Summary

- Factory Functions (aka Factory Object Creation Pattern) instantiate and return a new object in the function body
    - They allow us to create new objects on a pre-defined template and have two major downside:
    - There is no way to tell based on a returned object itself whether the object was created by a factory function
    - All objects created by factory functions that share behavior have an owned copy or copies of the methods, which is redundant
- Constructor Functions are meant to be invoked with the `new` operator
    - They instantiate a new object behind the scenese and allow the developer to manipulate it using the `this` keyword
    - A typical constructor is used with the following pattern:
    - Constructor invoked with `new`
    - A new object is created by the JS runtime
    - The new object inherits from the constructor's prototype
    - The new object is assigned to `this` inside the function body
    - The code inside the function is executed
    - `this` is returned unless there's an explicit return
- Every object has a `__proto__` property that points to a special object, the object's prototype
    - The object's prototype is used to lookup properties that don't exist on the object itself
    - `Object.create` returns a new object with the argument object as its prototype
    - `Object.getPrototypeOf` and `obj.isPrototypeOf` can be used to check for prototype relationships between objects
- The prototype chain property lookup is the basis for "prototypal inheritance" or property sharing through the prototype chain
    - Downstream objects "inherit" properties and behaviors from upstream objects
    - Downstream objects can "delegate" properties or behaviors upstream
    - A downstream object shadows an inherited property if it has same-named property (overriding)
    - `Object.getOwnProperityNames` and `obj.hasOwnProperty` can be used to test whether a given property is owned by an object
- Every Function has a `prototype` property that points to an object iwth a `constructor` property that points back to the function itself
    - If the function is used as a constructor, then the returned object's `__proto__` will be set to the constructor's prototype property
    - This behavior allows us to set properties on the constructors prototype object that will be shared by all objects returned by it

    - This is called the "Prototype Pattern" of object creation

- The Pseudo-Classical Pattern of object creation generates objects using a constructor function that defines state, then defines shared behaviors on the constructor's prototype
- The Object Linking to Other Objects (OLOO) pattern of object creation uses a prototype object and `Object.create` to generate objects with shared behavior
    - An optional `init` method on the prototype object is defined to set unique states on the returned objects