MIDTERM REVIEW

WEDNESDAY:
IN-CLASS MIDTERM

THINGS YOU SHOULD KNOW ABOUT THE MIDTERM

Anything from lecture and MPs is fair game

One-sheet of handwritten notes (front and back)

Expect to write code: **Javascript**, HTML, CSS, SASS, JQuery, AngularJS, Mongo Query Language

Will test your ability to apply what you've learned in new situations -- NOT regurgitate memorized facts (i.e., history of HTML)

HOW TO STUDY FOR MIDTERM

Go through all the questions on slides

Go through all code examples on slides/CODE PEN

Review the challenging aspects of the MPs

Be sure to review the following topics...

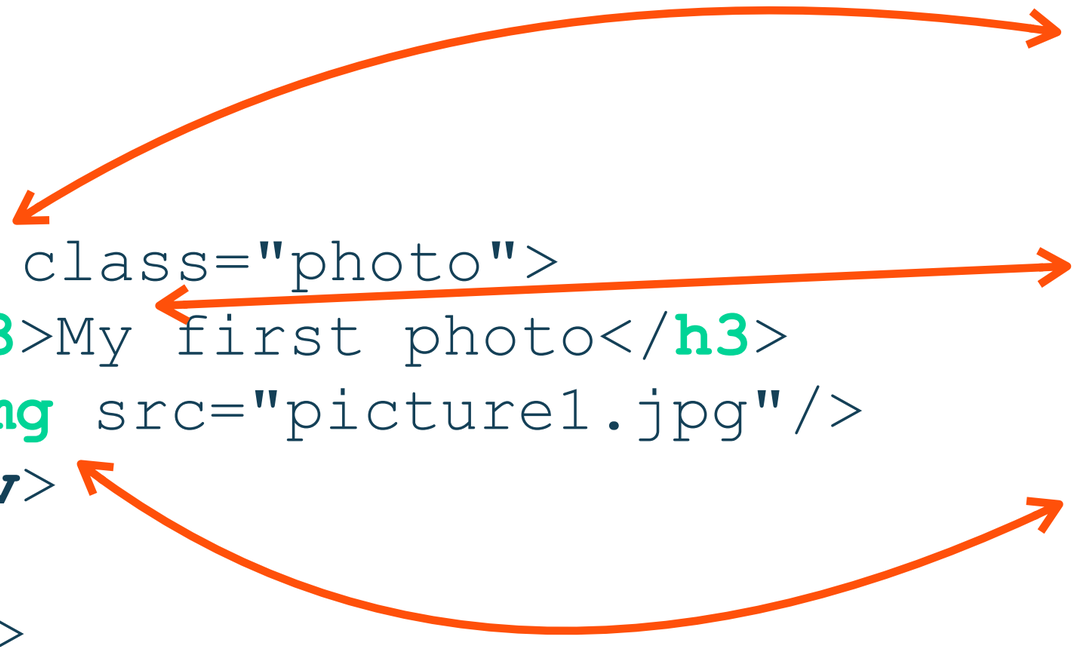
STRUCTURAL SEMANTIC TAGS

```
<body>
   <header>
      <h1>How to Get a PhD</h1>
      <nav>...</nav>
   </header>
   <article>
      <section>
         <figure><img src="benfranklin.jpg"></figure>
         <h3>Bribing your Committee</h3>
         <p>When blackmail fails...</p>
      </section>
      <aside>
         <h4>Useful Links></h4>
         <a href="www.bevmo.com">Research Supplies</a>
      </aside>
   </article>
</body>
```

CSS SELECTORS

```
<!DOCTYPE html>
<html>
  ...
  <body>
    <div class="photo">
      <h3>My first photo</h3>
      <img src="picture1.jpg"/>
    </div>
  ...
  </body>
</html>
```

