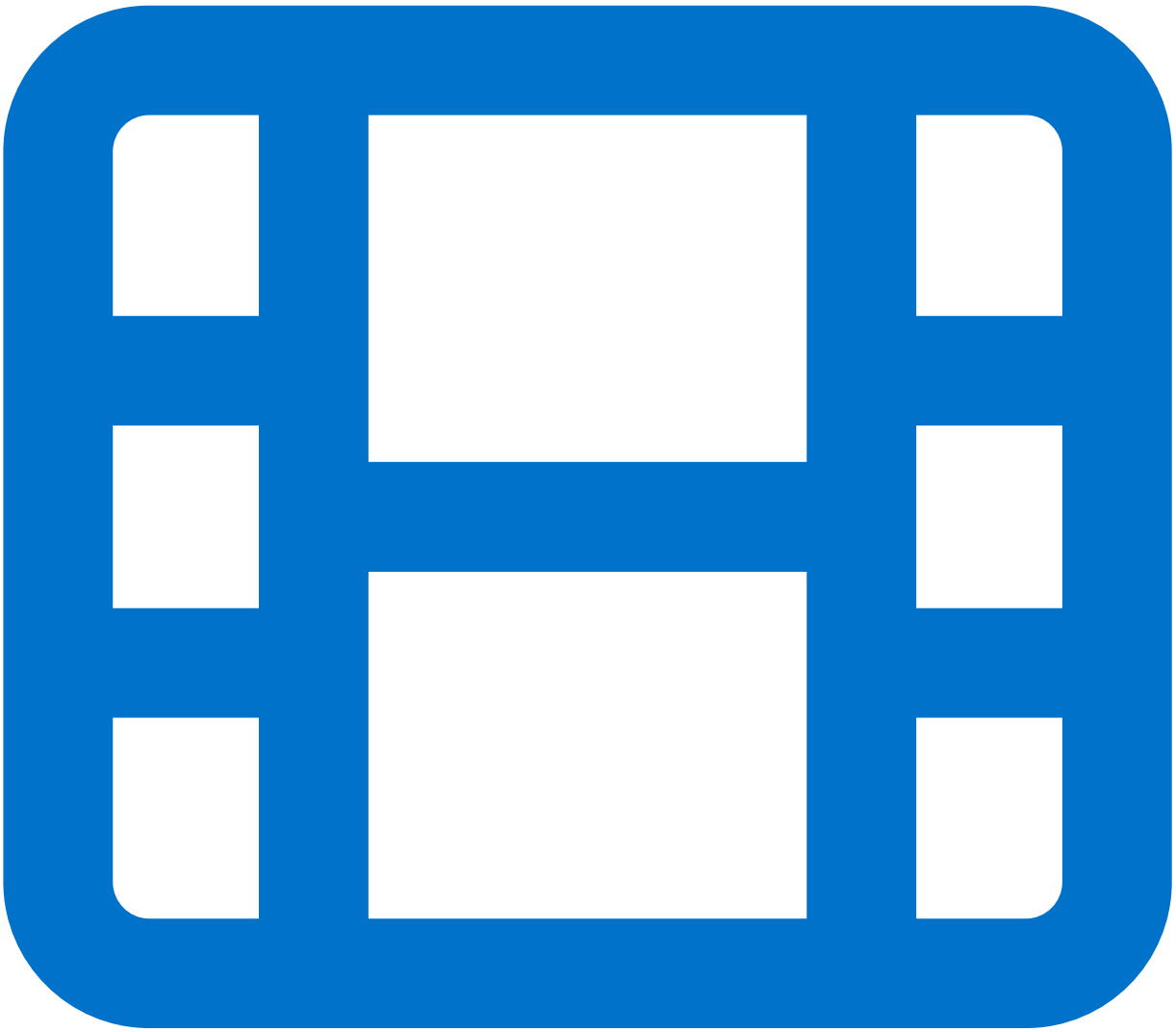
