Programación con JavaScript II

Transpiladores

Introducción

Conforme avanza la tecnología, existen nuevos lenguajes de programación con interfases gráficas más potentes, diseñados para mejorar la experiencia del usuario.

Estos avances en la infraestructura tecnológica y en los lenguajes de programación provocan que algunos navegadores web no soporten algunas de las nuevas funcionalidades, lo que los convierte en obsoletos e incluso pone en desventaja a los mismos desarrolladores del software.

Ante esta situación se han creado herramientas que permiten hacer la “traducción” de las líneas de código más actuales a líneas de código que sí son posibles de interpretar por cualquier plataforma web. Esto representa una ventaja para el programador, ya que puede adaptar sus desarrollos con facilidad. En este tema se verán los transpiladores y ejemplos de la forma en que pueden utilizarse para que los nuevos desarrollos no enfrenten dificultades al correr en distintas plataformas.

Explicación

¿Qué es la transpilación?

Son herramientas que permiten reescribir código en una estructura que pueda ser reconocida por diferentes plataformas web, de forma que sea posible aprovechar los avances del lenguaje de programación sin tener que codificar de manera manual una versión que se adapte a intérpretes más viejos.

En otras palabras, un transpilador permite traducir un código a otro. Su nombre es una adaptación del término en inglés transpiler, conformado por *Translator + Compiler o un S2S Compiler (Source to Source Compiler).*

El **transpilador** es de gran ayuda para el desarrollador, pues su trabajo puede ser convertido de una sintaxis y estructura modernas a otra más antigua en muy poco tiempo y sin gran esfuerzo.

Kantor (2022) ofrece el ejemplo que se muestra a continuación, donde un desarrollador utiliza la siguiente línea de código y el transpilador lo convierte de manera automática a otra estructura de código para un navegador desactualizado.

Código actual	Código actual
height = height ?? 100	(height !== undefined && height !== null) ? height : 100.

Las siguientes herramientas son bastante útiles para reconocer la compatibilidad del código:

- https://caniuse.com/
- https://kangax.github.io/compat-table/es6/

¿Por qué nos debería interesar usar transpiladores?
Desde que se lanzó la primera edición de la publicación de ECMAScript en 1997, que contiene las especificaciones de lenguaje de scripting en la que se basa JavaScript, los profesionales de la ingeniería de software publican periódicamente una nueva edición con mejoras sobre el rendimiento y la eficacia de este código. En junio del 2022 se publicó la 13ª versión con algunos conceptos como ***AWAIT***, que permite esperar un "promesa", dentro de la función ***ASYNC***, nuevas clases de elementos públicos y privados, instancias privadas de métodos, por mencionar algunos (ECMA International, 2022).

Esta evolución hace difícil que compañías dueñas de navegadores de Internet estén a la vanguardia en estos cambios, además de que los desarrolladores se preocupan al adoptar estos avances, pues pueden poner en riesgo el rendimiento de sus desarrollos actuales. Es aquí donde intervienen los transpiladores.

Si, por ejemplo, existe un desarrollo que adopte las novedades de ECMAScript, es posible usar los transpiladores para hacer la traducción de manera automática y minimizar el riesgo de que no funcione correctamente en diferentes plataformas.

Explicación

Babel

Entre las herramientas **“transpiladoras”** más populares se encuentra Babel, que, haciendo alusión a la historia bíblica, funciona como traductor. Se considera una “herramienta en cadena” (toma información de una cadena de caracteres y evalúa su contenido) que permite trasladar sintaxis de código fuente de las versiones de JavaScript más recientes para que se utilicen en ambientes web de versiones anteriores.

Ejemplo:
Babel ofrece la funcionalidad de conversión de código desde una página web en donde se pueden agregar archivos completos, o bien, bloques de código para ser traducidos a líneas de código compatibles con otros navegadores: https://babeljs.io/repl

En este ejemplo puedes observar cómo se prueba con el operador boleano *“nullish coalescing”* (operador de uso combinado NULL) introducido en la versión ECMAScript 2020, que permite devolver el valor del operando izquierdo en caso de que sea *“null”*, de lo contrario, evaluará el operando derecho y regresará su resultado:

```
// ES2020 nullish coalescing
function greet(input) {
    return input ?? "Hello world";
  }
```

