Reading Week
Feb 20th - 2017.
It begins.

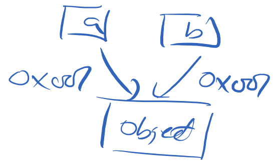
** Javascript: understanding the weird parts.

Section 4, lecture 36:

By Value vs By Reference:

Copying into two different memory locations.

points to same memory location.



← By reference.

(*) Primitives (by Value)

def

Mutate: to change something.

Immutable: can't be changed.

(*) All objects (incl functions) ← By Reference

so if d = c, they aren't copies, they just point to the same location in memory

* even as parameters to functions, its still passed by reference.

* Equals operator sets up new memory space (new address)

c = {greeting: 'howdy'};

c & d no longer point to the same memory location

as "changes things";

** In JS you have no choice,

primitives: by Value.

objects: by reference.

Lec 37, Sec 4 — objects functions and this.

Monday, February 20, 2017 8:52 PM

function called → new execution context is created (remember).

- this?? - under the hood.

- global object is Window at the global level.

- if you are simply invoking a function this still refers to global object (Window)

```
func a() {
    cons.log(this);
}
a();
```

← points to global object.

- when a function is actually a method attached to an object,

  this → object, so it points to the object.

  You can mutate, using this.

- To avoid that "weird" bug, in regards to the **this** keyword in JS, usually create a var called 'that' or 'self' and set it equal to 'this', that way you avoid any funny mistakes or bugs.

```
var self = this;
self.name = "Fayzah"
// this.name = "Fayzal";
```

← better practice due to the 'this' bug.

this way, you don't have to worry, if you're pointing to the right object. As self points to the same memory location as 'this' - cause by reference silly

Remember

No programming language is 'Perfect'.

↳ see the let keyword kind of helps out but thats for later eh?

Arrays in Javascript.

```
var arr = new Array();
var arr = [];    or var arr = [1,2,3];
```

Arrays in JS are 0 indexed.

So whats different, with Arrays in Javascript??

① JS is dynamically typed.

② So you can mix and match whats in an array.

```
var arr = [
    1,
    false,
    {
        name: 'Fayadh',
        address: "174"
    },
    function(name){
        var greet = "hello";
        console.log(greeting + name);
    },
    "hello"
];
```

Completely Valid JS Array,

Cool huh?

[illegible]

JS arrays can hold collections of anything.

```
arr[3](arr[2].name);
```

← calling the function, and passing it name from the same array.

↑ this is insane eh?