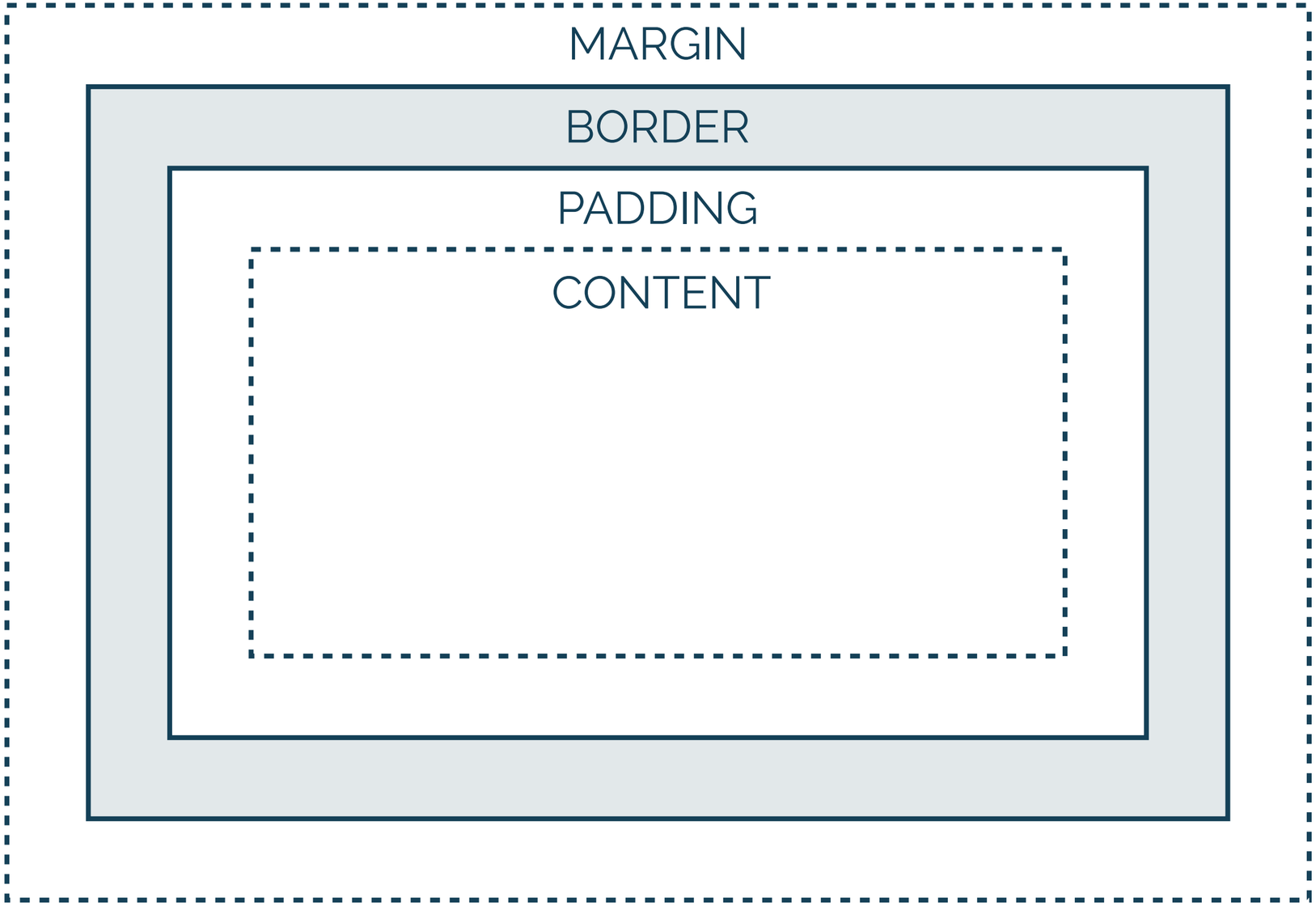
```
.photo {
 width:300px;
}
.photo h3 {
 font-weight:bold;
}
img {
 border:1px solid black;
}
...
```



map HTML elements to CSS rules

Learn correct usage of common CSS properties

Box Model



control over white space

position

VALUE	DESCRIPTION
`static`	default. positioned by the flow model; unaffected by top, bottom, left, right
`fixed`	positioned relative to browser window; will not move when window is scrolled
`relative`	positioned relative to its normal position
`absolute`	positioned relative to the first ancestor where `position!=static`

