LOOPS

WHILE LOOPS

`while` will repeat the same code over and over until some condition is met.

```
var bottlesOfBeer = 99;

while (bottlesOfBeer > 0) {
    console.log(bottlesOfBeer + ' bottles of beer on the wall');
    bottlesOfBeer = bottlesOfBeer - 1;
}
```

WARNING: INFINITE LOOPS

Make sure something changes in the loop or your loop will go on forever.

ACTIVITY: WHILE LOOP

- Write a while loop that gives you the 9 times table, from **9 x 1 = 9** to **9 x 12 = 108**.
- Bonus: Try using a loop inside a loop to write all the times tables, from 1 to 12.

FOR LOOPS

`for` loops are very similar, but you declare a counter in the statement

```
// will count 1 to 10
for (var i = 1; i <= 10; i++) {
        console.log(i);
}
```

ACTIVITY: FOR LOOP

- Write a for loop that gives you the 9 times table, from **9 x 1 = 9** to **9 x 12 = 108**.
- Bonus: Try using a loop inside a loop to write all the times tables, from 1 to 12.

LOOPS AND LOGIC

You can add other statements or logical operators inside loops.

```
// Count from 1 to 100

for (var i = 1; i <= 100; i++) {
        if (i % 3 === 0) {
                // Says 'Fizz' after multiples of three
                console.log(' Fizz');
        } else if (i % 5 === 0) {
                // Says 'Buzz' after multiples of five
                console.log(' Buzz');
        } else {
                console.log(i);
        }
}
```

ACTIVITY: LOGIC IN LOOPS

- Write a for loop that will iterate from 0 to 20.
- For each iteration, check if the current number is even or odd, and report that to the screen (e.g. "2 is even", "3 is odd")

Hint: Remember that modulus operator?

BREAK STATEMENT

To exit a loop, use the `break` statement.

```
// Count from 100 to 200
for (var i = 100; i <= 200; i++) {
    console.log('Testing ' + i);

    //Stop at the first multiple of 7
    if (i % 7 == 0) {
        console.log('Found it! ' + i);
        break;
    }
}
```

ACTIVITY: BREAKING LOOPS

- Go back to your times table loop.
- For some reason, you really hate the number 6.
- Break the loop before you print out the number 6.

Bonus: console.log the phrase "I hate the number 6" before breaking the loop.

ARRAYS

ARRAYS

Ordered lists of values.

```
var arrayName = [value0, value1];

var rainbowColors = ['Red', 'Orange', 'Yellow', 'Green',
    'Blue', 'Indigo', 'Violet'];

var lotteryNumbers = [33, 72, 64, 18, 17, 85];

var myFavoriteThings = ['Broccoli', 1024, 'Sherlock'];
```

ACTIVITY: CREATE AN ARRAY

- Create an array of your favorite foods.
- console.log the array.

ARRAY LENGTH

`length` property tells you how many items are in an array.

```
var rainbowColors = ['Red', 'Orange', 'Yellow', 'Green',
'Blue', 'Indigo', 'Violet'];

console.log(rainbowColors.length); // outputs 7
```

ACTIVITY

console.log the length of your favorite foods array.

USING ARRAYS

Access items in an array with **bracket notation** by using the position of the item you want.

```
var rainbowColors = ['Red', 'Orange', 'Yellow', 'Green',
'Blue', 'Indigo', 'Violet'];

var firstColor = rainbowColors[0];  // outputs "Red"
var lastColor  = rainbowColors[6]; // outputs "Violet"
```

JS arrays are zero-indexed. Counting starts at 0.

ACTIVITY

1. Make sure your favorite foods array has 5 items.
2. Log the 3rd item in your array.

CHANGING ARRAYS

Use **bracket notation** to change an item in an array.

```
var myFavoriteThings = ['Ice Cream', 16, 'Doctor Who'];

myFavoriteThings[0] = 'Apples';

console.log(myFavoriteThings); // outputs ['Apples', 16, 'Doctor Who']
```

ACTIVITY

1. Replace the 3rd food item in your array with "Asparagus".
2. Log your array.

EXPANDING ARRAYS

Arrays have no fixed length. You can use `push` to add an item to the array.

```
var myFavoriteThings = ['Ice Cream', 16, 'Doctor Who'];

myFavoriteThings.push('Apples');

console.log(myFavoriteThings); // output ['Ice Cream', 16, 'Doctor Who', 'Apples']
```

