A Note About Strings

Manipulating Strings in JS

- String Object
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String
- let str = "hello my name is Inigo Montoya";
- str.length; will return the length of the string (how many char, including spaces)
- str.indexOf('a'); will return the position of the first 'a' in the str
  - str.lastIndexOf('a') returns the position of the last 'a'
- str.substring(6,16); returns "my name is"
  - str.substring(17); returns "Inigo Montoya"
- let newStr = str.replace('Inigo', 'Inconceivable')
  - newStr will now contain the string 'hello my name is Inconceivable Montoya"

Manipulating Strings in JS

- `let str = ‘Inigo Montoya’;`
- We can convert a string to uppercase and lowercase
- `let bigStr = str.toUpperCase();`
  - `bigStr will be ‘INIGO MONTOYA’`
- `let lilStr = str.toLowerCase();`
  - `lilStr will be ‘inigo montoya’`
- We can get a single character out of string
- `str.charAt(0); will return I`

```
Ex.
let name = “”;
name += bigStr.charAt(0);
name += bigStr.charAt(6);
name += lilStr.substring(7,11);

Now what is stored in name?
```

Loops!

What’s a Loop?

- Functionality
  - A segment of code execute repeatedly when the loop’s Boolean expression is true
- Loops are useful for
    - Processing lists
    - Repeating steps, possibly with different parameters each time
    - Generating values, possibly with a different parameter each time.
- If statement allows us to skip code
- loops allows use to **repeat** code

- **Infinite loops** are loops that do no terminate
  - sometimes this is intentional, most of the time this is a programming error.

3 Kinds of Loops

- While loop
  - Used when we do NOT KNOW how many times we want the loop to run
- Do While loop
  - Used when we know we want the loop to run AT LEAST ONCE
- For loop
  - Used when we know EXACTLY how many times we want the loop to run

- NOTE: just like if statements, if you don’t use {} the loop naturally only associates with the first statement following it

While loop

- While loops are used if the number iterations that is needed to be executed is unknown
- While loops often require us to create a counter
  - This counter allows us to track how many times this loop has executed
- If we can’t or don’t want to use a counter, while loops are also appropriate
  - Ex. Stop when you find a value
  - Ex. Run till you reach the end of the string

While Loop Syntax

```
while(<boolean condition>)
{
   //loop body goes here
}
```

- Execution order

1. If the expression is false, run the statement directly after the loop body
2. Else run the loop body
3. Go back to step 1

- Modifiers
  - A **break** statement inside the body will cause the execution to leave the loop, and execute the line directly after the loop body
  - a **continue** statement inside the body will cause execution to immediately jump to step 1, ignoring the rest of the body

Sample While Loop

- Suppose we want to ask the user for a bunch of different numbers, and when the user enters 0 we want to print out the sum of all the user entered numbers

```
let userInput = +prompt("Enter some numbers, 0 
to stop: ");
let sum = 0;
while (userInput !== 0){
   sum += userInput;
   userInput = +prompt("Enter some numbers, 0 
to stop: ");
}
```

Pestering Users

- While loops can be used to repeatedly ask user for input until the give you valid input
- Think back to our Leap Year example from worksheet 3

```
let userInput = +prompt("Enter a year greater than 1582");
while (userInput < 1582){
  userInput = +prompt("Less than 1582. Try again!");
}
```

Solving Problems with Unknown Iterations

- While loops can also be used to solve problems where
  - We know the stop condition
  - But don't know how many iterations to run the loop
- Example problems:
  - Finding the first prime number after N
  - Checking if a word is contained inside a string

The Do While Loop

- The do while loop is usually used if you know you want the loop to execute AT LEAST ONCE
- Very similar to while loops, except while loops may execute zero times
- Do while loops execute **one** or **more** times

```
do{
   /// Body goes here
}while (<Boolean condition>);
```

Execution Order of a Do While Loop

```
do{
   /// Body goes here
}while (<Boolean condition>);
```

1. Execute the code body
2. If condition is not true, terminate the loop and execute the statement directly after the loop body
3. Else return to step 1

Remember Our Sample While Loop?

