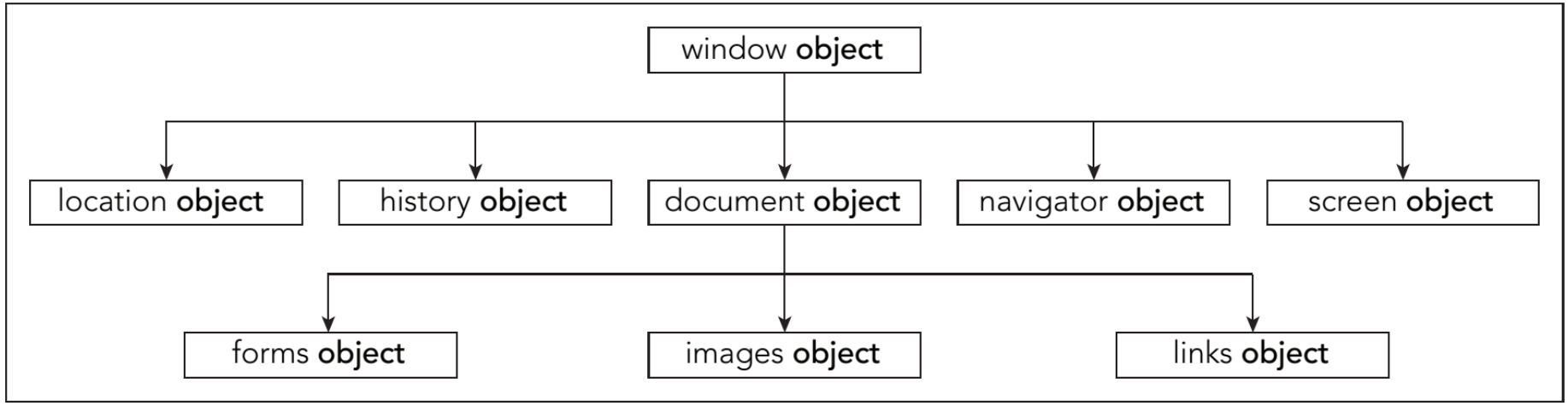
BROWSER OBJECTS

BROWSER OBJECTS

Examples

- window.alert()
- window.prompt()
- document.write()

BROWSER OBJECT MODEL



THE WINDOW OBJECT

- A global object (you don't need to use its name to access its properties and methods).

```
alert("Hello");
window.alert("Hello");
```

WHAT CAN YOU DO?

- find out what browser is running
- see the pages the user has visited
- find out the size of the user's screen
- change text in the browser's status bar
- change the page that is loaded
- open new windows

THE HISTORY OBJECT

- keeps track of each page that user visits
- enables Back and Forward buttons to revisit pages

```
history.length; // how many pages are in the history stack
history.back(); // go back 1 page
history.forward(); // go forward 1 page
history.go(-2); // goes back 2 pages
history.go(3); // goes forward 3 pages
```

ACTIVITY: BROWSER HISTORY

- Visit a few different pages in your browser
- In the console, type `history.back()`
- Next, try `history.forward()`
- What happens if you try `history.go(-2)`?

THE LOCATION OBJECT

- information about the current page's location
  - URL
  - server hosting page
  - port number
  - protocol

  (need to load a page from a server to see some of these)

```
location.replace("myPage.html"); // removes current page from history stack
and replaces it with new page
location.href = "myPage.html"; // goes to new page and adds it to the top of
the history stack
```

ACTIVITY: LOCATION LOCATION LOCATION

1. Create an index.html page.
2. Add this code between your body tags.

```
<button onclick=locationHref()>Href</button>
<button onclick=locationReplace()>Replace</button>
<script>
function locationHref() {
   location.href = "#top";
}

function locationReplace() {
   location.replace("#top");
}
</script>
```

3. Navigate to your index.html page. Hold your back button to see your browser history.
4. Click Href. Check your browser history. What has changed?
5. Hit the back button. Then, click Replace. Check your browser history. What's different this time?

THE NAVIGATOR OBJECT

- information about the browser and the operating system in which it's running
- often used to handle browser differences because it lets you see browser, version, OS the user has (browser sniffing)
- geolocation

ACTIVITY: BROWSER INFO

- Create a script.js file and link it to your index.html page.
- Console.log the user agent string for your browser `navigator.userAgent`;
- Console.log the vendor string for your browser `navigator.vendor`;
- Console.log the platform string for your browser `navigator.platform`;

GEOLOCATION

- obtain and use the position of the device or computer

```
function success(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var altitude = position.coords.altitude;
    var speed = position.coords.speed;
}

navigator.geolocation.getCurrentPosition(sucess);
```

GEOLOCATION ERROR

- getCurrentPosition() accepts a second parameter.
- use to handle errors

```
function geoError(errorObj) {
    alert("Uh oh, something went wrong");
}

navigator.geolocation.getCurrentPosition(success, geoError);
```

ACTIVITY: GEOLOCATION

- Create a new index.html page.
- Use the geolocation object to retrieve latitude and longitude of the device/computer and write it to the page.
- Create a success function and an error function.

THE SCREEN OBJECT

- contains information about the display capabilities of the client machine

```
screen.height; // height of the screen in pixels
screen.width; // width of the screen in pixels
screen.colorDepth; // number of bits used for colors on client's screen
screen.orientation // orientation of the screen (landscape, portrait)
```

ACTIVITY: SCREEN INFORMATION

- Go back to your script.js file.
- Log the screen height, width, colorDepth, and orientation.

