JavaScript Essentials 1 (JSE): Módulo 1

Sección 1

Cómo comunicarse con el ordenador

By usando un lenguaje que es comprensible para ambas partes. Usando tal lenguaje, escribimos un programa, que puede ser ejecutado por la computadora.

Desafortunadamente, un lenguaje que es directamente comprensible por una computadora será absolutamente ilegible para un ser humano normal. Sería una secuencia de instrucciones extrañas, escritas en forma numérica, que se refieren a componentes de computadora que ni siquiera sabíamos que existían.

Entonces, para ayudar en la comunicación con la computadora, se inventaron lenguajes de programación que son algo similares a los lenguajes naturales. Hay miles de lenguajes de programación, y difieren en propósito, nivel de abstracción, facilidad de uso, efectividad de los programas escritos en ellos, etc.

JavaScript como lenguaje interpretado

JavaScript es un lenguaje interpretado. Si ejecutamos un código escrito en JavaScript en un navegador web, el intérprete será el motor JavaScript integrado en el navegador. Esta no es la única forma de ejecutar código JavaScript.

Tal vez hayas oído hablar de node.js. También es un intérprete, pero se instala independientemente de los navegadores como un entorno (SANDBOX?) en el sistema operativo de la computadora. El uso de node.js le permite escribir programas en JavaScript que, por ejemplo, convertirán su computadora en un servidor.

Y de hecho, ejecutar un programa escrito en JavaScript parece como si estuviéramos ejecutando nuestro código fuente paso a paso.

La mayoría de los motores JavaScript modernos utilizan la técnica Just In Time Compilation (Compilación JIT). Esta técnica consiste en compilar fragmentos de código durante la ejecución del programa y permite aumentar su rendimiento.

En primer lugar, una etapa importante de este proceso es definir correctamente el problema que queremos resolver. Solo entonces tratamos de encontrar la solución óptima, que finalmente presentaremos en forma de un programa escrito en el idioma elegido.

Algunas palabras más sobre JavaScript

A principios de los años 90, todas las páginas web eran estáticas. Las cosas cambiaron en 1995 cuando la corporación Netscape contrató a Brendan Eich, y le encargó desarrollar un nuevo lenguaje para su producto, el navegador web Netscape Navigator. El nuevo lenguaje se llamó LiveScript, pero poco después su nombre fue cambiado a JavaScript. Su tarea principal era agregar dinámicas a los sitios web, lo que permitiría, por ejemplo, una interacción más compleja con el usuario. Y así comenzó la carrera de JavaScript.

Programación del lado del cliente frente a la del lado del servidor

El uso de JavaScript en sitios web, se llama programación del lado del cliente. El código a ejecutar se carga junto con la página en el navegador, en el lado del usuario, y el intérprete que forma parte del navegador web permite su ejecución.

Hoy en día, JavaScript es el único lenguaje compatible con todos los principales navegadores web, y alrededor del 95% de las páginas web en todo el mundo incrustan código JavaScript dentro de ellos. Desde el principio, las páginas web utilizaron JavaScript en el lado del cliente para agregar interactividad y cambiar dinámicamente el contenido.

JavaScript ofrece muchos marcos excelentes sobre los cuales construir aplicaciones web y redes sociales enormes y complejas.

Con el tiempo, JavaScript comenzó a aparecer en otras áreas, como la programación de las partes del lado del servidor de aplicaciones web complejas, también llamada back-end. Estos programas se ejecutan en servidores, procesando datos (por ejemplo, de bases de datos), que después del procesamiento estarán disponibles en el lado del cliente.

¿Es este el lenguaje de programación perfecto? - desventajas

Decimos que JavaScript es un lenguaje maduro, lo que significa que la mayoría de las características ya están implementadas y estables, y probablemente no veremos grandes cambios en el lenguaje. Desde 2015, muchos aspectos de JavaScript han cambiado y se han agregado muchas características nuevas. Muchos de estos cambios se introdujeron para facilitar la migración a JavaScript a los programadores que conocen otros lenguajes populares.

Pero... no hay soluciones ideales, por lo que no hay buenos lenguajes de programación para todas las aplicaciones. Cada uno de ellos tiene sus propias limitaciones, y no es diferente con JavaScript. Debido a su naturaleza, no es adecuado para ciertas aplicaciones. Por ejemplo, no tiene sentido usarlo para escribir programas que requieren cálculos matemáticos avanzados o un rendimiento muy alto.

Algunas limitaciones se deben al propio concepto del lenguaje, pero la gran mayoría están relacionadas con la plataforma en la que lo utilizamos. Esto es especialmente visible cuando se escribe código para ser ejecutado en un navegador, que como dijimos anteriormente se llama del lado del cliente. En tal situación, JavaScript está limitado en funcionalidad por el hecho de que los navegadores, por razones de seguridad, ejecutan código de script en un entorno sandbox (un entorno separado del mundo exterior), que no permite el acceso a archivos y recursos locales (es decir, aquellos archivos que se encuentran en la computadora donde se inicia el navegador).

Otro inconveniente es que como el código no está compilado, entra en el navegador en la misma forma, o muy similar, a lo que escribimos nosotros mismos. ¿Por qué es esto una desventaja? Esto se debe a que todos pueden ver nuestra solución en una forma fácil de leer y usarla sin nuestro permiso para escribir su propio programa.

