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.
- **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

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/index allows us to track how many times this loop has executed
    - Ex. Run till you reach the end of the string
- If we can't or don't want to use a counter, while loops are also appropriate
    - Ex. Stop when you find a value

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

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 numbers, 0 to stop: ”);
}
console.log(sum);
```

PESTERING USERS

- While loops can be used to repeatedly ask user for input until they give you valid input
- Think back to our Leap Year exercise

```
let userInput = +prompt(“Enter year 1582 or later”);
while (userInput < 1582){
    userInput = +prompt(“Invalid year. 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 problem:
  - Finding the first prime number after N
  - Letting the user enter as many values as they want

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

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?

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

REMEMBER OUR SAMPLE WHILE LOOP?

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

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

SAME WITH PESTERING USERS

- It also makes more sense to solve our leap year problem using a do while loop

```
let userInput = 0;
do{
     userInput = +prompt(“Enter greater than 1582.”);
}while(userInput < 1582);
```

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

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

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

OPERATORS REVISITED

- The Unary Operators
  - Effect ONE expression, not two like binary operators do
- !
  - negation
- + and –
  - Positive (or numerical type conversion) and negative
- ++ and --
  - Increment and decrement
  - x++ means x = x + 1;

```
let bool = 15 + 2 > 20;


let user = +prompt("enter a number.");
user = -user;


user++;
user--;
```

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 print that to the console
  Sample Output:
  ```
  0 is even
  1 is odd
  2 is even
  3 is odd
  …
  20 is even
  ```

SOLUTION

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

LOOPS TO BUILD STRINGS

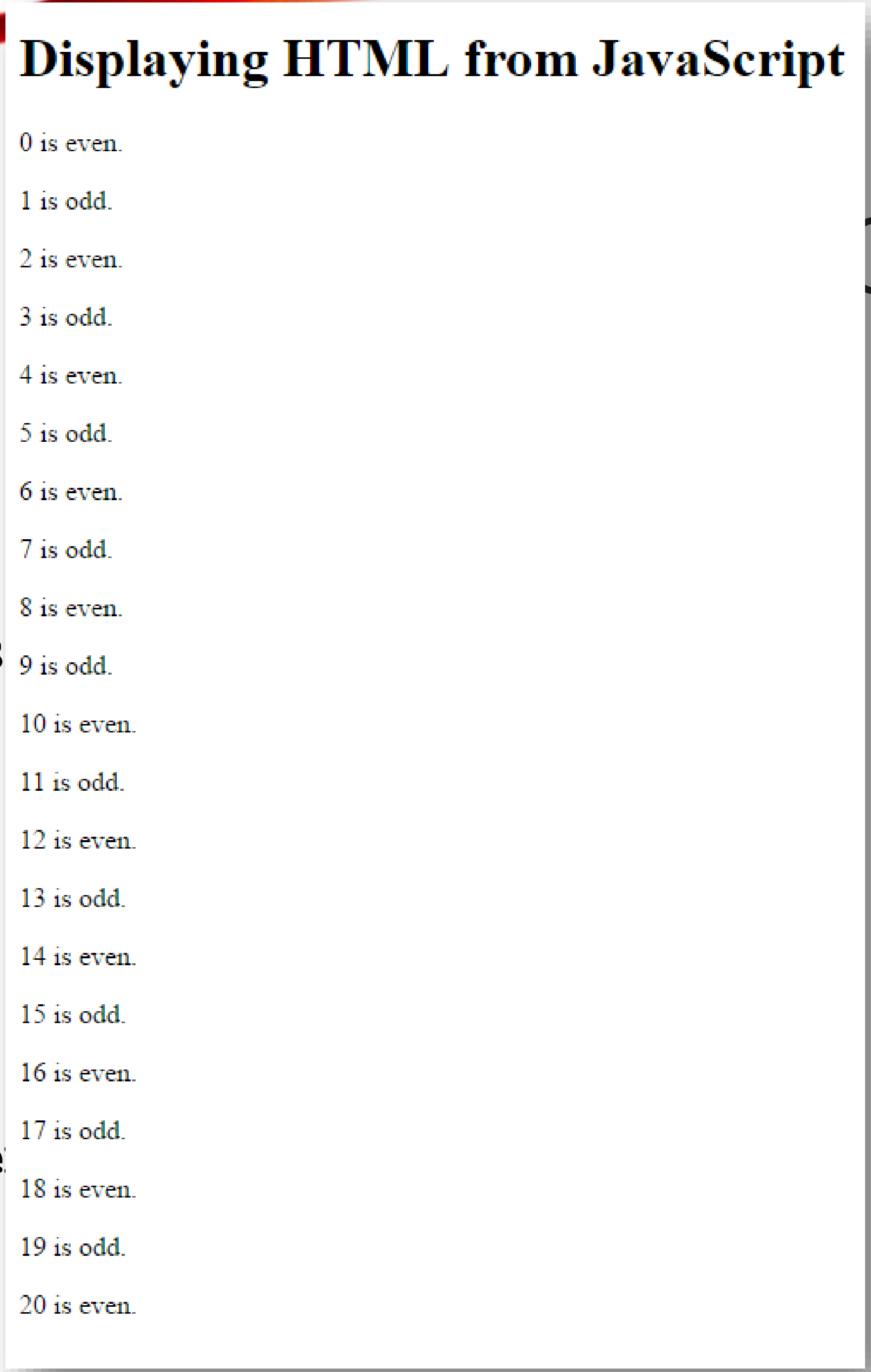
- We can use loop to build strings.
- A simple example is to add something multiple times to a string.
- We can also use it to build up a large chunk of HTML to be put into the HTML page.
- What if we had wanted to display the results of the previous exercise in paragraphs on the HTML page instead of the console?

