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)

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;
```

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!

THE DOM

ANATOMY OF A WEBSITE

Your Content
+ HTML: Structure
+ CSS: Presentation
+ JS: Behavior
= Your Website

IDS VS CLASSES

ID - Should only apply to one element on a webpage.

```
<nav id="nav"></nav>
```

```
#nav {
    /* CSS here */
}
```

Class - Lots of elements can have the same class.

```
<ul>
    <li class="list-item">Content Here</li>
    <li class="list-item">Content Here</li>
    <li class="list-item">Content Here</li>
    <li class="list-item">Content Here</li>
    <li class="list-item">Content Here</li>
</ul>
```

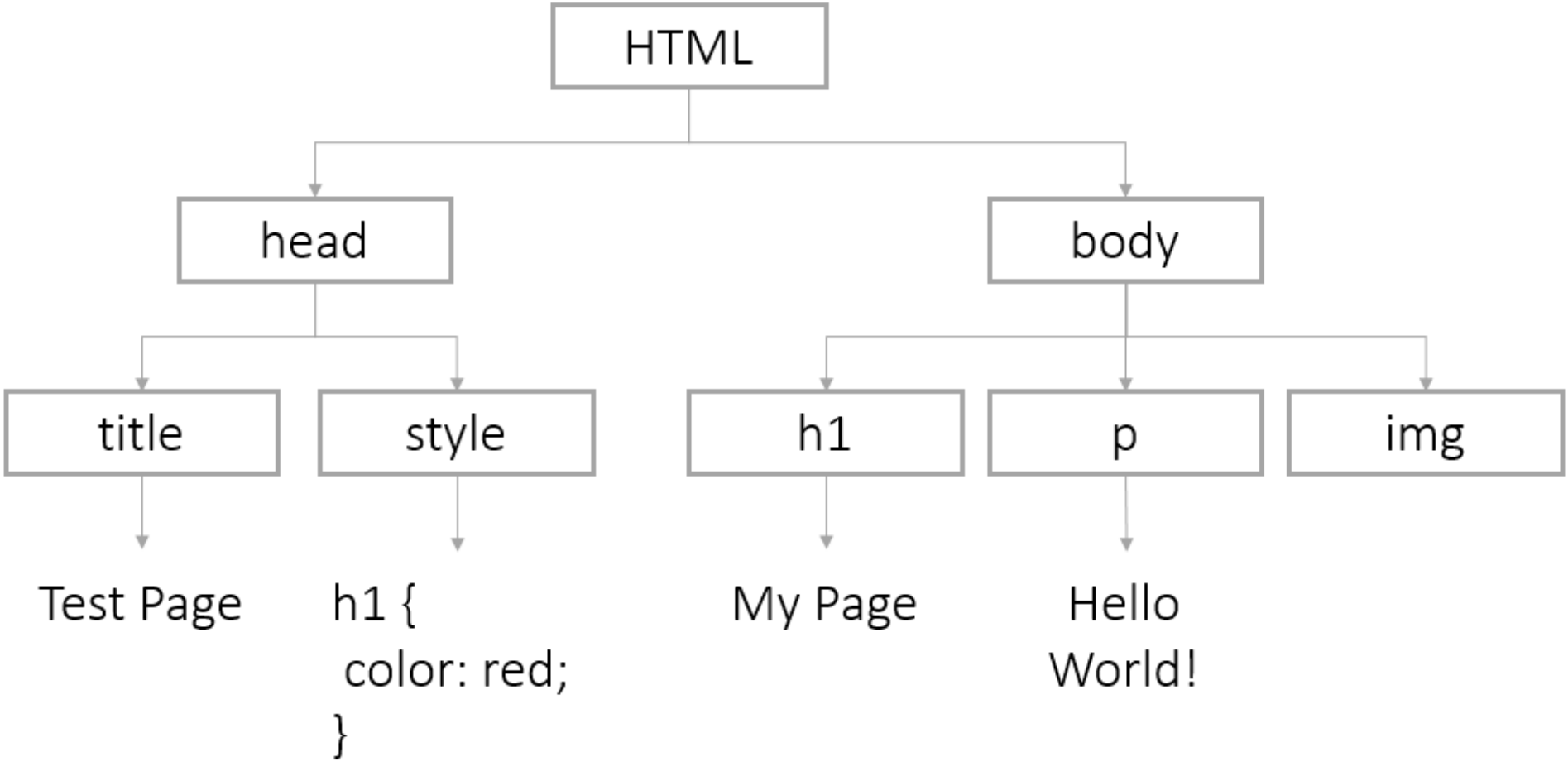
```
.list-item {
    /* CSS here */
}
```

THE DOM TREE: SAMPLE CODE

