Advanced AJAX
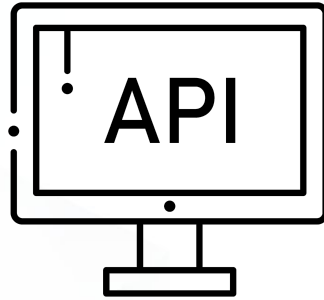
API Recap

What Is an API?

An **application programming interface (API)** is a set of features and rules that exist inside a software program (the application) enabling interaction with it through software—as opposed to a human user interface.

— MDN Web Docs, developer.mozilla.org

API Recap

In software development, APIs are often the bridge between different components.

3rd Party APIs

Your Browser

Browser APIs

Core JavaScript Language
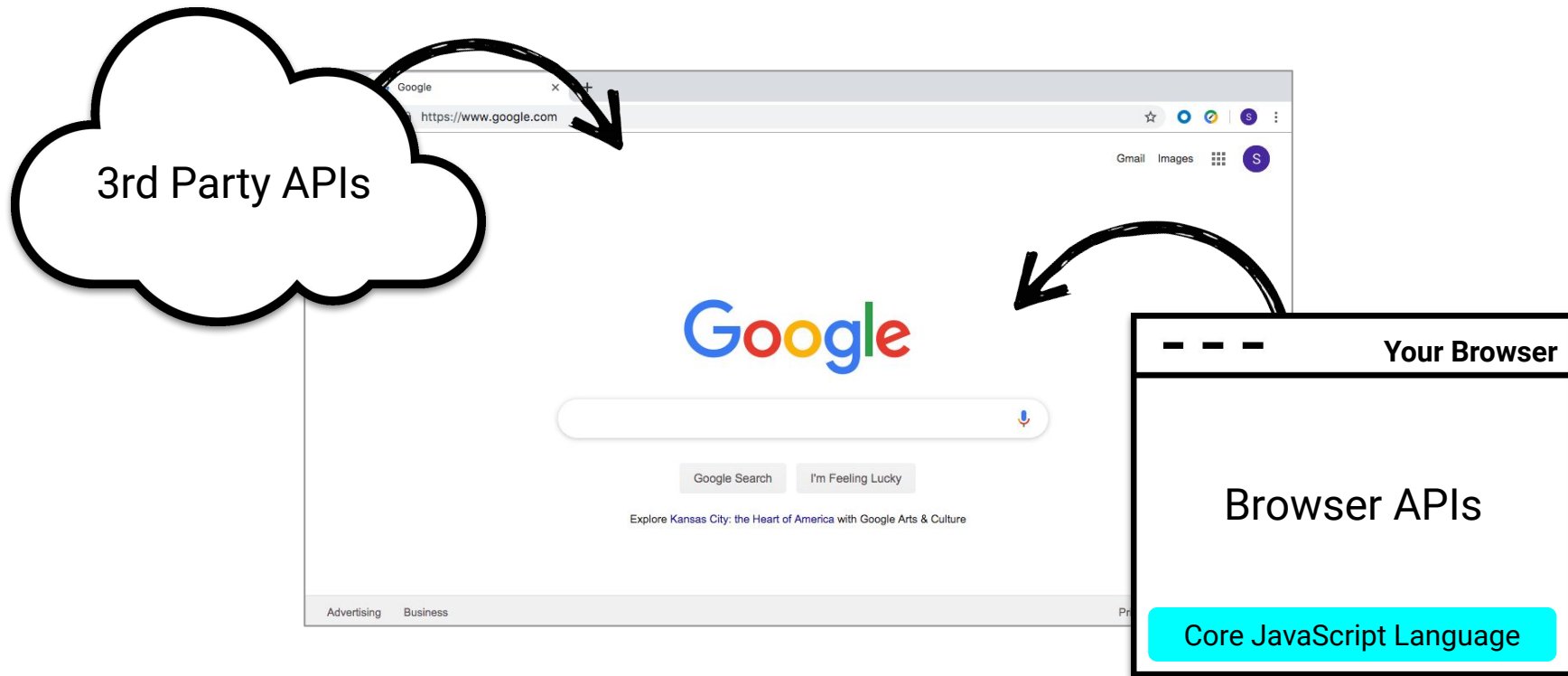
API Use Cases
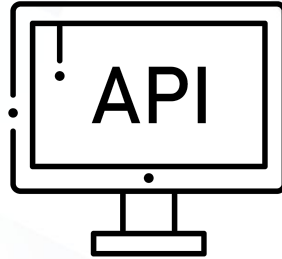
Three Common Use Cases for APIs

01 Provide pre-built code for accessing and sending data to a centralized database (e.g., weather data, movie data from IMDB).

