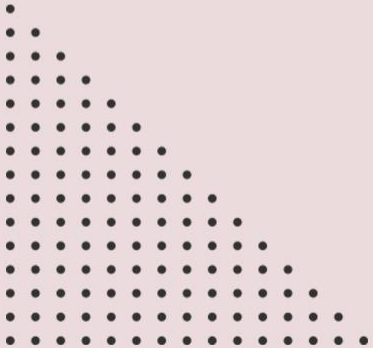
4.1

Javascript

Programación dinámica

Introdución



@mihifidem 01/2024

Contenidos

- ¿Qué es JavaScript?
- Conceptos básicos: contenedor y sus elementos
- Propiedades del contenedor
- Propiedades de los elementos

Javascript_ ¿Qué es?

JavaScript fue creado para
“dar vida a las páginas web”.

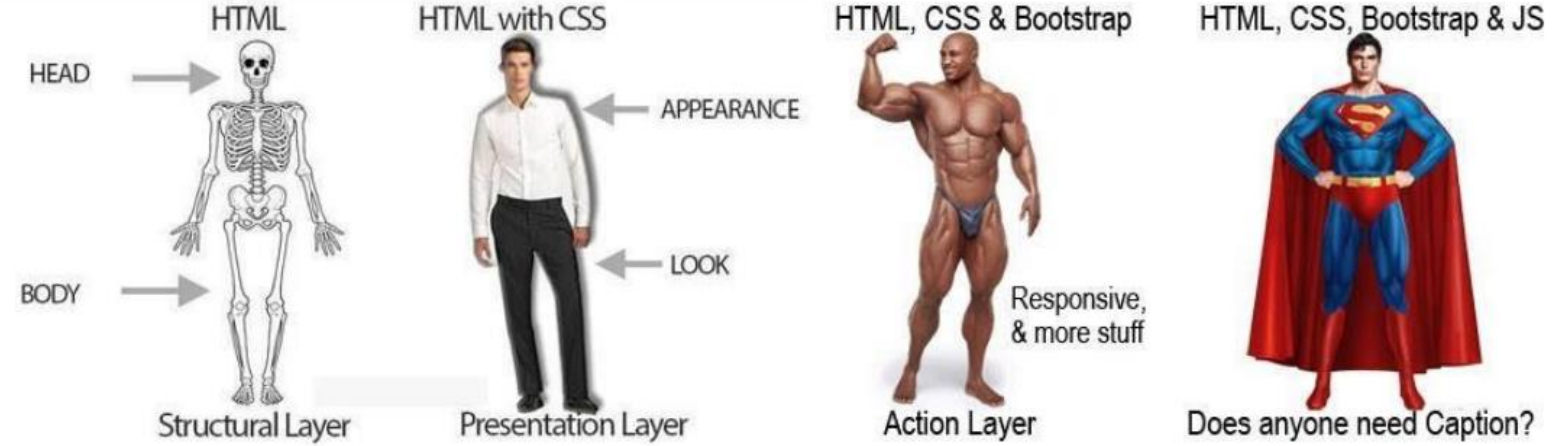
HTML

HTML + CSS

HTML + CSS + JavaScript

Javascript_ ¿Qué es?

Apareció en: 4 de diciembre de 1995


HTML
HEAD
BODY
Structural Layer
HTML with CSS
APPEARANCE
LOOK
Presentation Layer
HTML, CSS & Bootstrap
Responsive,
& more stuff
Action Layer
HTML, CSS, Bootstrap & JS
Does anyone need Caption?

Javascript_ ¿Qué es?

Los programas en este lenguaje se llaman scripts.

Javascript_ ¿Qué es?

JavaScript es un lenguaje de programación **interpretado**, dialecto del estándar **ECMAScript**.

Inicialmente tenía otro nombre: “**LiveScript**”.

Javascript_ ¿Qué es?

Se define como **orientado a objetos**, basado en **prototipos**, **imperativo**, **débilmente tipado** y **dinámico**.

Javascript_ ¿Qué es?

Los scripts se proporcionan y ejecutan como texto plano. No necesitan preparación especial o compilación para correr.

Javascript_ ¿Qué es?

JavaScript es muy diferente a otro lenguaje llamado Java.

Javascript_ ¿Qué no puede hacer?

Restricciones en el Acceso a Archivos y el Sistema Operativo:

- No puede leer/escribir arbitrariamente en el disco duro.
- No puede copiar ni ejecutar programas.
- No tiene acceso directo a funciones del Sistema Operativo.