```
<!DOCTYPE html>
<html>
    <head>
        <title>Test Page</title>
        <style>
        h1 {
            color: red;
        }
        </style>
    </head>
    <body>
        <h1>My Page</h1>
        <p>Hello World!</p>
        <img src="http://placekitten.com/g/200/300" alt="cat"/>
    </body>
</html>
```

THE DOM TREE: SAMPLE MODEL

Any HTML document is a tree structure defined by the **DOM (Document Object Model)**.



DOM ACCESS

Your browser automatically loads the content of a webpage into a `Document` object which serves as the entry point into a web page's content.

Using the `document` you can:

1. Change the content tree any way you want.
2. Build an HTML document from scratch.
3. Access or replace any existing DOM nodes (HTML elements in the DOM).

ACTIVITY: HTML

Create a simple HTML page or use this sample code.

```
<!DOCTYPE html>
<html>
<head>
    <title>Test Page</title>
</head>
<body>
    <div id="wrapper">
        <div id="header">
            <h1>JavaScript Test Site</h1>
            <nav>
                <ul>
                    <li class="nav-item">About</li>
                    <li class="nav-item">Services</li>
                    <li class="nav-item">Contact</li>
                </ul>
            </nav>
        </div>
        <div id="main">
            <p>I learned about JavaScript in a SAIT class.</p>
            <img src="https://picsum.photos/500/300?random
```

DOM ACCESS: BY ID

You can find nodes by `id` using the method:

```
document.getElementById(id);
```

For example, to find:

```
<img id="kittenPic" src="http://placekitten.com/g/200/300" alt="cat"/>
```

We would use:

```
var imgKitten = document.getElementById('kittenPic');
```

ACTIVITY: GET ELEMENT BY ID

- Create and link a script.js file to your index.html page
- create a variable `header`
- get the header element by id and assign it to `header`
- console.log(header);

DOM ACCESS: BY TAG NAME

You can also get HTML elements by their tag using this method:

```
document.getElementsByTagName(tagName);
```

To find:

```
<ul>
    <li>Daisy</li>
    <li>Tulip</li>
</ul>
```

We would use:

```
var listItems = document.getElementsByTagName('li');

for (var i = 0; i < listItems.length; i++) {
    var listItem = listItems[i];
}
```

ACTIVITY: GET ELEMENT BY TAG NAME

- Create variable `listItems`
- Get your list elements by tag name and assign it to `listItems`

DOM ACCESS: HTML 5

In newer browsers, you can use methods `getElementsByClassName`, `querySelector`, and `querySelectorAll`.

Available in IE9+, FF3.6+, Chrome 17+, Safari 5+:

```
document.getElementsByClassName(className);
```

Available in IE8+, FF3.6+, Chrome 17+, Safari 5+:

```
document.querySelector(cssQuery); // gets the first item that matches that selector
document.querySelectorAll(cssQuery); // gets all items that match the selector
```

ACTIVITY: GET ELEMENTS

- Get your list elements by class name and assign it to `listItems`
- Get your list elements by `querySelectorAll` and assign it to `listItems`
- Create a variable `firstItem` and use `querySelector` to assign the first list item

DOM ACCESS: BY CLASS NAME

You can also get HTML elements by their class using this method:

```
document.getElementsByClassName(className);
```

To find:

```
<ul>
    <li class="list-item">Daisy</li>
    <li class="list-item">Tulip</li>
</ul>
```

We would use:

```
var listItems = document.getElementsByClassName('list-item');

for (var i = 0; i < listItems.length; i++) {
    var listItem = listItems[i];
}
```

GETELEMENT VS. GETELEMENTS

Any method that starts with `getElement` will return a **single** node.

