OOPS JS NOTES:

* This is known as **abstraction** — creating a simple model of a more complex thing, which represents its most important aspects in a way that is easy to work with for our program's purposes.
* In OOP, we can create new classes based on other classes — these new **child classes** can be made to **inherit** the data and code features of their **parent class.**
* JavaScript uses special functions called **constructor functions** to define and initialize objects and their features.





* this keyword being used here as well — it is basically saying that whenever one of these object instances is created, the object's name property will be equal to the name value passed to the constructor call.



* hey are using their own name value that was assigned to them when they were created; this is one reason why it is very important to use this, so each one uses its own value, and not some other value.
* the **new** keyword is used to tell the browser we want to create a new object instance



* we are defining *greeting ()* every time, which isn't ideal. To avoid this, we can define functions on the **prototype** instead.
* The JavaScript prototype property allows you to add new properties to object constructors.
* A **constructor** in **JavaScript** is just a plain old function that returns an object.
* **Prototypes vs**. **Classes** based inheritance
* **class** defines a **type** which can be instantiated at runtime
* **prototype** is itself an object instance
* In JavaScript, almost **"everything" is an object**. Objects are **Mutable**
* JavaScript Classes are templates for JavaScript Objects.
* Different ways to create new objects:
* Define and create a single object, using an object literal.
* Define and create a single object, with the keyword new.
* Define an object constructor, and then create objects of the constructed type.
* Object.create() for ES5 and above
* Display JavaScript objects are:
* Displaying the Object Properties by name
* Displaying the Object Properties in a Loop
* Displaying the Object using Object.values()
* Displaying the Object using JSON.stringify()
* Access Properties of Object in Loop



* Access Properties of Object using **Object.values()** returns array of values of object.



* First, access the greeting() method of a person object.
* Second, returns the function definition.
* Method vs Getter:
* Method is accessed as function.
* Getter is accessed as property.