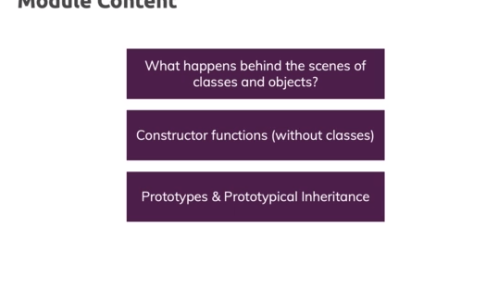
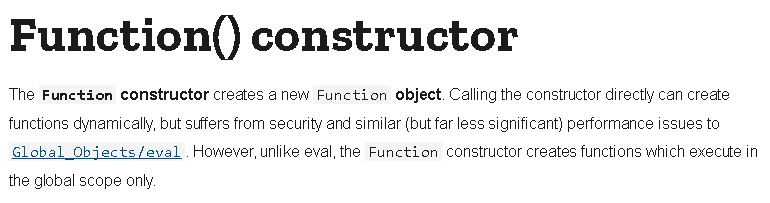
**Constructor Functions & Prototypes**

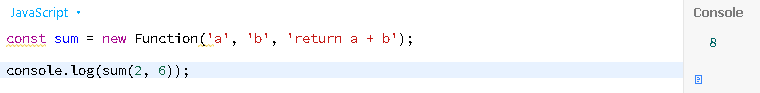


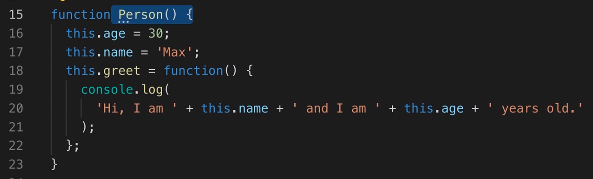
**Introducing Constructor Functions**

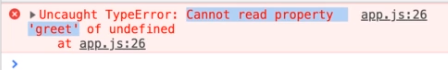


https://paper-attachments.dropbox.com/s_07877671CC63741771FC8B296982F7540C8A7B735C6F6F1E581D3280F9B5703C_1632753084759_gfhfghfg.JPG



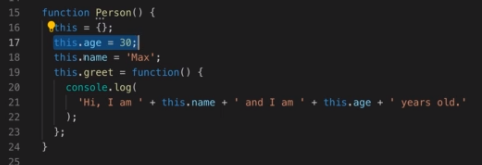






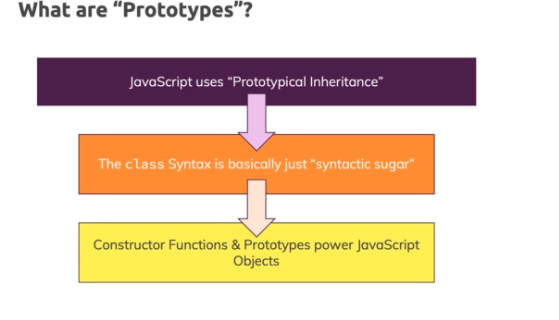
**Constructor Functions vs Classes & Understanding "new"**





* Creates this = {};
* stores all the other objects into it
* and proceeds with the invoked finction

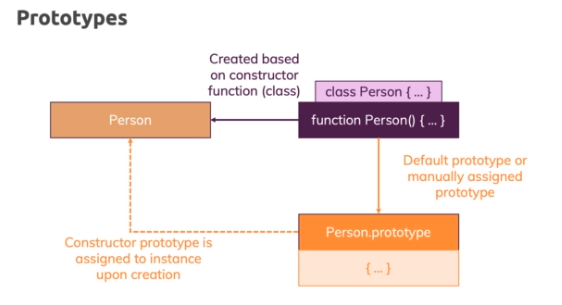
**Introducing Prototypes**



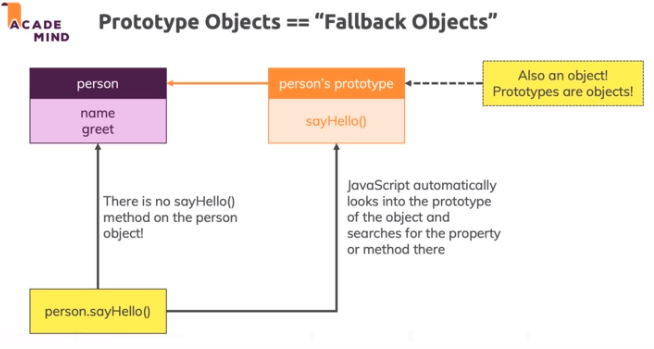
* Every constructor function you build has a special prototype property which is not added to the objects you create based on it because it's not part of the function body but a property of that function object
* functions are objects and that prototype is there by default
* it is then automatically assigned as a prototype to the object which is created when you instantiate that constructor function
* a prototype is an object itself,
* there are objects and every object has such a prototype