02 Provide pre-built code for creating or utilizing other software components (e.g., Google Maps, Spotify tools).

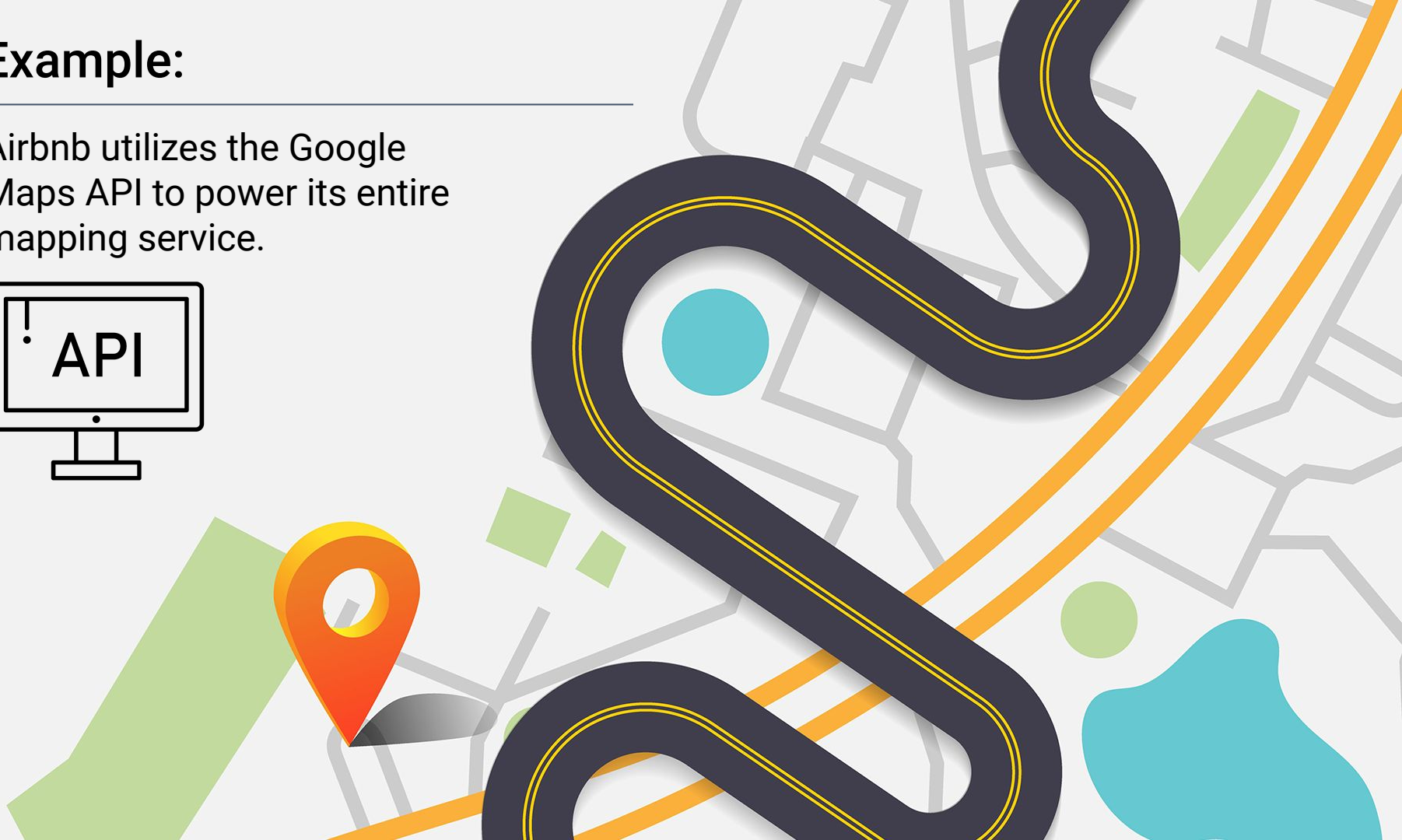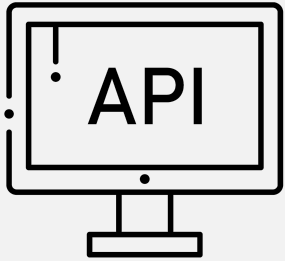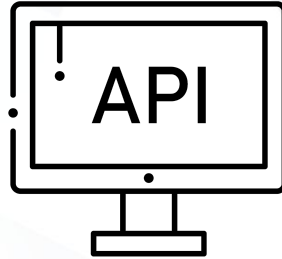03 Interface with physical sensors or hardware devices (e.g., Nest thermostats, Philips Hue).

Use Case #1:
Accessing and Sending Data

Example:

Airbnb utilizes the Google Maps API to power its entire mapping service.

