Question 1:

Formally, an API is a previously defined interface to make two apps exchange information. In the real world there will be a client who sends a request to a server through the internet where it will perform a desired function and return the data, usually in the form of JSON or XML. Most provide synchronous operations such as create, read, update, and delete. While others rely on asynchronous subscriber/publisher/broker concept. An example would be how appications use PayPal to handle all their transactions which applies a form of abstraction. (Connects frontend and backend).

Question 2:

The formats used to send information between machines for RPC APIs are protocol buffers, XML and JSON. JSON being an alternative to XML for its ease of reading and writing data. Generally protocol buffers are used and defined in a .proto file which contains messages and name-value fields. Code is then generated from your .proto file.

Question 3:

```
if response.ok:
        date1 = response.json()['properties']['periods'][0]['startTime'])
        temp1 =  (response.json()['properties']['periods'][0]['temperature'])
        day1 = (response.json()['properties']['periods'][0]['name'])

        date2 = response.json()['properties']['periods'][1]['startTime'])
        temp2 =  (response.json()['properties']['periods'][1]['temperature'])
        day2 = (response.json()['properties']['periods'][1]['name'])

       return f"{day1} {date1} the expected temperature is {temp1} and {day2}
{date2} the expected temperature is {temp2}."
```

Question 4:

```html
<!DOCTYPE html>
<html>

<head>
    <style>
        .div1 {
            width: 450px;
            height: 100px;
            background-color: yellow;
        }

        .div2 {
            width: 900px;
            height: 200px;
            background-color: blue;
        }

        .div3 {
            width: 1800x;
            height: 400px;
            background-color: red;
        }
    </style>
</head>

<body>
    <div class='div3'>
        <div class='div2'>
            <div class='div1'></div>
        </div>
    </div>
</body>

</html>
```

Question 5:

```
<!DOCTYPE html>
<html>

<head>
    <title>Welcome to my webpage!</title>
</head>

<body>
    <h1>Welcome to my webpage!</h1>
    <button onclick="changeBackground()">Change background</button>
    <button onclick="helloWorld()">Display Message</button>
    <h2 id="message"></h2>
    <script>
        function changeBackground() {
            document.body.style.backgroundColor = 'orange';
        }
        function helloWorld() {
            var message = document.getElementById('message');;
            message.innerText = 'Hello, World!';
        }
    </script>
</body>

</html>
```

Question 6

The HTML needs to be adapted to have a meta viewport tag in the documents head. The content will then be set to “width=device-width”, which matches the screens width in device independent pixels. “Initial-scale=1’ can also be added to create a 1:1 relationship with CSS pixels when rotating the device.

Code Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
  </head>
```

Question 7:

A. Kotlin
B. Dart
C. Swift

Question 8:

Developers need to understand code generators limitations and that it is only as good as the one using it. Algorithms need to be designed by the programmer themselves and always expect basic replies in this regard since the generators are unable to do this. Code needs to be understood and tested since the generators are unreliable and may cause problems if you do not fully understand what is going on in the code.