Front-End Development Skills Evaluation Framework

Technical Brief

CodeSignal

Introduction

Front-End engineers are responsible for the planning, designing, building and implementation of user interface systems for web-based applications and/or software programs. The demand for this role is projected to grow by 13% from 2020 to 2030[1] as there is more emphasis across the industry around what product excellence means and how to design for an intuitive and excellent user experience. While many companies are looking for front-end engineers who have experience in one of the common frameworks such as *React*, *Angular*, and *Vue.js*, this framework paper will dive into the importance of mastering the fundamental building blocks of front-end development that include HTML, CSS, JavaScript.

Although front-end engineers share many of the core skills and knowledge with Software engineers such as problem solving and code-writing, the Front-End Development Framework is designed specifically to assess additional skills that are unique to Front-End engineers such as being able to translate UX design into functional applications, coordinating and handling back-end APIs, and understanding how to design intuitive and responsive applications.

This paper goes into detail about our guidelines for creating the framework based on consultation with subject matter experts with an emphasis on common core skills. We will then illustrate how the score is calculated as well as how the results map to the core skills for Front-End engineers.

Framework Specifications

This framework is designed to model closely to what the engineer would be expected to perform on the job. It can be utilized across different methods of delivery, assessment or interview, while preserving its objectivity by automatically calculating the final score.

The maximum allowed completion time for the framework is **90** minutes and it consists of **4** levels that are progressive in nature in mimicking a real world scenario. The scenario and what is being assessed should be common applications and/or features that ideally would not create any unfair advantages or disadvantages to candidate populations (e.g. asking a candidate to implement a short order or support limit feature are not ideal since it can create unfair

1 Bureau of Labor Statistics, U.S. Department of Labor, Occupational Outlook Handbook, Web Developers and Digital Designers, at https://www.bls.gov/ooh/computer-and-information-technology/web-developers.htm

advantages for candidates who may have prior trading knowledge as well as the concepts require too much elaboration in a limited time setting). Possible scores range from 200 to 600.

Level 1 – Common Layout and Basic Rendering

The average time for solving this level should be **15** minutes.

Expected Knowledge

- HTML layout and structure
- Document-Object Model (DOM)
- CSS styling

Can Include

- Standard HTML tags and attributes that are supported by common browsers (Chrome, Safari, Firefox, Edge)
- Simple scripting with JavaScript (e.g., rendering UI elements based on statically provided JSON-style data)
- Straightforward use of CSS for styling

Should Exclude

- Complex scripting with JavaScript (e.g., user interactions, calling APIs)
- CSS animations/transitions
- Tricky CSS cascading/selector specificity logic
- Responsive design
- SVG markup
- Cross-browser compatibility
- Knowledge of particular HTML/ templating or CSS/styling frameworks, including jQuery

Level 2 – Dynamic Interaction

The average time for solving this level should be **25** minutes.

Expected Knowledge

- Everything from the prior level
- Dynamic interactions using JavaScript and advanced CSS

Can Include

- Handling user actions in JavaScript
- Mutating the DOM using JavaScript
- Implementing data inputs and data validation

Should Exclude

- CSS animations/transitions
- Responsive design
- Interaction with APIs
- Cross-browser compatibility
- Knowledge of particular HTML/ templating or CSS/styling frameworks, including jQuery

Level 3 – Consuming an API

The average time for solving this level should be **30** minutes.

Expected Knowledge

- Everything from the prior levels
- Using asynchronous JavaScript to interact with an API

Can Include

- Knowledge of REST API standards
- Knowledge of how to asynchronously call remote APIs from the browser
- Basic/simple auth strategies like putting a token or password in HTTP headers
- Handling API errors and response

codes

- Working with APIs that include pagination
- Calling multiple, possibly dependent, API endpoints asynchronously

Should Exclude

- Less commonly used API protocols (e.g., WinForms, SOAP, RPCs, GraphQL)
- HTML form submissions (non-AJAX)
- Complex auth strategies like OAuth
- Websockets
- Server-side knowledge or implementation of the API
- Knowledge of particular JS libraries used for API interaction (e.g., axios, jQuery)

Level 4 – Extending Design Functionality

The average time for solving this level should be **20** minutes. Level 4 does not introduce any new fundamental skills or knowledge that are being evaluated, but it is focused on evaluating how candidates have approached the requirements up to this point and evolve its design to accommodate the expanded, yet related, requirements.

