PEC 2
Lenguajes de desarrollo front-end

Desarrollo front-end con frameworks Javascript

Máster Universitario en Desarrollo de sitios y aplicaciones web

Semestre 20211

Estudios de Informática, Multimedia y Telecomunicación

Introducción

Los lenguajes de tecnologías web han evolucionado y crecido exponencialmente en los últimos años. En el lado del servidor se pueden encontrar multitud de lenguajes de programación según diferentes paradigmas tales como PHP, Python, JAVA, Scala, .NET, NodeJS, Go…

Esto es debido a que la programación de lado de servidor es más cercana a la programación que se ha desarrollado a lo largo de la historia para sistemas informáticos. Por su parte, los lenguajes de lado de cliente han sido muy pocos y hoy en día existe un único y claro vencedor JavaScript. No obstante, a lo largo de la historia han existido diferentes intentos de crear un lenguaje de programación de lado de cliente que hiciera frente a JavaScript tales como VBScript, CoffeScript o Dart. Sin embargo, ha sido un superconjunto sobre JavaScript el que se está imponiendo, hoy en día, en el desarrollo de lado del cliente: TypeScript. TypeScript es un superconjunto sobre JavaScript, lo que permite desarrollar en JavaScript y, por ello, ser totalmente compatible. Una de las características que diferencia TypeScript de JavaScript es que TypeScript solamente existe en tiempo de programación, ya que los navegadores web no saben interpretar dicho lenguaje. Por este motivo, el código escrito en TypeScript debe ser transpilado/transformado a JavaScript para poder ser ejecutado/interpretado en los navegadores web.

El hecho de utilizar TypeScript en vez de JavaScript durante la fase de desarrollo es debido a que, entre otras cosas, este lenguaje ayuda al programador a hacer un código más fiable. Por ejemplo, TypeScript permite asignar un tipo a las variables, p.ej. `boolean`, `number`, etc. Así si a una variable declarada como numérica le asignamos un texto, al hacer la transpilación a JavaScript, el transpilador nos dará un error.

En esta PEC se van a presentar las novedades y cambios que ha sufrido el lenguaje JavaScript en su constante maduración a través de los estándares ES6-ES11 (2015-2020). Finalmente, se hará una breve introducción a TypeScript, puesto que éste es el lenguaje de la web hoy en día al ser adoptado por los principales frameworks para el desarrollo de aplicaciones Web.

TEORÍA

1. Evolución de ECMAScript

ECMAScript (abreviado como ES) es una especificación/estándar de lenguaje de programación publicada por ECMA Interantional. Este desarrollo comenzó en 1996 basado en el lenguaje de programación JavaScript. ECMAScript define un lenguaje de tipos dinámicos inspirado en Java y C. Además, soporta características de la programación orientada a objetos mediante objetos basados en prototipos y pseudo-clases. La mayoría de los navegadores realizan una implementación de este estándar ECMAScript e incorporan una API propia para manejar el DOM (*Document Object Model*).

Desde el lanzamiento en junio de 1997 del estándar han existido las versiones 1, 2, 3 y 5 (la 4 nunca fue finalizada). No obstante, desde junio de 2015 (versión 6) se lanzó la mayor versión de este estándar dando comienzo a una versión por año. De este modo tenemos las siguientes versiones ES7 (2016), ES8 (2017), ES9 (2018), ES10 (2019), ES11 (2020) y ES12 (2021)

Por lo tanto, ECMAScript es un estándar de un lenguaje. Mientras que JavaScript es la implementación más popular de este estándar. Otras implementaciones del estándar son: SpiderMonkey, V8 y ActionScript.

En esta PEC se profundizará en los avances y novedades aportados a JavaScript debido a los avances hechos en ECMAScript en los recientes años, transformándolo de un lenguaje de script a un lenguaje de aplicaciones Web.

