intro to JavaScript

class four

course adapted from:
Teaching Materials: http://www.teaching-materials.org/javascript/
Oz Girl Develop It: http://cathylill.net/courses/js101/
Girl Develop It: http://www.girldevelopit.com/materials

homework!

recipe card

- Create an object to hold information on your favorite recipe. It should have properties for title (a string), servings (a number), and ingredients (an array of strings).
- On separate lines (one console.log statement for each), log the recipe information so it looks like:
  - Mole
  - Serves: 2
  - Ingredients:
  - cinnamon
  - cumin
  - cocoa

reading list

- Create an array of objects named `readingList`, where each object describes a book and has properties for the title (a string), author (a string), and readYet (a boolean indicating if you read it yet).
- Create function named `myReadingList` and iterate through the array of books. For each book, `alert` the book title and book author like so: "The Hobbit by J.R.R. Tolkien".
- Now use an if/else statement to change the output depending on whether you read it yet or not. If you read it, alert a string like 'You already read "The Hobbit" by J.R.R. Tolkien', and if not, alert a string like 'You still need to read "The Lord of the Rings" by J.R.R. Tolkien.'

a little review

The story thus far:

We've learned about ways to name and process data - we can declare **variables** (that have certain **data types**) that can be manipulated in predefined ways. Super simple.

We made **arrays** - a more complex structure that holds multiple pieces of data. Still pretty simple.

We learned **if/else** statements and **loops** which are simple sets of instructions that execute under certain conditions.

We made **functions**, which are named entities that can be reused and given different inputs.

We made **objects**, which have a complex structure, since object properties can range from primitive values all the way up to methods (a.k.a. functions). Remember, things like arrays and functions are objects.

objects on a webpage

We can also access everything on a web page as an object called `document`. This is called the

`Document Object Model (DOM).`

When the browser reads a web page, it builds up a **tree structure** of objects that represent all the elements on the page.

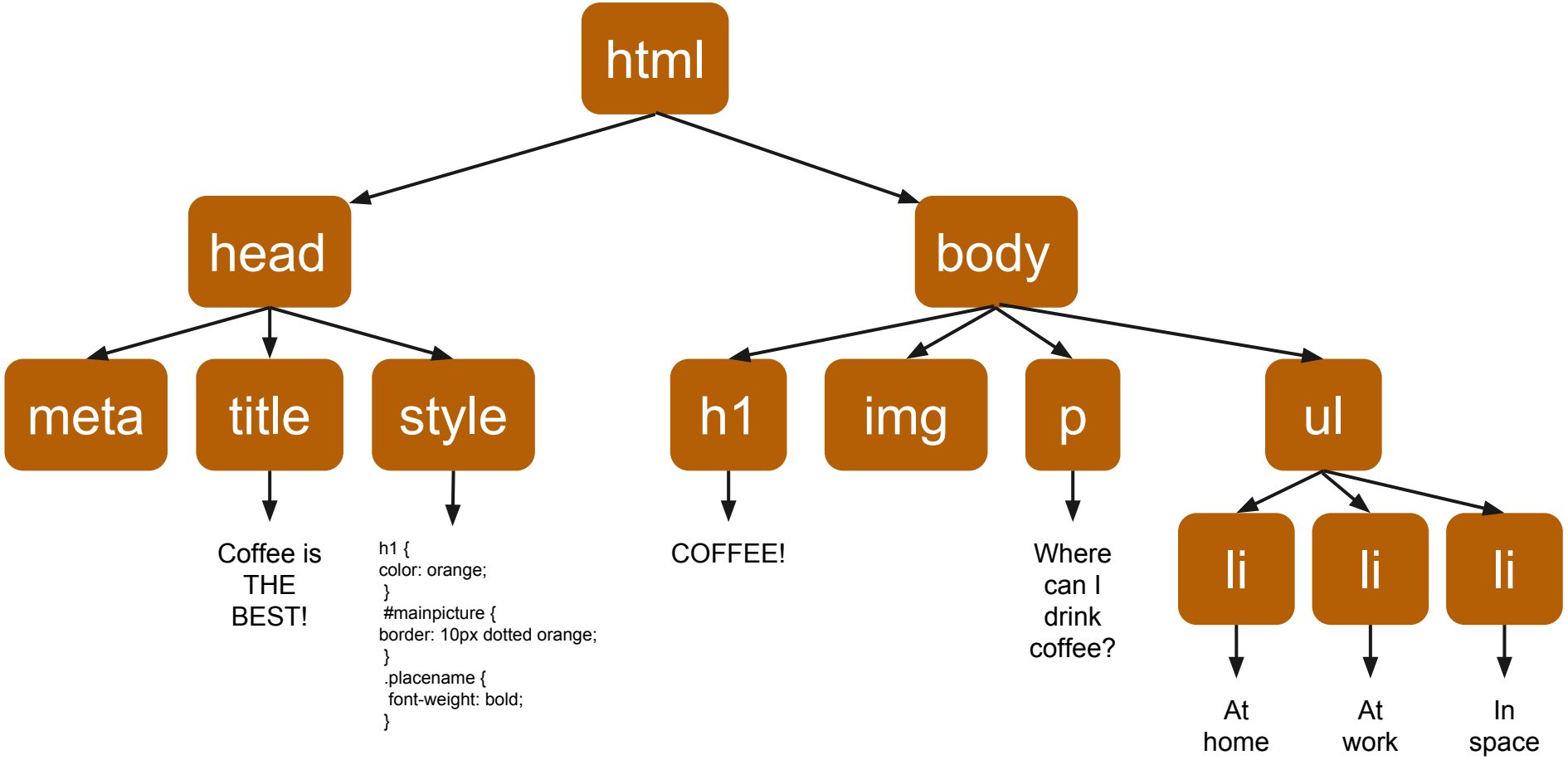
for example...

