Last administrative notes!

- Remember the final – April 16 @8:30 am in FSC 1005
  - Expect a seating chart
- Projects are due today at 6pm
  - I'll put up a link to all the projects after the deadline. If you do NOT want your project put on the webpage, send me mail.
- Please do your course evaluations
- Download the review template at http://www.ugrad.cs.ubc.ca/~cs101/2013W2/examples/review-template.html
Review!
As mentioned last week, my goal is to review during this class

- We'll do most of this with exercises
- I'll add in a bit of explanation where necessary
- The things that I heard the most that people wanted reviewed were functions, conditionals, for loops, and arrays
- That's not to say that other materials won't be on the final, or aren't important…
As a brief recap, we've covered

- computing basics and html
- interfaces
- programming concepts
- art & image representation
- networking
- minds & machines
- inside the machine
- digital DNA
- diversity
There are MANY practice questions on the web

• If you try to do them by looking at the solution and making sure that you understand the solution rather than trying to solve the problem, you are in for a world of pain on the final.

• The best way to study is to do the practice problems

• That's what we'll do in class today based on feedback on what people would like to cover.
Let's start with functions

- A function is a bit of code that we **declare** in one place, and then **call** in another.
- You can think of this as kind of like the declaration creates a blueprint for a house, and then calling it actually creates the house.
**Anatomy of a Javascript Function Declaration**

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)
}
```
Our goal: calculate RQ grades

The syllabus states: "The marks will be summed up and counted out of n * 1.2, where n is the final number of reading questions. A score of n * 1.2 or above earns full credit on the reading question portion of your grade. For example, if there are 15 RQs, a score of 18 or higher earns full credit. If there are 18 RQs, a score of 22 or higher earns full credit."
Let's create a function where we can calculate the goal to get 100%

To do that, we need to multiply the # of RQs by 1.2. Well call that full credit

Since there were 14 RQs, we'll pass in 14 as the parameter, so our call might look like:

calculateFullCredit(14);
function calculateFullCredit(numRQs) {
    return numRQs * 1.2;
}
//end function calculateFullCredit
Next, we can calculate the percentage grade

We can calculate a percentage score by taking the sum of the students RQ grades (we'll calculate that later— for now assume it's a given) and dividing it by the full credit value, and then multiply by 100 to get the percentage. We'll start with an easy version where we're given the full credit value.

A call might look like:

```
calculateRQPercentGivenFull(12, 16.8);
```

meaning that the sum of the student's RQ grades was 12, and the full credit RQ value is 16.8
calculateRQPercentGivenFull might look like:

```javascript
function calculateRQPercentGivenFull(sumRQMarks, fullCreditValue) {
    return sumRQMarks/fullCreditValue * 100;
}//end function calculate RQPercentGivenFull
```
Let's put these together

• We have a function that takes the #s of RQs and computes the full credit score.
• We have a function that takes the full credit score and the sum of a students RQ marks and finds the student's RQ percentage
• Let's write a function that takes the # of RQs and the sum of a students RQ marks and returns the percentage a call might look like calculateRQScoreGivenNumRQs(12, 14); where 12 is the sum of the student's RQ marks and 14 is the number of RQs.
function calculateRQScoreGivenNumRQs(sumRQMarks, numberRQs) {
    var fullCreditValue = calculateFullCredit(numberRQs);
    return sumRQMarks/fullCreditValue * 100;
}//end function calculateRQScoreGivenNumRQs
Well, that's great. But the top score for RQs in the class was 26… and the mode was 21!

• So what do we do with those?
• We'd like them to be given 100%
• To do that, we need conditionals
if/else Statements

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

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

• If the <Boolean expression>’s outcome is \textit{false}:
  • The <then-statement> is skipped
  • The <else-statement> is executed
Let's change calculateRQPPercentGivenFull

What we have so far:

```javascript
function calculateRQPPercentGivenFull(sumRQMarks, fullCreditValue){
    return sumRQMarks/fullCreditValue * 100;
}
```

//end function calculate RQPPercentGivenFull

Change it so that if the initial value is over 100, return 100
If we change it, it might look like:

```javascript
function calculateRQPercentGivenFull(sumRQMarks, fullCreditValue){
    var percent = sumRQMarks/fullCreditValue * 100;
    if (percent > 100)
    {
        percent = 100;
    }
    return percent;
}
//end function calculate RQPercentGivenFull
```
Now, make the change to calculateRQScoreGivenNumRQs

What we had so far was