API

Use Case #2:
Utilizing Pre-Built Code

Use Case #2: Utilizing Pre-Built Code

```javascript
var context;
  var request;
  var source;

  try {
    context = new AudioContext();
    request = new XMLHttpRequest();
    request.open("GET","http://jplayer.org/audio/mp3/RioMez-01-Sleep_together.mp3",true);
    request.responseType = "arraybuffer";

    request.onload = function() {
      context.decodeAudioData(request.response, function(buffer) {
        source = context.createBufferSource();
        source.buffer = buffer;
        source.connect(context.destination);
        // auto play
        source.start(0); // start was previously noteOn
      });
    };

    request.send();

  } catch(e) {
    alert('web audio api not supported');
  }
```
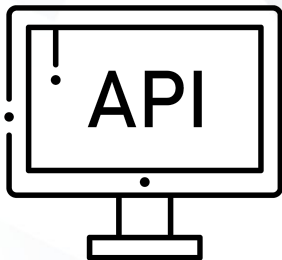
Use Case #3:
Controlling Physical Hardware

Nest Smart Thermostat API: developer.nest.com
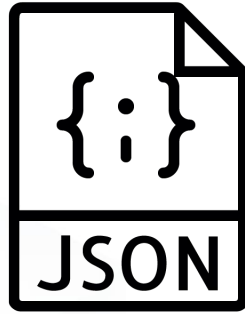Philips Hue API: developers.meethue.com
NodeBots (Cylon.js) API: cylonjs.com

JSON Recap

What Is JSON?

JSON stands for **JavaScript Object Notation** and is nothing more than simple JavaScript objects used as a **data interchange format.**

JSON Recap

JSON is a lightweight data interchange format used to correlate keys with values.

```
1   "apiVersion":"2.0",
2   "data":{
3       "updated":"2010-01-07T19:58:42.949Z",
4       "totalItems":800,
5       "startIndex":1,
6       "items":[
7           {"id":"hYB0mn5zh2c",
8            "uploaded":"2007-06-05T22:07:03.000Z",
9            "updated":"2010-01-07T13:26:50.000Z",
10           "uploader":"GoogleDeveloperDay",
11           "category":"News",
12           "title":"Google Developers Day US - Maps API Introduction",
13           "description":"Google Maps API Introduction ...",
14           "tags":["GDD07","GDD07US","Maps"],
15           "duration":2840,
16           "rating":4.63,
17           "viewCount":220101,
18           "favoriteCount":201,
19           "commentCount":22
20          }
21      ]
22  }
```

Getting Data

What **jQuery method** do we use to retrieve data from a URL database?

What **two parameters** do we pass into AJAX to retrieve data from online?

AJAX

What two parameters do we pass into AJAX to retrieve data from online?

```
 7  <script src='http://code.jquery.com/jquery-2.1.3.min.js'></script>
 8▼ <script type="text/javascript">
 9      var type = 'the+matrix';
10      var queryURL = "http://www.omdbapi.com/?t=" + type + "&y=&plot=short&r=json";
11
12▼     $.ajax({url: queryURL, method: 'GET'})
13        .done(function(response) {
14            console.log(response);
15      });
16
17  </script>
```

url   **'GET'** method

Knowledge Is Power

Welcome to Full-Stack Development

Full-stack development refers to the development of *every* aspect of the web application, from visuals and interactions to data transfer and processing.

USER

Client Browser (Front-End)

React.js

HTML/CSS/Image Assets

JavaScript (Client Side)

JSON/XML

User Interactions

Request

Response

Server (Back-End)

Node.js

Express
Web Server

Server-Side Scripts and Services

MongoDB
Database

Raw Files
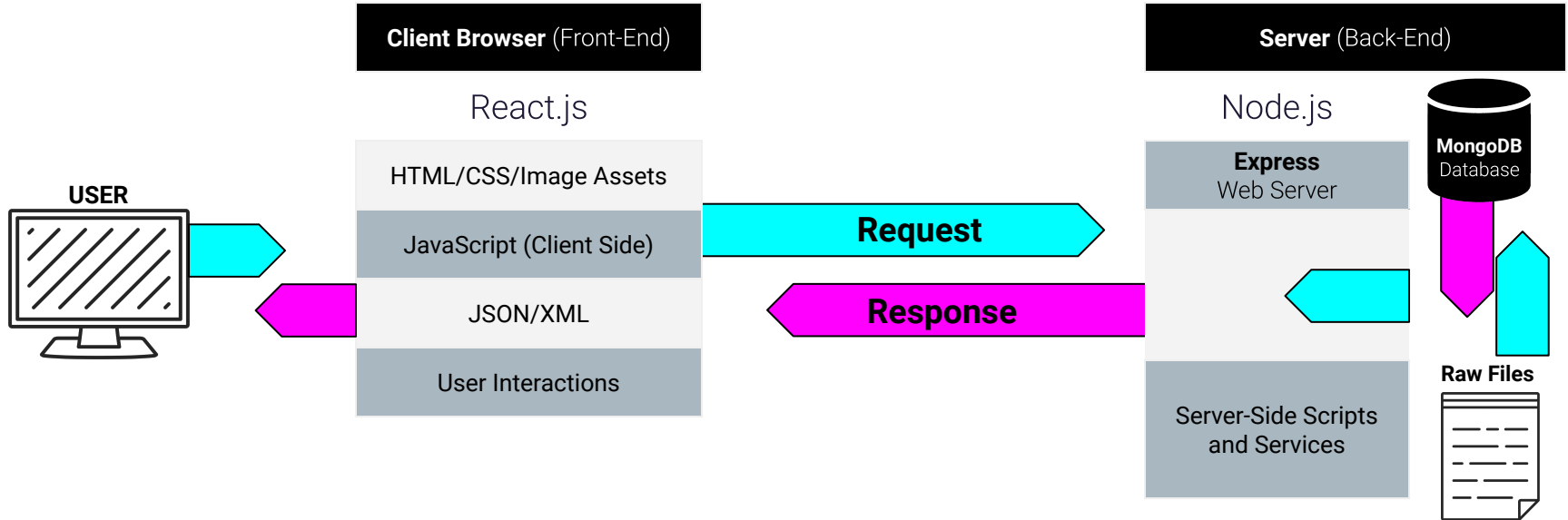
The front end displays the user interface, listens for user interaction events, and updates the display based on that interaction