GO HERE:

http://jsbin.com/penah/1/edit?html,js,output

the DOM tree

The DOM tree for that page looks like this:



The general strategy

1) Find the element we want in the DOM using an access method and store it in a variable (so we don't have to go looking for it every time we need it).

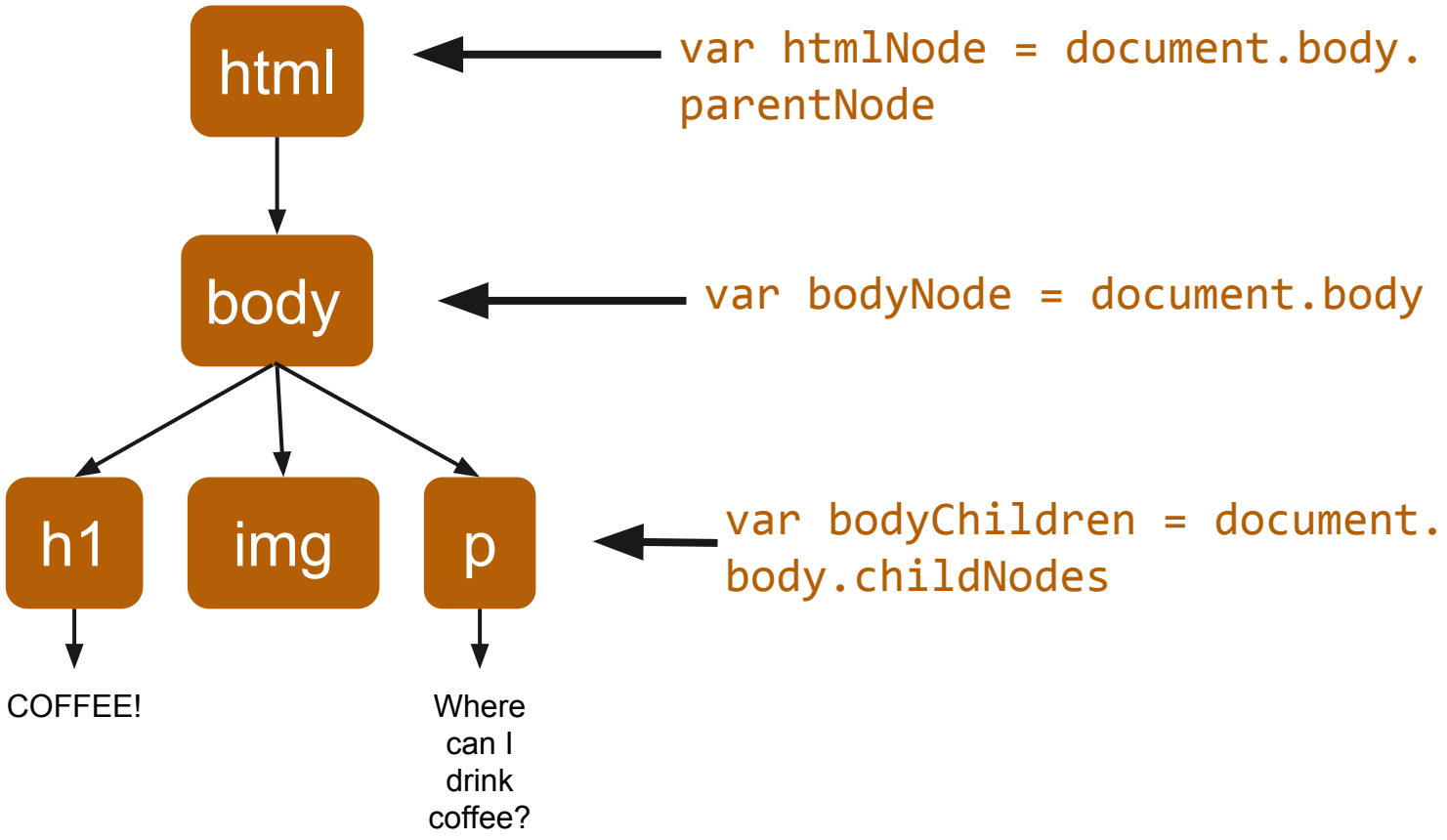
2) Manipulate the DOM by changing its attributes, style, inner HTML, or appending new nodes to it.

