**LOS 3 ESTÁNDARES PARA GENERAR PRODUCTOS WEB:**

* HTML
* CSS
* JavaScript

**HTML**

HTML (HyperText Markup Language) es un lenguaje de marcado de texto. Se utiliza para darle una estructura al sitio web que estás visitando.

Estructura de los documentos que se visualizan en los navegadores (Chrome, Edge, Firefox, Safari, Opera, entre otros).

Es el lenguaje con el que clasificamos o etiquetamos cada parte del documento, dejándole saber al navegador qué información va a visualizar y qué valor tiene para nosotros. De esa manera, presentamos: Barras de Navegación, Secciones, artículos, listas, títulos de diferentes tamaños, audios, videos, imágenes, entre otros.

**CSS**

Es el lenguaje con el que le damos estilos a todo el contenido previamente etiquetado con HTML.

Es el encargado o usado para darle todo el aspecto visual de manera profesional, usando estándares de diseño de documentos y logrando así la correcta visualización de textos y recursos audiovisuales en formatos digitales o, en nuestro caso, las páginas web.

**JAVASCRIPT**

Es lenguaje que se emplea para crear o brindar a las páginas web la capacidad de interactuar con el usuario y los recursos asociados a una página web, como lo son: bases de datos, servidores de diversa índole, información derivada de la interacción de cada página en sí.

**DIFERENCIAS**

* HTML es la estructura, como si fuese el esqueleto de un ser humano o los cimientos de un edificio. Por ejemplo: una página web que funciona, pero sin interactividad, colores ni diseño.
* CSS corresponde a lo estético, como la piel de ser humanos. La pintura y adornos de los edificios o la manera en que es posible que la estructura del HTML se vea mejor para los usuarios.
* JavaScript equivale a los músculos, lo que da esa interactividad a las personas para moverse y realizar acciones como correr.

**FRONTEND**

Es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios.

Un programador Frontend debe saber de código que entienda el navegador (HTML, CSS y JavaScript) para poder usar algunos frameworks o librerías que expanden sus capacidades para crear cualquier tipo de interfaces de usuarios. React, Redux, Angular, Bootstrap, Foundation, LESS, Sass, Stylus y PostCSS son algunos de ellos.

**HERRAMIENTAS QUE MANEJA UN FRONTEND**

Debido a que un frontend es el desarrollador (que puede ser o no Full Stack) que va a manejar las cosas del lado del cliente, las tecnologías con las que va a trabajar son:

* HTML: https://devdocs.io/html/
* CSS: https://devdocs.io/css/
* JavaScript: https://devdocs.io/javascript/

**FRAMEWORKS DE CSS PARA FRONTEND:**

* Bootstrap: https://getbootstrap.com/
* Foundation CSS: https://get.foundation/
* Materialize CSS: https://materializecss.com/

**LOS FRAMEWORKS DE JAVASCRIPT PARA FRONTEND:**

* React JS: https://es.reactjs.org/
* Angular JS: https://angular.io/
* Vue JS: https://vuejs.org/

**PREPROCESSADORES DE CSS:**

* Stylus: https://stylus-lang.com/
* SASS: https://sass-lang.com/

**COMPILADORES / EMPAQUETADORES DE JS:**

* BABEL: https://babeljs.io/
* Webpack: https://webpack.js.org/

**BACKEND**

Backend en programación corresponde al lado opuesto a un Front-end en un sitio web o aplicación, ya que el Backend trabaja en el lado del servidor, mientras el Frontend lo hace en el lado del cliente. Es el responsable de manejar toda la lógica que existe detrás de una petición dada por el navegador hacia el servidor. Existen programadores que manejan ambas especialidades y se les conoce como full stack.

Una característica que lo diferencia del Frontend es que no tiene estándares, puesto que tiene varios lenguajes de programación (Node.js, Python, PHP, Ruby, GO, Java, .NET entre otros) con los que debe trabajar. Cada uno de estos lenguajes tiene sus propios frameworks como:

* Django (Python)
* Lavarel (PHP)
* Rails (Ruby)
* Express (JavaScript)
* Spring (Java)

El Backend también tiene en cuenta la infraestructura donde va a realizarse el deploy de su aplicación (esto también puede ser tarea de un DevOps, un perfil dedicado a la infraestructura), con tecnologías como:

* Google Cloud
* DigitalOcean
* AWS
* Heroku, entre otras.

**QUE ES DEPLOY**

Deploy es un término famoso entre los desarrolladores web. Puede significar muchas cosas, dependiendo del ambiente y de la tecnología usada. Sin embargo, los significados que más se refieren a la práctica y pueden resumir su función son: implantar, colocar en posición, habilitar para uso o, simplemente, publicar.

Por último, entramos en bases de datos, que son las encargadas de almacenar toda la información del proyecto. Los principales tipos son:

* Bases de datos relacionales (como MySQL)
* Bases de datos no relacionales (como mongoDB).

**FULLSTACK**

Full Stack es un término utilizado para describir a los desarrolladores que conocen tanto los lenguajes de frontend como de backend. Principalmente, el desarrollo full stack se refiere al uso de JavaScript en el backend y de HTML/CSS/JavaScript en el frontend para crear nuevas plataformas.

El nacimiento de tecnologías que funcionan entre el frontend y el backend ha dado lugar a la proliferación de frameworks y herramientas de desarrollo full stack, que permiten a los desarrolladores construir sus propias aplicaciones web completas empleando un único lenguaje de programación, como Django para Python.

**¿QUÉ ES UN DESARROLLADOR FULLSTACK?**

Las y los desarrolladores Full Stack son profesionales que se encargan tanto de la parte visual y de interacción de un sitio (frontend), como de su lógica y funcionamiento del lado del servidor (backend).

Un stack (en inglés: pila o montón) se refiere al grupo de tecnologías que componen un sitio web en todos los aspectos (desde la base de datos, hacia el manejo lógico y la interfaz visual). Una o un desarrollador Full Stack en teoría es capaz de manejar la pila completa de un sitio, tanto de frontend como de backend, además sabe utilizar su base de datos.

**¿QUE HACE UN DESARROLLADOR FULLSTACK?**

Una desarrolladora o desarrollador Full Stack entiende muy bien cómo funciona un producto web de principio a fin, desde su diseño en mockup y deploy hasta producción.

Este tipo de programador o programadora no maneja por completo todas las tecnologías de ambas partes, pues cada una requiere conocimiento profundo. De hecho, no es recomendado profundizar en tantas especialidades y no es sano. El desarrollo web evoluciona muy rápido y cada dos o tres meses encontrarás algo nuevo.

**TECNOLOGIAS**

Aprender un stack conocido puede ayudarte a dar tus primeros pasos como desarrollador Full Stack y a escalar un proyecto a producción de una manera más rápida y con una mayor interacción. Para eso necesitas estas herramientas.