Alguna ayuda aquí puede ser la ofuscación del código, que consiste en transformar nuestro script listo en una forma un poco menos legible, pero el simple hecho es que si alguien quiere robar nuestro código JavaScript, hay muy poco que podamos hacer para detenerlos.

¿Es este el lenguaje de programación perfecto? - ventajas

Por otro lado, JavaScript tiene muchas ventajas sobre otros lenguajes de programación, y uno de los más grandes es una comunidad muy activa y solidaria. Es fácil encontrar soluciones a problemas comunes y encontrar ayuda en general. Esto también significa que las herramientas que funcionan con JavaScript se desarrollan activamente.

Otra gran ventaja es una gran cantidad de marcos y bibliotecas listos para usar que proporcionan la mayoría de las funcionalidades y características comúnmente requeridas.

Además, JavaScript no requiere que compre herramientas costosas para trabajar con él, y las herramientas realmente buenas ya están integradas dentro de su navegador web.

Por último, pero no menos importante, los grandes jugadores como Google, Facebook y Mozilla apoyan activamente las herramientas de JavaScript y su desarrollo.

Sin embargo, lo que es una ventaja para algunos puede resultar ser una desventaja para otros. Un ejemplo puede ser la característica de escritura dinámica de JavaScript. En definitiva, consiste en el hecho de que podemos almacenar datos de cualquier tipo en una variable.

Por ejemplo, durante la ejecución del programa, podemos almacenar el número 10 en una variable, y en el siguiente paso usar la misma variable para almacenar la cadena " abc", eliminando el valor anterior automáticamente.

Al agregar la escritura estática, donde una variable solo puede contener un tipo de variable (por ejemplo, números) durante la ejecución del programa, se introdujo un nuevo lenguaje llamado TypeScript.

Sección 2

Herramientas de desarrollo

Como cualquier otra tarea, la programación requiere las herramientas y el espacio de trabajo adecuados. El desarrollo de software, requiere un editor de código y un compilador o intérprete de un lenguaje determinado.

También podemos utilizar el depurador, que es una herramienta que nos permite, pausar el programa en el lugar indicado y analizar su estado actual

No se puede negar que el tamaño del monitor afectará la comodidad de su trabajo. Cuanto más grande sea el monitor, más fácil será colocar el editor de código, el intérprete y otros contenidos (por ejemplo, este curso) uno al lado del otro. En circunstancias normales de trabajo, los programadores muy a menudo usan varios monitores.

El sistema operativo no importa, ya que se puede encontrar la herramienta adecuada para Windows, macOS y Linux.

En este momento, hay dos opciones. Puede instalar todas las herramientas necesarias en su máquina y trabajar en el entorno local.

Otro enfoque es utilizar herramientas en línea. Estos pueden ser convenientes, ya que no necesita instalar o configurar

Entorno de desarrollo en línea

Los entornos en línea, comúnmente conocidos como áreas de juego de código, son sitios que actúan como un simple editor y entorno de tiempo de ejecución. Todos ellos tienen conjuntos similares de características. Le permiten escribir código, ejecutarlo con fines de prueba y, con mayor frecuencia, compartirlo con otros usuarios.

En el caso de JavaScript, donde preparar un entorno local completamente funcional en realidad se reduce a instalar un editor de código y ejecutar el navegador, no son tan importantes como los entornos de desarrollo normales. Se utilizan principalmente como plataformas de capacitación y prueba, o lugares para publicar soluciones de muestra a problemas de programación.

los más populares son los siguientes: JSFiddle, CodePen, JsBin, Plunker

Entorno de desarrollo local

Dependiendo del nivel de sofisticación, la complejidad del proyecto escrito o el entorno para el que escribimos nuestros programas (del lado del cliente, del lado del servidor, móvil), también se pueden necesitar otras herramientas.

Estos serán, entre otros:

* gestores de paquetes: permiten la gestión de bibliotecas (que contienen soluciones listas para usar que podemos utilizar en nuestros programas) o componentes del entorno de desarrollo (por ejemplo, npm o hilo)
* ejecutores de tareas y empaquetadores de módulos: utilizados, en términos simples, para automatizar el proceso de desarrollo de software y fusionar el código resultante de muchos archivos y bibliotecas (por ejemplo, Grunt o Webpack)
* Marco de pruebas: permite realizar pruebas automáticas de la corrección de nuestro programa en busca de posibles errores (por ejemplo, Mocha, Jasmine o Jest)
* analizadores de seguridad: como puede adivinar, se utilizan para controlar la seguridad de nuestra solución (por ejemplo, Snyk, RetireJS u OWASP Dependency Check)

La apertura de los entornos de desarrollo web es tanto una bendición como una maldición. Tenemos una selección de cientos de componentes, a partir de los cuales podemos crear el ambiente más cómodo para nosotros mismos.

Sin embargo, su cantidad, más los cambios dinámicos de herramientas particulares o incluso las tendencias entre los programadores hacen que sea difícil mantenerse al día con todo lo que está sucediendo dentro de estos entornos.

Pero para nosotros, este es un problema para el futuro lejano.

