JavaScript Objects

What do you **understand** the term **OOP**, to mean?

each **object** has it's own:

- If we want to make a **model** of a hotel or car, the **computer has no idea** what this looks like, how
  it behaves or what it does
- Programmers have to **create this** using code, feeding it the **right data** to make it behave in such a way that it is **useful**

classes	objects
Cannot do any real **work by themselves** (they need to be **instantiated** as objects)	An **instance** (copy) of a class automatically contains the **data** (properties) and **behaviours** (methods) of the class
Classes are **templates** that provide the **blueprint** of multiple objects	

each **object** has it's own:

- **properties** (variables)
  - *Hotel: Park Hotel*
- **methods** (functions)
  - *available rooms:* ***45***
    (totals rooms - let rooms)

Difference between a **property** and
a **variable** is that a property controls
a unique aspect of that object

- A hotel's **properties** would include: name, rooms, bookings, rating, gym, pool

Difference between a **method** and a **function** is that a method represents **something that can be done to an object**

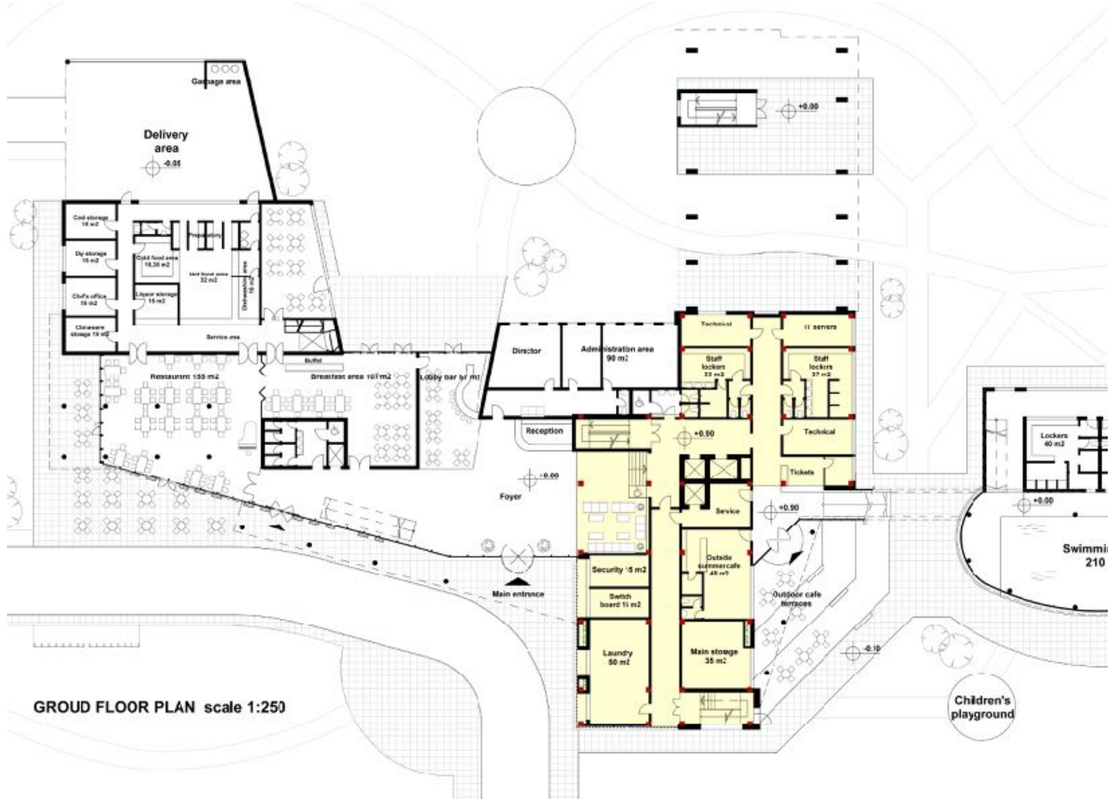
- A hotel's **method** would include:
  makeBooking();
  cancelBooking();
  checkAvailability():

objects but **not** **classes**:

- The above ways of creating an object are only slightly different (the first line)
- They show how to create the **object first** then add **properties** and **methods**
- **Neither** is a **class** (template to create other hotels, they are a finished hotel)
- Let’s create the hotel object **with** properties **and** methods

classes and **objects** :

- The literal notation did not create a **classes** but just an **object**
- The Object Constructor Notation **created a class** (template) that hotels can be made from
- The literal **class** **can do this as well**, but requires a little more work



JS **object** **literal notation** with **arrows**:

- Can we use **arrow functions** with these objects?
- Yes, but there is a **caveat**
- There can be issues using the keyword **‘this’** eg this.rooms
- *Back to our example to see it not working, and look at a way around it*

JS **object** in **ES6**:

- We now have access to a new keyword ‘**class**’ to avoid confusion
- We also have a real **constructor**
- Both **Object Constructor** and **Literal Notation** are here, they are very similar
- *Back to our example to see it not working, and look at a way around it*