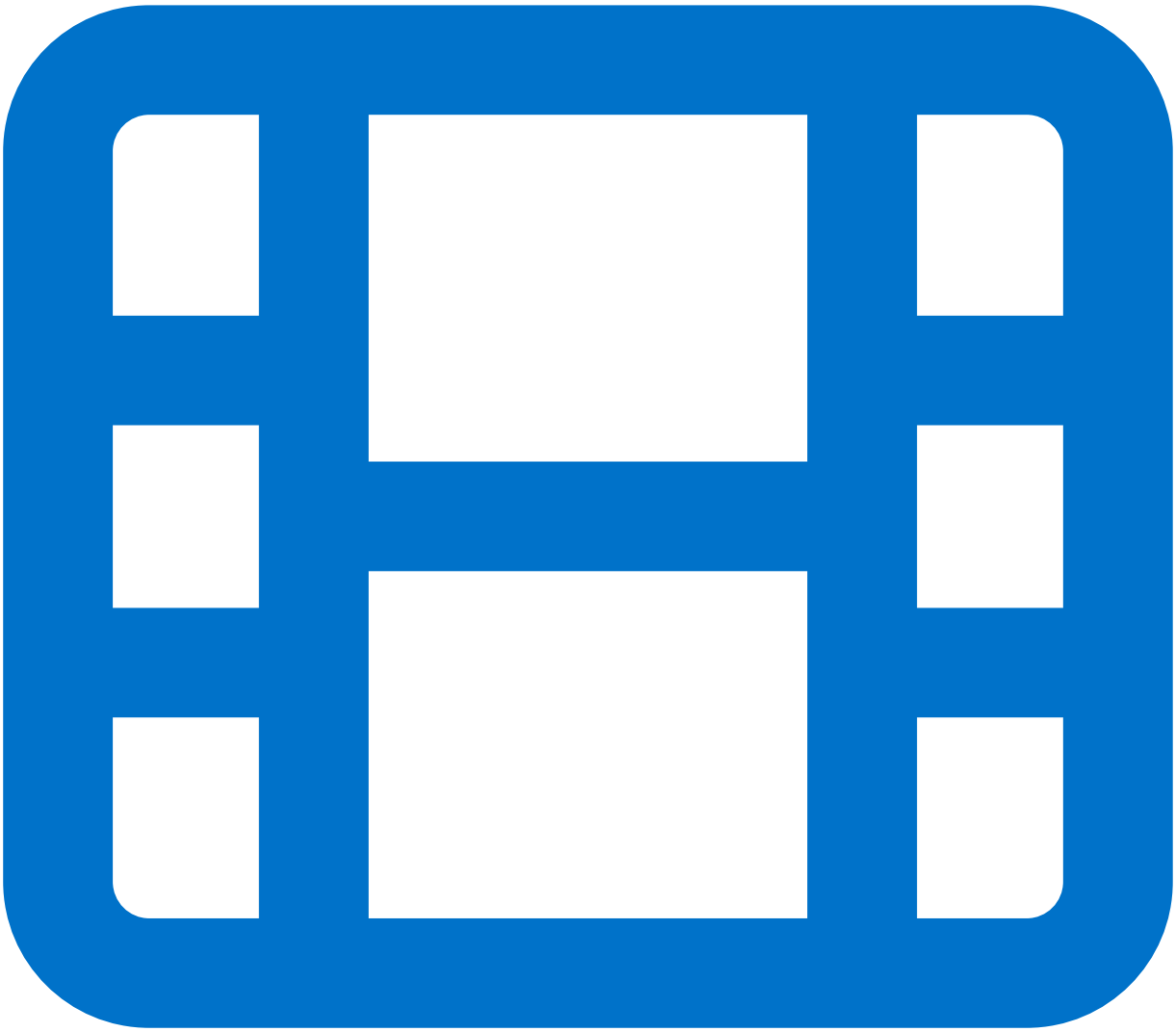
Iniciando con Objetos