Por ahora, necesitamos el trío mínimo: un editor de código, intérprete y depurador.

Editor de código

El código de casi todos los lenguajes de programación se compone de alguna forma de texto. Entonces, para escribir el código, necesitamos un editor de texto. Pero debe ser una aplicación que escriba texto sin formato (no puede ser un editor de texto enriquecido, como MS Word). En otras palabras, solo un simple bloc de notas que pueda escribir archivos .txt es suficiente para escribir código, aunque es mucho más fácil si usa un editor de código dedicado. El mercado está lleno de editores de código profesionales, tanto gratuitos como de pago. Algunos de ellos son universales, mientras que otros son exclusivos de idiomas específicos. La principal ventaja de usar un editor de código dedicado es el resaltado de sintaxis, el autocompletado de texto y la comprobación de errores. Esto mejora la eficiencia del trabajo y la comprensión del código, y reduce el número de errores y errores tipográficos. Hay muchos buenos editores de código, pero puede ser muy difícil seleccionar uno que funcione para usted.

Aquí hay algunos populares:



* Visual Studio Code [Windows, macOS, Linux]

Potente editor de código gratuito para uso personal y comercial. Se ha convertido rápidamente en uno de los favoritos cuando se trata de desarrollo web. Tiene características incorporadas como un depurador de JavaScript y herramientas para agilizar los proyectos web. También es altamente personalizable a través del sistema de extensión (hay muchas adiciones dedicadas especialmente al lenguaje JavaScript).

* WebStorm [Windows, macOS, Linux]

Un entorno de desarrollo comercial popular, en el que el editor de código es solo uno de los elementos más pequeños en un gran conjunto de herramientas que mejoran el desarrollo de código (por ejemplo, pruebas de soporte). Destinado a proyectos grandes, puede resultar demasiado pesado y complejo para programas pequeños. Aunque está destinado a uso comercial, es posible obtener una licencia educativa gratuita.

* Texto sublime [Windows, macOS, Linux]

Editor de código rápido y fácil de usar con muchas funciones avanzadas, como edición de varias líneas, búsqueda rápida y otras. Hay una versión de prueba disponible, pero para el uso a largo plazo, se debe comprar una licencia tanto para uso privado como comercial.

* Bloc de notas++ [Windows]

Editor de texto y código gratuito y ligero. El programa es pequeño y rápido, admite docenas de lenguajes de programación y se puede ampliar con complementos. Puede ser viejo y feo, pero sigue siendo agudo.

Existen muchos otros editores de código, tanto gratuitos como de pago, y puede usar el que prefiera. Muchos desarrolladores utilizan, entre otras cosas, editores de consolas, incluido el legendario vim. Los editores de consola no se ejecutan en un entorno gráfico, sino en una consola de texto. Sin embargo, solo puede buscar tales soluciones si las tareas que va a hacer resultan ser demasiado simples y desea hacer su vida un poco más difícil.

Intérprete

Ya hemos hablado un poco sobre el intérprete y su papel. Funciona como un entorno de tiempo de ejecución para nuestro programa. Comprueba si hemos cometido algún error formal, por ejemplo, haciendo un error tipográfico en el nombre de una función u olvidando cerrar un paréntesis, y luego ejecuta la instrucción del programa por instrucción.

La elección del intérprete de JavaScript dependerá de la plataforma para la que escribamos nuestro software. Por ejemplo, si queremos escribir una aplicación sencilla del lado del servidor, es casi seguro que elegiremos el entorno nodo.js, que tendremos que instalar directamente en nuestro sistema operativo. En el caso del software del lado del cliente, nuestro intérprete será simplemente el navegador web que ya tiene instalado (porque ¿de qué otra manera leería este curso?).

Nuestro curso trata sobre JavaScript central, es decir, aquellos elementos del lenguaje que serán igualmente útiles en soluciones del lado del cliente, del lado del servidor y móviles. Así que podemos practicarlos en cualquier entorno, utilizando cualquier intérprete. La forma más fácil de hacerlo es limitarse a un navegador web.

Como dijimos antes, prácticamente todos los navegadores tienen motores JavaScript incorporados (o intérpretes), pero recomendamos encarecidamente usar Chrome de Google o FireFox de Mozilla. Ambos son conocidos por su eficiencia y herramientas avanzadas integradas para desarrolladores web (ese eres tú). Están disponibles para Windows, macOS y Linux.

Recuerde actualizar regularmente su navegador elegido y usar la última versión. Esto es especialmente importante cuando se trabaja con JavaScript. El lenguaje cambia constantemente, con nuevas características y mecanismos que se agregan. Es posible que su navegador favorito pero algo anticuado no admita ciertas características del idioma. Su navegador ahora es una herramienta, así que trate de mantenerlo en buena forma actualizándolo regularmente.

Depurador

Los programas informáticos son bestias complicadas, miles o incluso millones de líneas de código (pero cálmate, comenzaremos con solo unas pocas). Con tal complejidad y tamaño, es imposible producir código sin ningún error. Algunos tipos de errores, especialmente los lógicos (formalmente, el programa está escrito correctamente, pero probablemente inventamos la solución incorrecta al problema), solo se pueden encontrar mientras el programa se está ejecutando, y a menudo solo en circunstancias especiales. Es realmente difícil averiguar qué está sucediendo exactamente dentro de un programa que se ejecuta a toda velocidad, y para esos problemas, existen depuradores.

