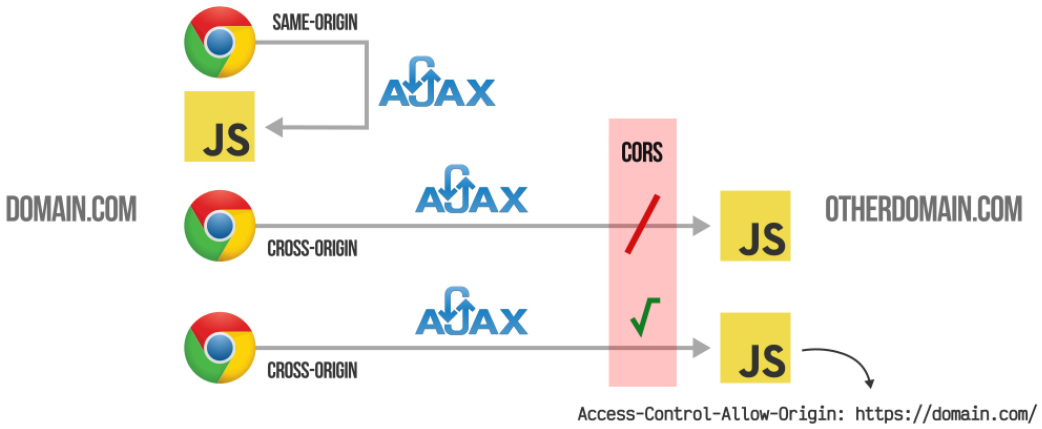
TEXT CLASS REVIEW

TEMAS A TRATAR EN EL CUE

- Implicancias del CORS y cómo resolverlas.
- Diferencia entre el modo desarrollo y el modo producción de Vue.js.
- Empaquetar una aplicación Vue.js usando Build Tools.
- Plataforma Github Pages para alojar una aplicación con Vue.js.
- Plataforma Firebase Hosting para alojar una aplicación Vue.js.

IMPLICANCIAS DEL CORS Y CÓMO RESOLVERLAS

Cuando se desarrolla o visita un sitio web, no sólo realizamos peticiones entre un sitio u otro para solicitar acceso al contenido audiovisual y datos, sino que también podemos modificarlos usando: `PATCH`, `PUT` y `POST`. Por defecto, los navegadores permiten las `same-origin request`, es decir, peticiones del mismo servidor donde se aloja la página web, ya que cumplen con lo reglamentado por la `SOP` (same-origin policy) o política de seguridad del mismo origen, que impide la descarga de contenido de otros servidores. Sin embargo, no ocurre lo mismo cuando se trata de peticiones `HTTP` asíncronas, es ahí cuando nace el concepto de las cross-origin requests (`COR`) o peticiones de origen cruzado. Si los administradores web de ambos sitios han acordado trabajar juntos, el intercambio se puede dar, y es regulado por el llamado cross-origin resource sharing (`CORS`); que es un mecanismo creado por la `W3C`, para otorgar permiso únicamente a clientes concretos, y cuenta con soporte en todos los navegadores actuales. Fue ideado para limitar el acceso a los recursos de un servidor foráneo, y establecer cómo éstos pueden ser modificados.



Si intentamos realizar una petición asíncrona hacia otro dominio diferente, sin haber establecido los permisos para `CORS`, probablemente obtendremos un error similar al siguiente:

Access to fetch at 'https://otherdomain.com/file.json' from origin 'https://domain.com/' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

Para evitarlo, en cada petición y respuesta entre los distintos servidores, debemos agregar una metadata llamada cabecera `HTTP` (HTTP Header), que contiene la información de quién consulta, bajo qué navegador lo hace, el tipo de contenido respuesta, entre otros. Existen diferentes cabeceras o `CORS headers`, y cada una aborda un aspecto distinto; para conocerlas, puedes revisar el siguiente enlace:

https://developer.mozilla.org/es/docs/Web/HTTP/CORS#las_cabeceras_http_de_respuesta

La que tiene mayor incidencia sobre decidir si un recurso es compartido o no, es `Access-Control-Allow-Origin`. Para incluir esta cabecera en la respuesta de la petición, debe indicarse el dominio al que se le quiere dar permiso:

```
Access-Control-Allow-Origin: https://domain.com/
```

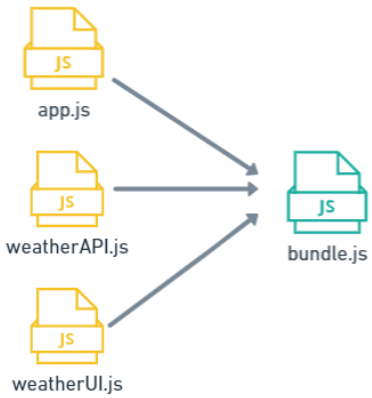
De esta forma, el navegador comprobará dichas cabeceras y si coinciden con el dominio de origen que realizó la petición, esta se permitirá.