ACTIVITY

1. Add another food item to the end of your favorite foods array.

ITERATING THROUGH ARRAYS

Use a `for` loop to easily work with each item in the array.

```
var rainbowColors = ['Red', 'Orange', 'Yellow', 'Green',
'Blue', 'Indigo', 'Violet'];

for (var i = 0; i < rainbowColors.length; i++) {
        console.log(rainbowColors[i]);
}
```

ACTIVITY: FOR LOOP

- Use a `for` loop to print a list of all your favorite foods.

OBJECTS

OBJECTS

Objects let us store a collection of properties.

```
var objectName = {
        propertyName: propertyValue  // key: value pair
};

var user = {
        hometown: 'Atlanta, GA',
        hair: 'Brown',
        likes: ['gaming', 'code'],
        birthday: {month: 06, day: 18}
};
```

ACTIVITY: CREATE AN OBJECT

Create an object to hold information on your favorite recipe.

It should have properties for:

- `recipeTitle` (a string)
- `recipeDescription` (a string with multiple sentences)
- `ingredients` (an array of strings)
- `directions` (a string)
- `rating` (a number between 1 and 5)
- `cook time` (a number to indicate how many minutes it takes to cook)

ACCESSING OBJECTS

You can retrieve values using dot notation

```
var user = {
        hometown: 'Atlanta, GA',
        hair: 'Brown'
};

var usersHometown = user.hometown;
```

Or using bracket notation (like arrays).

```
var usersHair = user['hair'];
```

ACTIVITY

Try displaying some information (values) about your recipe in the console.

CHANGING OBJECTS

You can use dot or bracket notation to change properties.

```
var user = {
        hometown: 'Atlanta, GA',
        hair: 'Brown'
};

user.hair = 'Blue';
```

Add new properties.

```
user.married = true;
```

Or delete properties.

```
delete user.married;
```

ACTIVITY: CHANGE YOUR RECIPE

1. Go back to your recipe object.
2. Add a `servings` property (a number)
3. Change the `rating` property value
4. Delete the `cook time` property

ARRAYS OF OBJECTS

Because arrays can hold any data type, they can also hold objects.

```
var users = [
        {name: 'Jolene', age: 21},
        {name: 'Alexa',  age: 18}
];

for (var i = 0; i < users.length; i++) {
        var user = users[i];
        console.log(user.name + ' is ' + user.age + ' years old.');
}
```

ACTIVITY: ARRAYS OF OBJECTS

1. Create a movies array with 2 objects that each have 2 properties:
    - movie name
    - movie rating
2. Write a for loop that logs out "I give [MOVIE NAME] [MOVIE RATING] stars" for each movie object in the array.

"I give Black Panther 5 stars."

"I give Snowpiercer -16 stars."

OBJECTS

Just like other data types, objects can be passed into functions:

```
var jolene = {
        age: 21,
        hairColor: 'Auburn',
        likes: ['pizza', 'tacos'],
        birthday: {month: 3, day: 14, year: 1995}
}

function describeUser(user) {
        console.log('You are ' + user.age + ' years old with '
        + user.hairColor + ' hair.');
}

describeUser(jolene);
```

ACTIVITY: OBJECTS IN FUNCTIONS

1. Go back to your recipe object.
2. Create a function that logs out the recipe title and servings.
3. Call your new function and pass in your recipe object as an argument.

OBJECT METHODS

Objects can also hold functions.

```
var jolene = {
        age: 21,
        hairColor: 'Auburn',
        talk: function() {
                console.log('Hello!');
        },
        eat: function(food) {
                console.log('Yum, I love ' + food);
        }
};
```

Call object methods using dot notation:

```
jolene.talk();

jolene.eat('pizza');
```

ACTIVITY: ADD A FUNCTION

- Go back to your recipe object.
- Add a function called `letsCook` that says "I'm hungry! Let's cook..." with the name of your recipe title.
- Call your new method.

BUILT-IN OBJECTS

JS provides several built-in objects:

- Array
- Number
- Date
- Math
- String

  Soooooooooo many useful things!