use with

`top`
`bottom`
`left`
`right`

Design Challenge:

vertically center a `<div>`
of unknown height

CODEPEN

SOLUTION

```
<div class="table-outer">
  <div class="outer">
    <div class="inner">
    </div>
  </div>
</div>
```

```
.table-outer {
  width: 100%;
  display: table;
}

.outer {
  height: 200px;
  background-color: #144057;

  display: table-cell;
  vertical-align: middle;
}

.inner {
  width: 100px;
  height: 50%;
  background-color: #B6C4C9;
}
```

CSS PREPROCESSORS

languages that extend CSS in meaningful ways

features: variables, nesting, mixins, inheritance

shrinks developer's codebase and compiles into CSS

popular CSS preprocessors: LESS and SASS

Functions are first-class objects

FUNCTIONS ARE OBJECTS

that are callable!

reference by variables, properties of objects

pass as arguments to functions

return as values from functions

can have properties and other functions

ANONYMOUS FUNCTIONS

create a function for later use

store it in a variable or method of an object

use it as a callback

see more examples next class

`this`

the other implicit parameter

a.k.a. function context

object that is implicitly associated with a function's invocation

defined by how the function is invoked (not like Java)

`apply()` *and* `call()`

two methods that exist for every function

explicitly define function context

```
apply(functionContext,arrayOfArgs)
```

```
call(functionContext,arg1,arg2,...)
```

implemented in Javascript 1.6

```
function forEach(list, callback){
  for (var n = 0; n < list.length; n++){
    callback.call(list[n],n);
  }
}

var numbers = [5,3,2,6];
forEach(numbers, function(index){
        numbers[index]= this*2;});
console.log(numbers);
```

Classes are defined through functions

OBJECT-ORIENTED PROGRAMMING

`new` operator applied to a constructor function creates a new object

no traditional class definition

newly created object is passed to the constructor as this parameter, becoming the constructor's function context

constructor returns the new object

CONSTRUCTOR INVOCATION

constructors are given the class name

```
function Llama() {
  this.spitted = false;
  this.spit = function() { this.spitted = true; }
}

var llama1 = new Llama();
llama1.spit();
console.log(llama1.spitted); true

var llama2 = new Llama();
console.log(llama2.spitted); false
```

scopes are declared through functions and not blocks { }

`closure` *scope created when a function is declared that allows the function to access and manipulate variables that are external to that function*

PRIVATE VARIABLES

```
var add = (function () {

  var counter = 0;

  return function () {return
  counter += 1;}

})();

add();
```