It is like a cycle

"Paint" a user interface

update UI elements

listen for interactions

Do something based on that interaction (e.g. make a data request to a remote server)

get user inputs

Handle the user interaction
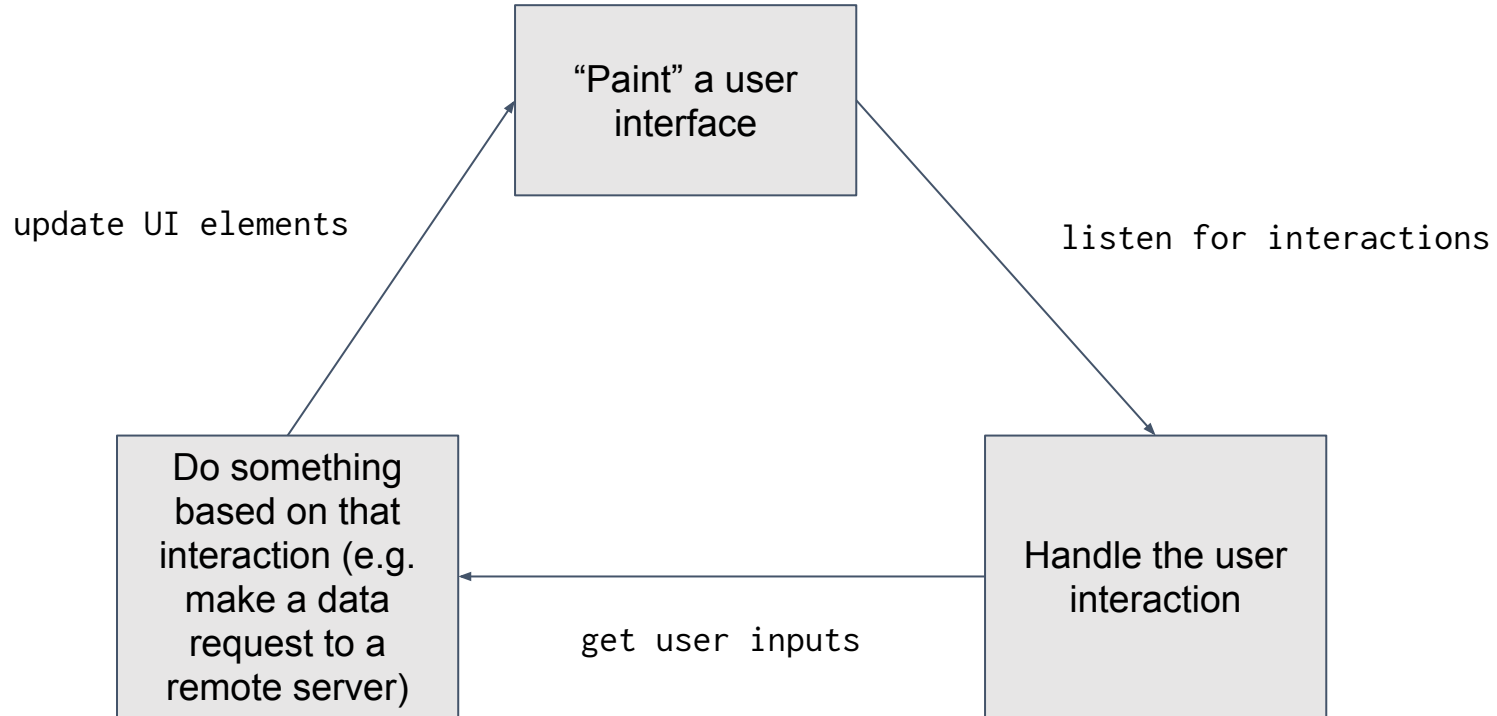
User Interaction: wait for something to happen