Un depurador es una herramienta que le permite ralentizar o incluso detener la ejecución de un programa, ejecutar instrucciones paso a paso y mirar y analizar el estado del programa en un momento dado.

Afortunadamente, en el momento en que decidimos usar el navegador web como nuestro entorno de arranque e intérprete de JavaScript, también nos conseguimos un depurador. Todos los navegadores modernos están equipados con las herramientas de desarrollo. Durante el funcionamiento normal, son invisibles, y tenemos que habilitarlos en las opciones del navegador (más sobre esto en el siguiente capítulo).

Dependiendo del navegador, encontraremos varias herramientas allí, pero sin duda habrá:

* el inspector, que nos permitirá, por ejemplo, analizar los elementos HTML individuales de un sitio web abierto;
* la consola de JavaScript, que en primer lugar muestra toda la información sobre los errores y, en segundo lugar, nos permite ejecutar comandos únicos de JavaScript en el contexto de la página actual;
* el depurador– que, entre otras cosas, muestra los valores actuales de las variables y le permite pausar la ejecución del código en el lugar indicado y realizar un trabajo paso a paso (es decir, ejecutar instrucciones únicas del programa).

¿Cómo se habilitan las herramientas de desarrollo? Desafortunadamente, no hay una respuesta única; depende del navegador que esté utilizando (a veces también de su versión) y del sistema operativo. Las interfaces del navegador cambian con bastante frecuencia, por lo que es mejor aprender los accesos directos correctos en lugar de buscar la opción correcta en el menú. Pruebe las siguientes combinaciones de teclas:

* Sistemas operativos Windows y Linux, todos los navegadores comunes excepto Internet Explorer y Edge: ctrl + shift + I
* Sistema operativo Windows, Internet Explorer y Edge: F12
* Sistema operativo macOS, todos los navegadores comunes: tecla comando + tecla opción + I

En el próximo capítulo, volveremos a este tema y aprenderemos algunas cosas más sobre estas útiles herramientas.

En el siguiente capítulo escribiremos nuestra primera pieza de código JavaScript. Lo probaremos en primer lugar en el entorno de tiempo de ejecución integrado con nuestra plataforma de entrenamiento. También lo usaremos para verificar cómo funciona nuestro entorno de desarrollo local. Por lo tanto, asegúrese de que las herramientas seleccionadas estén instaladas y que pueda iniciarlas. Si aún no sabe qué elegir, le sugerimos que utilice el entorno local con Visual Studio Code (nuestro editor de código) y Chrome (navegador web con intérprete y depurador de JavaScript).

Sección 3

El "¡Hola, Mundo!" Programa

¿Por qué "¡Hola, mundo!"? Durante casi 50 años, esta frase, y sus derivados, han marcado a alguien como aprendiendo un nuevo lenguaje de programación, aunque es más una tradición que otra cosa. La frase se usó hace mucho tiempo en un libro muy importante sobre el lenguaje C, pero el texto en sí no importa.

La idea es escribir algo en la pantalla usando un lenguaje específico. En primer lugar, nos permite ver la sintaxis básica del lenguaje y compararlo con otros lenguajes de programación. En segundo lugar, es un programa muy simple, y cualquiera puede escribirlo o copiarlo fácilmente de Internet y verificar si sus herramientas y entorno están configurados correctamente. En tercer lugar, es un programa que produce algo, por lo que proporciona retroalimentación sobre si se ejecutó correctamente o no.

En el caso de JavaScript del lado del cliente, mostrar algo en la pantalla se puede entender de dos maneras.

En primer lugar, JavaScript del lado del cliente siempre se ejecuta en el contexto de un sitio web y le permite manipular elementos de ese sitio web. Así podemos, por ejemplo, utilizar la función adecuada para insertar algún texto, cambiar un título, crear una tabla, etc. en la página. De esta manera, controlamos la parte visual del sitio web.

En segundo lugar, podemos usar la consola como una pantalla para escribir alguna información. La consola, como comentábamos en el capítulo anterior, forma parte de las herramientas de desarrollo. Por lo tanto, no es visible de forma predeterminada, y debe estar habilitado correctamente (también escribimos sobre esto en el capítulo anterior). Para nuestras necesidades, será mucho más conveniente utilizar la consola, ya que evitaremos la necesidad de un análisis exhaustivo de la estructura de la web.

Pero, ¿qué es realmente una consola? En primer lugar, es un lugar donde se muestran varios mensajes, normalmente invisibles para el usuario del navegador. Estos mensajes pueden, por ejemplo, ser generados por el intérprete de JavaScript tras encontrarnos con un error o si lo imprimimos, llamando a la función adecuada. En segundo lugar, podemos ejecutar comandos javascript individuales en la consola, que se ejecutarán en el contexto de la página web cargada actualmente (un poco más sobre eso en un momento).