* LAMP (Linux, Apache, MySQL, PHP)
* MERN (MongoDB, Express.js, React, Node)
* PERN (PostgreSQL, Express.js, React, Node)

**SITIOS WEB ESTÁTICOS**

La información que contiene se mantiene constante y estática. No se actualiza con la interacción del usuario. Es conveniente para realizar landing pages (páginas informativas o de aterrizaje) o blogs. Se mostrarán siempre iguales para todos los usuarios.

**SITIOS WEB DINÁMICOS**

También conocidos como aplicaciones web, actualizan su información con respecto a la interacción del usuario. Dependen de una base de datos, de donde extrae e ingresa información. Serán diferentes, dependiendo del usuario que la use y la información que se ingrese.

**Ejemplo de páginas estáticas:**

* Menú de un restaurante
* Blog de viajes
* Página informativa de un negocio

**Ejemplo páginas dinámicas:**

* Sistema de reporte de ventas
* Linkedin
* Banca en línea

**ESTRUCTURA BÁSICA DE HTML**

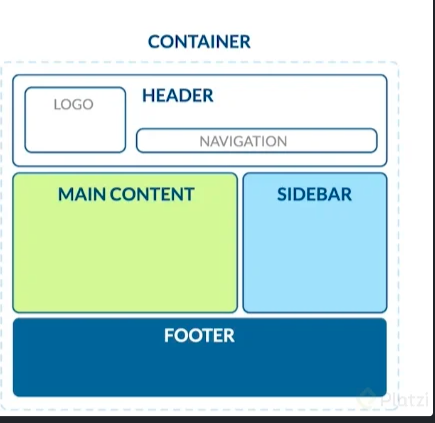
**Container**: contenedor principal

**Header**: cabecera de la página. Aquí usualmente encuentras el logo y el menú de navegación del sitio.

**Main content**: estructura principal. Por ejemplo, el feed o lista de publicaciones de una red social.

**Sidebar**: contenido secundario de una página, que usualmente se encuentra a los lados del contenido principal (o main).

**Footer**: pie de página. Esto se encuentra al fondo del sitio web, salvo en casos de sitios web donde el scroll (o navegación hacia abajo) es infinito, por ende, no tendría sentido ponerlo al fondo.



Las etiquetas en HTML nos ayudan a diferenciar en qué parte del contenido nos encontramos.

La web se conforma de tres conceptos:

* **URL**: Uniform Resource Locator. El identificador único del sitio en el navegador (por ejemplo: https://platzi.com).
* **HTTP**: Protocolo de transferencia de hipertexto. Es el estándar que se utiliza para enviar datos a través de paquetes entre el cliente y el servidor.
* **HTML**: es el código que se emplea para estructurar el contenido de tu web, y darle sentido y propósito.

HTML son siglas que corresponden a Hyper Text Markup Language (Lenguaje de Marcado de Hipertexto).

* **Hyper Text** significa que el texto tiene interactividad, conexión con otros documentos.
* **Markup** significa que le pone etiquetas a los elementos. Por eso también se le conoce como un lenguaje de etiquetas.
* **HTML** es un lenguaje interpretado. Además, HTML es un estándar, así que no importa desde qué navegador o dispositivo se ejecute, el código sigue siendo el mismo en cualquier sitio.

**BODY**

Body es la etiqueta que identifica la parte visible de nuestro sitio web. Dentro del body, se añadirán las etiquetas para marcar los elementos visuales del sitio web, como logotipo, menús de navegación, contenido principal, entre otrs. Es muy importante usar HTML semántico y no llenar todo de <div> para que nuestro sitio sea mejor interpretado por el navegador y, por lo tanto, más accesible.

**ETIQUETAS DEL CUERPO DEL DOCUMENTO (BODY):**

**article**: diferencia partes del contenido que pueden vivir por sí mismas.

**nav**: para hacer menús de navegación.

**aside**: contenido menos relevante, como publicidad, etc.

**section**: sirve para diferenciar las secciones principales del contenido.

**header**: cabecera del documento.

**footer**: pie de página del documento.

**h1 - h6**: títulos de nuestro sitio web.

**table**: tablas de contenidos, similar a la estructura de las hojas de calculo.

**ul y ol**: listas de items.

**div**: cualquier división para organizar el contenido.

**h1 a h6**: son etiquetas para indicar títulos con un estilo que destaca del resto.

**article**: es la parte de nuestro contenido que puede vivir por sí mismo. Pueden haber tantos artícle como proyectos o eventos tenga nuestro portafolio.

**p**: define el texto de un párrafo.

**small**: aplica una apariencia de texto reducido en tamaño.

**strong**: aplica al texto un formato de negritas.

**a**: corresponde a un ancla o enlace a una url interna o externa del documento.

**img**: con esta etiqueta podemos enlazar imágenes en el documento.

**figure**: le da un contexto semántico a las imágenes.

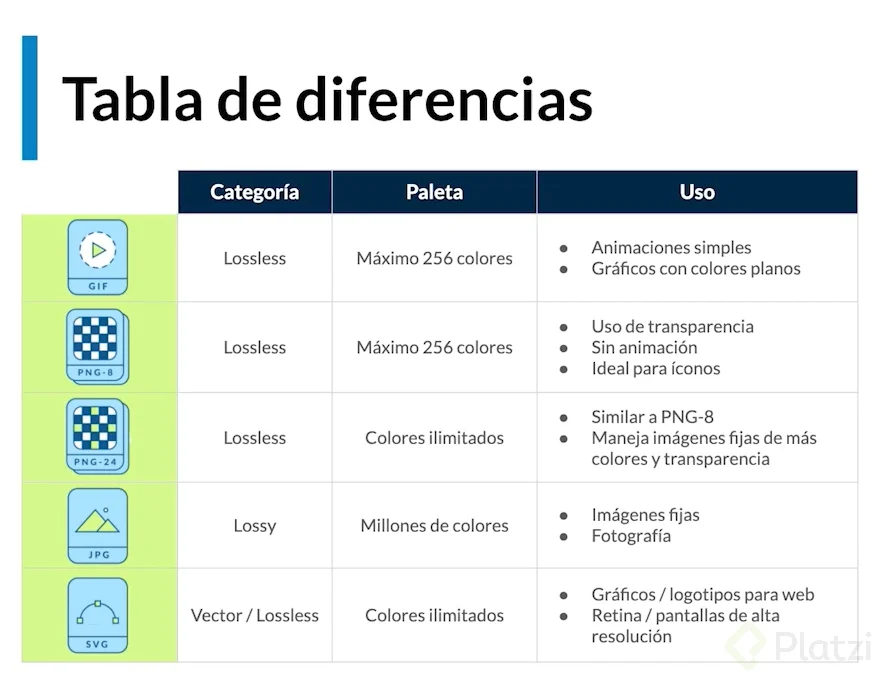
**TIPOS DE IMAGENES**

**LOSSLESS (SIN PERDIDA)**

* Capturan todos los datos del archivo original.
* No se pierde nada del archivo original.
* Puede comprimirse, pero podrá reconstruir su imagen al estado original

**LOSSY(CON PERDIDA)**

* Se aproximan a su imagen original.
* Podría reducir la cantidad de colores en su imagen o analizar la imagen en busca de datos innecesarios.
* Por consiguiente puede reducir su tamaño, lo que mejora el tiempo de carga de la página, pero pierde su calidad.
* Los archivos tipo lossy son mucho más livianos que los archivos tipo lossless, por lo que son ideales para usar en sitios en donde el tamaño del archivo y la velocidad de descarga son importantes.