ARRAY OBJECT

```
array.length // returns number of items in array
array.push(value) // adds new value to array
array.concat(array2) // merges two or more arrays
array.slice(beginningPosition)  // Extracts section of array and returns new array
array.join()  // joins all elements of array into a string
array.reverse() // reverses order of the elements in an array
array.indexOf(value) // returns the first index of an element (value)
array.lastIndexOf(value) // returns last index of an element

array.every(functionName) // do all elements in array pass the test
array.some(functionName) // do some elements in  array pass the test
array.filter(functionName) // creates new array from elements that pass the test
array.forEach(functionName) // performs an operation on each element
array.map(functionName) // performs an operation on each element; returns a new array
```

```
var array = ["cats", "dogs", "elephants", "rabbits"];

array.reverse();
console.log(array); // outputs ["rabbits", "elephants", "dogs", "cats"]
```

Array

ACTIVITY: ARRAY OBJECT METHODS

1. Create 2 arrays
   - list of favorite movies
   - list of favorite books
2. Log the length of each array
3. Create a new variable `moviesAndBooks` and assign the array created by merging your 2 arrays
4. Create a function `isTitleLong` that returns true if the argument (`title`) is longer than 6 characters
5. Create a new variable `moviesWithLongTitles` that assigns the array created with all movies that pass the isTitleLong test. (filter)

Use this resource for more details

NUMBER OBJECT

```
toFixed() // cuts off a number after a certain point

myNumber.toFixed(2);
```

Number

ACTIVITY: FIXED NUMBERS

1. Create a variable pi that equals 3.14159
2. Log the results of cutting off pi after 2 decimal points

Number

DATE OBJECT

```
getDate() // The day of the month
getDay() // The day of the week as an integer (Sunday 0, Monday 1)
getMonth() // The month as an integer (January as 0, February as 1)
getFullYear() // The year as a four-digit number
toDateString() // Returns the full date based on the current time zone as a human-
readable string, for example, “Wed 31 Dec 2003

setDate()
setMonth()
setFullYear()

var date = new Date("31 January 2014");

date.getDate();
```

Date

ACTIVITY: DATES

1. Create a new Date object with today's date and assign it to a variable `today`
2. Log the day of the month
3. Create a function `dayOfWeek` that takes a number between 0 and 6 and returns the associated day. For example, if you call dayOfWeek(0), the function returns "Sunday"
4. Use your new function to get today's day of the week and log it.

Date

MATH OBJECT

```
Math.pi
abs() // returns absolute value of a number
min() // returns number with the lowest value
max() // returns  number with the highest value
ceil() // rounds a number up to the next largest whole number or integer
floor() // removes any numbers after decimal point; returns whole number
round() // rounds up if the decimal is .5 or greater, else rounds down
random() // returns a random floating-point number (0-1, excluding 1)
pow() // raises a number to a specified power

console.log(2 * Math.pi);
```

Math

ACTIVITY: MATH

1. Create a function `randomNumber` that returns a random number between 1 and 100.
2. Create a variable called `randomPi` and assign it the value of a randomNumber between 1 and 100 multiplied by pi.
3. Log the value of that variable after rounding it up.

Math

STRING OBJECT

```
length // returns the number of characters in the string
indexOf() and lastIndexOf() // find character position in array
substr() and substring() // get part of string
toLowerCase() and toUpperCase()
charAt() // get the character at a particular position
fromCharCode() // convert Unicode number into character
trim() // remove whitespace from string

var string = "Hello World!";
console.log(string.toLowerCase()); // "hello world!"
```

ACTIVITY: STRINGS

1. Create a variable `myName` and assign it the value of your first and last name.
2. Find the length of your string.
3. Log the lowercase version of your name.
4. Use substr() to log your last name
5. Use substring() to log your last name.

ACTIVITY: RECIPE PAGE

- Go back to your recipe object.
- Add a function that:
    - Takes the recipe object as an argument.
    - Creates a variable with today's date.
    - Displays a recipe with:
        - Recipe Title
        - Date Published (today's date) in any format you choose
        - The first 50 characters of the directions with "..." at the end.

Salted Caramel Brownies

Published: January 2018

Perfectly fudgy brownies that have a great texture...