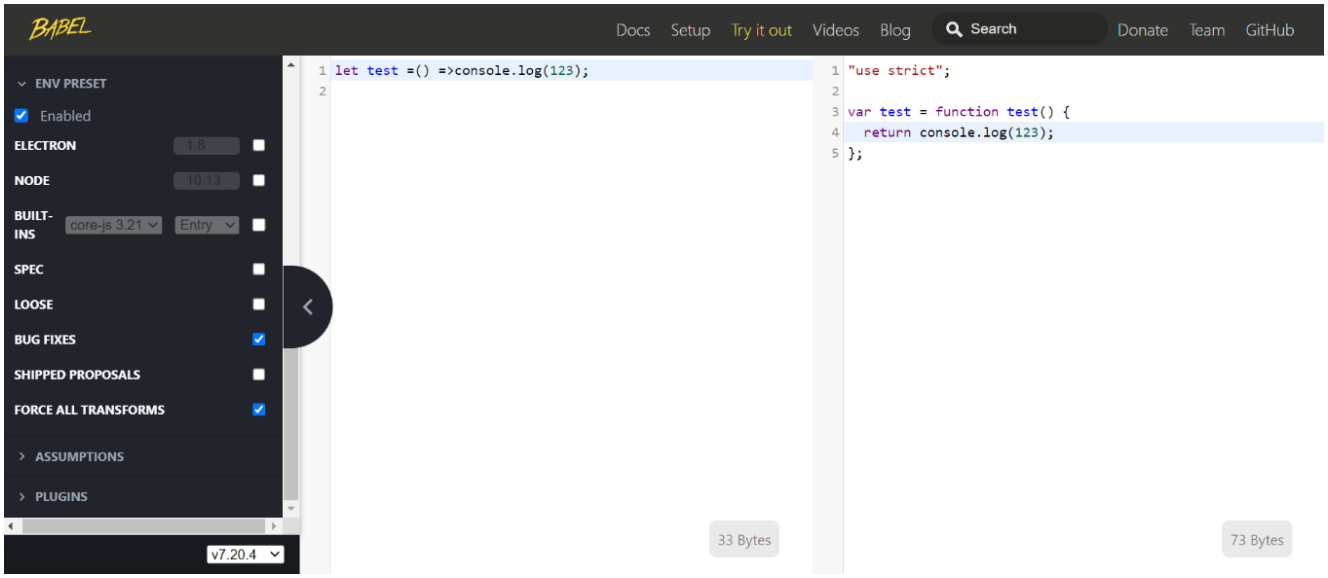
Al introducir este código en la página de Babel, se obtiene el siguiente resultado, que es un código JavaScript que navegadores no tan actuales pueden leer, como Internet Explorer 11:

```
"use strict";

// ES2020 nullish coalescing
function greet(input) {
  return input !== null && input !== void 0 ? input : "Hello world";
}
```

Explicación

En la siguiente imagen es posible observar los dos paneles de la página babeljs.io. A mano izquierda se encuentra el código original y a mano derecha la traducción.



Fuente: Babel. (s.f.). Babel Try it out. Recuperado de https://babeljs.io/repl

Nota: Asegúrate de que la opción ***Force All Transcripción*** que aparece al final del menú izquierdo se encuentre activa.