**FORMATOS DE IMAGEN:**

* **GIF** (Graphics Interchange Format): Formato de imagen sin pérdida, no se puede comprimir
* **PNG 8** (Portable Network Graphics): Formato de imagen sin pérdida, uso de colores de 256, se utiliza para logotipos e iconos para la página.
* **PNG 24** (Portable Network Graphics): Formato de imagen sin pérdida, utilización de colores ilimitados, alta calidad, si intentamos comprimir no ayudará demasiado por la gran cantidad de colores.
* **JPG / JPEG** (Photographic Experts Group): Formato de imagen con pérdida, perdemos calidad a la hora de comprimirlas, pero llegan a ser óptimas para la carga en la página web.
* **SVG** - Vector (Scalable Vector Graphics): Formato de imagen muy ligero sin pérdida, con svg no perdemos calidad, ya que está compuesta por vectores.
* **WebP**: Es un formato gráfico en forma de contenedor que sustenta tanto compresión con pérdida como sin ella. ​​Fue desarrollado por Google.

**PORQUE ES IMPORTANTE OPTIMIZAR IMÁGENES PARA LA WEB?**

Optimizar imágenes es crucial para cualquier proyecto web que se precie. Imagina que una imagen de tu sitio pesa tanto que ralentiza la carga de la página, provocando una mala experiencia al usuario. Esto es lo que ocurre cuando se usan imágenes pesadas sin procesar. Además, las imágenes pesadas ocupan más espacio de almacenamiento y pueden afectar negativamente tu posicionamiento en motores de búsqueda. Entonces, ¿cómo podemos optimizar las imágenes sin sacrificar calidad?

**CUAL ES EL TAMAÑO ÓPTIMO PARA UNA IMAGEN WEB?**

El tamaño recomendado para una imagen en un sitio web no debería superar los 70 kilobytes (KB). Sin embargo, muchas veces encontramos imágenes que pesan varios megabytes (MB), lo cual es excesivo. Un peso tan elevado no solo afecta el tiempo de carga sino también el rendimiento general de tu página web.

Peso óptimo: Alrededor de 70 KB

Límite máximo: Nunca más de 1 MB

**COMO OPTIMIZAR IMAGENES EFICIENTEMENTE?**

Existen varias herramientas y métodos para optimizar imágenes de manera eficaz. A continuación, te describimos dos opciones que te serán de utilidad para comprimir y optimizar imágenes sin perder demasiada calidad:

**TinyPNG**

TinyPNG es una solución web eficiente que permite comprimir imágenes de formatos PNG y JPG. Aunque su nombre lo asocia principalmente al formato PNG, es especialmente eficaz para fotos en formato JPG, ya que reduce significativamente el tamaño sin comprometer la calidad visual.

**Verifix**

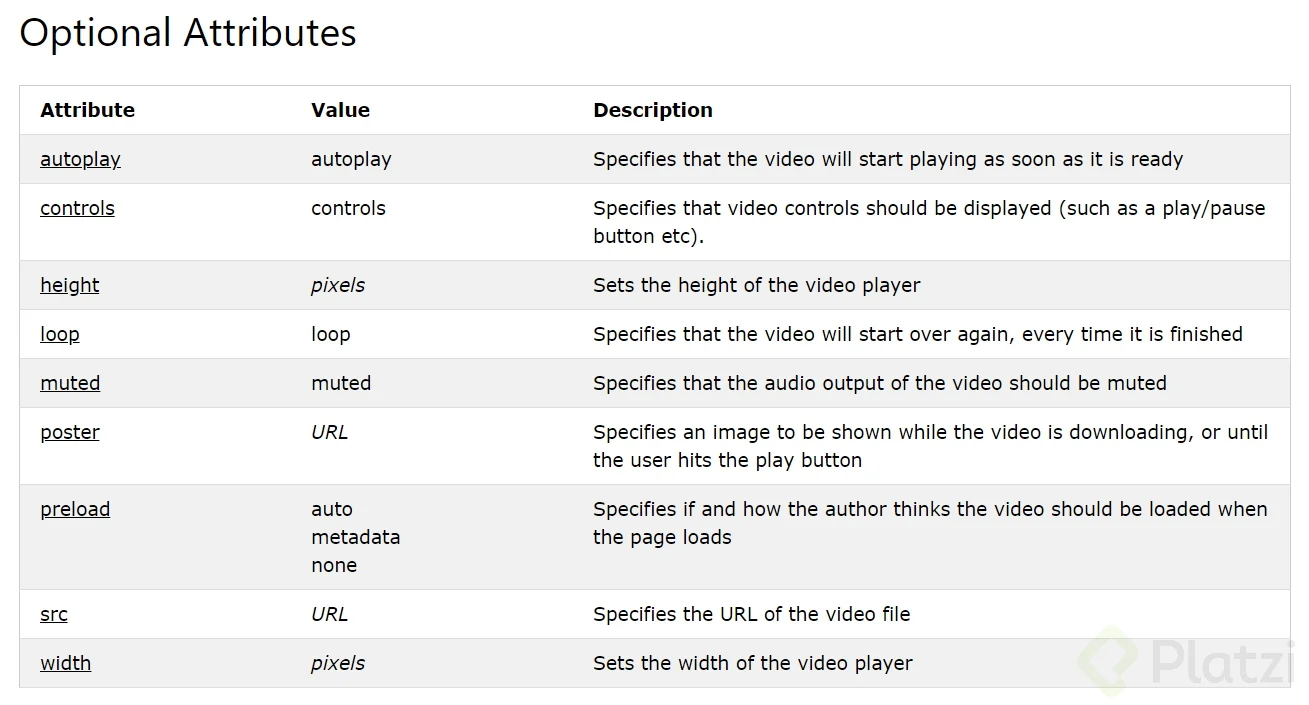
Se especializa en eliminar metadatos de las fotografías. Cuando tomas fotos con una cámara o un teléfono, estas a menudo almacenan metadatos como la hora en que se tomó, la ubicación, entre otros. Aunque útiles en ciertos contextos, estos datos pueden añadir peso innecesario a tus archivos.

**¿CÓMO AFECTA EL PESO DE LAS IMÁGENES AL RENDIMIENTO WEB?**

Imágenes pesadas son la razón por la que algunas páginas tardan en cargar, lo cual puede ser frustrante para los usuarios. Además del tiempo de carga, las imágenes de mayor tamaño pueden afectar:

* **Aranado en buscadores:** Las velocidades de carga más lentas pueden afectar tu SEO negativamente.
* **Excesivo uso de datos**: Esto es problemático particularmente para usuarios móviles.
* **Menor retención de usuarios**: Los visitantes tienden a abandonar páginas lentas rápidamente.

