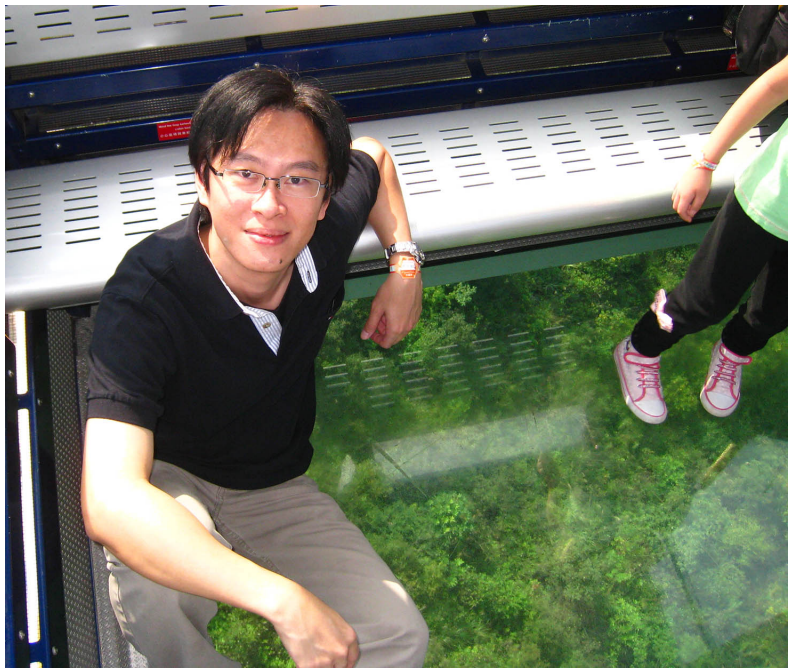
Introducing The Source Academy, crafting a coherent CS1101S experience

The Cast

Co-lecturer Low Kok Lim

Interesting fact

"I once had a dog named "Come Here!"



Co-lecturer Martin Henz

Interesting fact

"I don't have an undergraduate degree."

Avengers

Their role

The life blood of the module: Run discussion groups, mark 24-by-7, manage the module server, fine-tune and test the missions, handle “awards and incentives”, run contests, etc etc

Avenger Thenaesh Elango

Interesting fact

"I eat people who don't indent their programs properly."

Avenger Tran Tien Dat

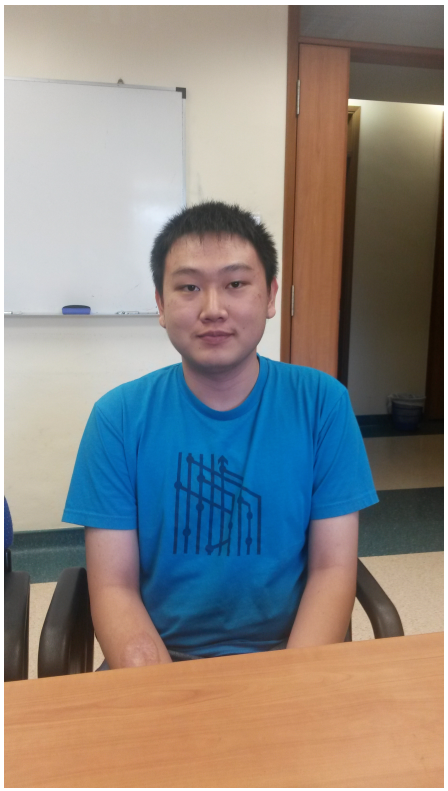
Interesting fact

"I learnt cooking over the summer and can now cook some simple Vietnamese dishes."

Avenger Evan Sebastian

Interesting fact

Suffers from arachnophobia and thinks fire can cure it!



Avenger Lei Mingyu

Interesting fact

"I've visited the northernmost city in the world."

Avenger Syed Abdullah

Interesting fact

"I'm not really fazed by blood and gore, maybe due to my 2 years of NS."

Avenger Cai Deshun

Interesting fact

"I like chocolate."

Avenger Kenneth Loh

Interesting fact

"I learnt Java from a C++ manual."

Avenger Lim Jia Yee

Interesting fact

"I still can't find my talent."

Avenger Advay Pal

Interesting facts

"An Avenger who prefers DC, and an Indian who can't tolerate spicy food."

Avenger Yuan Yuchuan

Interesting fact

"Prof. Martin once gave me ten trillion dollars."

Avenger Ray Yan

Interesting fact

"When I was a kid, I downloaded my first free anti-virus. Guess they meant 'free virus included' :(#PossibleReasonsToStudyCS"

Avenger Derek Nam

Interesting fact

"I am fascinated with bears."

Avenger Xiao Pu

Interesting fact

I am not afraid of termite as I have lived with them for one year (they are in my table!).

Avenger Souvik Kundu

Interesting fact

"I love travelling, but am scared of flying!"