SO LET’S MODIFY IT

```
let str = “”;

for(let j = 0; j <= 20; j++){
     str += “<p>”;
     if(j % 2 === 0){
          str += j + " is even";
     }else{
          str+= j + " is odd";
     }
     str += “</p>”;
}

let div = document.getElementById(“myDiv”);
div.innerHTML = str;
```

DIFY IT

```
let str = “”;

for(let j = 0; j <= 20; j
      str += “<p>”;
      if(j % 2 === 0){
            str += j + " is
      }else{
            str+= j + " is
      }
      str += “</p>”;
}

let div = document.getEle
div.innerHTML = str;
```



Displaying HTML from JavaScript

0 is even.

1 is odd.

2 is even.

3 is odd.

4 is even.

5 is odd.

6 is even.

7 is odd.

8 is even.

9 is odd.

10 is even.

11 is odd.

12 is even.

13 is odd.

14 is even.

15 is odd.

16 is even.

17 is odd.

18 is even.

19 is odd.

20 is even.

EXERCISE: REVERSING A STRING

- Write a function that accepts a string as a parameter and returns the reverse of that string
- Ex.

```
stringReverse(“watch me whip”);
should return “pihw em hctaw”
```

SOLUTION

```
function stringReverse(str){

   let reverse = "";

   for(let i = str.length - 1; i >= 0; i--){
      reverse += str.charAt(i);
   }
   return reverse;
}
```

ALTERNATIVE SOLUTION

```
function stringReverse(str){

   let reverse = “”;

   for(let i = 0; i < str.length; i++){
      reverse = str.charAt(i) + reverse;
   }
   return 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, where the height of the triangle is specified as a parameter
- Ex. **`printTriangle(5);`**

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

SOLUTION

```
function printTriangle(max){
   let toPrint = “”;

   for (let row = 1; row <= max; row++)
   {
      for (let star = 1; star <= row; star++)
            toPrint += "*";
      toPrint += “\n”;
   }
   console.log(toPrint);
}
```

EXERCISES

1. Write a JavaScript function that accepts a string as a parameter and returns true 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 function that accepts a number as a parameter and prints the triangle shown, where the height of the triangle depends on the number passed as a parameter.
    - **`printTriangle(5)`** would print:

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

SOLUTION - PALINDROME

```
function isPalindrome(str){
   let reverse = “”;

   for(let i = str.length - 1; i >= 0; i--)
   {
      reverse += str.charAt(i);
   }
   return reverse === str;
}
```

SOLUTION - PALINDROME

- But wait, aren’t we just doing the same work we did for stringReverse ?

```
function isPalindrome(str){
   let reverse = “”;

   for(let i = str.length - 1; i >= 0; i--)
   {
      reverse += str.charAt(i);
   }
   return reverse === str;
}
```

SOLUTION - PALINDROME

- Yep! Let's take advantage of the fact stringReverse already exists!

```
function isPalindrome(str){

   let reverse = stringReverse(str);

   return reverse === str;
}
```

ALTERNATE SOLUTION

```
function isPalindrome(str){
     let length = str.length;
     for(let i = 0; i < length / 2; i++){
          if(str.charAt(i) != str.charAt((length-i-1))){
               return false;
          }
     }
     return true;
}
```

SOLUTION – UPSIDE DOWN TRIANGLE

```
function printTriangle(max){
   let toPrint = “”;

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

OPERATORS REVISITED

- ? :
  - The ternary operator
  - Effects THREE expressions
  - Basically a short handed if else statement

```
(<CONDITION>)? //DO IF TRUE : //DO IF FALSE;

let user = prompt(“Enter your gender: “);
(user === “male”)? alert(“you are male”) : alert(“you are female”);
OR
alert((user === “male”)? “you are male” : “you are female”);
```

EXERCISE

- Write a statement using the ternary operator that checks if a variable called age is less than 50.
- If the age is less than 50 print the statement “You are young!”, otherwise print the statement “You are old!”.

HIGH-LOW GUESSING GAME

- 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