**VIDEO**



**FORM E INPUT**

Los formularios son esenciales en los productos web porque son la principal forma en que interactuamos con los usuarios. Nos permiten solicitar o recibir información de los visitantes, ya sea para registrar cuentas de usuario, completar transacciones de compras o recolectar retroalimentación. Sin embargo, a menudo pueden ser una barrera si no están bien diseñados. Formularios que son confusos o demasiado largos pueden alejar a los usuarios potenciales.

**MEJORAR LA EXPERIENCIA DEL USUARIO**

Para enriquecer la experiencia del usuario en los formularios, puedes seguir estas recomendaciones:

**Simplicidad**: Reduce los campos a los necesarios.

**Guías visuales**: Utiliza etiquetas y placeholders que orienten al usuario sobre qué información necesita proporcionar.

**Proporciona retroalimentación:** Después de enviar, confirma el éxito o indica errores de manera clara.

**QUE TIPOS DE INPUTS EXISTEN**

HTML ofrece una amplia variedad de tipos de inputs que permiten recibir diferentes formatos de datos. Entre los más comunes están:

**Text**: Para texto regular, como nombres o direcciones.

**Email**: Que incluye validación básica para formatos de correo electrónico.

**Date**: Permite al usuario seleccionar una fecha desde un calendario desplegable.

**Time**: Brinda una interfaz para que el usuario seleccione una hora específica del día.

**Number**: Permite inputs numéricos, lo que limita el tipo de información que puede ser ingresada.

**PlaceHolder:** proporciona un texto sugerido dentro del campo de entrada

**CSS**

CSS, o Cascading Style Sheets, es una tecnología crucial para definir la presentación visual de un sitio web. Este estándar se encarga de la apariencia de los elementos HTML, permitiendo a los desarrolladores aplicar estilos visuales sofisticados, desde colores y tipografía hasta el diseño y las dimensiones de los elementos. Queremos que nuestros proyectos no solo funcionen bien, sino que también se vean atractivos, y aquí es donde CSS entra en juego.

CSS se estructura mediante un algoritmo de "cascada" que el navegador utiliza para aplicar los estilos en un orden específico y lógico. Este mecanismo garantiza que se sobreescriban o ajusten adecuadamente los estilos cuando sea necesario, lo que permite personalizar y ajustar la presentación del contenido de manera eficiente.

Podemos comparar CSS y HTML con el proceso de escribir una carta. Si consideramos que HTML es el lápiz que nos permite plasmar la información en la hoja, CSS sería el conjunto de herramientas decorativas: los colores, adornos e incluso tijeras para personalizar y embellecer la carta. HTML estructura la información, mientras que CSS la hace visualmente atractiva.

**IMPORTANCIA DE CSS**

**Esteticismo Visual**: Cualquier proyecto web puede transformar texto aburrido y grandes imágenes en contenido visualmente apacible.

**Flexibilidad y Personalización**: Permite definir diferentes estilos y formatos, adaptándose a las necesidades específicas del proyecto.

**Compatibilidad Multidispositivo:** CSS ofrece métodos para que un sitio se vea bien en múltiples dispositivos y tamaños de pantalla.

**QUÉ SON LAS PSEUDO CLASES Y PSEUDO ELEMENTOS EN CSS?**

En el desarrollo web, el manejo adecuado de estilos en CSS es crucial para crear interfaces de usuario atractivas y funcionales. Dos de las características avanzadas que más se utilizan para estilizar componentes son las pseudo clases y los pseudo elementos. Estos nos permiten agregar estilos basados en ciertos estados o insertar contenido adicional sin cambiar el HTML.

**¿QUÉ SON LOS PSEUDO ELEMENTOS?**

Los pseudo elementos se utilizan para designar y estilizar partes específicas de un elemento. Se introducen utilizando dobles dos puntos (::)

**¿CÓMO AFECTA EL ESTILO PREDETERMINADO DEL NAVEGADOR?**

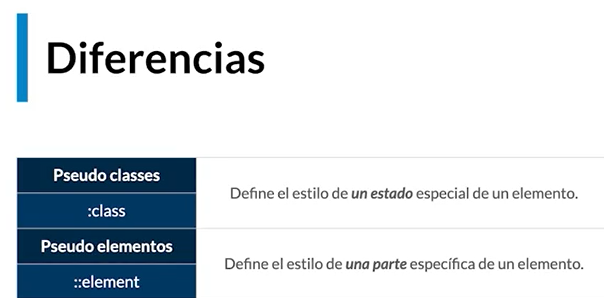
Es fundamental entender que los navegadores aplican algunos estilos por defecto a los elementos HTML. Estos estilos pueden incluir márgenes, paddings y decoraciones como subrayados en enlaces. Es vital dominar cómo sobrescribir estos estilos para que todos los elementos se comporten según tus especificaciones de diseño.

**¿QUÉ ES EL BOX MODEL Y CÓMO INFLUYE EL PADDING Y MARGIN?**

El box model de CSS es la base para entender cómo el espacio y el tamaño de los elementos se determinan en el navegador:

Padding: Espacio interno entre el contenido de un elemento y su borde. Al ajustar el padding, incrementas la distancia entre el contenido y los bordes.

Margin: Espacio externo que separa un elemento de otro en la página. Es útil para separar y alinear elementos.



**¿QUÉ ES UNA REGLA EN CSS?**

El mundo del CSS, o Cascading Style Sheets, es esencial para cualquier desarrollador web y comprende un conjunto de reglas formales que nos permiten transformar la apariencia de los elementos HTML en una página web. Una regla de CSS se compone de varias partes fundamentales y comprensión de esta estructura básica es clave para implementar estilos de manera efectiva.

**¿CÓMO SE COMPONE UNA REGLA DE CSS?**

**Selector**: El primer componente de una regla CSS es el selector, que indica cuál es el elemento en el que queremos aplicar estilos.

Ejemplos comunes de selectores son etiquetas HTML (p, div, h1), clases (precedidas por un punto, como .clase) o IDs (precedidos por una almohadilla, como #id).

**Declaración**: Después del selector, se utilizan llaves {} para contener las declaraciones de estilo.

Una declaración de estilo se compone de una propiedad y un valor.

Las propiedades son las características que deseas cambiar, como el color o la fuente.

Los valores determinan cómo quieres que se vea dicha propiedad.

**PORQUE ES FUNDAMENTAL SEGUIR LAS REGLAS DE SINTAXIS?**

**Punto y coma** (;): Cada declaración dentro de una regla debe terminar con un punto y coma. Omidir este detalle esencial puede romper la cadena de estilos y afectar la visualización.

