Topic/Title: Random Number Generation in JavaScript: Building a Love Calculator
Control Statements: Using If-Else Conditionals & Logic
Comparators and Equality
Combining Comparators

Keywords/Questions:

Math.random();

===

<=

>=

!==

>

<

==

!=

&&

||

!

Notes:

Math.random(); :- generates a random number between 0 - 0.9999999999999999. The 9s are 16 digits. It might be 0 but it will never reach 1.

=== :- used for testing for equality in JavaScript.

- known as the "Is equal to" sign. === Is equal to

- if (hale===5){console.log(5);} :-This means if hale is equal to 5, print 5.

!== :- used for testing for inequality in JavaScript.

-known as the "Is not equal to" sign.

-usually used and executed if the left side is not equal to the right

=== Is equal to
!== Is not equal to
> Is greater than
< Is lesser than
>= Is greater or equal to
<= Is lesser or equal to

The difference between == and ===. A === equal sign checks if the data types are matching whereas a == sign doesn't.

```
> var a = 1;
  var b = "1";
< undefined
> typeof(a);
< "number"
> typeof(b);
< "string"
```

```
> if (a === b) {
    console.log("yes");
  } else {
    console.log("no");
  }
no
```

-The same rule applies for the != sign and !===

```
> if (a == b) {
    console.log("yes");
  } else {
    console.log("no");
  }
yes
```

Summary:

Math.random(); :-generates a random number from 0 to 0.9999999999999999. Sixteen 9s.

Conditionals:- if (x>5){} -else if(x>3 && x<=5){} -else{}

===:- equal to !==:- not equal to > :-greater than <:-less than >=:-greater or equal to <=:-lesser or equal to

== & != :-don't check if they have same data types === & !== :- check if they have same data types

&& :- "AND" combiner

|| :- "OR" combiner

! :- "NOT" combiner

More Notes

-Best way to write if else statements

```
if (track === “clear”){
  goStraight();
} else {
  turnRight();
}
```

&&:-the "AND" combiner.

||:-the "OR" combiner.

!:- the "NOT" combiner.

```
if (loveScore > 30 && loveScore <= 70) {
    alert("Your love score is " + loveScore + "%");
}
```

-we only use one bracket inside the if clause

Topic/Title: Collections: Working with JavaScript Arrays
Adding Elements and Intermediate Array Techniques
Control Statements: While Loops
Control Statements: For Loops

Keywords/Questions:

array.length;

array.includes(item);

array.push(item);

array.pop;

while(codition){//Do sth}

for(i=0; i<2; ++i{//Do sth}

for(var i=0; i<2; ++i){//Do sth}

Notes:

```
var myEgg = eggs[1];
```

-an array

```
var eggs = [ , , , , ]
```

eggs.length;

-tells you the length of an array

-this returns the number 5 because there are 5 items inside the array

```
eggs.includes( )
```

eggs.inludes(item1)

-We get a boolean returned.

```
eggs.push(    );
```

- .push(item1); :- adds an element to the end of the list.

```
eggs.pop;
```

- .pop; :-removes the last array element

jitbit.com/alexblog/249-now-thats-what-i-call-a-hacker/ :- a Russian hacker who automated his life.

while(condition){
//commands
}

```
var i = 1;
while(i<2) {
  console.log(i);
  i++;
}
```

Summary:

array.length; :-used to find length of an array

array.includes(item) :-return a boolean after checking if the item is inside the array.

array.push(item); :-adds an item to the end of an array.

array.pop; :- removes the last item of the array

while(condition){//Do sth} :- a while loop.

for(i=0; i<2; ++i){//Do sth} === for(var i=0; i<2;++i){//Do sth} - Both are equivalent for loops.

draw.io :- used to create flow charts.

More Notes

For Loops

```
      start   end change
        |       |     |
for (i=0; i<2; i++) {
  //Do something
}
```

```
while (something is true) {
    //Do something
}

for (i=0; i<2; i++) {
  //Do something
}
```

State

Iterate

- The above for loop is the same with for(*var i=0;* i<2; i++) {//Do sth}.

www.draw.io

- good website for drawing up flow charts.
- For example, let us draw a flow chart for a fibonacciGenerator function.

```
function fibonacciGenerator (n) {
    if(n==1){
        return [0];
    }else if(n==2){
        return [0, 1];
    }
    var output=[0, 1];
    for(i=2; i<n; ++i){
        output.push(output[i-1]+output[i-2]);
    }
    return output;
}
```