DIFERENCIA ENTRE EL MODO DESARROLLO Y EL MODO PRODUCCIÓN DE VUE JS

Como ya lo hemos revisado anteriormente, es muy importante poder simular y probar nuestro producto antes de dejarlo disponible para el usuario final. Ya conocimos algunas herramientas que se encargan de automatizar las pruebas, y mencionamos que existe la posibilidad de hacer despliegue continuo, lo cual es muy útil cuando se trabaja en equipos y en aplicaciones grandes. Todo esto corresponde, principalmente, al modo de desarrollo, donde solo el equipo al cuál pertenecemos, podrá ver el producto final. También es importante recordar que, al ejecutar el comando `"npm run serve"`, se verá nuestro proyecto compilando, y su despliegue en el puerto local: http://localhost:8080. Una de las ventajas que tiene ejecutar un proyecto en modo de desarrollo, es que los cambios que aplicados, se compilan apenas los guardamos, sin necesidad de refrescar el contenido en el navegador. Cuando tenemos el proyecto listo, se hace uso del modo producción. Este nos permite empaquetar al máximo los archivos del proyecto, para que así la carga de la aplicación en el navegador sea lo más rápida posible. Al ejecutar nuestro proyecto en modo producción, tenemos que ir a la terminal de comandos, posicionarnos en la raíz, y ejecutar el comando: `"npm run build"`. Esto nos generará una nueva salida, que corresponde a la compilación de los archivos del proyecto en archivos estáticos, fáciles de almacenar en la caché del navegador del usuario, y que estarán preparados para ser desplegados a través de un servidor web.

EMPAQUETAR UNA APLICACIÓN VUE.JS USANDO BUILD TOOLS

`JavaScript` es un lenguaje que ha ido modernizándose constantemente con nuevas características. Esto ha generado un problema de incompatibilidad con las numerosas versiones de navegadores que existen, lo que en inglés se conoce como: browser `support`. Para evitarlos, existen herramientas que "traducen" nuestro código de `JavaScript` moderno, a uno más antiguo y compatible con todos los navegadores. Aquí es cuando entran los empaquetadores o herramientas de `bundling`; su idea es ofrecer un sistema de compactación con soporte para los distintos tipos de archivos o tecnologías, y así hacer más fácil y preciso el proceso de compilación.



Build Tools:

- **Webpack:** nació a finales de 2012, de la mano de Tobias Koppers. Es uno de los empaquetadores o `bundlers` más populares en la actualidad, y cuenta con la mayor cantidad de herramientas complementarias `open-source` desarrolladas por terceros. Nos permite empaquetar los diversos bloques de código, que se encuentran en los archivos `JavaScript` o módulos que componen una app, en uno sólo llamado comúnmente `"bundle.js"`. Así, nos libra de gestionar las dependencias entre archivos; y nos ofrece soporte para compilar otro tipo de archivos como: sass, scss, png, gif, html, entre otros, y sirve como servidor de desarrollo con `Webpack Dev Server`. Para funcionar, requiere que le configuremos `loaders` y `plugins`, para cada tipo de empaquetación que necesite nuestra aplicación.

- **Parcel:** es un empaquetador de archivos para aplicaciones web, que se diferencia por la experiencia ofrecida a los desarrolladores al ser el más fácil de usar. Tiene un rendimiento ultra-rápido, utilizando procesamiento multinúcleo; también dispone de soporte para la compilación de archivos como `css`, `javascript`, `HTML`, entre otros y, a diferencia de `Webpack`, no requiere configuración, por lo que en algunos tipos de proyectos, dará un resultado poco óptimo.

PLATAFORMA GITHUB PAGES PARA ALOJAR UNA APLICACIÓN CON VUE.JS

Como ya hemos visto durante el desarrollo de este curso, `GitHub` es una plataforma de desarrollo colaborativo, donde usando el sistema de control de versiones de Git, podemos gestionar nuestro código, además de guardarlo y compartirlo con la comunidad. Su servicio de hosting es `Github pages`, y en él podemos publicar páginas desarrolladas con `js`, `css`, `html`, realizadas usando distintas librerías o `frameworks`, tales como: `Angular`, `React` y `Vue`. No necesita configuración de servidores ni de base de datos, por lo que es muy simple de usar.

Antes de empezar, debes tener en consideración lo siguiente:

- Tener una cuenta de GitHub creada.
- Tener GIT instalado en tu equipo.
- El proyecto contar con un archivo index.html.

PLATAFORMA FIREBASE HOSTING PARA ALOJAR UNA APLICACIÓN VUE.JS

En uno de los CUE anteriores, conocimos y aprendimos a trabajar con `Firebase`. Entre los múltiples servicios de los que dispone, encontramos `Firebase Hosting`, que proporciona hosting de contenidos web, con nivel de producción orientado a desarrolladores. Es seguro y rápido para las aplicaciones web, el contenido dinámico y estático, y los microservicios. También se puede sincronizar con `Cloud Functions` o `Cloud Run`, para compilar y alojar microservicios en `Firebase`. Está respaldado por una red de distribución de contenidos (`CDN`) global. Además, se puede usar con un dominio propio, o en los subdominios gratuitos del proyecto en `web.app` y `firebaseapp.com`.