# Agregar dinamismo al carrito

En este paso, te enfocarás en generar el contenido de la vista de forma dinámica, Con los datos reales de los productos del carrito y la cuenta total a pagar por los mismos.

> 💡 ***Atención***
>
> *Antes de finalizar su sesión en equipo de hoy, un integrante deberá compartir su pantalla y explicar en detalle cómo resolvió los ejercicios propuestos. Puede ser el facilitador o cualquier compañero que se anime (puedes ser tú 💪).*
>
> *Aprovecha esta oportunidad para enriquecer tu propia resolución y participa activamente haciendo preguntas.*

---

## 🖍️ Actividad: Generar tarjetas de carrito dinámicas

Para este ejercicio tienes que seguir los siguientes pasos:

1. Desde Visual Studio Code, abre la carpeta **store** y crea el archivo cart.js en la carpeta scripts

2. Conecta el nuevo script en la vista del carrito.

3. Abre el archivo cart.js y define una variable cartproducts.

4. Asigna a esta variable un array con los productos del localStorage.

5. Analiza la tarjeta del producto del carrito diseñada en la actividad anterior e identifica las propiedades de cada producto.

6. Define las funciones necesarias para renderizar dinámicamente todos los productos agregados al carrito. A modo de ejemplo createCartCard para crecibir un objeto (producto) y retornar un template dinamico) y printCartCards para recibir un array de productos y luego imprimir en la vista las tarjetas.

7. Programa y verifica la vista la cantidad de veces que sea necesario para que la vista se renderice y funcione correctamente.

---

## 📝 Actividad: Calcular total a pagar

Para este ejercicio tienes que seguir los siguientes pasos:

1. Desde Visual Studio Code, abre la carpeta **store** y crea el archivo cart.js en la carpeta scripts

2. Conecta el nuevo script en la vista del carrito.

💡*Presta atención al orden de los scripts. El script cart debe tener acceso a las variables y funciones de layout y el script total tiene acceso a las variables y funciones de layout y cart.*

1. Abre el archivo total.js y define las funciones necesarias para renderizar dinámicamente el cuadro de cálculo de la cuenta a pagar. A modo de ejemplo createTotalTemplate para recibir como parámetro un array de productos y devuelva un template string con el cuadro de detalle del total y printTotal para recibir un array de objetos  y luego imprimir en la vista el template generado.

2. La función necesita iterar los productos para que cada vuelta sume el subtotal (precio * cantidad) con el de la siguiente vuelta.

```
function createTotalTemplate(arrayOfProducts) {
  let total = 0;
  arrayOfProducts.forEach(
    (each) => (total = total + each.price * each.quantity)
  );
  return `
     <h4 class="total-title"> Resumen del pedido </h4>
     <p class="total-p">Subtotal $ ${total}</p>
     <button id="buy" type="button">COMPRAR</button>
  `;
}
```

1. Programa y verifica la vista la cantidad de veces que sea necesario para que la vista se renderice y funcione correctamente.

---

# 📖 Material Complementario

Profundiza los conocimientos adquiridos a través de los siguientes recursos de aprendizaje opcionales:

👉 Format Price

👉 Método reduce

---

# 📖 Repositorio en GitHub

Puedes subir a un repositorio existente de github la práctica del día de hoy. Los comandos básicos son:

- **git add .** (para preparar todos los archivos creados, modificados o eliminados)
- **git commit -m “nombre del envío”** (para versionar en repositorio local)
- **git push origin main** (o master según corresponda para enviar la nueva versión del proyecto al repositorio remoto)