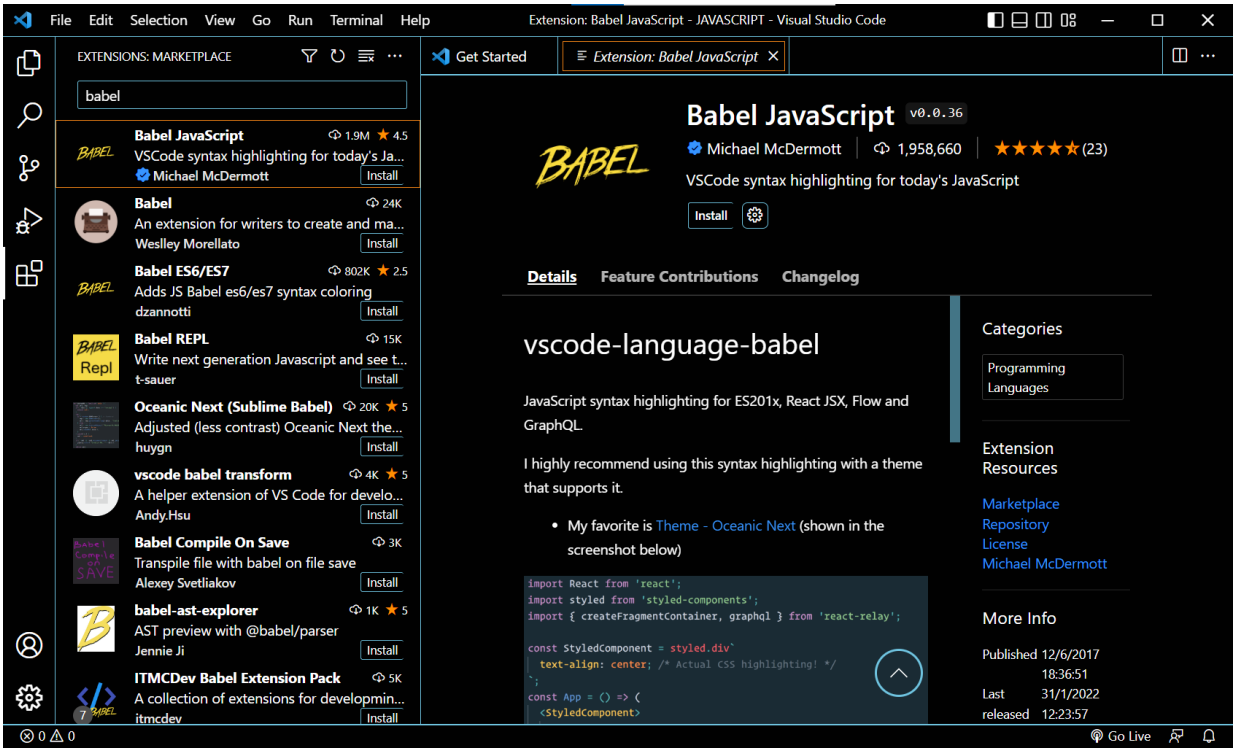
Algunas otras alternativas para realizar pruebas directamente en una página web como Babel son las siguientes:

- JSFiddle: https://jsfiddle.net/fh5whLfd/
- JSBin: http://jsbin.com/rokimopuse/edit?html,js,console,output
- Codepen: https://codepen.io/anon/pen/dOGgeO
- JSitor: https://jsitor.com/P1Br0ZbSF

Explicación

Uso de Babel en Visual Studio Code:

1. Antes de instalar Babel, considera que primero debes contar con Nodejs (un entorno para ejecutar JavaScript).
2. Instala la extensión Babel.



Fuente: Microsoft. (s.f.). Visual Studio Code. Recuperado de https://code.visualstudio.com/

3. Abre una ventana Terminal de Visual Studio y coloca la siguiente instrucción: npm inst -y.

4. Una vez instalado, se procede con el siguiente comando, asumiendo que el Node Package Manager registre Babel de manera local.

npm i @babel/core @babel/cli

Una vez que termine de registrarse, se observará que se ha incluido el siguiente bloque en el archivo **package.json** (la versión podría variar a una más actualizada, ya que depende del momento en el que se registró).

Package.json

```
{ "devDependencies": {
    "@babel/cli": "^7.0.0",
     "@babel/core": "^7.0.0" }
},
```

Explicación

5. Antes del siguiente paso, es necesario contar con un archivo de código cuyas características sean de una versión compatible con ECMAScript 2015 y que desee ser convertido a una versión compatible con todos los navegadores. Un ejemplo de esto es el que se muestra en la siguiente imagen, llamado index.js, que contiene declaraciones de constantes, operadores de deconstrucción, el operador rest y, en general, una sintaxis moderna que algunos navegadores no soportan.

index.js

```
const abc={ a: "a", b: "b", c: "c" };
const { a, ...bc} = abc;
console.log(bc);
```

6. El siguiente paso es abrir el archivo **package.json** e incluir la sección ***build*** dentro del campo ***scripts***, tal y como se muestra en la siguiente línea de código. Esta instrucción permitirá convertir el archivo index.js en código anterior a ECMAScript 2015 y lo guardará en el archivo **index-compiled.js**.

