Reminder: midterm next Thursday in class
- Chapter 17 will not be on the midterm

I posted some updated slides about the intersecting list on the schedule. You can also see more about it in the textbook.

You’ve all done the scratch lab. Let’s take a look at something that’s a teeny part of what you did, but makes a big change…
A **variable** is a named quantity.
Playing with variables

- The following examples work pretty much the same no matter which programming language you use.
- That being said, it’s helpful to have something to test it out for you.
- So we’re going to be using Javascript.
The Assignment Statement

\[ \text{<variable>} \ \text{<assignment symbol>} \ \text{<expression>;} \]

- \text{<variable>} is any declared variable in the program
- \text{<assignment symbol>} is the language’s notation for the assignment operation
- JavaScript’s \text{<assignment symbol>} is the equal sign (=)
- \text{<expression>} is a kind of formula telling the computer how to compute the new value
- Like any other statement, an assignment statement is terminated by a semi colon.
In computer science, the symbol “=” can be used in two ways:
1) to express equality (as in math),
   e.g., 3=5 (false) or 3=5-2 (true)
   (I read this as “equals” – as usual)
2) to assign values to variables,
   e.g., X=5 (after doing this, variable X
   has value 5, and the equality X=5 is true)
   (I like to read this as “gets”)

\[
\text{EUR\_per\_CAD} = 1.3597;
\]
\[
\text{balance\_in\_EUR} = 100;
\]
\[
\text{balance\_in\_CAD} = \text{balance\_in\_CAD} \times \text{EUR\_per\_CAD};
\]
Tips and Tricks

• In HTML a statement like X=5 always refers to a variable (actually, attribute) assignment.

• Likewise, in JavaScript, a statement like X=5 always denotes variable assignment, while equality is written differently (X==5).

• In other languages, ‘=’ is used to denote equality, and ‘:=‘ or ‘<-’ to denote variable assignment.
On readability of code...

**Note:** The following sequence of assignments has exactly the same function as the one shown previously, but is much less human-readable, and therefore inferior:

\[
\begin{align*}
x &= 1.32; \\
y &= 100; \\
z &= y \times x;
\end{align*}
\]

(In fact, you could use ‘alice’ for ‘x’, ‘bob’ for ‘y’, *etc.*, and still get the same function.)
And now programming with Javascript

• At this point, we’ve created webpages using HTML
• We’ve learned how to make them look better and have better coordination using CSS
• But we may want to do more…
Consider the class webpage

- The HTML for the top menu is in a different file
- This allows us to reuse it easily
- But what if we wanted to have the top menu tell you which page we’re on

Now:

Desired:

Set by using `class=“active”`
Let’s look at the menu now:

<!--the menu for each page, here in one spot for easy changes-->
<div id="header">
    <div id="header_inner" class="fixed">
        <div id="logo">
            <h1>CPSC101</h1>
            <h3>Connecting with Computer Science…</h3>
        </div>
    </div>
</div>

<div id="menu">
    <ul>
        <li><a href="schedule.shtml">Schedule</a></li>
        <li><a href="syllabus.shtml">Syllabus</a></li>
    </ul>
</div>
Let’s start by changing the menu list to Javascript

```html
<!-- HTML so far-->
<script> // Start script – Javascript by default
    // coments in Javascript
    document.write("<li><a href="schedule.shtml">Schedule </a></li>");
    //note the escaping of the "s
    //also note that x.y usually means x is a thing
    //and y is either something about x (an attribute)
    //or something that we can do to x (a function)
    document.write("<li><a href="syllabus.shtml">Syllabus</a></li>");
</script>
```

Connecting with Computer Science
www.ugrad.cs.ubc.ca/~cs101
Okay, so now it’s in Javascript, but it hasn’t done anything!

Right. In order for that to happen, we need to know what page we’re accessing:

```javascript
var index = location.pathname.lastIndexOf("/") + 1;
// create a variable called “index” that gets
// the value of where the last “/” appears +1
// in other words, where the directories end
var filename = location.pathname.substr(index);
// create another variable called filename that gets
// the substring of the URL that is before the /
document.write(filename);
```
But now we need to change what we’re doing based on which file is being loaded

We need a conditional statement
Conditional Statements

The conditional has the form:

if (<Boolean expression>)
<then-statement>;

The <Boolean expression> is an expression evaluating to true or false

The <then-statement> is any JavaScript statement

if (waterTemp < 0)
waterState = "Frozen";
if/else Statements

if (<Boolean expression>)
    <then-statement>;
else
    <else-statement>;

• If the <Boolean expression>’s outcome is true:
  • The <then-statement> is executed
  • The <else-statement> is skipped

• If the <Boolean expression>’s outcome is false:
  • The <then-statement> is skipped
  • The <else-statement> is executed
An important detail

• The “if statement” and “then statement” can only be ONE statement. To make them more than one, use {} – which we will call curly braces

• Example
  
  if (date == February 6) {
    midterm=true;
    bringID = true;
  }
  else{
    midterm=false;
  }

  
  *Always use the {}s or you will shoot yourself in the foot*
But back to our webpage

What we want is to say *if* the name of the page is “schedule.shtml” *then* we add in the information `class="active"`.