self-invoking

PRIVATE VARIABLES

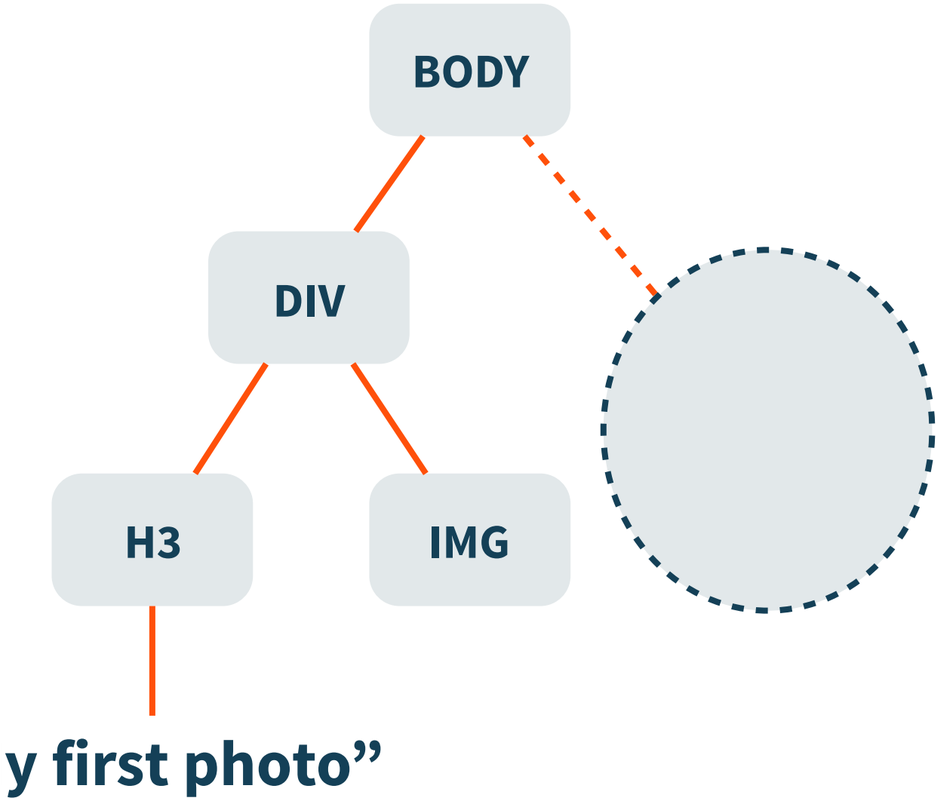
```
function Llama() {
  var spitted = false;
  this.spit = function() { spitted =
  true; }
  this.hasSpitted = function() { return
  spitted; }
}
```

private data member now!

DOCUMENT OBJECT MODEL

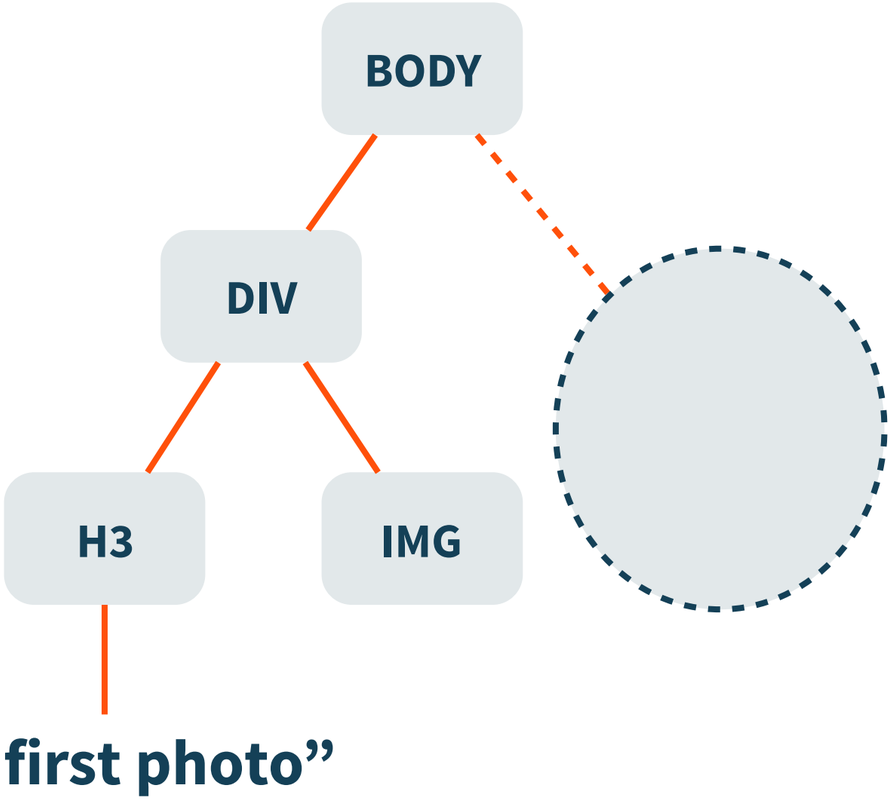
one-to-one correspondence between HTML elements and DOM nodes

```
<body>
  <div class="photo">
    <h3>My first photo</h3>
    <img src="picture1.jpg"/>
  </div>
...
</body>
```