DOM access

The `html` in our DOM tree is an object named `document` in JS.

One way to refer to any element in our HTML by working our way down the DOM, starting with `document`. Each element is called a `node`. We can reference the `parentNode` (the level above on the tree), `childNodes` (the level below on the tree), or siblings with `prevSibling` and `nextSibling` (nodes on the same level of the tree).

for example...


html
body
h1
img
p
COFFEE!
Where can I drink coffee?
var htmlNode = document.body.parentNode
var bodyNode = document.body
var bodyChildren = document.body.childNodes

easier DOM access

Finding elements by child or parent takes a lot of work. We can also find elements by ID or class:

in our HTML:

```
<img id="mainpicture" src="images/coffee1.jpg">
```

in our JS:

```
var img = document.getElementById('mainpicture');
```

in our HTML:

```
<li class="placename">At home</li>
```

in our JS:

```
var place= document.getElementsByClassName('placename');
```

Note the plural "`Elements`" - this makes an array, not a single-value variable!

easier DOM access: tags

We can find elements by tag name:

in our HTML:

```
<li class="placename">At home</li>
<li class="placename">At work</li>
<li class="placename">In space</li>
```

in our JS:

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

DOM node attributes

We can also access the attributes of any node:

in our HTML:

```
<img id="mainpicture" src="http://upload.wiki...">
```

in our JS:

```
var img = document.getElementById('mainpicture');
img.src = 'insert filepath/URL here';
```

OR:

```
var oldSrc = img.getAttribute('src');
img.setAttribute('src', 'insert filepath/URL here');
```

innerHTML

Every node has an `innerHTML` property, which overwrites the HTML of that node.

```
document.body.innerHTML = '<p>Oops, I just erased everything else on the page!<p>';
```

Adding content

The `document` object also gives you ways to add new nodes to the page.

```
document.createElement(tagName);
document.createTextNode(text);
document.appendChild();
```

for example...

Identify what you'll be adding things onto:

```
var pageNode = document.body;
```

Create a new element, specify attributes, and append it:

```
var newImg = document.createElement('img');
newImg.src = 'insert filepath/URL here';
newImg.style.border = '10px dotted orange';
newImg.setAttribute('width', '300px');
pageNode.appendChild(newImg);
```

for example...

Create new element, a new text node, and then append them all together:

```
var newParagraph = document.createElement('p');
var paragraphText = document.createTextNode('Kittens are a close second to coffee.');
newParagraph.appendChild(paragraphText);
pageNode.appendChild(newParagraph);
```

exercise time!

where do you drink coffee?