La función básica que nos permite escribir información en la consola es consola.log. Entonces, para referirnos al eterno "¡Hola, Mundo!", deberíamos llamarlo de la siguiente manera:

consola.log ("¡Hola, mundo!");

Podemos tratar la consola.log como una función\*. De hecho, la función es solo un registro, y la consola es el objeto al que pertenece la función.

\*Este tipo de función, que pertenece a un objeto, generalmente se denomina método. Pero una vez más, por el momento, para simplificar ciertas cosas, supongamos que esta es una función ordinaria: no nos molestará en absoluto (aprenderemos sobre objetos mucho más tarde).

La función es un fragmento de código que permite realizar una tarea concreta (en nuestro caso, mostrar algo en la consola). Las funciones a menudo toman argumentos, en otras palabras, datos que usarán durante la operación. En JavaScript, ejecutamos una función llamándola, y la llamamos escribiendo su nombre seguido de un par de paréntesis, donde se proporcionan los argumentos (si la función no necesita argumentos, los paréntesis se dejan vacíos). En nuestro ejemplo, el argumento es el texto que queremos mostrar. Tenga en cuenta que para indicar que "¡Hola, mundo!" es el texto, lo ponemos entre comillas.

Para que el intérprete sepa dónde termina el comando, ponemos un punto y coma al final de la llamada a la función. En este caso, el intérprete se las arreglaría sin tal ayuda, pero es un buen hábito terminar cada comando con un punto y coma, para que no lo olvide cuando realmente se necesita.

Ya sabemos qué escribir, y la única pregunta es, ¿dónde hacerlo?

Entorno de desarrollo en línea

Afortunadamente, nuestra plataforma utiliza un entorno en línea listo para usar, como mencionamos en el capítulo anterior. El entorno OpenEDG le permite editar y ejecutar programas escritos en JavaScript. Tenga en cuenta que la parte de la pantalla dedicada a este entorno se divide en tres partes. La parte superior es el editor, donde podemos elegir si editar un archivo JavaScript, HTML o CSS (diremos algunas palabras sobre HTML y CSS en un momento). Todos estos archivos juntos forman el código a ejecutar en nuestro entorno de entrenamiento. Nos interesará principalmente la pestaña de archivos JavaScript – app.js. En la parte inferior de la pantalla de la izquierda, hay una ventana que simula la consola, en la que aparecerán los mensajes del intérprete así como la información que escribamos. La ventana en el lado derecho está diseñada para mostrar la página en el contexto de la cual se ejecuta nuestro código JavaScript. Esta ventana será la menos útil en esta parte del curso.

En el editor, debería ver la pieza de código que se acaba de discutir, que contiene la función consola.log. Intenta ejecutarlo. Debe presionar el botón resaltado con el icono de reproducción, ubicado directamente encima del editor. Como resultado, la ventana inferior que simula la consola debería mostrar:

¡Hola mundo!

Ve al editor de nuevo y cambia la palabra "Mundo" por tu nombre. Inicie el programa de nuevo y compruebe lo que aparece en la ventana de la consola. Enhorabuena, acabas de modificar un programa escrito en JavaScript.

Podríamos discutir su primer programa JavaScript basado en este ejemplo. Has aprendido su sintaxis, la has ejecutado en línea, has comprobado su efecto e incluso la has modificado tú mismo. Puedes probar todos los ejemplos que discutimos en este curso de esta manera. Sin embargo, en el capítulo anterior, le instamos a configurar su entorno de desarrollo local. Por lo tanto, sería bueno mostrarle cómo se puede ejecutar este ejemplo en un entorno de este tipo. Y esto requerirá una introducción un poco más larga.

Entorno de desarrollo local

JavaScript del lado del cliente es un lenguaje de la web y existe solo en el ecosistema web. En esta configuración, JavaScript no puede existir por sí mismo. El código JavaScript debe estar incrustado en un documento HTML. Cuando utilizamos el entorno en línea para ejecutar nuestro programa, ciertos aspectos se nos ocultaron. Esta vez tendremos que mirarlos más de cerca.

Algunas palabras sobre HTML

HyperText Markup Language, o HTML para abreviar, es un conjunto de etiquetas utilizadas para describir la estructura de un sitio web. Nos permite dar a una página el formato de un documento que contiene secciones, encabezados, párrafos, listas y similares. HTML está definitivamente más allá del alcance del curso actual, por lo que presentaremos solo información básica al respecto, lo suficiente para que comprenda dónde y cómo podemos ejecutar el código JavaScript asociado con una página determinada.

Los tipos de etiquetas están predefinidos. Por ejemplo, la etiqueta que especifica un párrafo es <p> y la etiqueta para el encabezado de primer grado (más grande) es <h1>. El nombre de la etiqueta debe colocarse entre corchetes angulares. Las etiquetas se suelen utilizar en pares, limitando una determinada zona del documento (tenemos una etiqueta de apertura y otra de cierre). La etiqueta de cierre es diferente de la etiqueta de apertura, porque aparece una barra diagonal antes del nombre. Por ejemplo, un párrafo puede tener este aspecto:

<p>quite un párrafo ordinario</p>

A menudo, las etiquetas pueden (y a veces deben) colocarse dentro del rango de otras etiquetas. Por ejemplo, nuestro párrafo debe colocarse dentro de las etiquetas <body>, que separan la parte principal de nuestro documento.