"build": "babel index.js -o index-compiled.js"

package.json

```
{
  "name": "my-project",
  "version": "1.0.0",
 "scripts": {
   "build": "babel index.js -o index-compiled.js"
 },
  "devDependencies": {
    "@babel/cli": "^7.0.0"
  }
}
```

7. Aplica enseguida la siguiente instrucción:
npm install @babel/preset-env -D

8. El último paso de esta configuración es crear el archivo llamado **babel.config.json** y colocar el siguiente bloque:

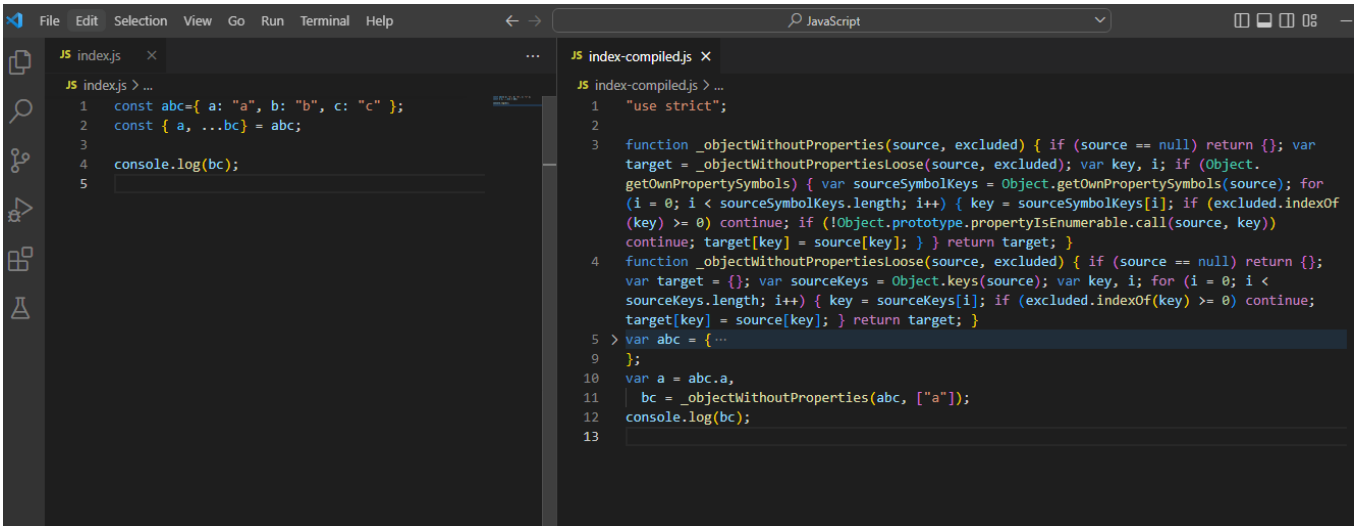
babel.config.json

```
{
  "presets": ["@babel/env"]
}
```

9. Finalmente, se ejecuta el comando que transformará el código index.js: npm run build.

Explicación

En la siguiente imagen puedes observar a mano izquierda el código JavaScript en su versión más actual, contenido en el archivo **index.js**, mientras que a la derecha aparece el código con una sintaxis que puede ser interpretada por navegadores no actualizados (archivo **index-compiled.js**).



Polyfills

Según MDN (s.f.-b), un *polyfill* es un término que se utiliza para describir a un extracto de código JavaScript que permite que a una funcionalidad moderna la acepten de manera nativa los navegadores antiguos. Una de sus funciones más importantes es su capacidad para adaptar componentes HTML5 a los navegadores, lo que le permite usar funciones como el canvas-element para gráficos, diagramas y animaciones.

El término fue acuñado por Remy Sharp en 2009, pensando en encontrar una forma de crear una API para cuando el navegador no pudiera interpretar JavaScript. De acuerdo con Sharp (2010), el término se le ocurrió juntando las palabras ***Poly***, dando la idea de que usaría muchas técnicas, y ***fill***, para llenar los huecos que requería el navegador para soportar el código JavaScript. Un término que es muy parecido al producto llamado ***Polyfillia***, el cual se utiliza para rellenar grietas en las paredes en Gran Bretaña.