- Using the DOM example page, create an array named `myPlaces` and fill it with three places you would drink coffee.
- In the HTML, add a button with the following markup: `<button onclick="changeLi()">Click me to change the list</button>`
- Create a function called `changeLi` which:
  - Accesses all the `li` elements on the page
  - Iterates through them and replaces them with the items from your list in `myPlaces`.
- Click the button and make sure it works!

bonus material!

events & animation

events

Most of the time, we want to manipulate a webpage (like we did last week) in response to a user's input. For this, we need **events**.

An `event` is a type of object created when a user interacts with a web page. To trigger a function to execute when a certain event happens, we need to create an **event listener**.

event listener

If we want something to be triggered when an event happens, we need to have the browser “listen” for that event. We do that like this:

```
target.addEventListener(type, listener);
```

- `type` = the type of event (click, mouseover, etc. See the full list)
- `listener` = the object that is notified when the event happens.

event types

The browser triggers many events. A short list:

- **mouse events** (MouseEvent): mousedown, mouseup , click, dblclick, mousemove, mouseover, mousewheel , mouseout, contextmenu
- **touch events** (TouchEvent): touchstart, touchmove, touchend, touchcancel
- **keyboard events** (KeyboardEvent): keydown, keypress, keyup
- **form events**: focus, blur, change, submit
- **window events**: scroll, resize, hashchange, load, unload

for example...

First, we make a var to hold the element we're going to listen to.

```
var counterBtn = document.getElementById('counter');

function onButtonClick(){
   counterBtn.innerHTML++;
    }

counterBtn.addEventListener('click', onButtonClick);
```

Second, we make a function which lays out what we want to happen

Then we make an event listener which triggers that function when a certain event happens

why not just `onclick`?

In previous classes, we've used `onclick` in our HTML to call JS functions.

This is bad practice:

1. It makes it harder to maintain your site by not separating JS and HTML
2. You lose the ability to attach multiple functions to a single event (onclick can be overwritten)

exercise time!

mad lib generator

- Go to: http://jsbin.com/xomuv/1
- In your JavaScript:
  - Make a var that accesses the button named `libButton`.
  - Make a function called `makeMadLib` which takes the `.value` from each text box and stores them in separate variables, then writes a story to the webpage in the story div, like “Cody really likes sassy squirrels.”
  - Add an event listener that will call the function when the user clicks on the right button.

the window object

`window` is an **object** created when you run JS in a browser. It has a lot of useful properties and methods.

It's the assumed global object on a page, so:

```
window.alert('hi!') = alert('hi!')
```

animation using window

Two ways of doing this:

Call a function after a delay:

```
window.setTimeout(function,delayMilliseconds);
```

Call a function repeatedly with a set interval in between:

```
window.setInterval(function,delayMilliseconds);
```

animating attributes

for example:

```
function makeImageBigger() {
  var img = document.getElementsByTagName('img')[0];
  img.setAttribute('width', img.width+10);
}
window.setInterval(makeImageBigger, 1000);
```

animating styles

```
var img = document.getElementsByTagName('img')
[0];
img.style.position = 'absolute';
img.style.top = '0px';
function kittyFall() {
  var oldTop = parseInt(img.style.top);
  var newTop = oldTop + 10;
  img.style.top = newTop + 'px';
}
window.setInterval(kittyFall, 100);
```

We have to strip and add units for the code to work correctly!

stopping animation

store timer in a variable and clear with one of the following methods:

```
window.clearTimeout(timer);
window.clearInterval(timer);
```

```
var fallTimer = setInterval(kittyFall,100);
window.clearInterval(fallTimer);
```

exercise time!

kitty moonwalk!

- Add a new img: <img src="http://gdriv.es/dpljs/cat_walking.gif">
- In your JS file, create a variable to store a reference to the img.
- Change the style of the img to have a "right" of "0px", so that it starts at the right hand of the screen.
- Create a function called catWalk() that moves the cat 10 pixels to the left of where it started, by changing the "right" style property.
- Call that function every 50 milliseconds. Your cat should now be moving across the screen from right to left. Hurrah!
- **BONUS:** When the cat reaches 400px, make it move backwards - and then have it move forward again when it hits 0px!

So what’s next?

Keep learning!

- Codecademy
- Code Combat
- Mozilla Developer Network
- Eloquent JavaScript
- OMG SO MANY BOOKS: check out the 005s in the stacks on the west side of the CTC. Here's what we have on JavaScript.