"Paint" a user interface

update UI elements

Do something based on that interaction (e.g. make a data request to a remote server)

get user inputs

Handle the user interaction

listen for interactions

**Event listeners, e.g.,
document.addEventListener("click")
or
$(document).on("click")**
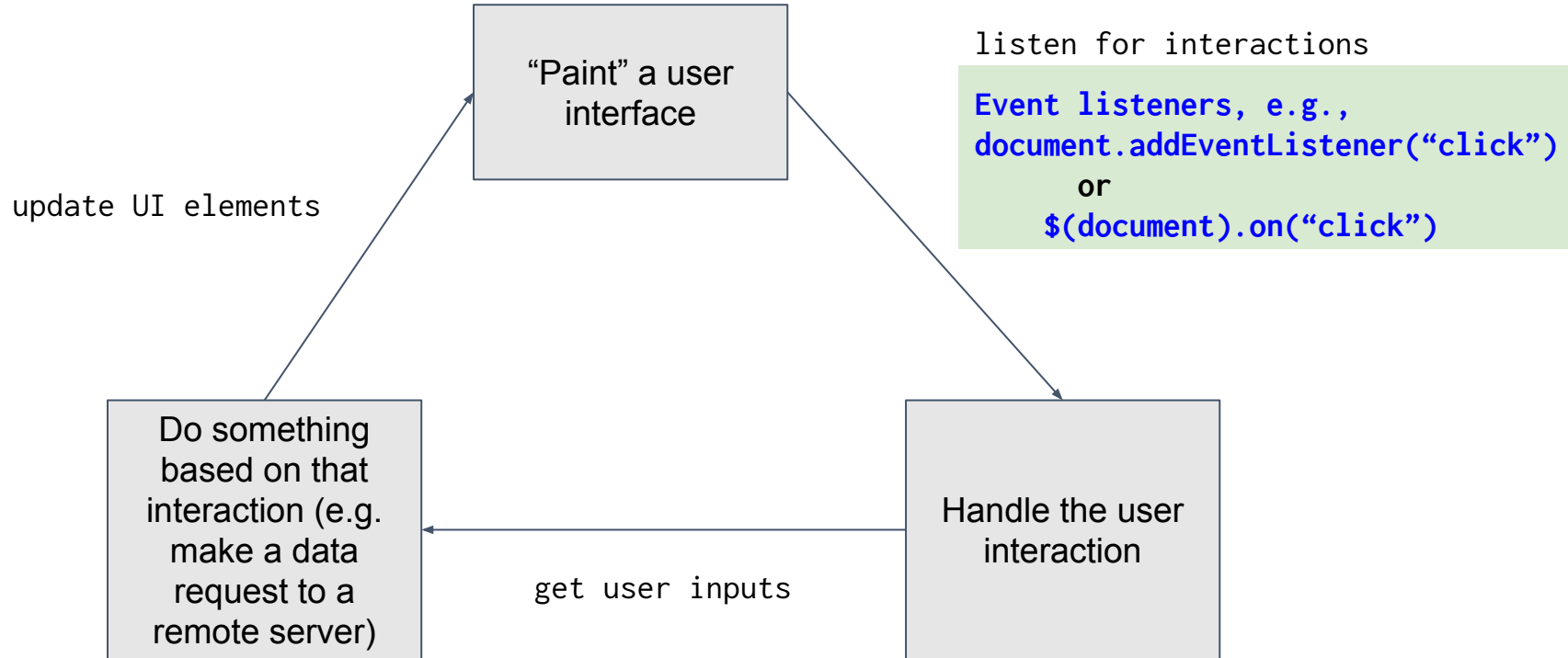
User Interaction: handle that event

"Paint" a user interface

update UI elements

listen for interactions

Do something based on that interaction (e.g. make a data request to a remote server)