TRAVERSING THE DOM

```
var body = document.body;
var div = body.children[0];
var h3 = div.children[0];
var textNode = h3.childNodes[0];
var textString = textNode.nodeValue;
```



www.w3schools.com/jsref/dom_obj_all.asp

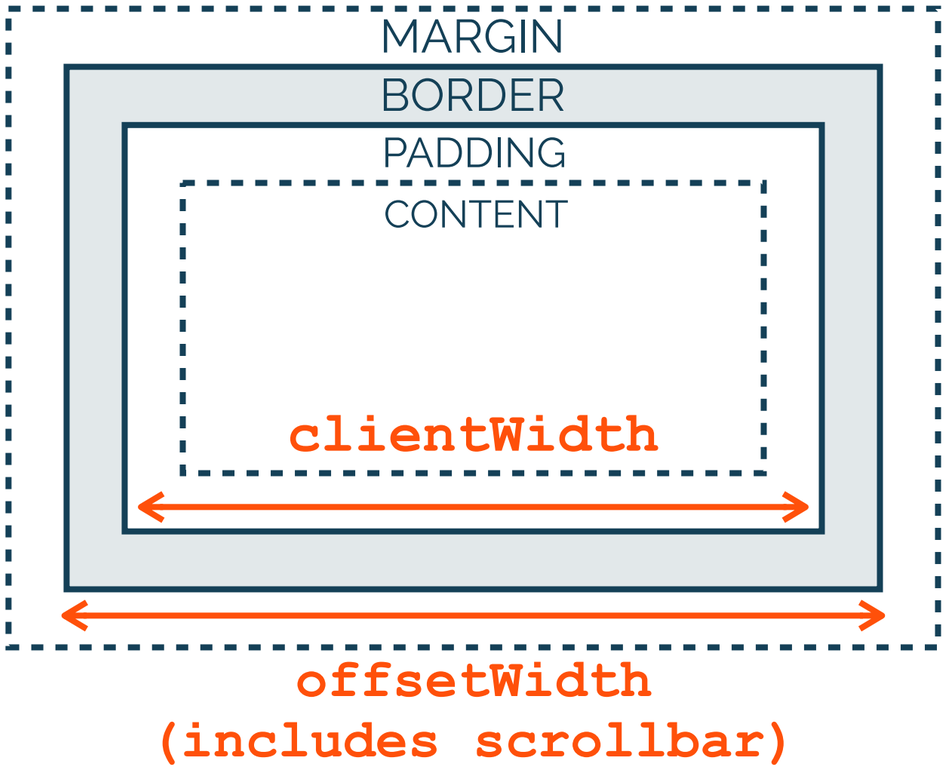
DOM ELEMENT OBJECT

relative to offsetParent

position: **element.offsetTop**, **element.scrollTop**, ...

dimensions: **element.clientWidth**, **element.offsetWidth**, ...

