i_X

Embrace Opportunity

Web Development

Week 1 | Day 3

i_X

Agenda

- Survey

- Intro to OOP

- Constructors and prototypes

- ES6 classes

- Exercises and homework

i_X

Introduction to OOP

An outline of OOP, the purpose and advantages.

i_X

OOP vs functional programming vs procedural

Procedural

- The steps to execute written in order

OOP

- Code in classes/objects
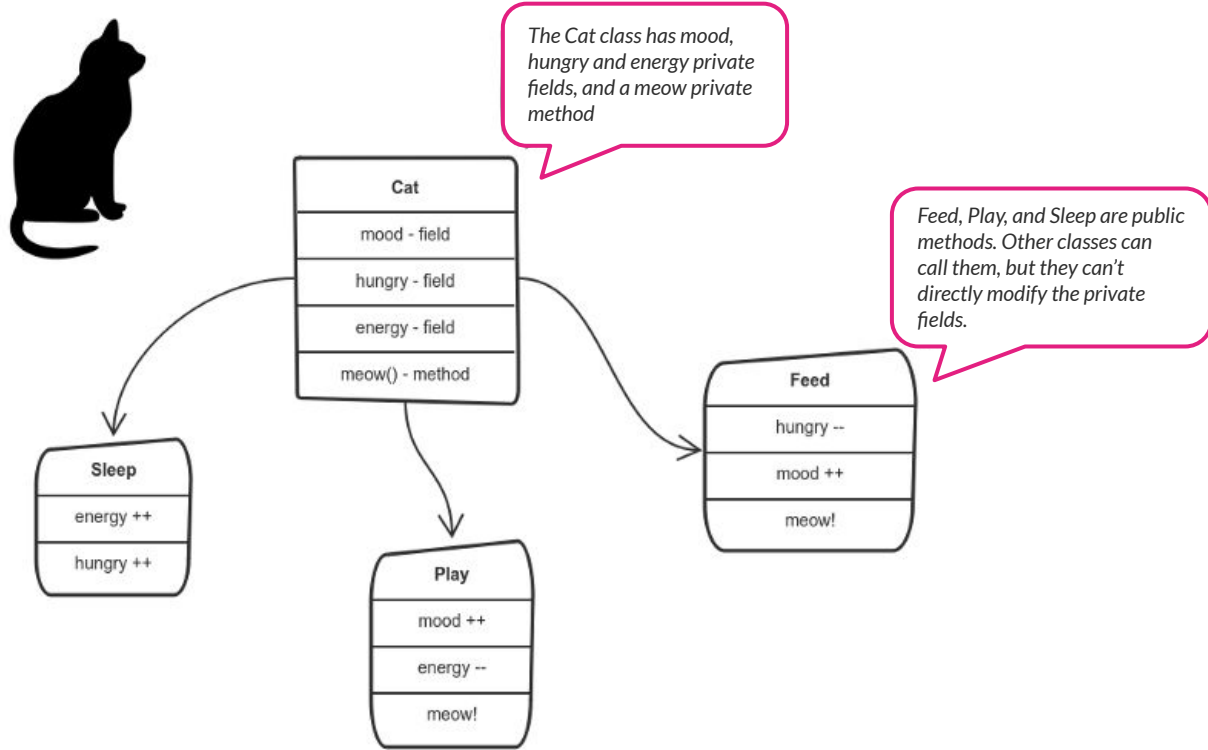- Data stored in properties, logic in methods