Handle the user interaction

get user inputs

Event handler: a function that executes because of an event trigger
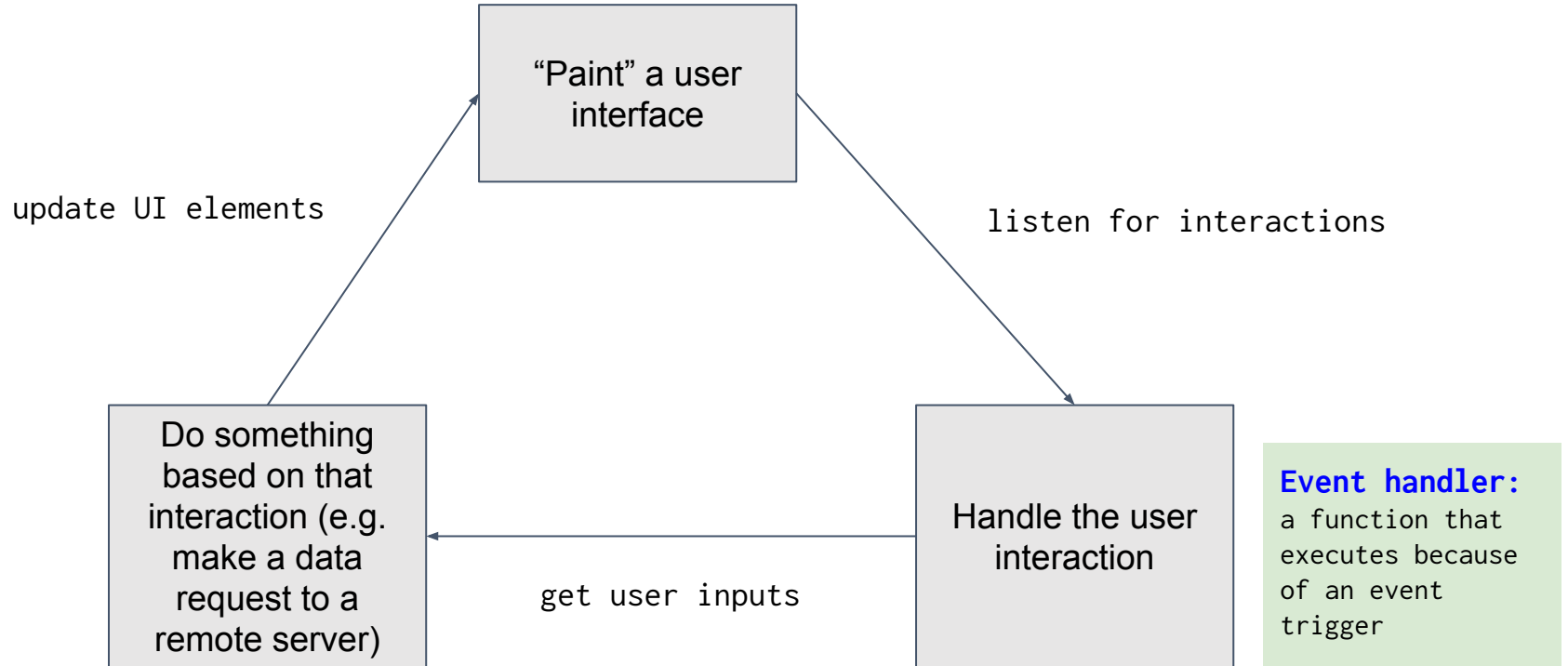
Event Handler

Plain Javascript

```
document.addEventListener("click", function(event){
    console.log("The event was triggered!");   ← Event Handler
});
```

jQuery

```
$(document).on("click", function(){
    console.log("The event was   ← Event Handler
triggered!");
});
```
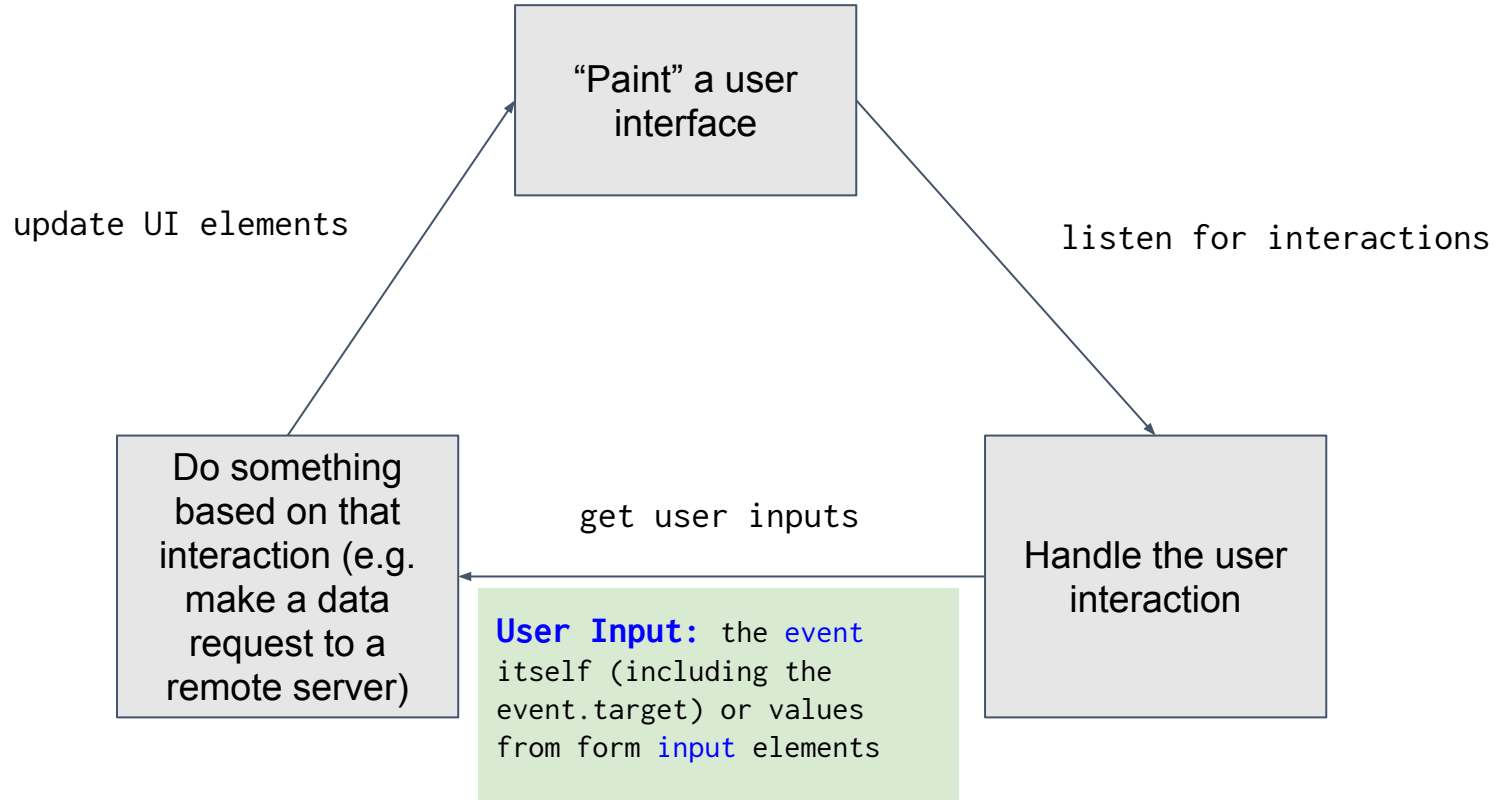
User Interaction: what did the user intend to do?