Avenger Quoc Binh Nguyen

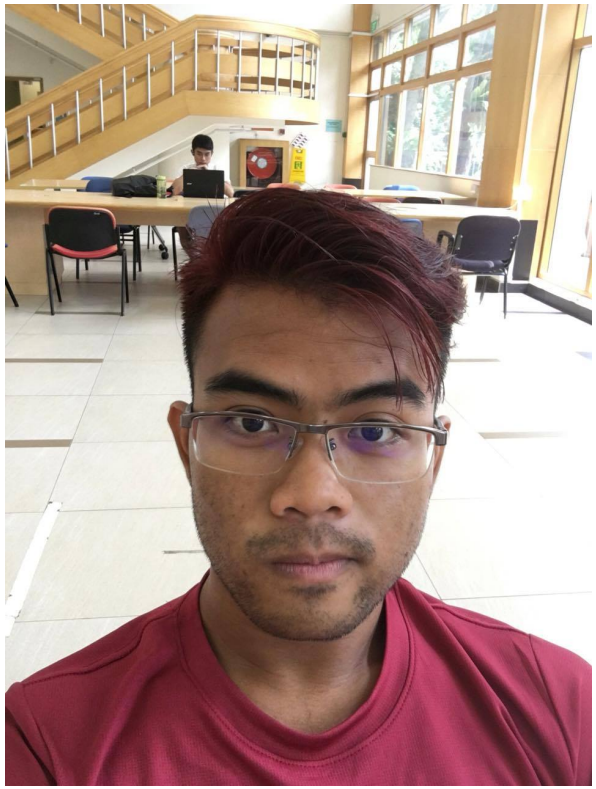
Interesting fact

"I also work in social sciences."

Avenger Yu Peng

Interesting fact

`#teamvalor`



The Flow

- Wednesday lecture with path and textbook sections
- Wednesday/Thursday recitation (hands-on material related to lecture)
- Friday lecture with path (tend to focus on “special” topics)
- Monday/Tuesday discussion groups (in-depth material related to lecture)
- Missions, side quests, contests: Any time, anywhere, anyhow
- Midterm on 28/9, PE on 10/11 or 11/11, final assessment on 23/11
- Other events: Sumobot Contest 12/10 (6pm till late), Deathcube Contest 28/10, 11am-12noon

The Source Academy—The Game

- 7 problem sets, containing 22 main missions
- contests on graphics, acoustics, robotics, game development
- 24-hour grading
- Unlock Achievements and aim for eternal fame in the Leader Board!

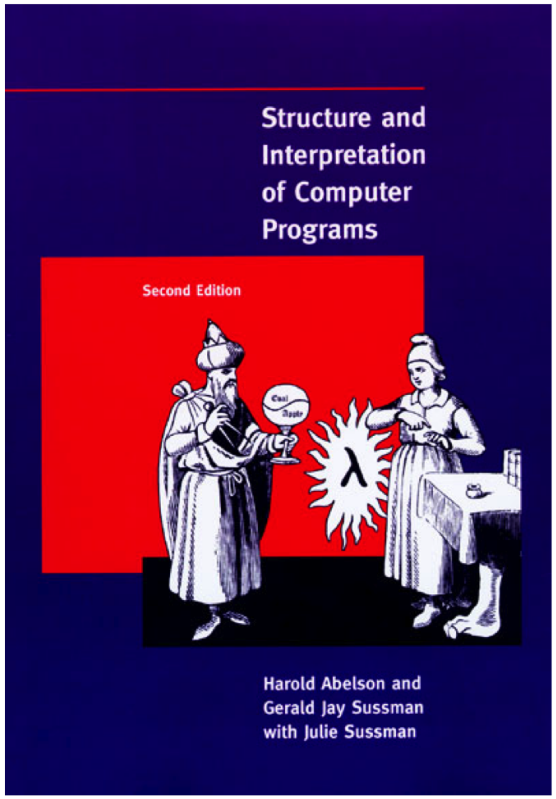
The Source Academy—Beyond the Game

Academy is emersive

It includes everything about CS1101S (exceptions: IVLE forum and IVLE notification, including SMS)

- Lecture slides and lecture paths
- Module calendar: lecture/recitation/DG times and venues, and mission deadlines
- Source Week X descriptions, style guide
- Textbook

Textbook



...with a twist: Translated into JavaScript!

IVLE Forum

Purpose

Students help each other out; Avengers and lecturers see common issues and address them

For the entire class

Allows for general discussion and broad questions

Stalker

Avengers are “stalking” the forum to help, and to see who deserves experience points

Forum Experience Points

Stalker is the ref

His decision is final. Complaints to lecturers, not him, please

Criteria

The stalker uses the following criteria, in decreasing importance:

- Relevance of post for classes / missions
- Impact: Help provided, discussions generated
- Effort put into writing
- Frequency of posts

If that's not enough...

Side quests

provide additional material and interesting challenges

Do they earn you Experience Points?

You bet they do! Yes!

Do they earn you academic credit?

Only when you complete (almost) all. Then you get to enrol in CS1010R in Sem 2, which gives you 1 MC without any work in Sem 2.

Experience Points—Summary

Earn your experience points by:

- Completing missions
- Completing side quests
- Winning contests
- Attending Discussion Groups
- Completing lecture paths (“Early Bird” bonus for same day or next day)
- Participating in Forum

...and **level up!**

Assessment Overview

- 35%: Missions
- 5%: Discussion group participation
- 15%: Midterm assessment (28/9)
- 15%: Practical Exam (10/11 or 11/11)
- 30%: Final Assessment (23/11)

Your grades

Are they “curved”?