Functional

- Multiple functions, each functions works on it's own
- Stateless and pure

i_X

4 Pillars of OOP

1. Abstraction

2. Encapsulation

3. Inheritance

4. Polymorphism

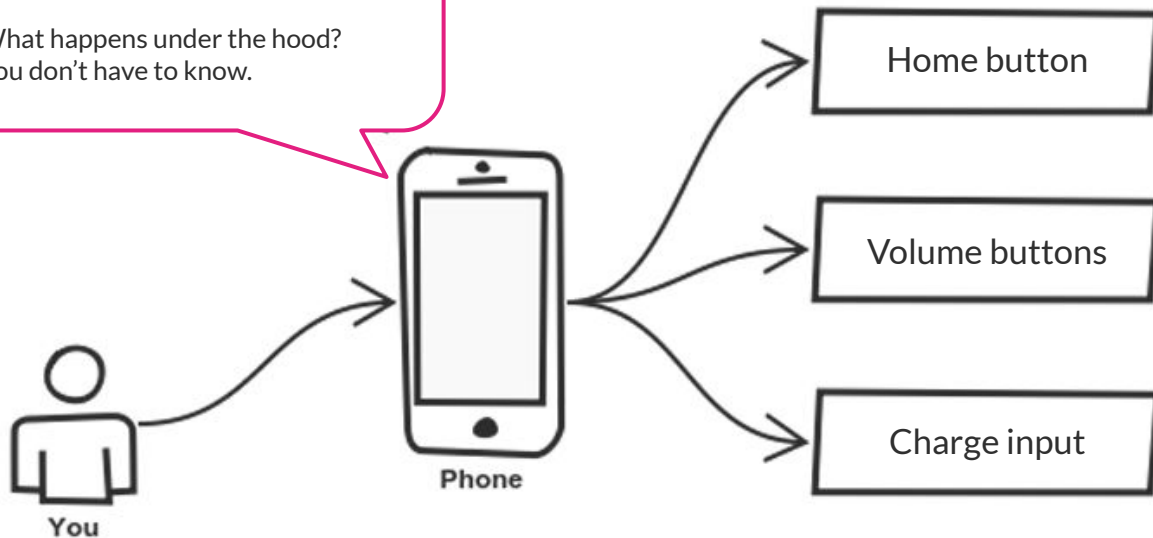i_X

Encapsulation

The Cat class has mood, hungry and energy private fields, and a meow private method

Feed, Play, and Sleep are public methods. Other classes can call them, but they can't directly modify the private fields.

Cat
mood - field
hungry - field
energy - field
meow() - method

Sleep
energy ++
hungry ++

Play
mood ++
energy --
meow!

Feed
hungry --
mood ++
meow!

You can feed the cat. But, you can't directly change how hungry the cat is.

i_X

Abstraction

Cell phones are complex. But using them is simple.

Inheritance

Person
name
email

Child classes take all private fields and methods from the parent class.

Teacher
everything from person
subject

Student
everything from person
classes
grades

Child classes can also add additional fields of their own.

Private teacher
everything from teacher
Students

Public teacher
everything from teacher
School
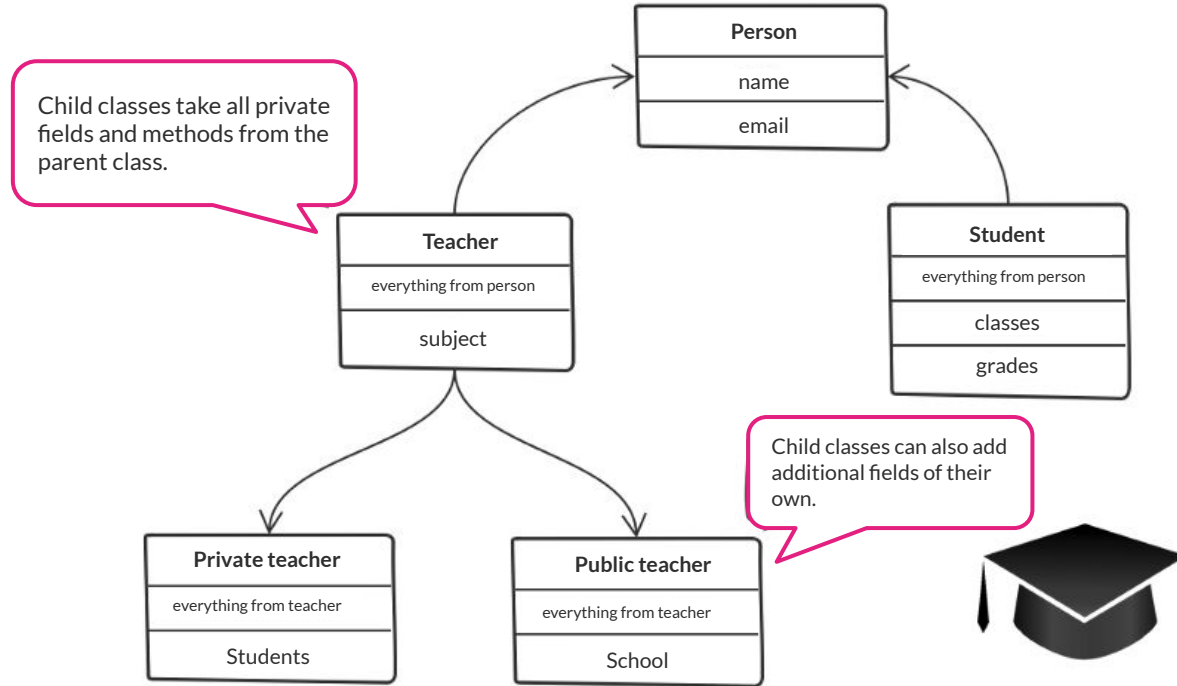
A private teacher is a type of teacher. And any teacher is a type of Person.

i_X

Polymorphism

Figure Interface
Whoever uses it must implement:
- CalculateSurface()
- CalculatePerimeter()

Triangle

Circle

Rectangle

Triangle, Circle, and **Rectangle** inherit the Figure interface or abstract class.