"Paint" a user interface

update UI elements

listen for interactions

Do something based on that interaction (e.g. make a data request to a remote server)

get user inputs

Handle the user interaction

User Input: the event itself (including the event.target) or values from form input elements

User Input: using the event target

Plain Javascript

```javascript
document.addEventListener("click", function(event){
    var itemId = event.target.getAttribute("data-itemId");
});
```
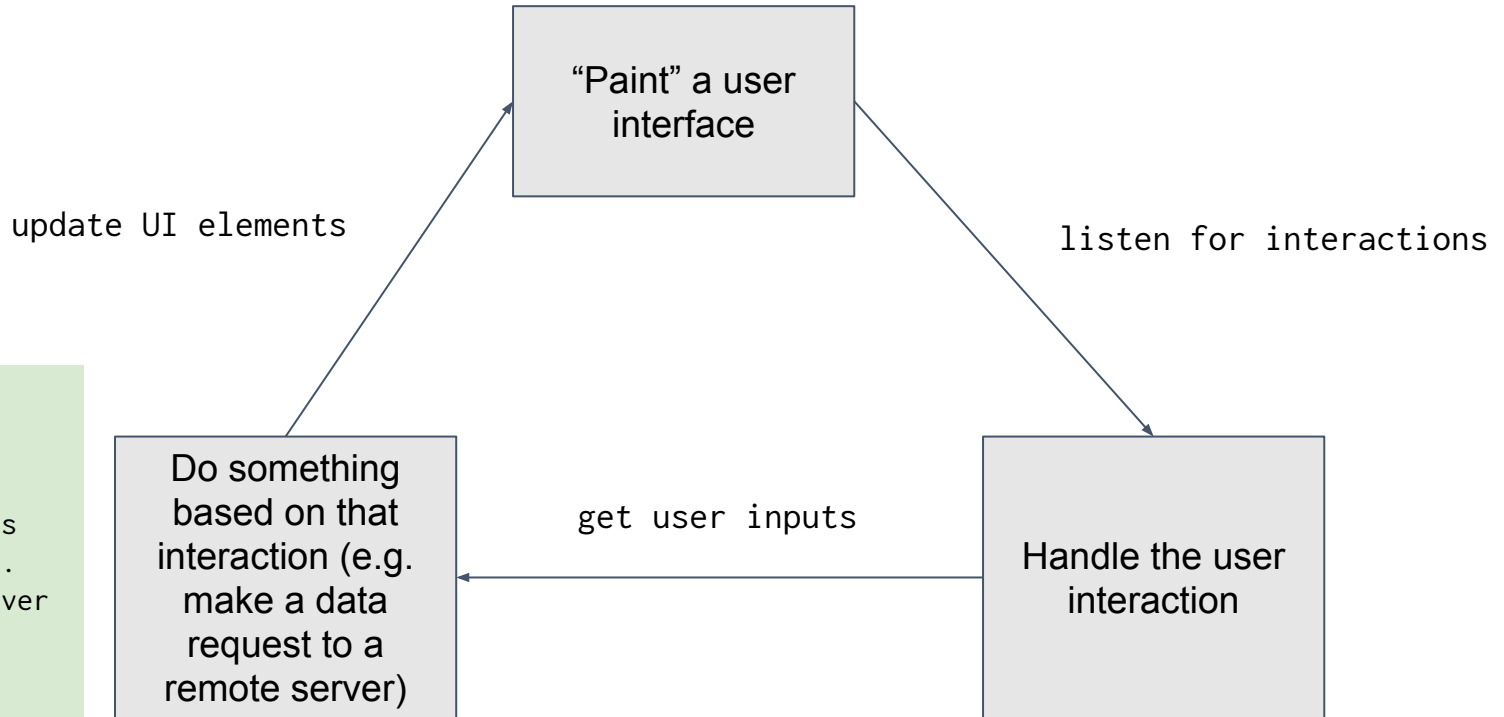
jQuery

```javascript
$(document).on("click", function(){
    var itemId = $(this).attr("data-itemId");
});
```