<>

<p>quite un párrafo ordinario</p>

</cuerpo>

Documento HTML mínimo

Intentemos crear un HTML mínimo que defina una página vacía.

<! DOCTYPE html>

<html>

<cabeza>

<título>Página exclusiva</título>

</cabeza>

<>

</cuerpo>

</html>

¡Comencemos con la declaración <! DOCTYPE html>. Esta no es una etiqueta típica, ya que se utiliza para informar al navegador de que todo el documento se ha preparado de acuerdo con HTML5. La descripción real del documento comienza con la etiqueta <html>, que junto con el </html>tag establece los límites del documento. Cualquier otra etiqueta debe estar dentro de estos. Si alguna etiqueta dada contiene otro contenido, habrá una etiqueta de cierre correspondiente, formando algún tipo de contenedor.

La siguiente etiqueta, <head>, contiene información adicional sobre el documento, que también debe colocarse en etiquetas. La más básica es la etiqueta <title>, que establece el título de la página mayormente visible en la barra de título del navegador. Después de <head> está el elemento <body>, y el contenido visible de la página web debe colocarse allí (por ejemplo, nuestro párrafo).

etiqueta <script>

El código JavaScript que ejecutará el navegador en la página debe adjuntarse al HTML utilizando la etiqueta <script>, y hay dos formas de hacerlo. El código se puede incrustar directamente dentro de las etiquetas <script> y </script>, pero esto solo se recomienda cuando el código es corto. Otro enfoque es usar el atributo "src" para apuntar a un archivo separado que contiene el código JavaScript. Esto es especialmente cierto cuando se va a usar el mismo código en varias páginas, porque repetir exactamente el mismo código muchas veces es una mala práctica, ya que cualquier cambio debe aplicarse a todos los archivos; y además, aumenta artificialmente el sze de la página. La extensión de archivo JavaScript es .js.

HTML es leído por el navegador línea por línea, y las etiquetas de script se ejecutan justo en el momento en que el navegador analiza la etiqueta <script> (el análisis de lenguajes de programación significa un análisis formal del código por parte de una máquina para comprender su estructura). Por lo general, las etiquetas <script> se insertan en el encabezado de la página entre las etiquetas <head> y </head>, y podemos insertar muchas de ellas en un archivo, por ejemplo, para incluir código JavaScript de diferentes archivos. Este comportamiento se puede cambiar para scripts externos apuntados por el atributo "src" mediante los atributos "defer" o "async".

diferir: significa que el script debe ejecutarse después de cargar toda la página;

asincrónico: significa que el script se ejecutará inmediatamente, pero en paralelo al análisis del resto de la página.

... y un poco de CCS

CSS, o cascading Style Sheets, es un lenguaje utilizado junto con HTML para describir la apariencia de una página y sus elementos. En pocas palabras, HTML describe la estructura de un documento, mientras que CSS describe su presentación.

Por ejemplo, en HTML, podemos describir una página que tiene un encabezado, dos párrafos y una tabla de datos.

En CSS, podemos definir qué fuente se usará en toda la página, qué color tendrá el fondo o si el cursor del mouse, cuando se mueve sobre la tabla, debe cambiar de forma.

Así que podemos tratar CSS como algún tipo de configuración de la capa visual de la página. Por lo tanto, el sitio web se construirá con mayor frecuencia sobre la base de un archivo HTML (es decir, una descripción de la estructura), código JavaScript (lo que nos permite agregar, por ejemplo, algunos mecanismos de interacción) y un archivo CSS (que describe la capa de presentación de la página). Sin embargo, lo importante es que no habrá página sin un archivo HTML, pero podemos crear fácilmente una página sin usar archivos CSS. La descripción de CSS en sí está fuera del alcance del curso actual, y la mencionamos solo por el bien del orden.

¿Cómo podemos ejecutar nuestro código JavaScript?

Comencemos con un ejemplo simple, donde el navegador obtiene una página simple (tal vez incluso vacía) de https://test.org. La dirección está hecha para este ejemplo, así que realmente no intente ingresarla. Mira la figura a continuación.

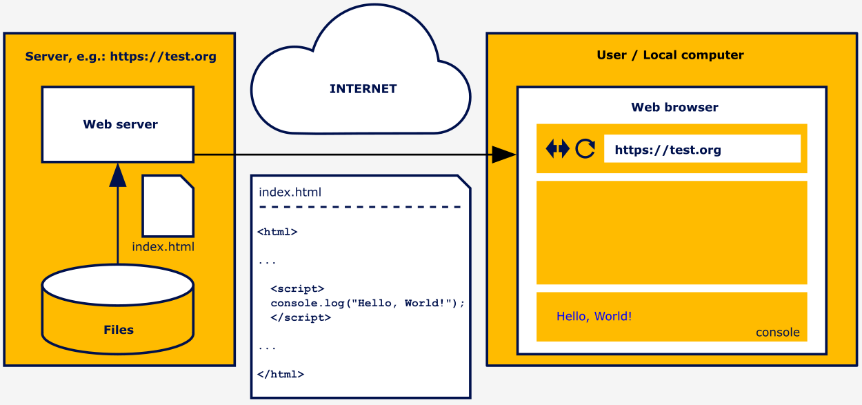
Comencemos por el lado derecho de la figura. El usuario ejecuta un navegador web en su computadora (por ejemplo, Chrome). Usando el atajo de teclas de acceso rápido apropiado, activan las herramientas de desarrollo (consulte el capítulo anterior) para poder usar la consola. Recuerde que estas herramientas no son necesarias para el uso normal del navegador y, por lo tanto, están ocultas de forma predeterminada. Luego, el usuario escribe https://test.org (la URL de nuestro sitio falso) en la barra de direcciones.