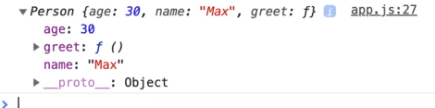
**What exactly is the idea behind a prototype?**

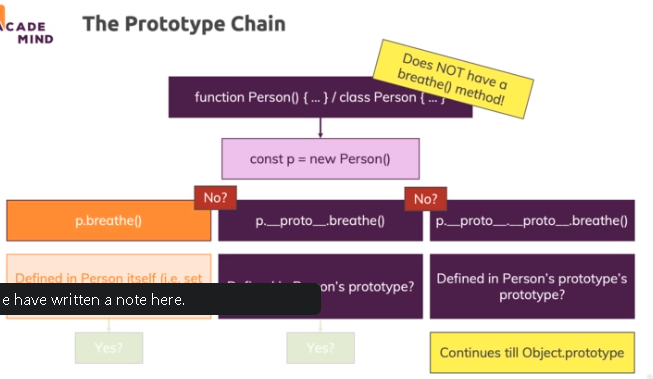
* It's how Javascript shares code in the end



* a prototype is basically a connected object which is used as a fallback object







https://paper-attachments.dropbox.com/s_07877671CC63741771FC8B296982F7540C8A7B735C6F6F1E581D3280F9B5703C_1632755138967_screenshot-www.udemy.com-2021-09-27-20-35-34-456.png

**Prototypes - Summary**

Prototypes can be a confusing and tricky topic - that's why it's important to really understand them.

A prototype is an object (let's call it "P") that is linked to another object (let's call it "O") - it (the prototype object) kind of acts as a "**fallback object**" to which the other object ("O") can reach out if you try to work with a property or method that's not defined on the object ("O") itself.

**EVERY object in JavaScript by default has such a fallback object** (i.e. a prototype object) - more on that in the next lectures.

It can be especially confusing when we look at how you configure the prototype objects for "to be created" objects based on constructor functions (that is done via the .prototype property of the constructor function object).

Consider this example:

    1. function User() {

    2.     ... // some logic, doesn't matter => configures which properties etc. user objects will have

    3. }

    4. User.prototype = { age: 30 }; // sets prototype object for "to be created" user objects, NOT for User function object

The User function here also has a prototype object of course (i.e. a connected fallback object) - **but that is NOT the object the prototype property points at**. Instead, you access the connected fallback/ prototype object via the **special \_\_proto\_\_ property** which EVERY object (remember, functions are objects) has.

The prototype property does something different: **It sets the prototype object, which new objects created with this User constructor function will have.**

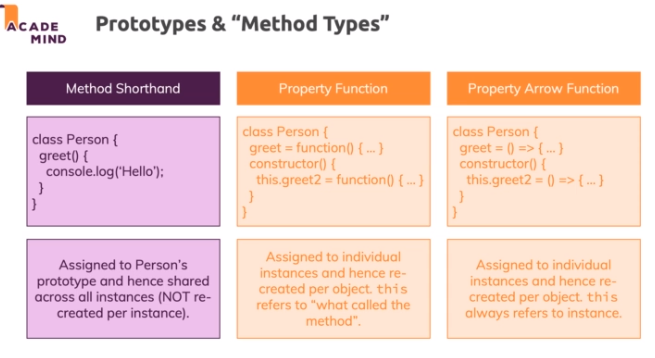
That means:

    1. const userA = new User();

    2. userA.\_\_proto\_\_ === User.prototype; // true

    3. userA.\_\_proto\_\_ === User.\_\_proto\_\_ // false

**Methods in Classes & In onstructors**



**Built-in Prototypes in JavaScript**



**Setting & Getting Prototypes**