style: **element.style**



www.w3schools.com/jsref/dom_obj_all.asp

DOM MANIPULATION

programmatically change the structure and modify element properties

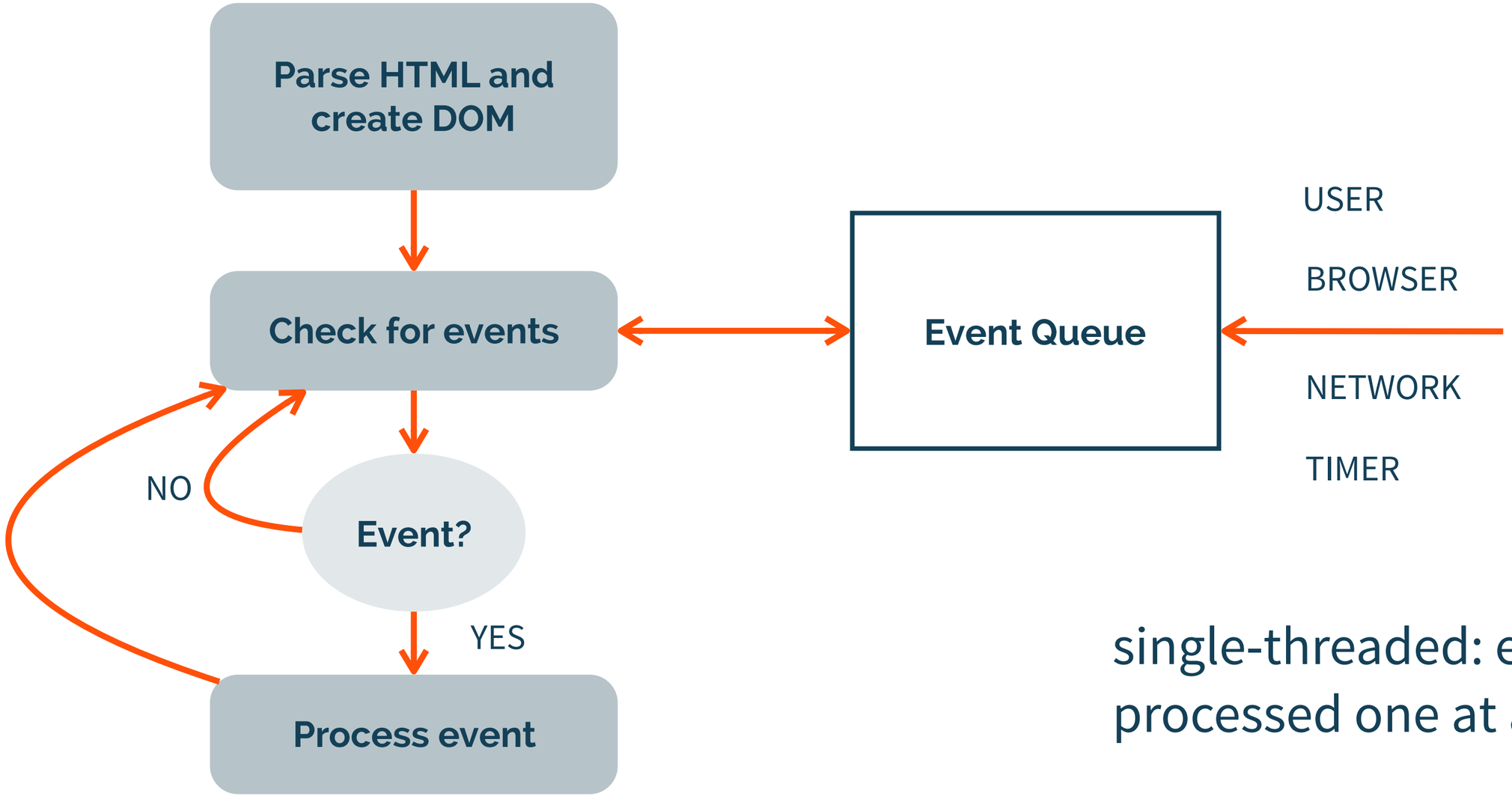
```
element.style.backgroundColor = “red”;
element.innerHTML = “<div><h3>Llama!</h3>…</div>”
```

augment DOM structure:

```
element.appendChild(),element.removeChild(),...
```

www.w3schools.com/jsref/dom_obj_all.asp

THE BROWSER EVENT LOOP



single-threaded: events processed one at a time

EVENT PROCESSING

events propagate in two phases

capture phase: root to innermost element

bubble phase: innermost element to root

DOM standard: *capture* then *bubble*

EVENT PROCESSING

```
element.addEventListener(event,
function, useCapture)
```


```
event.stopPropogation()
```