Data attributes

User Interaction: do some work

"Paint" a user interface

update UI elements

listen for interactions

Perform task:
usually involves updating the app's *state*: the variables that represent data. Local or from a server (i.e., make an ajax call)

Do something based on that interaction (e.g. make a data request to a remote server)

get user inputs

Handle the user interaction

Do a task: an AJAX request

jQuery

```
var itemName = "";
$(document).on("click", function(){
    var itemId =$(this).attr("data-itemId");
    $.ajax("https://myapi.com/api/getItem?id=" + itemId,
        function(response){
            console.log("Got a response!", response);
            itemName = response.data.itemName;
        }
    );
});
```
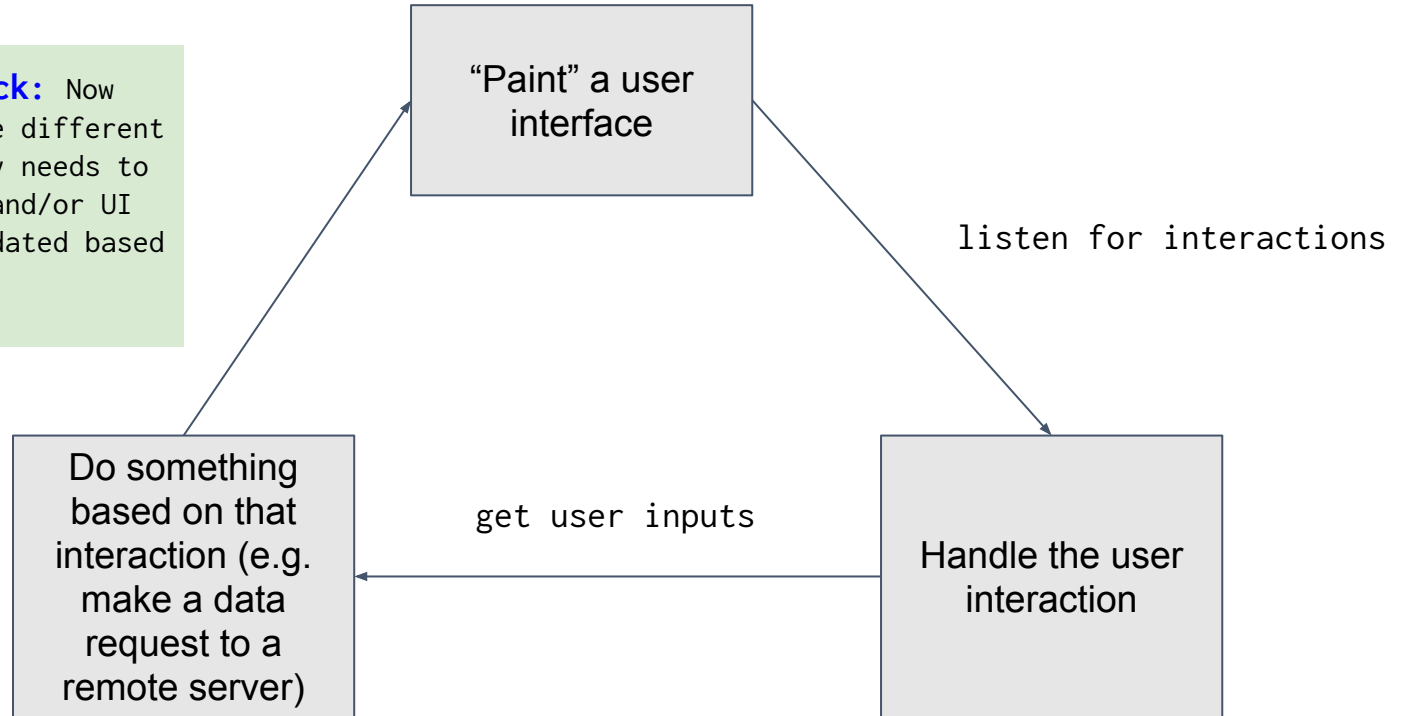
AJAX call

User Interaction: communicate back to the user

update UI elements

Provide user feedback: Now that our variables have different values, the UI probably needs to reflect that. Content and/or UI elements need to be updated based on the task result.

"Paint" a user interface

listen for interactions

Do something based on that interaction (e.g. make a data request to a remote server)

get user inputs

Handle the user interaction

Provide user feedback: update the UI:

jQuery

```
    var itemName = "";

    var itemNameEl = $("#item1 .name");

    $(document).on("click", function(){

        var itemId =$(this).attr("data-itemId");

        $.ajax("https://myapi.com/api/getItem?id= " + itemId,

            function(response){

                itemName = response.data.itemName;

                itemNameEl.text(itemName);

            }

        );

    });
```

AJAX call

AJAX and API Overview

At this point, you should know:

| 01 | The definition of API |

| 02 | The definition of JSON |

| 03 | Why AJAX methods are used (i.e., retrieving data in databases). |

| 04 | How to create a basic AJAX get request using jQuery (i.e., include URL and 'GET'). |

Questions?