```javascript
function calculateRQScoreGivenNumRQs(sumRQMarks, numberRQs) {
    var fullCreditValue =
        calculateFullCredit(numberRQs);
    return sumRQMarks/fullCreditValue * 100;
}//end function calculateRQScoreGivenNumRQs
```
calculateRQScoreGivenNumRQs might look like

function calculateRQScoreGivenNumRQs(sumRQMarks, numberRQs)
{
    var fullCreditValue =
        calculateFullCredit(numberRQs);
    var percent = sumRQMarks/fullCreditValue * 100;
    if (percent > 100){
        percent = 100;
    }
    return percent;
}
//end function calculateRQScoreGivenNumRQs
What if we wanted to find the student's total RQ score given the individual RQS?

- We'd need a for loop
- We'd also need an array
- So let's try something slightly easier to start with: let's just print to the screen what a perfect RQ run would look like. A row of as many 2s (separated by spaces) as there are RQs. This uses loops, but not arrays
- Let's review for loops first.
for Loop Syntax

for(<initialization>;<continuation>;<next iteration>)
{
    < statement list>
}

<continuation> has the same form as the predicate in a conditional statement

If the <continuation> test is false outcome, the loop terminates and <statement list> is skipped

If <continuation> has a true outcome, the <statement list> is performed

<next iteration> defines what happens at the end of the loop to start the next round
Great! Now we're ready to write our function

We might call it as

printNTwos(4);
The function might look like:

```javascript
function printNTwos(numRQs) {
    var index;
    for (index=0; index < numRQs; index++){
        document.write(2 + ' ');
    } // end for
} // end function printNTwos
```

NOTE: no return value!
Your turn

• Write a function that adds together the numbers one to N (where N is a parameter)
• This will help us when we're calculating the sum of the RQ grades
• A call to it might look like `sumOneToN(4);`
sumOneToN might look like

```javascript
function sumOneToN(number) {
    var sum = 0;
    var index;
    for (index=1; index <= number; index++){
        sum+= index;
    } //end for
    return sum;
} //end function sumOneToN

Note: this loop begins at 1 and ends at number
Okay, great. Now let's get back to adding all of a student's RQs

• For that, we're going to need arrays
Think of arrays as being a row of post office boxes.

As with any variable, each box can fit exactly one thing.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;red&quot;</td>
<td>&quot;green&quot;</td>
<td>&quot;blue&quot;</td>
<td>&quot;purple&quot;</td>
</tr>
</tbody>
</table>
Array indices

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>&quot;red&quot;</td>
<td>&quot;green&quot;</td>
<td>&quot;blue&quot;</td>
</tr>
</tbody>
</table>

We use indices to access array values

```
colours[2] = "purple";
```

After this assignment statement, the value stored in slot *numbered* 2 of the array changes to "purple". Note because we start counting at 0, this is the *third* slot of the array.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>&quot;red&quot;</td>
<td>&quot;green&quot;</td>
<td>&quot;purple&quot;</td>
</tr>
</tbody>
</table>
SumRQScores

To sum up our total RQ score, we'll have an array of those scores. A call might look like:
`sumRQScores([1, 2, 1, 0, 1, 2]);`
sumRQScores might look like

```javascript
function sumRQScores(RQScores)
{
    var index;
    var sum = 0;
    var length = RQScores.length;
    for (index = 0; index < length; index++)
    {
        sum = sum + RQScores[index];
    }
    return sum;
}
```
Let's assume that you had a list of everyone's percent scores. Write a function to find the class average. You might call this as:

calculateAverage([97, 22, 71, 40, 91, 42]);
calculateAverage might look like

```javascript
function calculateAverage(scores){
    var size = scores.length;
    var sum = 0;
    var index;
    for (index = 0; index < size; index++){
        sum = sum + scores[index];
    } //end for
    return sum/size;
} //end function calculateAverage

Note: we can also use this to calculate an average RQ score for a student!
```
Now we can put all this together

- Let's create `calculateRQPercentFromRaw`
- Given an array of RQ scores, we can calculate the total RQ score based on the things we've done above.
- We might call it as:
  ```
calculateRQPercentFromRaw([1, 2, 1, 0, 1, 2])
```
calculateRQPercentFromRaw might look like

```javascript
function calculateRQPercentFromRaw(scores){
    var total = sumRQScores(scores);
    var numScores = scores.length;
    return calculateRQScoreGivenNumRQs(
        total,numScores);
}
```
Last, but not least, please evaluate your TAs

- Only evaluate TAs you've had contact with
Your TAs & their pictures – plus Paul Bucci, who is here (even better than a picture!)

<table>
<thead>
<tr>
<th>Alexia Lou</th>
<th>Andrey Novitskiy</th>
<th>Étienne Hossack</th>
<th>Rebecca Lee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jenny Lian</td>
<td>Rebecca McKnight</td>
<td>Sunny Liu</td>
<td>Yuzan Yang</td>
</tr>
<tr>
<td>RQs</td>
<td>L2C</td>
<td>L2B</td>
<td>L2F</td>
</tr>
</tbody>
</table>