Javascript_ ¿Qué no puede hacer?

Interacción Limitada con Archivos del Usuario:

- Posible trabajar con archivos bajo acciones específicas del usuario (ejemplo: arrastrar archivos, usar etiqueta <input>)

Javascript_ ¿Qué no puede hacer?

Uso de Dispositivos con Permiso del Usuario:

- Interacción con cámara y micrófono requiere permiso explícito.
- No puede activar dispositivos de forma encubierta para vigilancia.

Javascript_ ¿Qué no puede hacer?

Restricciones entre Pestañas y Ventanas:

- Pestañas/ventanas desconocidas entre sí en la mayoría de los casos.
- Excepciones bajo ciertas condiciones (ejemplo: una ventana abierta por JavaScript).
- Política del mismo origen limita el acceso entre diferentes sitios.

Javascript_ ¿Qué no puede hacer?

Política del Mismo Origen (“Same Origin Policy”):

- Comunicación entre páginas de diferentes orígenes requiere acuerdo y código especial.
- Protege contra el acceso no autorizado a datos de otros sitios.

Javascript_ ¿Qué no puede hacer?

Limitaciones en la Comunicación de Red:

- JavaScript puede comunicarse con el servidor de origen.
- Restricciones en recibir información de otros sitios y dominios.
- Requiere acuerdo explícito a través de encabezados HTTP..

Javascript_ ¿Qué no puede hacer?

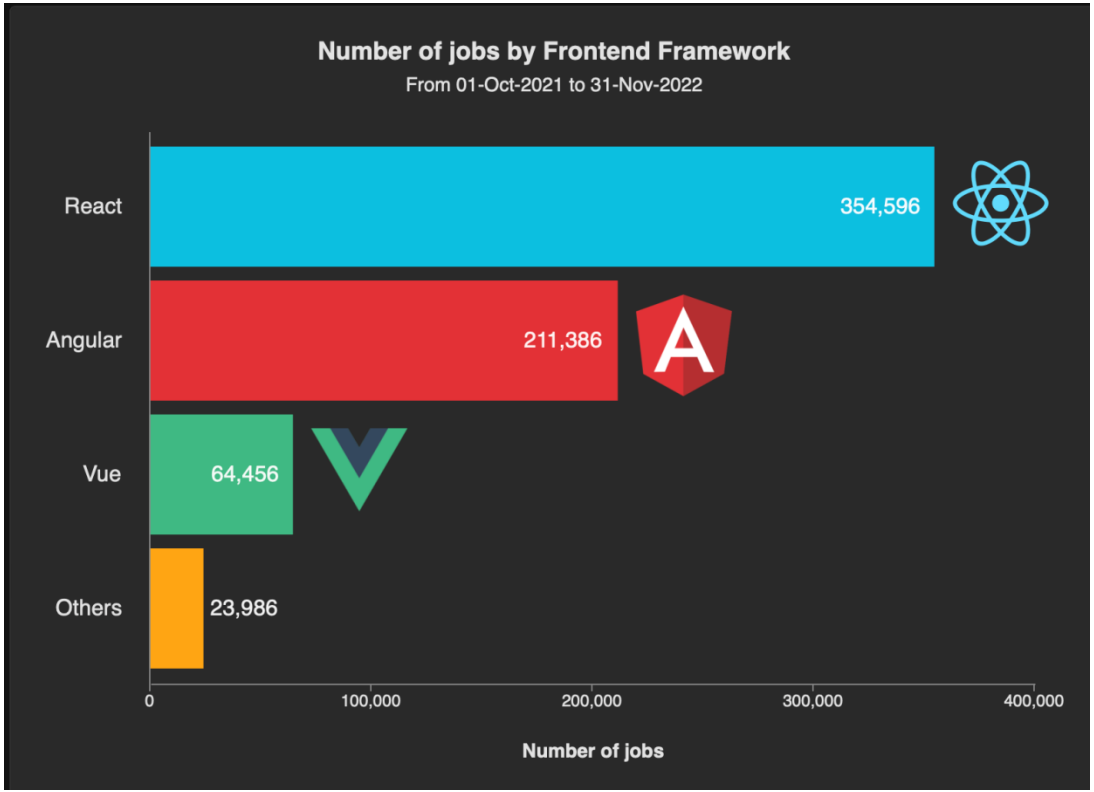
Sin Estas Limitaciones Fuera del Navegador:

- JavaScript en servidores no tiene estas restricciones.
- Navegadores modernos con complementos y extensiones pueden tener permisos extendidos.