THE DOCUMENT OBJECT

- one of the most used objects in the BOM
- gain access to HTML elements, their properties, and methods

```
document.bgColor; // get and set the background color of the page
document.images; // get a list of images on the page
```

https://developer.mozilla.org/en-US/docs/Web/API/Document

ACTIVITY: BACKGROUND COLOR

- Go back to your script.js file.
- Change your background color

COLLECTIONS

- document object has array-like structures called collections
- forms, images, links

LINKS AND IMAGES COLLECTIONS

- each image on your page is stored in the images collection
- each link is stored in the links collections

```
document.images[0]; // 1st image on the page
document.images[1]; // 2nd image on the page
document.images.length; // returns how many images are on the page
document.links.length; // returns how many links are on the page
```

FEATURE DETECTION

FEATURE DETECTION

Example:

- All modern browsers support navigator.geolocation, but IE8 doesn't!

```
if (navigator.geolocation) {
    // use geolocation
}
```

```
if (typeof navigator.geolocation != "undefined") {
    // use geolocation
}
```

ACTIVITY: FEATURE DETECTION

- Go back to your geolocation code
- Add feature detection to only geolocate your user's device if that feature is available for their device.
- If the feature isn't available, display a nice message or alternative.

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>
```

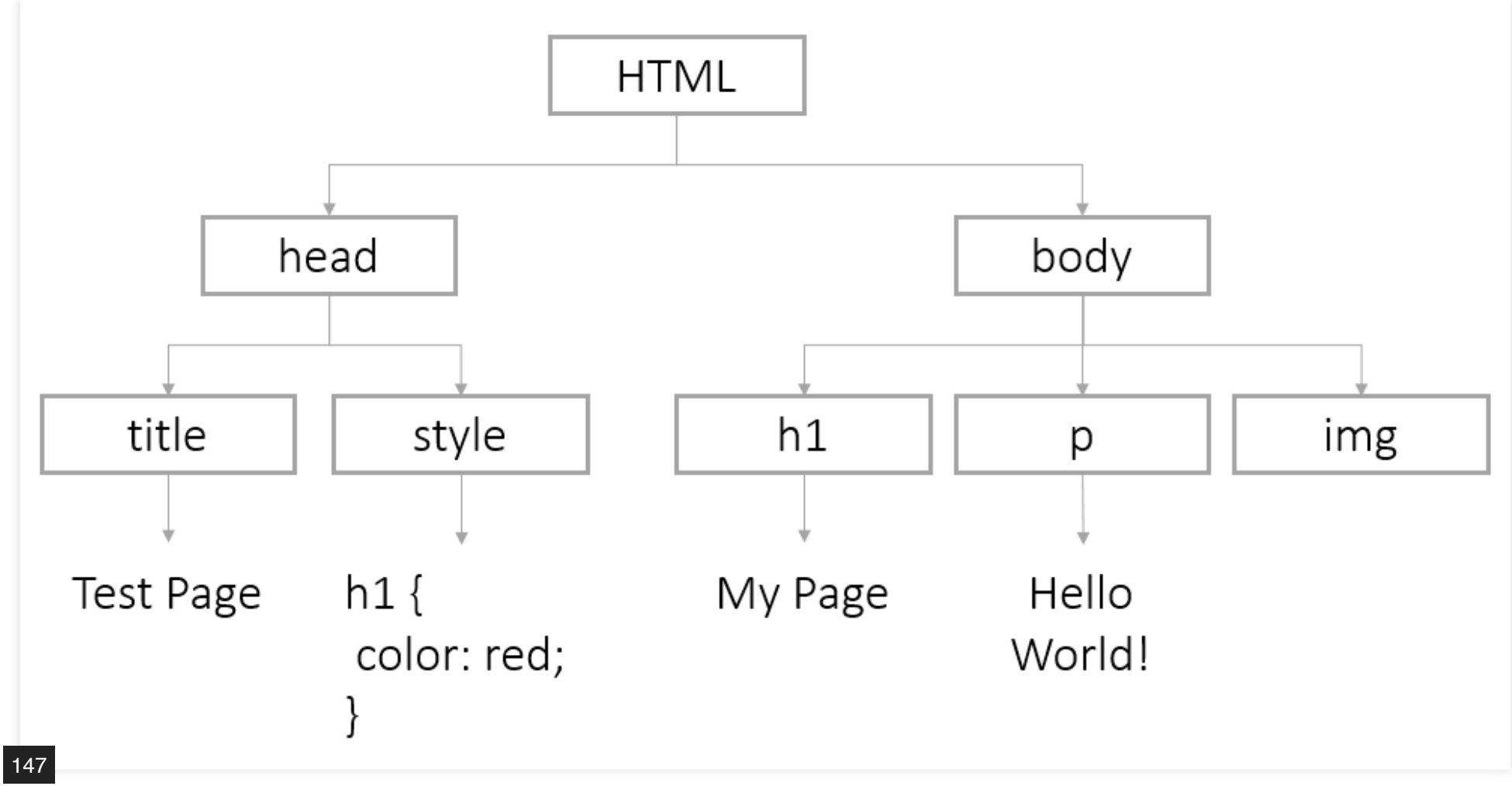
```
.list-item {
    /* CSS here */
}
```

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>About</li>
            <li>Services</li>
            <li>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
```

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

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.