JavaScript Callback function

Imagine you have a favourite toy car, and you want to play with it with your friend. But before you start playing, you need to ask your mom to get it from the shelf for you.

So, you tell your mom, "Mom, can you please get my toy car from the shelf? When you have it, let me know." Here, you're giving your mom a job **(getting the toy car)** and telling her what to do when she finishes **(letting you know)**.

In this example, you're like the main program, your mom is like the function you're calling **(fetchData)**, and what you tell her to do after she finishes **(letting you know)** is like the callback function **(processData)**.

Once your mom gets the toy car, she tells you, "I got your toy car, here it is!" This is like the callback function being called after the main task **(fetching the toy car)** is done.

So, a callback is just like telling someone what to do once they finish their task, just like you tell your mom what to do after she gets your toy car.

Practical:

```javascript
// Imagine you have a function called "fetchData" which is like asking your mom to get your toy car.
function fetchData(callback) {
  // This is like your mom going to get the toy car, but it takes a little time.
  setTimeout(function() {
    const toyCar = "red car"; // Here, "toyCar" is like the toy car your mom got.
    // This line is like your mom telling you she got the toy car.
    callback(toyCar); // Here, "callback" is like telling your mom what to do after she gets the toy car.
  }, 2000); // SIMULATION This is like how it takes 2 seconds for your mom to get the toy car.
}

// Now, you have another function called "processData" which is like what you want to do with your toy car.
function processData(toyCar) {
  console.log('Yay! I got my', toyCar); // This line is like you being happy because you got your toy car.
}

// When you ask your mom (call the function "fetchData"), you also tell her what to do once she has the toy car.
fetchData(processData); // This line is like you telling your mom to let you know when she got the toy car.
```

- **fetchData** is like asking your mom to get your toy car.
- **processData** is like what you want to do with your toy car.
- **setTimeout** is like the time it takes for your mom to get the toy car.
- **callback** is like telling your mom what to do after she gets the toy car.
- **fetchData(processData)** is like asking your mom to let you know when she got the toy car.

Why do we use the **callback** function?

When you're working with JavaScript, you often encounter situations where **things take time to finish**, like loading data from a website or waiting for a user to click a button. Now, while JavaScript is waiting for these things to happen, it doesn't want to just sit there doing nothing. Instead, it wants to keep doing other stuff, like updating the page or responding to other actions.

Here's where callbacks come in. They're like little notes you give to JavaScript, telling it what to do once those time-consuming tasks are finally done. For example, you might tell JavaScript, **"Hey, when you're finished loading that data, let me know, and I'll decide what to do with it."**

Callbacks are handy because they **keep your code running smoothly even when it has to wait for things to happen**. Plus, they let you **decide** what to do with the **results** when they're **ready**, giving you a lot of **flexibility** in how your programs behave. So, callbacks are basically a way for JavaScript to stay productive while it's waiting for important tasks to finish up.

Callbacks in APIs enable JavaScript to handle asynchronous tasks efficiently, ensuring smooth performance and responsiveness. They allow developers to specify actions to take once API data is available, promoting flexibility and modular code. In essence, callbacks empower JavaScript to remain productive by managing tasks effectively during wait times for important operations to complete.

```javascript
// Define a function to handle the retrieved launch data
function handleLaunchData(data) {
  // Log the retrieved launch data to the console
  console.log("SpaceX launches:", data);

  // Display the launch data on a webpage
  const launchList = $("#launch-list");
  data.forEach(function(launch) {
    const listItem = $("<li></li>");
    listItem.text(`${launch.mission_name} - ${launch.launch_date_local}`);
    launchList.append(listItem);
  });
}

// Make an API call to retrieve launch data and provide a callback function
$.ajax({
  url: "https://api.spacexdata.com/v4/launches",
  dataType: "json",
  success: handleLaunchData, // Pass the callback function to handle successful response
  error: function(jqXHR, textStatus, errorThrown) {
    // Log an error message if the API request fails
    console.error("Error retrieving launch data:", textStatus, errorThrown);
  }
});
```