En el servidor remoto (lado izquierdo del dibujo), asociado a la dirección https://test.org, se lanza un servidor web, que, tras recibir una solicitud de nuestro usuario, preparará una respuesta para ellos. En el caso más simple, la respuesta solo contendrá un archivo html, que se puede almacenar en el mismo servidor. El archivo html (en este ejemplo, index.html) se envía de vuelta al usuario y el navegador lo procesa. Si algún contenido (por ejemplo, un párrafo con texto) está definido en él, se mostrará en la ventana del navegador.

Sin embargo, estamos más interesados en el hecho de que el archivo index.html contiene las etiquetas <script> y </script>, con un fragmento de código JavaScript entre ellas. ¿Lo reconoces? Esto es obviamente un intento de mostrar nuestro "¡Hola, Mundo!" en la consola. Al cargar la página, se debe ejecutar el código colocado dentro de las etiquetas <script> y, si las herramientas de desarrollo están habilitadas y el panel de la consola está visible, la consola mostrará "¡Hola, Mundo!".

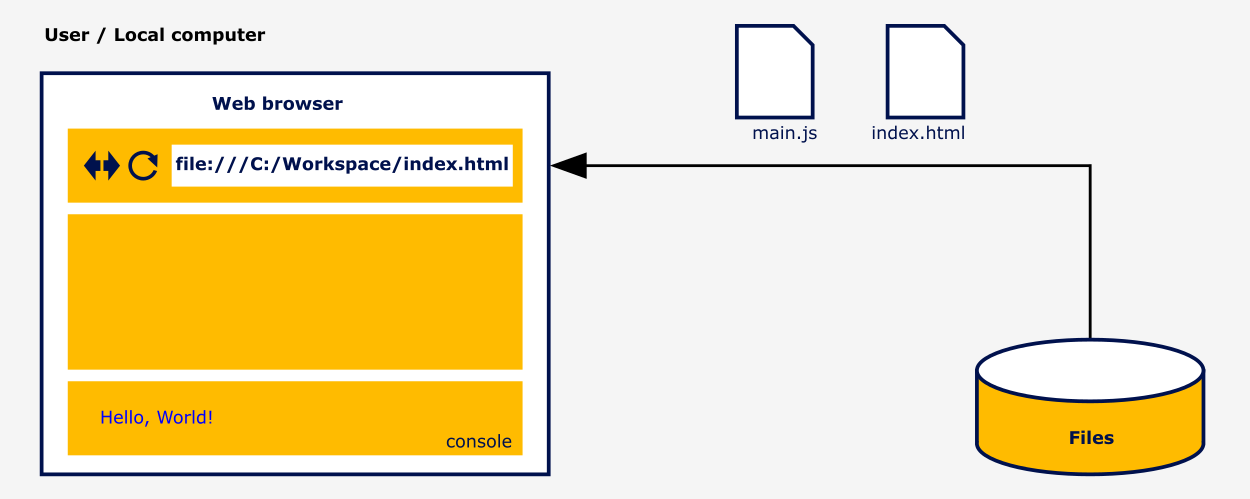
Como dijimos antes, la etiqueta <script> se puede usar de una manera diferente, no solo para restringir el lugar donde escribimos código JavaScript directamente. Si usamos el atributo "src" en esta etiqueta, podemos indicar un archivo JavaScript separado que se adjuntará aquí.

La siguiente imagen muestra un ejemplo de tal escenario. En el archivo index.html vuelve a aparecer la etiqueta <script>. Esta vez no hay ningún código JavaScript colocado después de él, pero al usar el atributo "src", se indica que el código del archivo principal.js debe adjuntarse aquí.



Todo funcionará exactamente igual que en el escenario anterior, excepto que el servidor web proporcionará el archivo principal.js además del índice.html. El usuario no notará ninguna diferencia. Por supuesto, colocar nuestro código en un servidor remoto solo para probarlo sería un poco engorroso.

Tenemos otra posibilidad, que es que podemos cargar un archivo html local (es decir, uno que está en nuestro ordenador) en el navegador. Si este código contiene una etiqueta <script> que indica algún archivo JavaScript, este archivo también se cargará desde los recursos locales.

La imagen muestra un escenario simple en el que el usuario carga un archivo de índice.html local en el navegador, en el que hay una referencia a main.js (por lo que este archivo también se cargará automáticamente).

Puede cargar un archivo html local escribiendo su ruta local después de file:/// en la barra de direcciones, o simplemente abriéndolo en su navegador usando el comando Abrir del menú. Dado que el menú en los navegadores a menudo está oculto, una forma más simple puede ser usar un acceso directo para abrir documentos existentes en las aplicaciones. El acceso directo es universal, no solo para navegadores y probablemente ya lo hayas visto:

Ctrl + O

o en el caso de macOS: comando +O

De acuerdo, tal vez finalmente podríamos ejecutar algo ...