La gran ventaja del uso de los polyfills es que pueden incrustarse en proyectos de código HTML, por ejemplo, en el siguiente código puedes observar el uso del método **startWith**. Este método determina si una cadena de caracteres empieza con otra cadena de caracteres. Si el navegador no soporta este método, es decir, si el resultado de la siguiente función es verdadero: **if (!String.prototype.startsWith)**, entonces el método es sobrescrito con **substr** para lograr el mismo objetivo.

```
if (!String.prototype.startsWith) {
    String.prototype.startsWith = function(BuscaCadena, posicion){
      return this.substr(posicion || 0, BuscaCadena.length) === BuscaCadena;
  };
}
```

Explicación

En el siguiente ejemplo se aplica un **polyfill** para la función ***Math.trun***. Esta función se utiliza para regresar la parte entera de un número, eliminando la parte decimal.

```
if (!String.prototype.startsWith) {
    String.prototype.startsWith = function(BuscaCadena, posicion){
      return this.substr(posicion || 0, BuscaCadena.length) === BuscaCadena;
  };
}
```

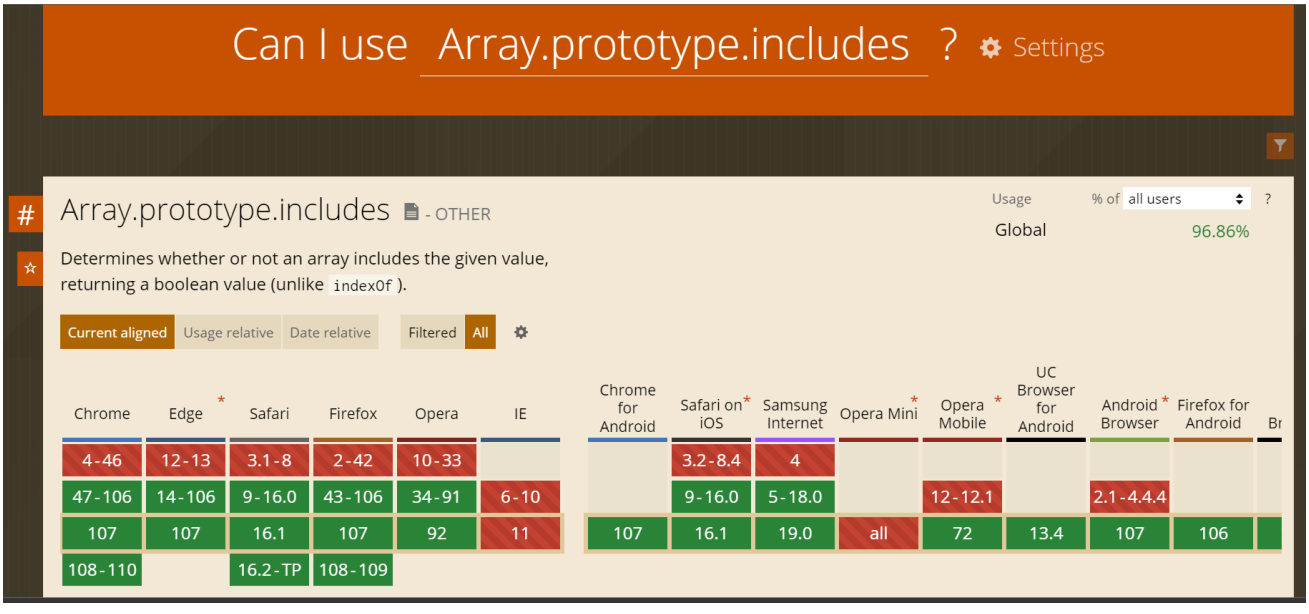
Como habrás notado, tanto las estructuras Polyfills como el código Babel son la mancuerna perfecta para asegurar que el desarrollo web funcione en diferentes navegadores. Solo será necesario convertir funciones modernas con un transpilador.

Herramienta *"Can I Use"*

Como es lógico, existen funciones de JavaScript ECMAScript con actualizaciones que posiblemente desconoces si son soportadas o no por los navegadores. La siguiente página puede ayudarte a tener más certeza al respecto: https://caniuse.com/

Esta página permite escribir una función JavaScript y reconocer las versiones de los navegadores que la soportan. De esa manera te puedes asegurar de que es necesario implementar estructuras polyfills que eviten el riesgo de lanzar un desarrollo que no opere de manera adecuada.