1.1. Novedades de ES2015 (o ES6)

ES2015 o ES6 ha supuesto el mayor cambio en JavaScript y el comienzo de la revolución de este. Esta versión es la que tiene mayor cantidad de nuevas características y, por tanto, a veces, es tomada que la versión de referencia de JavaScript hoy en día. El soporte de las características que presenta esta versión es prácticamente del 100% por parte de la mayoría de los navegadores.

Muchas características de ES2015/ES6 no son imprescindibles saberlas para un buen manejo del lenguaje, ya que se tratan de mejoras realizadas en aspectos avanzados del lenguaje donde se carecía de estas características.

La lista de características de ES6 se puede encontrar con un ejemplo de código en la siguiente Web: http://es6-features.org/

A continuación, vamos a listar los elementos más relevantes de ES6 que deben manejarse con fluidez:

- Declaración de variables (uso de `let` y `const`).
- Ámbito (de funciones y variables, de ahí, la importancia de `let` y `const`).
- Lambda funciones/`Fat arrow`
    - Azúcar sintáctico (expresiones con cuerpo y sin cuerpo).

    - Valor léxico de `this` (mejora para las clausuras).

      https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/Arrow_functions

- Manejo de parámetros
    - Parámetros por defecto.
    - Operadores `rest`/`spread`.

      https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Functions/rest_parameters
- Interpolación de cadenas de texto.
- Mejoras en las propiedades de los objetos
    - Shothand en la definición de propiedades/valor
    - Propiedades dinámicas
    - Métodos como propiedades
- Desestructuración
- Módulos (los trabajaremos con `TypeScript`).
- Clases (conceptos de Programación Orientada a Objetos).
- Promesas
    - Uso y creación de estas.

Tal y como puedes ver de la lista anterior (y no es completa), prácticamente se ha redefinido el lenguaje para darle dinamismo y frescura a la hora de crear código con el mismo.

Para profundizar en los puntos anteriores existen 2 libros gratuitos compartidos por la comunidad que son buenas referencias para leer:

- ExploringJS: https://exploringjs.com/es6/

- Understanding ECMAScript 6:

  https://leanpub.com/understandinges6/read/

Aparte de estos dos estupendos recursos podemos usar el siguiente recurso proporcionado por O'Reilly:

- https://learning.oreilly.com/library/view/es6-for-humans/9781484226230/

1.2. Novedades de ES2016 (ES7)

Tras la avalancha de cambios sufridos en ES2015 (ES6) era lógico que la siguiente versión del lenguaje solamente tuviera ligeras mejoras. En este caso, ES2016 sólo aporta dos características:

- **`Array.prototype.includes()`.** Este método comprueba que el *array* contiene el valor pasado como argumento. En caso de existir retorna true, en caso contrario false. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/includes

- **`Exponentiation operator`.** El operador exponencial permite calcular el exponencial de un número utilizando los caracteres **. Es exactamente igual que utilizar **`Math.pow()`**. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Arithmetic_Operators#Exponentiation

1.3. Novedades de ES2017 (ES8)

En ES2017 (ES8) se retornó a la mejora del lenguaje, siendo la mayor aportación el azúcar sintáctico `async/await`. Las funciones `async/await` permiten una mayor legibilidad a las funciones que gestionan promesas. Además, de ello se incorporaron métodos para obtener las claves, valores o el par claves/valor de los objetos entre otras mejoras.

Las características de ES2017 que son de mayor importancia son las siguientes:

- Async functions (`async/await`)
- `Object.entries()` && `Object.values()`

Para comprender las funciones asíncronas se recomienda leer el capítulo correspondiente del siguiente libro: https://exploringjs.com/es2016-es2017/

1.4. Novedades ES2018 (ES9)

ES2018 continuó mejorando el asincronismo de JavaScript, permitiendo realizar iteraciones asícronas con la estructura de control `for-await-of`. Además, el operador `rest` que funcionaba sobre la estructura de datos *array* se incorpora a los objetos, permitiendo desestructurar las propiedades de los objetos.