**Llaves ({ }):** Estas deben estar correctamente cerradas. Una llave que no se cierra adecuadamente puede impedir que los estilos se apliquen.

**Selector correcto**: Usar un selector que no existe o con errores tipográficos hará que los estilos sean ignorados.

**¿QUÉ PASA SI NO SIGUES LAS NORMAS?**

La importancia de seguir correctamente la sintaxis de CSS no puede subestimarse. Un pequeño error puede llevar a:

* Estilos no aplicados correctamente.
* Discrepancias visuales.
* Dificultades en el mantenimiento del código.

**¿QUÉ ES EL MODELO CAJA?**

Cuando trabajas con CSS, es esencial comprender el concepto de "modelo de caja", ya que es la base para aplicar estilos visuales a los elementos HTML. Imagina que cada elemento renderizado se comporta como una caja, la cual está compuesta por cuatro áreas principales: el contenido, el padding, el borde y el margin.

**¿CÓMO SE DEFINE LOS MÁRGENES, BORDES Y PADDING?**

**Margin**: Es el espacio externo de la caja hacia afuera. Funciona como una separación entre la caja y otros elementos. Aunque el margin es transparente y no visible, asegura que la caja tenga un respiro alrededor de otros elementos en la página.

**Borde**: Es la línea que rodea el contenido de la caja. El borde puede tener color y grosor, permitiendo destacarse si es necesario. Por defecto, es transparente, pero lo puedes personalizar.

**Padding**: Este es el espacio interno de la caja hacia adentro. Ayuda a posicionar el contenido dentro de la caja, brindando un margen interno entre el contenido y los bordes del elemento.

**¿CÓMO SE ESTABLECE EL TAMAÑO Y EL POSICIONAMIENTO?**

**Width y Height**: Dictan el ancho y el alto de la caja, respectivamente. Puedes manipular estos valores para ajustar tanto el tamaño del contenedor como el del contenido dentro de él.

**Posicionamiento:** Utilizando propiedades como top, bottom, right y left, puedes mover la caja en cualquier dirección, según se requiera.

**¿CÓMO PREVENIR PROBLEMAS CON EL SCROLL?**

A menudo, cuando aplicas padding y border, estos pueden causar que la caja se desborde, generando barras de scroll no deseadas. Una solución eficaz es utilizar box-sizing: border-box, lo cual asegura que el tamaño total del elemento respete las dimensiones del padding y el borde. Así, al establecer un ancho del 100%, el navegador automáticamente reajusta el contenido para evitar el scroll horizontal.

**INICIAR UN CSS**

Al comenzar con CSS, es crucial reiniciar los estilos predeterminados del navegador, lo que incluye margin y padding, aplicando el selector universal \*. Esto garantiza que no tengas comportamientos inesperados en las dimensiones de los elementos:

Resetear Padding y Margin: Usar \* { margin: 0; padding: 0; }.

Box-sizing Universal: Agregar box-sizing: border-box; en el selector universal para un manejo más uniforme de las dimensiones.

**¿QUÉ ES LA HERENCIA EN CSS?**

La herencia en CSS se refiere al fenómeno por el cual algunos estilos establecidos en un elemento HTML se transmiten automáticamente a sus elementos hijos. Por ejemplo, al definir un tamaño de fuente en la etiqueta <html>, ese tamaño puede propagar a etiquetas como <p> o <span> que no tengan un tamaño de fuente definido explícitamente.

**Propiedades herederas comunes:** tamaño de fuente (font-size), color de texto (color).

**Propiedades no herederas:** márgenes (margin), posición (position), anchura (width).

**¿CÓMO AFECTA LA HERENCIA?**

La herencia en CSS es un concepto clave que debe entenderse para dominar cómo los estilos se aplican en un documento HTML. Fundamentalmente, se refiere a la capacidad de ciertos estilos de "fluir" desde un elemento padre hasta sus hijos. Comprender este mecanismo ofrece un control más preciso sobre la presentación visual de nuestras páginas web, evitando sorpresas y permitiendo diseñar con intencionalidad.

**PORQUE ES IMPORTANTE COMPRENDER LA HERENCIA?**

Entender la herencia permite evitar errores comunes y optimiza el código CSS. Algunas razones para dominar la herencia incluyen:

**Consistencia en el diseño:** Asegura que los estilos se apliquen uniformemente en toda la página.

**Menor repetición de código:** Reducir la necesidad de redefinir estilos para cada elemento.

**Flexibilidad y control:** Facilita el ajuste preciso del estilo donde sea necesario, rompiendo la herencia cuando ciertos elementos requieren un tratamiento diferente.

**ORDEN DE DECLARACIÓN CSS**

Entender el orden de declaración y especificidad en CSS es vital para cualquier desarrollador web. Puede ser frustrante ver estilos que no se aplican como se espera, pero con una comprensión clara de la herencia y especificidad de CSS, estas frustraciones pueden minimizarse eficazmente. Aquí descubrirás cómo controlar las declaraciones de CSS y cómo afectan el resultado visual final de tu proyecto.

**¿CÓMO FUNCIONA LA CASCADA DE CSS?**

**Importancia**: El navegador carga estilos de distintas fuentes. Primero aplica los estilos predeterminados del usuario (navegador), luego los estilos escritos por los desarrolladores, y por último aplica aquellos con la etiqueta !important al final de una declaración. La recomendación es evitar el uso de !important pues puede complicar la gestión de estilos.

**Especificidad**: La especificidad es crucial y se mide de derecha a izquierda:

* !important (evitarlo por malas prácticas).
* Estilos inline.
* Estilos aplicados a IDs.
* Estilos aplicados a clases y pseudoclases.
* Estilos aplicados a selectores de elementos HTML.

**CÓMO AFECTAN LA ESPECIFICIDAD Y EL ORDEN DE LAS REGLAS CSS?**

El orden en el que se cargan los archivos CSS también importa. CSS sigue una lógica de cascada, donde el último estilo declarado tiene más peso. Aquí algunas recomendaciones:

Asegúrate de que los estilos más generales estén declarados primero y las especificaciones posteriores se apliquen a lo largo del archivo.

Mantén en mente que estilos externos importados después de tus declaraciones podrían sobrepasar tus estilos sin importar su especificidad.

**¿QUÉ SON LOS COMBINADORES EN CSS?**

Los combinadores en CSS son esenciales para definir estilos con especificidad sin depender de los IDs. Nos permiten seleccionar y aplicar estilos a elementos basándonos en sus relaciones con otros elementos. Los cuatro combinadores más utilizados son:

Hermano adyacente (+): Selecciona el primer hermano inmediato que sigue a un elemento específico.

Hermano general (~): Selecciona todos los hermanos posteriores a un elemento específico.

Hijo directo (>): Selecciona el hijo directo de un elemento determinado.

Descendiente (espacio): Selecciona a todos los descendientes de un elemento, independientemente de cuán profundamente anidados estén.