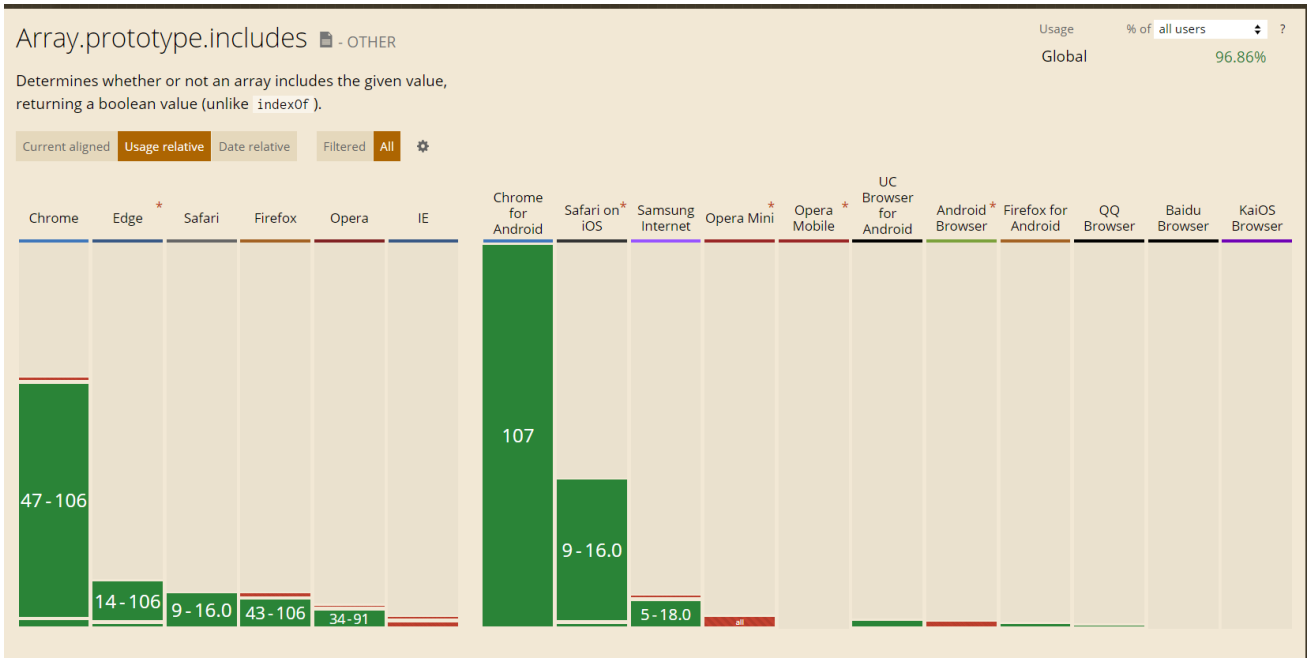
En la siguiente imagen se observan las versiones de cada navegador que soportan el método ***Array.prototype.***includes, la cual es una función que determina si un arreglo incluye un determinado elemento, devolviendo un falso o verdadero.



Fuente: Caniuse.com. (s.f.). *Can I Use*. Recuperado de https://caniuse.com/

Explicación

Esta herramienta también incluye el uso relativo de la función en diferentes navegadores (**opción: "usage relative**").



Fuente: Caniuse.com. (s.f.). *Can I Use.* Recuperado de https://caniuse.com/

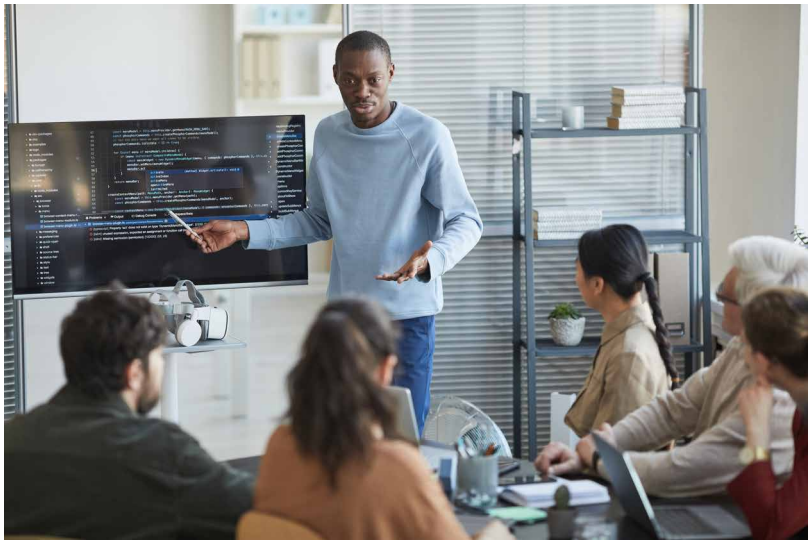
Te invitamos a usar esta herramienta cada vez que tengas dudas sobre la estabilidad del desarrollo web de JavaScript.