They implement their own versions of **CalculateSurface()** and **CalculatePerimeter()**.

They can be used in a mixed collection of Figures.
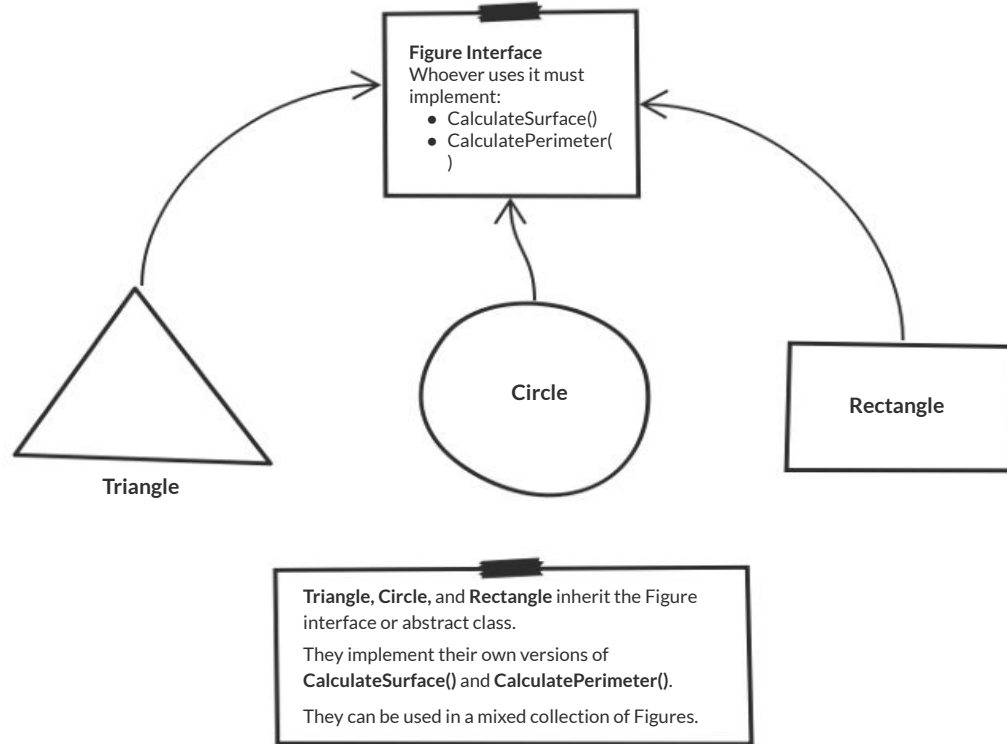
Triangle, Circle, and Rectangle now can be used in the same collection

i_X

ES5 vs ES6

ES5
- Functions
- This
- Prototypes

ES6
- Classes

i_X

Constructors and prototypes

The old way of doing things.

i_X

ES5

We use prototype to add methods to our function

```
function Person(firstName, lastName, dob) {
        this.firstName;
        this.lastName;
        this.dob;
        this.getFullName
    }
Person.prototype.getFullName() = function(){
    return this.firstName +" "+this.lastName;
    }
```

i_X

ES6 classes

The new standard.

i_X

Classes

A couple things we all struggle with at first:

- A class is just a blueprint

- A Person{} class would describe the blueprint of a person (notice that Person has a capital P)

- To instantiate means to create an object using the class blueprint

- ie: let james = new Person()

- james is then an instance (an object we created) of the Person class

Don't worry, the aha moment comes as you keep practising

i_X

ES5

- We can add methods within our class
- Methods are just functions in a class

```
class Person{
    constructor(firstName,lastName, dob) {
            this.firstName;
            this.lastName;
            This.dob;
    }
    getFullName(){
    return this.firstName +" "+this.lastName;
        }
    }
```

i_X

Exercise and homework

Contact Book comparing old and new methods.

Looking ahead and homework

Exercise

- Contact Book ES5
- Contact Book ES6

Homework

- Task List using ES6 Classes

i_X

Thanks for listening!

Tomorrow we'll work on asynchronous JavaScript.

i_X

See you tomorrow