Javascript_ ¿Qué lo hace único?

1. Completa integración con HTML y CSS.
2. Las cosas simples se hacen de manera simple.
3. Soportado por la mayoría de los navegadores y habilitado de forma predeterminada.

Javascript_ frameworks



Consola de desarrollador_

1. **Naturaleza Propensa a Errores del Código**:
    1. Los errores son comunes y esperados en el desarrollo.
    2. Incluso los desarrolladores experimentados cometen errores.
2. **Comportamiento Predeterminado del Navegador**:
    1. Los errores no se muestran automáticamente al usuario.
    2. Los fallos en scripts pueden ser invisibles sin herramientas adecuadas.
3. **Importancia de las Herramientas de Desarrollo**:
    1. Incorporadas en los navegadores modernos.
    2. Permiten ver errores y proporcionan información detallada sobre los scripts.

Consola de desarrollador_

4. **Preferencias de Navegadores entre Desarrolladores**:
    1. Chrome y Firefox son populares por sus avanzadas herramientas de desarrollo.
    2. Otros navegadores también tienen herramientas, pero pueden estar menos avanzadas.
5. **Uso de Diferentes Navegadores para Solución de Problemas**:
    1. Los desarrolladores suelen tener un navegador "favorito" para el desarrollo.
    2. Cambian a otros navegadores para problemas específicos de compatibilidad..

Consola de desarrollador_

6. **Características de las Herramientas de Desarrollo**:
    1. Ofrecen una amplia gama de funcionalidades.
    2. Incluyen la visualización de errores, ejecución de comandos JavaScript, inspección de elementos, etc.
7. **Aprender a Utilizar Herramientas de Desarrollo**:
    1. Comenzar por aprender a abrir las herramientas.
    2. Progresar a observar y entender errores.
    3. Practicar la ejecución de comandos JavaScript y explorar otras características.

Fundamentos_ ¡Hola, mundo!

```
<!DOCTYPE HTML>
<html>

<body>

  <p>Antes del script...</p>

  <script>
    alert( '¡Hola, mundo!' );
  </script>

  <p>...Después del script.</p>

</body>

</html>
```

Fundamentos_ implementación

Existen tres maneras principales de implementar JavaScript en un documento HTML

- **JavaScript en Línea (Inline JavaScript)**
- **JavaScript Interno (Internal JavaScript)**
- **JavaScript Externo (External JavaScript)**

Fundamentos implementación_
JavaScript en Línea(Inline JavaScript)

- Se escribe directamente en los atributos de los elementos HTML, generalmente para manejar eventos.

Por ejemplo, puedes usar JavaScript en un botón HTML:

```
<button onclick="alert('¡Hola Mundo!')">Haz clic aquí</button>.
```

Fundamentos implementación_
(Inline JavaScript)

onclick—Este atributo responde al evento **`click`**. El evento se ejecuta cuando el usuario hace clic con el botón izquierdo del ratón. HTML ofrece otros dos atributos similares llamados **`ondblclick`** (el usuario hace doble clic con el botón izquierdo del ratón) y **`oncontextmenu`** (el usuario hace clic con el botón derecho del ratón).

onmousedown—Este atributo responde al evento **`mousedown`**. Este evento se desencadena cuando el usuario pulsa el botón izquierdo o el botón derecho del ratón.

onmouseup—Este atributo responde al evento **`mouseup`**. El evento se desencadena cuando el usuario libera el botón izquierdo del ratón.

onmouseenter—Este atributo responde al evento **`mouseenter`**. Este evento se desencadena cuando el ratón se introduce en el área ocupada por el elemento.

onmouseleave—Este atributo responde al evento **`mouseleave`**. Este evento se desencadena cuando el ratón abandona el área ocupada por el elemento.

onmouseover—Este atributo responde al evento **`mouseover`**. Este evento se desencadena cuando el ratón se mueve sobre el elemento o cualquiera de sus elementos hijos.

onmouseout—Este atributo responde al evento **`mouseout`**. El evento se desencadena cuando el ratón abandona el área ocupada por el elemento o cualquiera de sus elementos hijos.

onmousemove—Este atributo responde al evento **`mousemove`**. Este evento se desencadena cada vez que el ratón se encuentra sobre el elemento y se mueve.

onwheel—Este atributo responde al evento **`wheel`**. Este evento se desencadena cada vez que se hace girar la rueda del ratón.