Do we need an else?
Putting it all together

... <script>
    var index = 
    location.pathname.lastIndexOf("/") + 1;
    var filename = 
    location.pathname.substr(index);
    document.write("<li><a 
    href="schedule.shtml\"">
    if (filename=="schedule.shtml"){
        //Note uses ==, not =
        document.write(" class="active\"");
    }
    document.write(">Schedule </a></li>"");
</script>
Yay! We made the website better! What else could we possibly want?

Well, one problem is that we could want a button to do something when we click on it. Currently, we can make a button, but it doesn’t do anything.

...  

<h1>Can Stimpy resist?</h1>
<button style="background-color:red">History Eraser Button</button>

<!--http://www.youtube.com/watch?v=NITBfc1EOBo-->

...
Which brings us to a point

So far, the stuff we’ve been writing has either just been used from the top down (as in HTML) or it’s been used every time we need it (as in CSS).

But sometimes that’s not enough. Sometimes we might want our code to react to a specific event, or we might want to reuse it.

Then we need functions.
Key concept: Functions

• Break up a big process into smaller parts
  – Handle the complexity of big processes

• Give a name to part of a process
  – Improve the readability of your processes

• Allows you to re-use components (share) parts between processes, and between people
  – Independence: you can change in one place for everything!
  – Abstraction / Teamwork: you don't have to know how it works, and someone else can write your functions
Function: Build a House

• Build a house
Function: Build a House

• Build a house
  – foundation
  – build frame
  – build floors
  – electrical work
  – plumbing
  – drywall
  – ...

79
Function: Build a House

• Build a house
  – foundation
  – build frame
    • ...
      – ...
        » hammer a nail
  – build floors
    • ...
      – ...
        » hammer a nail
Function: Build a House

• Build a house
  – foundation
  – build frame
    • ...
    • ...
    • hammer a nail
  – build floors
    • ...
    • ...
    • hammer a nail

Re-Use
Function: Build a House

- Build a house
  - build frame
    - ...
      - ...
        » hammer a nail

- Build a dog house
  - ...
    - ...
      - hammer a nail
Anatomy of a Javascript Function

A **function declaration** includes a **name** and possibly **parameters**. A parameter is a variable that is passed in to the function.

The **body** includes **variable declarations**, **other instructions** and sometimes a **return statement** to return information back to whatever called it.

```javascript
function calculateTip(amount) {
    var tip;
    tip = amount * 0.12;
    return(tip);
}
```
Let’s create a function for Stimpy’s button

<script>
// note that this is not html, so we have to
// do it in Javascript

function pushIt()
// our function is named pushIt. It has no
// parameters
{
    alert("DOOOOOMMMM!");
    // creates a popup that tells us what
    // happened.
}
</script>
Well that was anticlimactic

Why did nothing happen? Let’s look at what we did:

- We wrote HTML which created a button
- We wrote a function that creates an alert

Did we tell the button what to do when it’s clicked? Uh… no. (something that happens that you want to respond to is called an event, btw.)

```html
<button style="background-color:red"
    onclick="pushIt()"
    >
    History Eraser Button</button>
```

Note that we’re back in HTML for this. So we put the function call in quotes because html expects its attributes to be quoted – just like the background color for the button. The empty ()s is because the function takes no parameters
Victory!

But why not just have that code attached to the button?

Well, what if you wanted TWO history erase buttons?

<button style="background-color:red" onclick="pushIt()">First History Eraser Button</button>

<button style="background-color:yellow" onclick="pushIt()">Second History Eraser Button</button>
Well, that’s a little better. But what if we want them to do something a little different.

Let’s say that we wanted to have a different message if we clicked on the first history eraser button rather than the second. We need to use a parameter in the function:

```javascript
function pushIt(textToPrint)
{
    alert("DOOOOOOOOOOOOOOOOOOMMMMMMM!\n" + textToPrint);
    //note that \n is javascript for newline
    //since we’re dealing with text, the “+”
    //concatenates the text rather than adding
}
```
We also need to change how we call the function

```html
<button style="background-color:red"
onclick="pushIt('First')">First History Eraser Button</button>

<!-- normally it doesn’t really matter if you use ' or ", but since it’s better to use 's-->

<button style="background-color:yellow"
onclick="pushIt('Second')">Second History Eraser Button</button>
```