Expected Knowledge

- Everything from the prior levels

Framework Example Content

Below is an example of a question that is developed based on the structure of the framework. Similar questions are also created and monitored on an ongoing basis to ensure overall consistency as well as preventing widespread cheating and plagiarisms.

Scenario: Implement a simple task tracking application

Level 1 – Common Layout and Basic Rendering

The designer has provided you with the starting HTML template, the requirement is to take the provided JSON file data.json (see below) to properly assign the provided tasks into their corresponding columns based on the returned values: “To-do”, “In Progress”, “Done”.

Starting HTML Template

```
<div className="board">
  <h2 className="board__title">Tasks</h2>
  <div className="board__columns">
    <div className="column">
      <h2 className="column__title">TO_DO</h2>
      <div className="column__cards">
        <div className="card">
          <h3 className="card__title">Task 1</h3>
          <p className="card__description">Detailed task 1 description</p>
        </div>
      </div>
    </div>
  </div>
</div>
```

data.json

```json
{
  "todoItems": [
    {
      "title": "Task 3",
      "description": "Detailed task 3 description",
      "status": "TO_DO",
      "userId": "userId2"
    }
  ],
  "inProgressItems": [
    {
      "title": "Task 1",
      "description": "Detailed task 1 description",
      "status": "IN_PROGRESS",
      "userId": "userId1"
    }
  ],
  "doneItems": [
    {
      "title": "Task 2",
      "description": "Detailed task 2 description",
      "status": "DONE",
      "userId": "userId2"
    }
  ]
}
```

Level 2 – Dynamic Interaction

The designer has provided you with the following HTML template to add a form so that new tasks can be added to the board:

New Task HTML Template

```html
<form>
  <div class="input-container">
    <input name="taskTitle" placeholder="Task title*" value=""></div>
  <div class="input-container">
    <textarea name="taskDescription" placeholder="Task description*"></textarea></div>
  <div>
    <input type="submit" value="Create task"></div>
</form>
```

You are asked to support the following requirements:

- Enable users to create and append a new task that by default gets added to status TO_DO.

Level 3 – Consuming an API

The back-end API is now available to consume which allows you to get a list of all tasks to display on the board. The response of the API is slightly different format compared to the static JSON that is provided in Level 1:

Response to https://contentapi.codesignal.com/tasks

```json
{
  "data": [
    {
      "title": "Task 1",
      "description": "Detailed task 1 description",
      "status": "IN_PROGRESS",
      "assignedUser": "userId1"
    },
    {
      "title": "Task 2",
      "description": "Detailed task 2 description",
      "status": "DONE"
    },
    {
      "title": "Task 3",
      "description": "Detailed task 3 description",
      "status": "TO_DO",
      "assignedUser": "userId2"
    }
  ]
}
```

You are asked to use the updated task card template so that you can appropriately display the assigned user by consuming a separately API endpoint:

`Response to https://contentapi.codesignal.com/users/{userId}`

```json
{
  "id": "userId1",
  "firstName": "Andrew",
  "lastName": "Quill"
}
```

Level 4 – Extending Design Functionality

Enable users to move the tasks between different columns on the board - “Move right” and “Move left”. When the user presses the “Move right” button, the card must be moved to the column on the right unless it is at the right-most column. Apply the same logic to “Move left” as well. The very left column tasks shouldn’t include the “Move left” button, as well as the very right column tasks shouldn’t include the “Move right” button. The designer has provided you with the HTML template for the new buttons rendering:

`Move card buttons HTML template`

```html
<div className="card__buttons">
  <button
    aria-label="button left"
    className="card__button card__button--left"
    type="button"
  />
  <button
    aria-label="button right"
    className="card__button card__button--right"
    type="button"
  />
</div>
```