CODEPEN

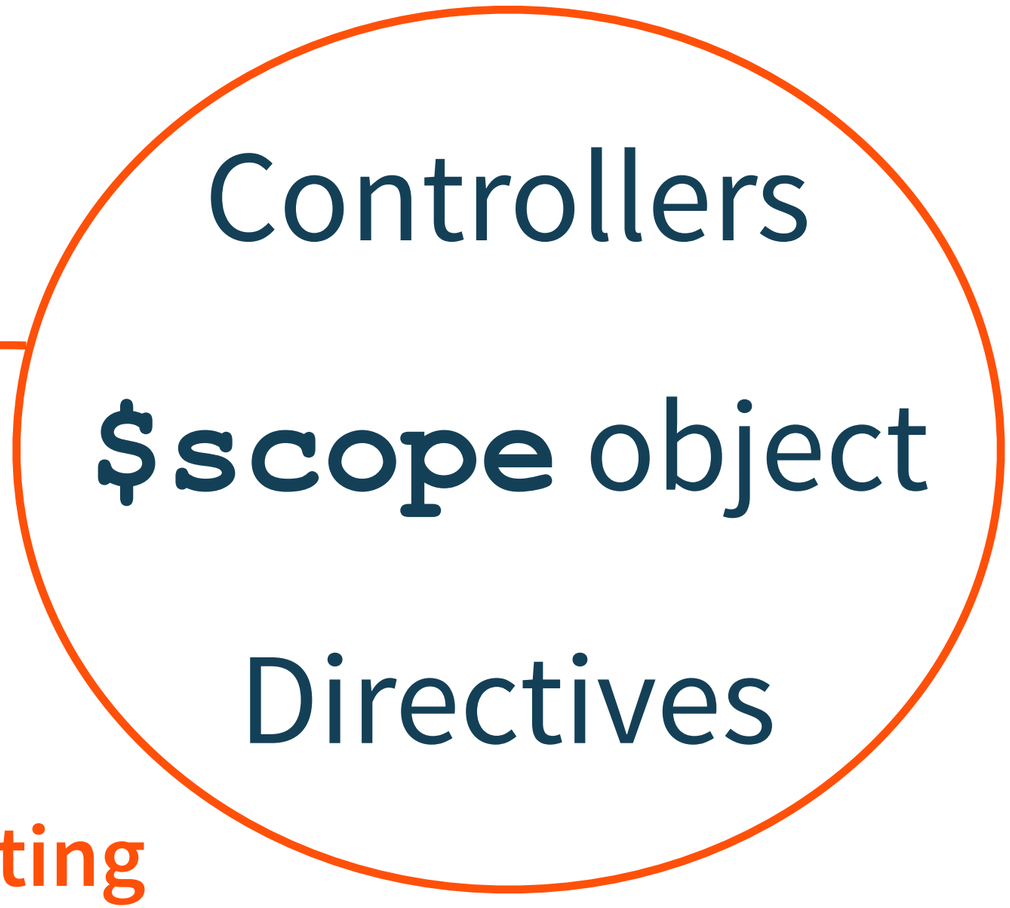
JQUERY

cross-browser

use for all DOM manipulation:
(*e.g., positioning relative to document and not offsetParent*)