No

Are they fair, compared to CS1010?

We hope so!

“S/U”: Satisfactory/Unsatisfactory

Can I S/U CS1101S”?

Yes

Caveat (“watch out”:

CS2020 has a requirement of at least A- in CS1010/CS1101S.

Need help?

- PM your Avenger in The Academy (or in real-life WhatsApp/FB/you-name-it)
- Discussion forum
- Email/SMS/FB your avenger, or lecturer
  - Martin: 96991279, `henz@comp.nus.edu.sg`
  - Kok Lim: 93388638, `lowkl@comp.nus.edu.sg`

The Source Academy

Immersive experimental environment

for learning programming methodology, starting in AY2016/17

Caveat

There will be hick-up and problems. Please bear with us.

The Up-Side

You can tell your grandchidren: “It started with my batch!”

Kudos

The Source Academy team, led by Evan Sebastian, the creative team, led by Ng Tse Pei, the developers of Coursemology, V2 Beta, led by Joel Low, and the mastermind behind the original CS1101S game and behind Coursemology, Ben Leong

Why program?

- Necessary: We need to give computers very precise instructions
- Possible: Computers follow such instructions precisely (and quickly!)
- Fun...
- Some people make a living doing it

What is programming?

communicating
a computational process

Why Source?

- For learning programming, the language isn't that important (same same...but different)
- JavaScript is the most executed programming language on earth (counting the number of individual program executions)
- JavaScript is ubiquitous and powerful, but quite ugly (and with serious design flaws, see Douglas Crockford: JavaScript, the Good Parts)
- Our solution: pare the language down to a simple and beautiful core: The Source
- The Source Week 3, The Source Week 4, etc: 5, 6, 8, 9, 10, 12

History of JavaScript

- In 1995, Brendan Eich started to work on an interpreted language for the Netscape browser, called Mocca, then LiveScript. Eich designed and implemented the language within a few weeks, apparently without much collaboration with anyone.
- In December 1995, Sun Microsystems and Netscape announced the language under the name JavaScript.
- Standardized as ECMA-262, first edition in June 1997: “ECMAScript”
- Latest version: JavaScript 1.8.5: ECMAScript-5-compliant, supported by all major browsers, and most likely the phone in your pocket

Use of ECMAScript

- Client-side interpretation; embedded in web pages; the “machine language” of the world-wide web
- ActionScript: programming of Flash animations (remember Flash?)
- JScript: scripting language for Windows/.NET
- Server-side execution: Node.js

Primitive expressions

- Numerals: 4, -42, 824.3
- Numerals use decimal notation
- Our interpreter can evaluate numerals, resulting in the numbers they represent

A small complication

Expressions are not programs in Source

Instead they can be turned into programs using a semicolon.

Example

```
-42;
```

is a program.

Means of Combination: Operators

Examples

```
5 + 8;

25 - (4 + 2) * 3;
```

Notation as in school

operator between operands: *infix notation* with *precedences*

Means of Abstraction: Naming

Example

```
var a = 55;
a + 8;
```

Environment

Purpose

The interpreter of JavaScript keeps track of an environment (table) that associates symbols with their values.

Variable statements

The keyword **`var`** adds a symbol-value entry to the table or changes the value if the symbol is already defined.

Means of Abstraction: Functions

Example

```
( function(x) { return x * x; } )
```

What does this expression mean?

This function definition expression evaluates to a function that takes an argument `x` and produces (returns) the result of multiplying `x` by itself.

Means of Abstraction: Functions

Applying a function

We can apply a function by supplying its arguments in parentheses.

Example

```
( function(x) { return x * x; } )(4);
```

Combining Function Definition and Naming

Example

```
var square = function(x) { return x * x; };
```

Two concepts

We *define* a function using a function definition expression, *and* give it a name, using a variable declaration statement.

Now, applying the function becomes easier:

```
square(4);
```

Function Definition *Statements*

```
var square = function(x) { return x * x; };
```

another way of writing it:

```
function square(x) { return x * x; }
```

Evaluation of Expressions

```
(3 * 4) + (7 - 8)

12 + (7 - 8)

12 + -1

11
```

Making use of Function Definition

```
square(21);

square(2 + 5);

square(square(3));
```

Your turn: In-class exercise

The task

Find a way to easily compute the sum of the squares of two given numbers

Our problem solving habit

First think, then program!

The Program

```
// computing sum of squares of numbers x and y
function sum_of_squares(x, y) {
    return square(x) + square(y);
}
```

Summary

- `3 + 4;` a combination with an operator and two numerals
- **`function`**`(...) ...` forms a value that describes a sequence of actions
- **`var`** `x = ...;` associates the name `x` with a value
- **`var`** `f =` **`function`**`(...) ...;` combines the two abstraction techniques.
- **`function`** `f(...) ...` is practically equivalent and easier to write.

Welcome to CS1101S

Let The Show Begin