Cierre

Imagina que estás a punto de hacer una presentación del proyecto de desarrollo web a los inversionistas de la compañía que contrató tus servicios y en el que estuviste trabajando con un equipo de desarrolladores por tres meses. Te encuentras ya en la sala de reuniones haciendo la demostración cuando uno de los ejecutivos te solicita hacer la prueba en su propio equipo de cómputo. En ese momento te das cuenta de que usa el navegador Internet Explorer y que la aplicación web no carga porque ya es obsoleto, situación que requerirá de tiempo para explicar.

Los proyectos de desarrollo web deben asegurar su funcionalidad ejecutándose sobre diferentes plataformas, de lo contrario, podrías perder la oportunidad de llegar a usuarios que utilizan navegadores obsoletos, impactando en la imagen de la compañía fabricante del software o de la misma organización a la que representan.

En este tema revisaste el concepto de transpilador y de polyfills, herramientas que aseguran la experiencia del usuario en los desarrollos web creados con JavaScript.



Referencias bibliográficas

- Babel. (s.f.). *Babel Try it out.* Recuperado de https://babeljs.io/repl
- Caniuse.com. (s.f.). *Can I Use.* Recuperado de https://caniuse.com/
- ECMA International. (2022). *ECMAScript Language Specification* (13ª ed.). Suiza: ECMA International. Recuperado https://www.ecma-international.org/wp-content/uploads/ECMA-262_13th_edition_june_2022.pdf
- Kantor, I. (2022). *Polyfills y transpiladores.* Recuperado de https://es.javascript.info/polyfills
- Microsoft. (s.f.). *Visual Studio Code.* Recuperado de https://code.visualstudio.com/
- MDN. (s.f.-a). *Usar Promesas.* Recuperado de https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Using_promises
- MDN. (s.f.-b). *Polyfil.* Recuperdo de https://developer.mozilla.org/es/docs/Glossary/Polyfill
  Sharp, R. (2010). *What is a Polyfill?* Recuperado de https://remysharp.com/2010/10/08/what-is-a-polyfill

Para saber más

Lecturas
Para conocer más acerca de **transpiladores**, te sugerimos leer lo siguiente:

- Kantor, I. (2022). *Polyfill y transpiladores*. Recuperado de https://es.javascript.info/polyfills
- Riyaa7vermaa. (2022). *How to use polyfill in JavaScript?* Recuperado de https://www.geeksforgeeks.org/how-to-use-polyfill-in-javascript/

Videos
Para conocer más acerca de **transpiladores**, te sugerimos revisar lo siguiente:

- Midulive. (2022, 22 de mayo). *¿Qué es un Polyfill? ¿Cómo crear un polyfill desde cero? Típica pregunta de entrevista* ✅ [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=DyfX6ZWVky4.

- Steve Griffith - Prof3ssorSt3v3. (2018, 20 de julio). *Creating JavaScript Polyfills* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=qTCM8td2klQ

Checkpoints

Asegúrate de:

- Entender las funciones de un transpilador para asegurar la correcta ejecución de las funcionalidades modernas de código en navegadores antiguos.
- Aplicar las funcionalidades de la herramienta Babel para convertir estructuras de código modernas a un código que pueda ser interpretado por navegadores obsoletos.
- Comprender la necesidad de usar Polyfills en el desarrollo web de JavaScript para asegurar la funcionalidad en diferentes navegadores.

Requerimientos técnicos

- Uso de Visual Studio Code
- Acceso a Internet

Prework

- Deberás revisar el tema 7.
- Para realizar la actividad guiada se requiere:
    - El uso de Visual Studio Code.
    - Instalar la extensión de Babel sobre Nodejs (https://nodejs.org/)