Material Audiovisual

Mira los siguientes videos para reforzar el contenido teórico y ayudar a resolver las actividades más rápido:



Objetos | Egg



📝 Actividad: Creación de un producto

En esta actividad, mejorarás el programa **store** de las secciones anteriores para definir los primeros productos de la aplicación web. Sigue estos pasos:

1. Desde Visual Studio Code, abre la carpeta **store**

2. Comenta el script **index** utilizado en secciones anteriores.

3. Dentro de esta carpeta crea un archivo **products.js** y conéctalo con el html. A modo de ejemplo:

```
<!-- <script src="./index.js"></script> -->
<script src="./scripts/products.js"></script>
```

4. En **products.js,** define la variable **producto1** y asignale un objeto con { } y las propiedades nombre, stock y precio. A modo de ejemplo:

```
const producto1 = {
nombre: "celular",
precio: 100000,
stock: 2
};
```

5. Agrega las propiedades id y foto, probando estas formas.

```
producto1.id = "id123";
producto1[foto] = "https://i.postimg.cc/Jn2C5W84/galaxy1.webp";
```

6. Programa la impresión en consola de **producto1**.

7. Programa la impresión de la propiedad **nombre** de **producto1**

8. Abre con live server el archivo index.html y verifica las salidas programadas.

📝 Actividad: Creación de un producto

En esta actividad, mejorarás el programa **store** de las secciones anteriores para definir los primeros productos de la aplicación web.

A partir de este punto, comenzarás a definir las variables en inglés, ya que es una práctica habitual en el mundo de la programación. Sigue estos pasos:

1. Desde Visual Studio Code, abre la carpeta **store**

2. En products.js, define la clase Product con las propiedades:

1. id y title (nombre): deben ser una cadena de texto.

2. price (precio) y stock (cantidad) deben ser de tipo numérico.

3. images (imágenes) por ahora asigna []

4. y onsale (en oferta): debe ser un booleano para definir si está o no está en oferta.

```
dclass Product {
  constructor(id, title, price, stock, images, onsale) {
    this.id = id;
    this.title = title;
    this.price = price;
    this.stock = stock;
    this.images = images;
    this.onsale = onsale;
  }
}
```

3. Define la variable prod1 e implementa el constructor new para crear una instancia de Product vacía. A modo de ejemplo:

```
const prod1 = new Product();
```

4. Define la variable prod2 para crear una instancia de Product con las propiedades id, title y price.

5. Define la variable prod3 para crear una instancia de Product con todas las propiedades que requiere la clase.

6. Programa la impresión en consola de prod1, prod2 y prod3.

7. Programa la impresión de la propiedad title de prod2 y de la propiedad onsale de prod3.

8. Abre con live server el archivo index.html y verifica las salidas programadas.

📝 Actividad: Obtención y modificación de propiedades de una instancia

En esta actividad, vas a definir propiedades privadas de la clase Product y vas a crear getters y setters para poder leer y modificar las mismas. Para este ejercicio tienes que seguir los siguientes pasos:

1. Desde Visual Studio Code, abre la carpeta **store**

2. En products.js, modifica el constructor de la clase Product para agregar una propiedad privada supllier. A modo de ejemplo:

```
class Product {
constructor(id, title, price, stock, images, onsale, supplier) {
...
this._supplier = supplier;
        }
}
```

3. Define el método getter y el método setter de esta propiedad. A modo de ejemplo:

```
get getSupplier() {
return this._supplier;
}
set setSupplier(newName) {
this._supplier = newName;
}
```

4. Define la variable prod4 para crear una instancia de Product con todas las propiedades que requiere la clase.

5. Modifica el proveedor implementando el setter definido de prod4.

6. Programa la impresión en consola del getter de prod4.

7. Abre con live server el archivo index.html y verifica las salidas programadas.

📝 Actividad: Venta de productos

En esta actividad, crearás un método que, por cada venta de un producto, restará la cantidad vendida al stock del producto de la clase Product. Sigue estos pasos:

1. Desde Visual Studio Code, abre la carpeta **store**

2. Dentro de la clase Product del archivo products.js, define el método sellUnits para que cumpla los requerimientos. A modo de ejemplo:

```
class Product {
constructor(id, title, price, stock, images, onsale) {
                ...
}
        sellUnits(units) {
                this.stock = this.stock - units
        }
}
```

3. Define la variable prod5 para crear una instancia de Product con todas las propiedades que requiere la clase. Debe tener 12 unidades de stock.

4. Ejecuta el método sellUnits para vender 10 unidades y luego 5 unidades.

5. Programa la impresión en consola de prod5.

6. Abre con live server el archivo index.html y verifica las salidas programadas.

7. Mejora el método para que si no hay stock suficiente, no se descuenten unidades y se devuelva un mensaje de error.