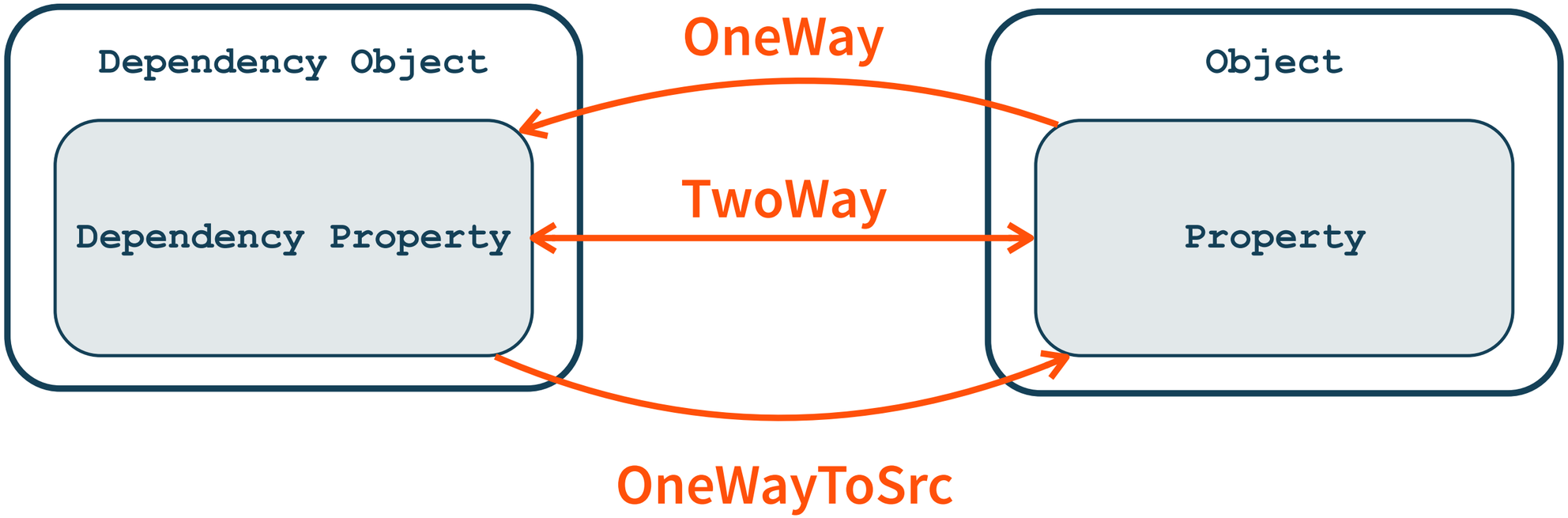
ANGULAR CONCEPTS

1.3



MVC
Client-side Templating
Data Binding

WAYS OF DATA BINDING



MADLIBS TEMPLATE

```
<div ng-app>
  <div ng-controller='MadlibsController'>
    <div>Hola
      <span class="madlib">{{madlibs.animal}}</span>,
    </div>
    <div>Se llama
      <span class="madlib" ng-bind="madlibs.name"></span>!
    </div>
    <form>
      <input ng-model="madlibs.name">
    </form>
   </div>
</div>
```

CODEPEN

Example 1

CODEPEN

```
var bar = 20;
var foo = (function() {
  var bar = 10;
  return function(llama) {
    return llama + bar;
  }
})();

var result = bar + foo(5);
console.log(result);
```

what will this print?

Example 2

CODEPEN

```
function familyFunction() {
  var name = "fruit bats";
  console.log(name + " are " + family);
  if (name === "fruit bats") {
    var family = "pteropodid";
  } else if (name === "camels") {
    var family = "camelid";
  }
  console.log(name + " are " + family);
};

familyFunction();
```

what will this print?

Example 3

CODEPEN

```
var fun = (function first() {
  var name = "camels";
  var family = "camelid";
  return (function second() {
    return function() {
    console.log(name + " are " + family);
  }
  var family = "pteropodid";
  })();

 var name = "fruit bats";
})();

fun();
```

what will this print?

Example 4

CODEPEN

```
fun = (function outside() {
  var name = "cat";
  var cry = "meow";
  return (function inside() {
    return function() {
      console.log(name + " goes " + cry);
    }
  var name = "dog";
  })();
  var cry = "woof";
})();

fun();
```

what will this print?

Example 5

```
 <div class="row">
    <div class="medium-3 columns">3</div>
    <div class="medium-3 columns end”>3</div>
   <div class="medium-3 columns end">3 end</div>
</div>
```

A. 3 divs aligned to left
B. 3 divs center aligned
C. 2 divs aligned to left, 1 div aligned to the right
D. 1 div aligned to the left, 2 divs aligned to the right

Example 6

```
function foo() {
  var result = '';
  for (var i = 0; i < arguments.length; i++) {
    result += arguments[i];
  }
  return result;
}

function bar() {
  return arguments.join('');
}

console.log(foo('a', 'b', 'c'));
console.log(bar('a', 'b', 'c'));
```

https://stackoverflow.com/a/2091159

Example 7

```
<ul id="awesome">
  <li class="favorite">
    My name is:
    <span class="highlight">Camelid</span> </li>
</ul>
```

How do we turn just “Camelid” red?