```
document.getElementById('uniqueID'); // returns a single node
```

Any method that starts with `getElements` will return an **array** of nodes. To modify a single node, you will need to use bracket notation to get the correct one.

```
document.getElementsByTagName('p'); // returns multiple nodes
var specificParagraph = document.getElementsByTagName('p')[2];
```

ACTIVITY: GET THE RIGHT ELEMENT

- Use `getElementsByTagName` and bracket notation to console.log the 2nd paragraph element

DOM NODES: ATTRIBUTES

You can access and change attributes of DOM nodes using dot notation.

To change this element:

```
<img id="kittenPic" src="http://placekitten.com/g/200/300" alt="cat"/>
```

We could change the src attribute this way:

```
var imgKitten = document.getElementById('kittenPic');

// will return src attribute on image
imgKitten.src

// will set our src to a new src
imgKitten.src = 'http://placekitten.com/g/600/500';
```

DOM NODES: GETTING AND SETTING ATTRIBUTES

You can also use `getAttribute` or `setAttribute`

```
<img id="kittenPic" src="http://placekitten.com/g/200/300" alt="cat">
```

We could change the src attribute this way:

```
var imgKitten = document.getElementById('kittenPic');

// will return src attribute on image
imgKitten.getAttribute('src');

// will set our src to a new src
imgKitten.setAttribute('src', 'http://placekitten.com/g/600/500');
```

DOM NODES: STYLES

You can change page css using `style`

To make this CSS:

```
body {
color: red;
}
```

Use this JavaScript:

```
var pageBody = document.getElementsByTagName('body')[0];

pageBody.style.color = 'red';
```

DOM NODES: MORE STYLES

The rule of thumb in JavaScript is to change CSS styles with a "`-`" to camelCase.

To make this CSS:

```
body {
background-color: pink;
padding-top: 10px;
}
```

Use this JavaScript:

```
var pageBody = document.getElementsByTagName('body')[0]

pageBody.style.backgroundColor = 'pink';

pageBody.style.paddingTop = '10px';
```

ACTIVITY: CHANGE AN ATTRIBUTE

Create a simple HTML page or use this sample code.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Test Page</title>
    </head>
    <body>
        <div id="wrapper">
            <div id="header">
                <h1>JavaScript Test Site</h1>
            </div>
            <div id="main">
                <p>I learned about JavaScript in a SAIT class.</p>
            </div>
            <div id="footer">
                    <p>This is my awesome footer.</p>
            </div>
        </div>
    </body>
    <script src="script.js"></script>
</html>
```

Isolate a node (an element on the page) and change an attribute or add a new style.

DOM INNERHTML

Each DOM node has an `innerHTML` property. Use this property to view or change the HTML of a node.

For example, you can overwrite the entire body:

```
var pageBody = document.getElementsByTagName('body')[0];

pageBody.innerHTML = '<h1>Oh Noes!</h1><p>I changed the whole page!</p>'
```

Or just add some new content to the end

```
pageBody.innerHTML += '...just adding this at the end of the page.';
```

DOM INNERHTML

You can also target one specific element's content

To put content in this paragraph element:

```
<p id="warning"></p>
```

We can select the node and modify it

```
var warningParagraph = document.getElementById('warning');

warningParagraph.innerHTML = 'Danger Will Robinson!';
```

CREATING NEW NODES

The `document` object also has methods to create nodes from scratch:

```
document.createElement(tagName);

document.createTextNode(text);

element.appendChild(element);
```

CREATING NEW NODES: SAMPLE CODE

```
var pageBody = document.getElementsByTagName('body')[0];

// create our image tag with attributes
var newImg = document.createElement('img');
newImg.src = 'http://placekitten.com/g/500/200';
newImg.style.border = '1px solid black';

// add our image to the body
pageBody.appendChild(newImg);

// create a paragraph tag with content
var newParagraph  = document.createElement('p');
var paragraphText = document.createTextNode('KITTY!');
newParagraph.appendChild(paragraphText);

// add our new paragraph to the body
pageBody.appendChild(newParagraph);
```

ACTIVITY: CREATE A PARAGRAPH

Create a new paragraph element and add it to a `div` on your page.