this is week 2

fall 2013

"sections provide you with
opportunities to explore
the course's material
in a more intimate environment
as well as to dive into
hands-on activities"

agenda

norms

arrays

functions

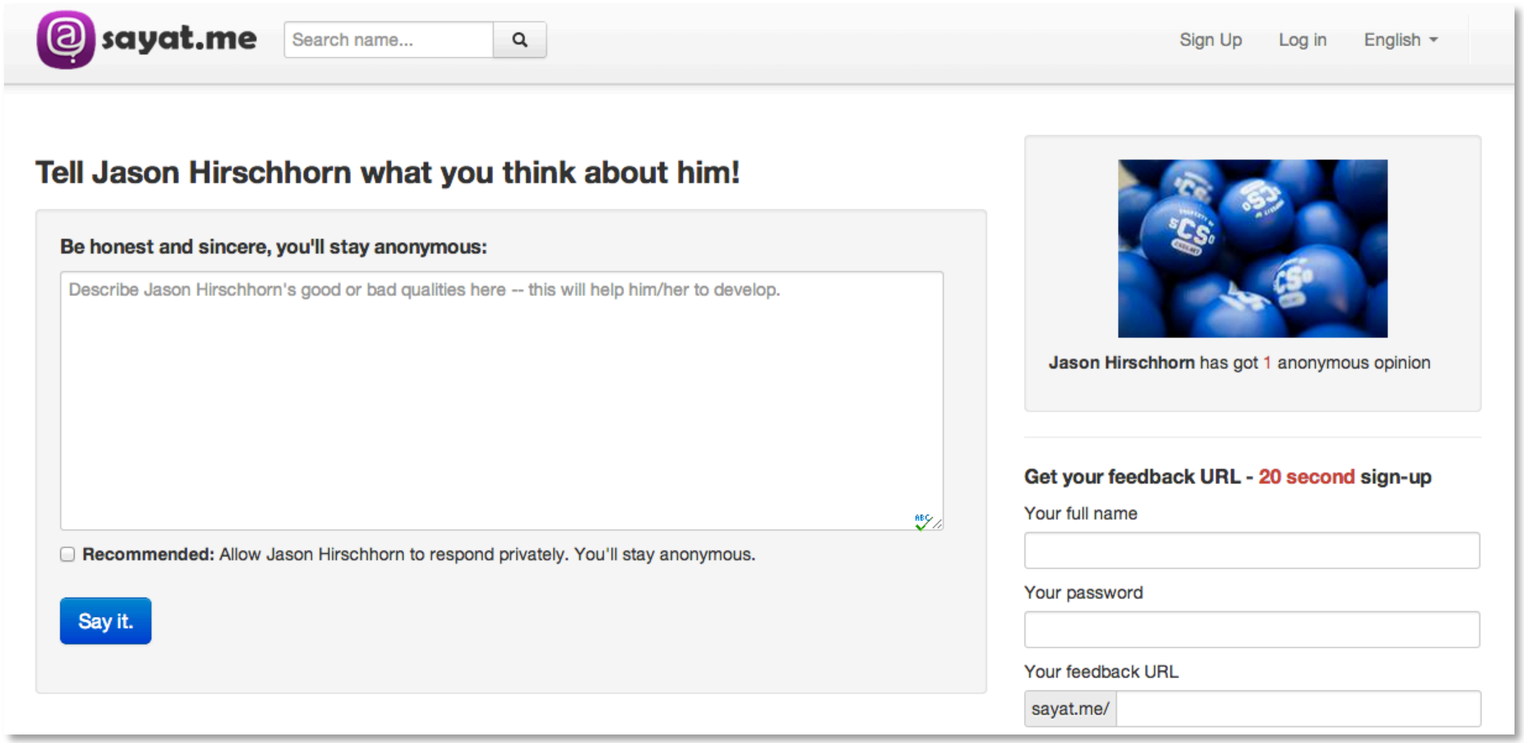
command-line arguments

norms

support

meeting in the middle

high expectations



Tell Jason Hirschhorn what you think about him!

Be honest and sincere, you'll stay anonymous:

Describe Jason Hirschhorn's good or bad qualities here -- this will help him/her to develop.

☐ **Recommended:** Allow Jason Hirschhorn to respond privately. You'll stay anonymous.

Say it.

Jason Hirschhorn has got 1 anonymous opinion

Get your feedback URL - 20 second sign-up

Your full name

Your password

Your feedback URL

sayat.me/

http://sayat.me/cs50

norms

support

meeting in the middle

high expectations

fun

arrays

make an array

```
<data type> <name>[<size>];

char alphabet[26];

int scores[3];
```

make an array

```
int scores[3];

scores[0] = 1;

scores[1] = 2;

scores[2] = 3;
```

make an array

```
int scores[3];

scores[0] = 1;

scores[1] = 2;

scores[2] = 3;

scores[3]; // what's this?
```

make an array

```
int scores[3];
scores[0] = 1;
scores[1] = 2;
scores[2] = 3;
// alternative initialization
int scores[3] = {1, 2, 3};
```

iterate through an array

```
int scores[3] = {1, 2, 3};
for (int i = 0; i < 3; i++)
{
    printf("%d\n", scores[i]);
}
```

iterate through an array

```c
int scores[3] = {1, 2, 3};
// is this okay?
for (int i = 0; i <= 3; i++)
{
    printf("%d\n", scores[i]);
}
```

your turn: count.c

Write a program that creates an array with the integers 1 through 5 and then prints out each integer on a new line.

strings

arrays of chars

end with a '\0'

to iterate, use i < strlen(s)

your turn: spell.c

Write a program that asks the user for a string then prints out each character on a new line.

your turn: students.c

Write a program that asks the user for fives names then randomly chooses and prints out one of the names.

functions

black boxes

take things in (parameters)

do something (side effects)

spit something out (return value)

why use functions?

anatomy of a function

```
<return type> <name>(<parameters>)
{
    <code>
}
```

anatomy of a function

```
int main(void)
{
    printf("ohai, world!\n");
    return 0;
}
```

anatomy of a function

```
int main(void)
{
    printf("ohai, world!\n");
    // is the return necessary?
    return 0;
}
```

scope

every variable has a certain scope

where the variable may be referenced

what happens in the braces,

stays in the braces

scope

```
int a;
int main(void)
{
    int a;
    {
        int a;
        a = 4; // which "a" is this?
    }
    a = 2; // which "a" is this?
}
```

your turn: function.c

Write a program in which main calls another function that prints out a greeting to the user.

function declaration

```
void hello(void);
int main(void)
{
        // code here
}
void hello(void)
{
        // code here
}
```

command-line arguments

Command-line arguments

one way to pass information into a program

```
int main(int argc, string argv[])
```

argc = "argument count" (# of arguments)

argv[] = "argument vector" (arguments)

example

```
./ohai cs50 section

argc is 3

argv[0] is "ohai"

argv[1] is "cs50"

argv[2] is "section"
```

multi-dimensional arrays

arrays of arrays

"rows and columns"

```
./ohai cs50 section
```

argv[1] is "cs50"

argv[1][2] is '5'

your turn: personalized.c

Write a program that takes a user's full name via the command-line (two and only two words). Next, print out a greeting to the user that includes their first name.