**¿CÓMO FUNCIONA EL COMBINADOR DE HERMANO ADYACENTE?**

El combinador de hermano adyacente (adjacent sibling) nos permite seleccionar un elemento que sigue inmediatamente después de otro en el mismo nivel jerárquico. Veamos un ejemplo que lo ilustrará mejor.

**PORQUE USAR COMBINADORES EN PROYECTOS GRANDES?**

En proyectos de gran envergadura, donde pueden haber múltiples niveles de anidamiento y repetidas estructuras de elementos similares, los combinadores permiten:

**Especificidad sin IDs:** Un enfoque limpio y fácil de mantener para aplicar estilos específicos.

**Flexibilidad**: Permite hacer ajustes rápidamente sin necesidad de alterar el HTML subyacente.

**Reducción del uso de clases**: Disminuye la sobrecarga de crear y manejar clases para cada caso particular.

**¿QUÉ ES EL COMBINADOR GENERAL?**

El combinador de hermano general, simbolizado por la tilde (~), es un selector CSS que aplica estilos a todos los elementos que comparten al menos un elemento previo en común en su jerarquía de documento. No te preocupes si no sabes cómo escribir este carácter en tu computadora; existen atajos de teclado específicos para distintas plataformas como Windows, Mac o Ubuntu.

**CONCLUSIONES**

Al usar el combinador de hermano general, puedes:

Estilizar múltiples elementos de forma eficiente sin tener que nombrar todos los elementos individualmente.

Mantener un CSS limpio y legible.

Estructurar mejor tu código HTML para aplicar estilos dinámicos adaptables a tu contenido.

**QUÉ SON LOS COMBINADORES HIJOS Y DESCENDIENTES EN CSS?**

Los combinadores en CSS son herramientas cruciales para un diseño eficiente y dirigido. En este artículo, exploraremos en profundidad dos combinadores esenciales: child combinator (hijo directo) y descendant combinator (descendiente).

**COMBINADOR HIJO (CHILD)**

El combinador hijo en CSS se utiliza para estilizar elementos que son hijos directos de un contenedor. Utiliza el símbolo > para definir que el estilo se aplicará solo a los elementos que sean hijos directos de otro.

**COMBINADOR DESCENDIENTE (DESCENDANT)**

El combinador descendiente es más generalizado y se usa frecuentemente en CSS. Selecciona todos los elementos que estén dentro de un elemento específico, sin importar su nivel de anidación.

**CUANDO USAR CADA COMBINADOR?**

**Child combinator (>):** Úsalo cuando necesites estilizar solo los elementos que son hijos inmediatos y directos de un elemento contenedor. Esto es útil para mantener el control sobre estilos específicos donde la jerarquía de HTML es importante.

**Descendant combinator (espacio)**: Al ser más general, este combinador es ideal para aplicar estilos globales a todos los elementos de un tipo específico dentro de un contenedor. Es sencillo, rápido, pero menos preciso.

**CONSEJOS**

Conocimiento del DOM: Es fundamental entender la estructura del DOM (Document Object Model) para usar estos combinadores eficazmente. Conocer cómo están anidados tus elementos HTML te ayudará a elegir el combinador correcto.

Evitar redundancias: Evita sobrecargar tus estilos usando combinadores innecesarios. Si un estilo global es suficiente, no es necesario que uses un combinador hijo directo.

Práctica interactiva: Pon en práctica lo aprendido con recursos interactivos como juegos o ejercicios que desafíen tus habilidades con selectores y combinadores. Esto no solo te ayudará a afianzar el conocimiento, sino que hará el aprendizaje más ameno.

**¿QUÉ SON LAS MEDIDAS EN CSS?**

Las medidas en CSS juegan un papel crucial en la creación de interfaces de usuario receptivas y efectivas. La comprensión de las diferencias entre medidas absolutas y relativas es fundamental para diseñar páginas web que se vean bien en cualquier dispositivo. Las medidas absolutas no cambian dependiendo del medio de visualización; por ejemplo, 18 píxeles seguirán siendo 18 píxeles. Por otro lado, las medidas relativas, como porcentajes o unidades em y rem, se ajustan al entorno en el que se visualizan, ofreciendo flexibilidad y adaptabilidad.

**¿CUÁLES SON LAS MEDIDAS ABSOLUTAS EN CSS?**

Las medidas absolutas son aquellos parámetros fijos que no cambian sin importar el medio en que se visualice la página web. Aquí se incluyen las medidas en píxeles.

Píxeles (px): Definidos de forma constante, no cambian sin importar el dispositivo. Otorgan predictibilidad, pero pueden carecer de flexibilidad cuando se trata de dispositivos de diferentes tamaños.

**¿QUÉ SON LAS MEDIDAS RELATIVAS EN CSS?**

A diferencia de las absolutas, las medidas relativas ajustan su tamaño basado en el contexto del objeto padre o del tamaño de la pantalla, lo que las hace altamente flexibles y recomendadas para diseños adaptativos.

**Porcentajes (%):** Ajustan su tamaño en relación al contenedor padre.

**Em:** Basado en el tamaño de fuente del elemento padre.

**Rem**: Basado en el tamaño de fuente del elemento raíz del documento HTML.

**Viewport width (vw) y viewport height (vh):** Miden el ancho y alto del viewport (la ventana visible de la página).

**USO DE MEDIDAS MAX Y MIN**

Además de las medidas estándar, existen las variantes max-width, min-width, max-height y min-height, que permiten establecer límites máximos o mínimos a las dimensiones de un elemento.

**¿QUÉ ES LA MEDIDA RELATIVA “EM”?**

La medida "em" es una unidad de longitud en CSS que se utiliza con frecuencia para especificar tamaños de fuente, márgenes y rellenos. Su característica distintiva es que depende del tamaño de fuente del elemento padre inmediato, lo que la convierte en una medida relativa que puede generar algunas situaciones confusas para los desarrolladores si no se utiliza adecuadamente.

**CONFUSIONES COMUNES AL USAR “EM”**

El escenario donde un <div> tiene un font-size de 1.5em tomará el tamaño de fuente del padre inmediato (no ancestro), y así sucesivamente, multiplicando acumulativamente si hay anidaciones de tales situaciones.

**COMO EVITAR PROBLEMAS CON “EM”**

**Uso cuidadoso:** Asegúrate siempre de saber cuál es el elemento padre inmediato cuando usas "em", para poder prever el tamaño calculado.

**Validación visual:** Utiliza herramientas como el inspector de elementos del navegador para verificar rápidamente si los tamaños de fuente se comportan como esperas.

**Estado inicial definido:** Establece intencionadamente tamaños de base (font-size) en

elementos raíz para mantener consistencia y previsibilidad.

**QUE ES REM?**

El rem es una unidad relativa en CSS que siempre se refiere al tamaño de fuente del elemento raíz (html), que por defecto suele ser 16 píxeles en la mayoría de los navegadores. A diferencia de em, que depende del tamaño de fuente del elemento padre inmediato, el rem es consistente y predecible. Esto te permite tener un control absoluto sobre cómo aparecen los textos y demás elementos en tu página.

