Reviewing JS-Cart

Goal: Practice how to mingle

- Events
- State
- HTML updates/changes

...with front end JS

Different than server-side

- Not request/response

Key Concepts

- State + Project Architecture
- Managing HTML
- Separating Concerns
- Source of Truth

What is Architecture?

- Planning how the pieces connect

Key concept there is *planning*

- Advanced developers will tell you the majority of coding involves no typing of code
- Even when you are skilled, don't jump straight to ChatGPT!

Data Models and Structures

> *Bad programmers worry about the code. Good programmers worry about data structures and their relationships.*
>
> - *Linus Torvalds, Creator of Linux kernel and Git*

- What does your inventory look like?
- What does your cart data look like?
- What impact do these decisions have?

Project Files and Structure

- Where is your "main code"?
    - Runs on page load
    - Imports and uses other code
- If I'm looking for a bit of code
    - Do I know what file to look in?
- If I have a variable/function/file name
    - Do I know what it represents from the name?

Improving your architecture

- Get past a project by
    - Get it working by the deadline
- Improve as a coder by
    - Not being content with "working"
    - Notice "pain points"
        - Find better solutions
    - Ask questions
        - Seek answers
    - Pace yourself
        - Can't learn it All

Understand not just the How, but the Why

Following a best practice

- Makes you good to work with

Understanding WHY the best practice

- Makes you great to work with
- Able to solve changes
- Able to apply nuance
    - say and understand "it depends"

6250 Assignments are Tough!

You should have questions

- I can answer some
- Other answers come with practice

Intent is to get you used to tackling problems where

- You know enough to make a working solution
- But not so much that you have no questions
- This mimics what should happen on the job
- Use the uncertainty to get better

Separation of Concerns

- Data Models
    - Is the quantity in the inventory or the cart?
    - Where is the price?
- Generation of HTML?
    - No shared scope between files
    - Requires passing state to generating function
        - That's good!

Source of Truth

Code should always have a **single source of truth**

- Otherwise when sources of truth don't agree
    - Subtle bug
    - Disagreement more likely

A single source of truth can still be wrong

- A bug
- But a more OBVIOUS bug

Truth in JS Cart

What is in your cart data?

- Is it a copy of the inventory data?

Ideal is cart only has its own data + references

- `quantity`
- Product index (if products is an array)
- Product key (if products is an object)

When cart needs name, pic, and price

- All pulled from the single source of truth
    - products

Common Issue: Object vs Array

- If you repeatedly loop through array for 1 item
    - RED FLAG that you shouldn't use an array!
- Common interactions should be easy
    - Not just "possible"
- Complexity doesn't just slow computer
    - Makes it harder for us to think about

Sample State using Objects, not Arrays

```
const products = { // by "id"
  jorts: {
    id: "jorts",  // in record for convenience, not required
    name: "Jorts",
    price: 0.99,
    image: "https://placehold.co/150x150?text=Jorts",
  },
  jean: {
    id: "jean",
    name: "Jean",
    price: 3.14,
    image: "https://placehold.co/150x150?text=Jean",
  },
  nyancat: {
    id: "nyancat",
    name: "Nyancat",
    price: 2.73,
    image: "https://placehold.co/150x150?text=Nyancat",
  },
};
const cart = {  // by id
    jorts: { quantity: 3 }, // price and image in products!
};
```

Examples with Red Flags from using Arrays

```
export function addToCart(productId) {
  for (let i = 0; i < cart.length; i++) { // loop
    if (cart[i].id === productId) {
      cart[i].quantity += 1;  // Add to existing in cart
      productInCart = true;
      break;
    }
  }
  if (!productInCart) {   // Create new in cart
    for (let i = 0; i < products.length; i++) { // loop
      if (products[i].id === productId) {
        cart.push( /*...*/);
//...

export function plusQuantity(productId) {
  for (let i = 0; i < cart.length; i++) { // loop
    if (cart[i].id === productId) {
      cart[i].quantity += 1;  // Update existing in cart
      updateCart();
      return;
    }
  }
}
```

What if cart was an object?

```
export function addToCart(productId) {
  if(cart[productId]) {
    cart[productId].quantity += 1; // Add to existing in cart
  } else {
    cart[productId] = { quantity: 1 }; // Create new in cart
  }
}
//...
export function plusQuantity(productId) {
  cart[productId].quantity += 1;  // Update existing in cart
  updateCart();
}
```

Complexity is the enemy!

This looks sophisticated:

```
// Reducing quantity in cart
data.totalCost = parseFloat(
  Math.max(
    (data.totalCost -
      products.find(product => product.name === id)?.price
    ).toFixed(2),
    0
  )
);
```

But could also be:

```
data.totalCost -= products[id].price;
```

...or even not exist at all!

Derived State

Total Cost is known as **derived state**

- You don't want to store it as part of your state
- Violates single-source-of-truth
- Complicates changes
    - Miss one recalculation === subtle bug

Calculate derived state when you need it

- Usually render
- Not when a basis value changes
- Don't store derived values in state

Number or Text?

- All HTML-based values are text (Strings)
    - Even "numbers"
- We want to do math with Numbers
- We want to display as Strings (`.toFixed()`)

Always be clear what a value is!

- Convert to number when storing
- Math with numbers
- Convert to string when displaying
- Typescript can force/track
    - But you can/should do it regardless!

Parts of JS Cart should "feel" wrong

- Event to State to Render should feel good
    - But likely still new and unfamiliar
- But rendering can feel clumsy
    - Writing HTML in JS
    - Replacing a LOT of HTML for any state change

These are the right responses!

- We'll address these "pain points"

You are learning state management

State separated from Presentation

- Best practice
- Handles changes without complexity explosion
- Unnatural and inhuman

You have to learn to think this way

- Learning isn't instant

Writing HTML feels clumsy

Writing HTML in JS

- Not ideal

There are templating libraries

- Make it a little easier
- Do the same thing

We are skipping such libraries

- Understand what it is doing
- Will jump past to React very soon

Rendering feels wasteful

We rewrite a LOT of HTML on ANY state change

We could track which HTML depends on which state

- Write wrapper functions to change that state
- Trigger re-render of just those parts of HTML
- Would be a lot of work
    - Lots of edge cases and bugs to fix
- People have already done this work
    - Such as React
- For now we focus on learning OTHER aspects
    - So don't worry about efficiency for now