Fundamentos implementación_
JavaScript Interno Internal JavaScript

- Se coloca dentro de las etiquetas <script> en el cuerpo del documento HTML.

```
<script>
  function saludo() {
    alert("¡Hola Mundo!");
  }
</script>
```

Fundamentos implementación_
JavaScript Externo (External JavaScript)

- Se escribe en un archivo separado con la extensión .js y luego se referencia en el archivo HTML utilizando la etiqueta <script> con un atributo src.

```
<script src="ruta/al/archivo.js"></script>
```

Sentencias

Las sentencias son construcciones sintácticas y comandos que realizan acciones.

```
alert('Hola'); alert('Mundo');

alert('Hola');
alert('Mundo');
```

Punto y coma

Se puede omitir un punto y coma en la mayoría de los casos cuando existe un salto de línea.

```
alert('Hola')
alert('Mundo')
```

Fundamentos_ estructura del código

Comentarios

Los comentarios de una línea comienzan con dos caracteres de barra diagonal //.

```
// Este comentario ocupa una línea propia.
alert('Hello');

alert('World'); // Este comentario sigue a la sentencia.
```

Comentarios

Los comentarios de varias líneas comienzan con una barra inclinada y un asterisco /* y terminan con un asterisco y una barra inclinada */.

```
/* Un ejemplo con dos mensajes.
Este es un comentario multilínea.
*/
alert('Hola');
alert('Mundo');
```

Comentarios

La directiva se asemeja a un string: "use strict".

Cuando se sitúa al principio de un script, el script entero funciona de la manera “moderna”.

```
/* Un ejemplo con dos mensajes.
Este es un comentario multilínea.
*/
alert('Hola');
alert('Mundo');
```

Variables_ declaración de variables

var: Es la forma más antigua de declarar variables en JavaScript. Las variables declaradas con var tienen un ámbito de función (function scope) y pueden ser redeclaradas dentro de su ámbito.

let: Introducido en ES6 (ECMAScript 2015), let permite declarar variables con un ámbito de bloque (block scope). A diferencia de var, una variable declarada con let no puede ser redeclarada dentro de su ámbito.

const: También introducido en ES6, const se utiliza para declarar constantes. Una vez que se asigna un valor a una constante, no puede ser reasignado. Al igual que let, tiene un ámbito de bloque.

Variables_ Asignación de Valores:

Las variables pueden almacenar diferentes tipos de datos, como números, cadenas de texto (strings), booleanos, objetos, funciones, etc.

```
var nombre = "Juan";
let edad = 30;
const PI = 3.14;
```

Variables_ Ámbito de las Variables (Scope)

Ámbito de Función: Las variables declaradas con var son accesibles dentro de toda la función en la que se declaran, o globalmente si se declaran fuera de una función.

Ámbito de Bloque: Las variables declaradas con let y const están limitadas al bloque (por ejemplo, un bucle o una declaración condicional) en el que se declaran.

Variables_ Hoisting

Las variables declaradas con var son "elevadas" (hoisted) al inicio de su función o ámbito global, pero no su asignación.

let y const también son elevadas, pero no se puede acceder a ellas hasta que la ejecución del código llega a su declaración, lo que se conoce como "temporal dead zone".

Variables_ Tipos de Datos

JavaScript es un lenguaje de tipado dinámico, lo que significa que no necesitas especificar el tipo de dato de la variable al declararla. El tipo de dato de una variable puede cambiar durante la ejecución del programa.

Variables_ Tipos de Datos

- Números (sin distinción entre enteros y flotantes)
- Cadenas de texto (Strings)
- Booleanos (true/false)
- Objetos
- Funciones
- Undefined
- Null

Variables_ Buenas Prácticas

- Prefiere el uso de let y const sobre var para una mejor claridad del código y control de ámbito.
- Utiliza nombres de variables descriptivos y sigue las convenciones de nomenclatura (como camelCase en JavaScript).
- Declara las variables en la parte superior de su ámbito para evitar confusiones relacionadas con el hoisting.

Variables_ Buenas Prácticas

- Prefiere el uso de let y const sobre var para una mejor claridad del código y control de ámbito.
- Utiliza nombres de variables descriptivos y sigue las convenciones de nomenclatura (como camelCase en JavaScript).
- Declara las variables en la parte superior de su ámbito para evitar confusiones relacionadas con el hoisting.