Para ejecutar esto localmente, deberá abrir el editor de código de su elección. Cree un nuevo archivo con la extensión .html (el nombre del archivo no importa, pero es una buena práctica evitar espacios en el nombre del archivo). Coloque el código siguiente en este archivo y guárdelo.

<! DOCTYPE html>

<html>

<cabeza>

<título>Página exclusiva</título>

<script src="main.js"></script>

</cabeza>

<>

</cuerpo>

</html>

Luego, en el mismo editor, crea otro archivo, esta vez llamado main.js (este es el nombre que usamos en nuestro archivo html). Debe contener una línea que hayas visto antes:

consola.log ("¡Hola, mundo!");

Guarde los cambios y vaya al navegador. Abra una nueva pestaña, habilite las herramientas de desarrollo (se abren para una pestaña en particular) y seleccione la herramienta de consola. Tómese un momento para acostumbrarse al diseño de las herramientas de desarrollo (cada herramienta, incluida la consola, debe colocarse en un panel separado, que se puede seleccionar).

Por si acaso, asegúrese de que el enfoque esté configurado en la ventana del navegador (es decir, la nueva pestaña) haciendo clic en ella. Luego, usando el atajo de teclado apropiado, abra el archivo html que acaba de crear. Si todo se ha hecho correctamente, debería ver nuestro signo "¡Hola, mundo!" en la consola. Asegúrese de que funciona y de que puede ver un mensaje que se muestra en la consola de depuración. Si es necesario, hágalo de nuevo, siguiendo cuidadosamente los pasos.

Y ahora un pequeño reto. Intente modificar el archivo html usted mismo para que no se refiera al archivo principal.js. En su lugar, el mismo código JavaScript que escribimos en main.js debe colocarse directamente después de la etiqueta <script>. Si tiene problemas, vuelva al primer dibujo de esta sección.

Para ejecutar este código en el entorno en línea, colóquelo dentro de las pestañas HTML y, si es necesario, presione el botón ejecutar.

Ejecución del código directamente en la consola

Tenemos otra opción bastante conveniente cuando se trata de ejecutar piezas cortas de código JavaScript en el navegador (y nuestro programa, que consiste en una instrucción, es definitivamente corto). Como dijimos antes, la consola no solo se usa para mostrar información, sino que también le permite ejecutar instrucciones individuales de JavaScript. Estas instrucciones deben ejecutarse en el contexto de alguna página HTML. Sin embargo, no necesariamente tienes que escribir tu página mínima, como lo hicimos hace un momento. Intente abrir una nueva pestaña y escriba about:blank en la barra de direcciones. Esta es una pseudo dirección que le dice a su navegador que genere y cargue una página HTML en blanco mínima.

A continuación, ejecute las herramientas de desarrollo. Al principio, podemos comprobar cómo es el HTML generado por el navegador. Para ello, selecciona la primera herramienta del panel (en Chrome, será Elements, en Firefox Inspector). Debería ver un código html absolutamente mínimo:

<html>

<cabeza></cabeza>

<></cuerpo>

</html>

Ahora elija la consola de las herramientas de desarrollo. Debería ver un mensaje, generalmente un signo > o >> seguido de un cursor parpadeante (si no hay cursor, haga clic en el mensaje). Luego puede ingresar la instrucción que mostrará "¡Hola, Mundo!" en la consola (usando la función .log consola).

De hecho, independientemente del navegador, deberíamos obtener el mismo efecto: la consola mostrará el texto que especificamos. En el caso de Chrome (que se ejecuta en el sistema operativo Windows), la consola debería verse algo así después de completar esta tarea:

Para ambos navegadores, las ventanas del depurador que contienen la consola pueden variar mínimamente dependiendo de la versión del software y del sistema operativo que lo ejecute. Las herramientas de desarrollo se pueden mover. Se pueden ubicar en la parte inferior del navegador, como en los ejemplos que se muestran, pero también se pueden colocar en el lado izquierdo o derecho de la ventana (o como una ventana separada por completo). Así que no se sorprenda si el diseño de su navegador es ligeramente diferente de las imágenes.

Summary

Nuestro primer programa se lanzó en un entorno en línea al principio. Este entorno nos permite ocultar ciertos detalles que no son importantes para nosotros en esta etapa del curso. Todos los ejercicios y ejemplos que discutiremos deben hacerse en este entorno.

Sin embargo, de vez en cuando, sería bueno que intentes hacer el ejemplo elegido también en el entorno local. Está mucho más cerca de lo que realmente se utiliza en el trabajo de un desarrollador web. Ejecutar código JavaScript en el entorno local puede parecer un poco engorroso al principio, pero afortunadamente esto es solo una primera impresión. Recuerde, para probar instrucciones simples, solo necesita usar la consola con una página vacía (por ejemplo, about:blank). Si desea probar una pieza de código un poco más grande, es mejor crear un archivo html mínimo que se refiera al archivo que contiene nuestro código JavaScript utilizando la etiqueta <script>.