- It actually makes more sense to write this as a do while loop!

```
let userInput = +prompt(“Enter numbers, 0 to stop: ”);
let sum = 0;
while (userInput !== 0){
   sum += userInput;
   userInput = +prompt(“Enter again, 0 to stop: ”);
}
```

```
let userInput = 0;
let sum = 0;
do{
        userInput = +prompt(“Enter numbers, 0 to stop: ”);
        sum += userInput;
}while (userInput !== 0);
```

The For Loop

- The for loop is usually used if you know the number of iterations or maximum number iterations is known
- Typically only one loop index is required
- Unless the condition is wrong, for loops rarely result in an infinite loop
- For loops can execute **zero** or more times

```
for(initializer; condition; increment/decrement)
{
   /// Body goes here
}
```

Execution Order of a For Loop

```
for(initializer; condition; increment/decrement)
{
   /// Body goes here
}
```

1. Run initialization code.
2. If condition is not true, terminate the loop and execute the statement directly after the loop body
3. Else run the body
4. Run the increment or decrement code
5. Go back to Step 2

Sample For Loop

- For example, if we wanted to print the numbers 1 to 10 to the console

```
let i;
for(i = 1; i <= 10; i++){
  console.log(i);
}
```

You Try

- Write a for loop that will iterate from 0 to 20. For each iteration, it will check if the current number is even or odd, and report that to the screen (e.g. "2 is even")

```
let j;
for(j = 0; j <= 20; j++){
  if(j % 2 === 0)
    console.log(j + " is even");
  else console.log(j + " is odd");
}
```

Loops to build strings

- We can use loop to build strings.
- A simple example is to add something multiple times to a string.

- We can also use it to build up a large chunk of HTML to be put into the HTML page.

Reversing a String

- Write some JavaScript that reads a string from the user, then alerts the reverse of that string back to the user
- Ex. If the user enters “watch me whip” you will alert “pihw em hctaw”

```
let input = prompt("enter a string for me to reverse");

let current;
let reverse = '';

for(current = input.length - 1; current >= 0; current--)
   reverse += input.charAt(current);

alert(reverse);
```

Nested For Loops

- You can use loops inside of loops
- When you use nested for loops think of it like a clock's minute and hour hand
- Example: Use nested for loops to print a right triangle of asterisks to the console like this:

```
*
**
***
****
*****
```

- Where the height of the triangle is specified by the user (so in this case the user entered 5)

Solution

```
let row, star;
let toPrint = '';
const MAX_ROWS = +prompt('Enter the number of rows of stars to print');

for (row = 1; row <= MAX_ROWS; row++)
{
    for (star = 1; star <= row; star++)
        toPrint += "*";
toPrint += '\n';
}
console.log(toPrint);
```

Exercises

1. Write a JavaScript that reads a string from the user, and determines if the string is a palindrome.
   - A palindrome is a string that is the same forwards and backwards. Ex. RACECAR.
   - If the user has entered a palindrome you should alert “You entered a palindrome!”, otherwise you should alert “This is not a palindrome”.
2. Write the JavaScript to read a number from the user and then generate the triangle given below, where the height of the triangle depends on the number entered by the user. So if the user enters 5, you should print:

```
*****
****
***
**
*
```

Solution - Palindrome

```
let input = prompt("enter palindrome");
let current, reverse = '';

for(current = input.length - 1; current >=
0; current--)
   reverse += input.charAt(current);

if(reverse === input)
   alert(“This is a palindrome”);
else alert(“This is not a palindrome”);
```

Solution – Upside Down Triangle

```
let row, star;
let toPrint = '';
const MAX_ROWS = +prompt('Enter the number of rows of stars to print');

for (row = MAX_ROWS; row >= 0; row--)
{
    for (star = 1; star <= row; star++)
        toPrint += "*";
toPrint += '\n';
}
console.log(toPrint);
```

Rock Paper Scissors

- Write the JavaScript to simulate the computer playing Rock, Paper, Scissors with a user.
- Your JavaScript should generate the computer's selection of either rock, paper, or scissors.
- Then you should read in the user's choice of either "Rock", "Paper", or "Scissors".
- You should print to the screen what these two selections were, and who won, before asking if the user wants to play again.
- The user should be able to play the game as many times as they choose.

Exercise

- Write the JavaScript to read a number from the user and then generate the triangle given below, where the height of the triangle depends on the number entered by the user. So if the user enters 5, you should print:

```
*****
 ****
  ***
   **
    *
```

Exercise

- Design and implement an application that plays the Hi-Lo guessing game with the user
- The program should pick a random number between 1 and 100 (inclusive), then repeatedly prompt the user to guess the number.
- On each guess, report to the user that he or she is correct or that the guess is high or low.
- Continue accepting guesses until the user guesses correctly or chooses to quit. Use a sentinel value to determine whether the user wants to quit.
- Count the number of guesses and report that value when the user guesses correctly.
- At the end of each game (by quitting or a correct guess), prompt to determine whether the user wants to play again.
- Continue playing games until the user chooses to stop