**VENTAJAS**

* **Consistencia**: Rem siempre está basado en el tamaño de fuente de html.
* **Previsibilidad**: Evita variaciones indeseadas causadas por tamaños de fuente anidados.
* **Simplicidad**: Facilita la gestión de dimensiones sin cálculos complejos.

**USAR REM**

* **Reemplaza píxeles por rem** en tus proyectos actuales y futuros.
* **Establece el font-size inicial en el html** como hemos indicado para simplificar la conversión.
* **Adopta este método en conjunto con otras buenas prácticas** para un diseño web más eficiente, flexible y adaptable.

**CÓMO DEFINIR MEDIDAS EN CSS USANDO MIN, MAX, VIEWPORT WIDTH Y VIEWPORT HEIGHT?**

Definir las medidas y tamaños de los elementos que construimos en una página web es fundamental para asegurar una experiencia de usuario adecuada. Este enfoque permite que las interfaces sean más fluidas y adaptables a distintos dispositivos. En esta clase, exploraremos cómo el uso conjunto de las unidades de medida min, max, viewport width (vw), y viewport height (vh) nos ayuda a lograr un diseño flexible y responsivo.

**QUE ES EL VIEWPORT WIDTH(VW) Y VIEWPORT HEIGHT(VH)?**

viewport width (vw) y viewport height (vh) son unidades de medida relativas al tamaño de la ventana gráfica del navegador:

**vw**: Un vw es igual al 1% del ancho de la ventana gráfica.

**vh**: Un vh es igual al 1% de la altura de la ventana gráfica.

**MIN-MAX WIDTH MIN-MAX HEIGHT**

Las propiedades min-width, max-width, min-height, y max-height permiten restringir el tamaño mínimo y máximo que puede alcanzar un elemento. Esto ayuda a controlar el crecimiento y contracción del contenido de manera ordenada dentro de las restricciones de diseño.

**SOLUCIÓN AL PROBLEMA DE OVERFLOW EN CONTENIDO**

El overflow, o desbordamiento, ocurre cuando el tamaño del contenido excede el espacio disponible en el contenedor. Esto es común cuando se define una altura fija y el contenido es dinámico.

Para resolver el desbordamiento podemos utilizar min-height:

section {

min-height: 500px;

/\* Otros estilos \*/

}

Al utilizar min-height para el contenedor, permitimos que crezca más allá de sus 500 píxeles originales si el contenido lo requiere, proporcionando espacio adicional.

**QUE ES POSITION?**

Position es una propiedad fundamental en CSS que nos permite controlar la ubicación de los elementos dentro de una página web. Comprender cómo funciona esta propiedad es crucial para manipular el diseño de nuestros proyectos de manera efectiva. Todos los elementos HTML vienen con position: static por defecto, lo que significa que se mantienen en el lugar asignado originalmente en el flujo del documento, y no se moverán aunque hagamos scroll en la página.

Existen otras propiedades de position, como absolute, relative, fixed y sticky, que ofrecen comportamientos distintos y nos permiten crear diseños más dinámicos. Explorar estas propiedades y aprender a utilizarlas nos permite agregar interactividad y estilo a nuestros diseños.

**QUÉ DIFERENCIAS HAY ENTRE STATIC, ABSOLUTE, RELATIVE?**

**static**: Es la posición por defecto de todos los elementos. No permite modificar la posición del elemento en relación a su posición original en el documento.

**absolute**: Esta propiedad permite a un elemento posicionarse respecto al contenedor posicionado más cercano, lo que significa que puede salir del flujo normal del documento y aparecer sobre otros elementos.

**relative**: Permite desplazar un elemento desde su posición original sin alterar el flujo del documento. Puedes especificar el desplazamiento usando propiedades como top, bottom, left y right

**QUE ES EL DISPLAY EN CSS?**

El concepto de display en CSS es fundamental para entender cómo se renderizan los elementos HTML en un navegador. El display determina cómo se muestra un elemento y cómo interactúa con otros en la página. Las propiedades más comunes de display son block, inline, y inline-block.

**¿CÓMO FUNCIONA EL DISPLAY BLOCK?**

El display block es uno de los valores de display más utilizados. Cuando un elemento tiene display block, este ocupa todo el ancho disponible, sin importar el contenido que tenga dentro.

Características de display block:

* Ocupa el 100% del espacio horizontal disponible.
* Siempre comienza en una nueva línea.
* Se pueden aplicar propiedades de margin, padding, width, y height.

**¿QUÉ ES DISPLAY INLINE?**

El display inline es el valor predeterminado para muchos elementos de texto, como span o a. A diferencia de los elementos que usan display block, los elementos inline solo ocupan el espacio que requiere su contenido.

Características de display inline:

* No comienzan en una nueva línea.
* Ocupan solo el espacio necesario para el contenido.
* No se pueden aplicar width ni height.
* Solo se pueden aplicar margin y padding en los lados izquierdo y derecho, no arriba ni abajo.

**DISPLAY INLINE-BLOCK**

El display inline-block es una combinación poderosa de los modelos block e inline. Permite que los elementos se alineen al mismo tiempo que se aplican las características de los elementos block.

Características del display inline-block:

* Se comporta como un elemento inline.
* Permite aplicar margin, padding, width, y height.
* Permite que otros elementos se posiciones en línea horizontalmente si hay espacio suficiente.

**RECOMENDACIONES**

**Formato adecuado del lay-out:** Elige el modelo de display adecuado según el resultado visual que busques. Considere block para estructuras completas y inline para elementos en línea de texto.

**Mezcla propiedades con inline-block**: Aprovecha las ventajas de inline-block para ganar flexibilidad con las propiedades de tamaño y alineación.

**Modelo de caja en acción**: Familiarízate con el modelo de caja de CSS para manejar margin y padding con eficiencia. Esto te ayudará enormemente a diseñar elementos que se comporten y se alineen de manera correcta.

**Experimenta en tu navegador:** Utiliza herramientas de desarrollo en el navegador para ver en tiempo real cómo los cambios en las configuraciones de display afectan a los elementos de tu página.

**QUÉ ELEMENTOS DEBE INCLUIR EL LAYOUT?**

Al trabajar en un layout web, es importante distinguir las diferentes secciones que componen la página:

Header: Generalmente incluye el logo y la barra de navegación.

Sección principal: Área de contenido principal con texto e imágenes de ejemplo.

Sidebar: Espacio para elementos complementarios como noticias, blog posts o multimedia.

Footer: Contiene información básica de la empresa y enlaces a redes sociales.

**¿QUÉ ES EL DISPLAY FLEX?**

El uso de Display Flex, o Flexbox, en CSS es una herramienta esencial para el diseño web moderno. Ofrece una manera avanzada y más sencilla de posicionar elementos dentro de un contenedor. Esta técnica permite que los contenedores se comporten de forma flexible adaptándose automáticamente según el espacio disponible, lo que ayuda a crear diseños más responsivos y menos complicados.