Los puntos más relevantes que deben conocerse de ES2018 son los siguientes:

- Asynchronous iteration
- `Rest/Spread` properties
- `Promise.prototype.finally()`

Para comprender estas tres características recomienda leer el capítulo correspondiente del siguiente libro: https://exploringjs.com/es2018-es2019/toc.html

1.5. Novedades ES2019 (ES10)

ES2019 (ES10) incorpora una serie de mejoras orientadas a depurar errores que existen de compatibilidad de JavaScript en formateo de datos y ligeras mejoras en nuevos métodos.

Los puntos más relevantes que deben conocerse de ES2019 son los siguientes:

- `Array.prototype.{flat, flatMap}`
- `Object.fromEntries()`
- `String.prototype.{trimStart, trimEnd}`
- Optional catch binding.

Para comprender estas características se recomienda leer el capítulo correspondiente del siguiente libro: https://exploringjs.com/es2018-es2019/toc.html

También recomendamos leer el post escrito por el consultor Carlos Caballero donde se presentan las características de ES10 en pequeños ejemplos: https://carloscaballero.io/12-es10-features-in-12-simple-examples/

1.6. Novedades ES2020 (ES11)

En junio 2020 se publicó la versión ES2020 (ES11). Como principal novedad de este año, se incorpora el nuevo tipo de dato BigInt y también se incorporan algunas nuevas funcionalidades, las más destacadas serian:

- `BigInt`
- `Import()`

- `Promise.allSettled()`
- `globalThis`
- Nullish Coalescing Operator ??
- Optional Chaining Operator ?.

Al igual que en el caso de ES10, recomendamos leer el post escrito por el consultor Carlos Caballero donde se presentan las características de ES2020 (ES11) en pequeños ejemplos: https://carloscaballero.io/es2020-features-in-simple-examples/

1.7. Novedades ES2021 (ES12)

Como es norma a partir de ES2015, en este año 2021 se ha publicado la nueva versión ES2021 (ES12), de esta manera se sigue con la norma de año a año ir incorporando pequeñas nuevas funcionalidades que mantengan el lenguaje vivo, como principal novedad de este año, se incorporan los siguientes 3 operadores nuevos `??=`, `&&=`, `||=` y también se incorporan algunas nuevas funcionalidades, las más destacadas serian:

- `String.prototype.replaceAll`
- `Promise.any`
- `WeakRef`
- Logical Assignment Operators
- Numeric separators

Al igual que en el caso de ES11, recomendamos leer el post escrito por el consultor Carlos Caballero donde se presentan las características de ES2021 (ES12) en pequeños ejemplos: https://carloscaballero.io/es2021-features-in-simple-examples/

NOTA: Como miembros de la Comunidad UOC, tenéis disponible un gran catálogo de recursos de la editorial O'Reilly. Para acceder debéis seguir los siguientes pasos:

* Acceder a https://learning.oreilly.com

* Cuando te pida el *login* y *password*, escribir en el *campo Email Address or Username* vuestro correo electrónico de la UOC, p.ej. dgarciaso@uoc.edu

* Después pulsáis fuera con el ratón y desaparecerá el campo *password*. Además el botón de *Sign In* cambiará por un botón rojo con el texto *Sign In with Single Sign On*.

* A partir de aquí seguid los pasos que os vaya pidiendo, seguramente os llevará a una página de autenticación con el logo de la UOC. Además os llegará un e-mail a vuestra cuenta de correo electrónico de la UOC avisando de que os habéis suscrito a este catálogo.

* Una vez tengáis acceso al catálogo de O'Reilly, para ver el contenido de los enlaces que os proporcionamos, sólo tenéis que estar autenticados en O'Reilly y abrir una pestaña nueva en vuestro navegador y copiar el enlace que os facilitamos.