Flexbox es especialmente útil si te has enfrentado a desafíos complicados tratando de alinear elementos con las propiedades de display y position. Flex viene al rescate y evita frustraciones médicas, simplificando así la vida de los diseñadores.

**¿CÓMO CAMBIAR LA ORIENTACIÓN DE LOS CONTENEDORES?**

Para cambiar la orientación de los contenedores y llevar el contenido al centro o bordes del contenedor padre, se puede usar Flexbox con propiedades específicas:

**Altura del contenedor principal:** Primero, es necesario establecer la altura del contenedor principal. Por ejemplo, asignar una altura mediante unidades relativas, como 50vh, asegurará que ocupe el 50% de la altura de la pantalla.

**Propiedad align-items**: Esta propiedad centraliza el contenido verticalmente con opciones como:

**center**: Alinea el contenido al centro del contenedor.

**flex-start**: Coloca el contenido en la parte superior.

**flex-end:** Envía el contenido al final del contenedor.

**PROPIEDADES DE FLEXBOX**

Flexbox también te permite manipular el crecimiento de los contenedores y su orden sin cambiar el HTML, lo que simplifica el diseño adaptativo.

**Propiedad align-items:** stretch: Se utiliza para estirar el contenido a llenar el espacio del contenedor padre. Esto funcionará únicamente si no se define una altura fija para los elementos hijos.

**Propiedad flex-grow:** Una de las características más útiles, permite ajustar el área en el que puede crecer un contenedor específico cuando hay espacio extra

**Propiedad order**: Permite cambiar la posición de los elementos hijos dentro de un contenedor. Un elemento con un orden más bajo se ubicará antes que los que tienen un orden más alto.

**FLEX-WRAP**

Cuando el espacio se vuelve limitado, Flexbox facilita la agrupación de elementos en múltiples líneas sin complicaciones adicionales en CSS, gracias al uso de la propiedad flex-wrap.

Efecto wrap: Al activar esta propiedad, tus contenedores se distribuirán automáticamente en nuevas líneas conforme sea necesario.

**PORQUE ELEGIR FLEXBOX PARA DISEÑO RESPONSIVE?**

Flexbox es una herramienta versátil para layouts escalables que se adaptan automáticamente a diferentes dispositivos y tamaños de pantalla. Su capacidad para redistribuir elementos sin necesidad de desmantelar el CSS subyacente lo hace ideal para el diseño responsive, permitiendo que los elementos se comporten de manera predecible al cambiar el tamaño del viewport.

**CREAR VARIABLES PERSONALIZADAS EN CSS?**

Las variables personalizadas en CSS son una herramienta poderosa que nos permite definir y reutilizar valores en nuestros estilos de manera eficiente. Aunque las variables pueden sonar más asociadas a matemáticas o a lenguajes de programación como JavaScript, CSS también tiene esta capacidad. El uso de variables nos permite almacenar valores comunes que pueden ser reutilizados, evitando la repetición de código.

**CUALES SON LOS BENEFICIOS DE USAR VARIABLES EN CSS?**

Los beneficios de usar variables en CSS son especialmente notables en proyectos más grandes. Algunas de las ventajas incluyen:

* **Mantenimiento más fácil:** Cambiar un valor en un solo lugar y ver los cambios reflejados en todo el proyecto.
* **Consistencia en el diseño:** Evita discrepancias de color y tamaño de fuente.
* **Reusabilidad**: Facilita la reutilización de estilos a través de diferentes módulos o componentes.

**WEB FONTS**

Las Web Fonts son grupos familiares de fuentes, los navegadores web poseen fuentes predeterminadas y dependiendo del mismo cada uno de ellos posee estilos diferentes.

**GENERIC FAMILIES**

**Serif**: son un tipo de fuente de estilo formal o clásico como Times New Roman.

**Sans-serif**: No tienen acabado en las puntas, como: Verdana.

**Cursive**: Son las que tienen estilo cursivo.

**Monospace**: Son tipos de fuentes con espaciado entre las letras, como: Roboto mono.

¿Como puedo saber que tipo de fuente tengo instaladas en mi navegador?

Menú>Configuración>Diseño>Personalizar Fuentes>Fuente Serif/Fuente Sans-serif

**CÓMO IMPORTAR FUENTES**

Ir a Google Fonts.

Seleccionar la fuente.

Seleccionar Estilo de fuente.

Agregar al proyecto, se considera buena práctica agregar las fuentes utilizando la etiqueta <link>, ya que la fuente cambia la fuente una vez que se haya cargado la página.

**CÓMO IMPLEMENTAR MEDIA QUERIES EN UN PROYECTO MOBILE FIRST CON FLEXBOX**

Comenzar a diseñar sitios web con un enfoque mobile first es esencial para asegurar una experiencia de usuario óptima en dispositivos móviles y escalar adecuadamente a tabletas y escritorios. Usar media queries en CSS permite ajustar las reglas de estilos según el tamaño de la pantalla (breakpoints). A continuación, veremos cómo implementar media queries en un proyecto usando Flexbox, asegurando un diseño adecuado en distintos dispositivos.

**BUENAS PRACTICAS MEDIA QUERIES**

Cuando trabajas con media queries en un archivo CSS, es importante seguir ciertas buenas prácticas:

Ubicar siempre al final del archivo: Coloca todos los media queries en la parte inferior de tu archivo CSS. Esto facilita que los miembros del equipo identifiquen rápidamente los lugares donde se encuentran los diferentes breakpoints.

Sintaxis básica de un media query: El comando básico para escribir una media query es el siguiente:

@media (min-width: 600px) {

/\* Reglas CSS aquí \*/

}

Usa min-width para mobile first: Al utilizar min-width, defines reglas que se aplican a partir de un cierto tamaño, por lo que comienzas optimizando para pantallas pequeñas y luego escalas hacia pantallas más grandes.

**EJEMPLO PRÁCTICO DE MEDIA QUERIES**

Aquí hay un ejemplo de cómo puedes ajustar el layout al ir pasando de un dispositivo móvil a una tableta usando media queries. Imaginemos que tienes contenedores que ocupan el 100% del ancho en dispositivos móviles y quieres cambiar esto en una tableta:

CSS Base para Mobile:

.container {

width: 100%;

}

Media Queries para Tabletas:

En el CSS, agrega la media query para dispositivos de 600 píxeles en adelante:

@media (min-width: 600px) {

.container-2, .container-3, .container-4 {

width: 50%;

}

}

**CÓMO IMPLEMENTAR VARIOS BREAKPOINTS EN UN PROYECTO?**

Además de ajustar el diseño al pasar de un móvil a una tableta, puedes definir más breakpoints para dispositivos aún más grandes. Por ejemplo, cuando el ancho alcanza 800 píxeles, puedes hacer más ajustes:

@media (min-width: 800px) {

.c1 {

width: 60%;

}

.c2 {

width: 40%;

}

.c3, .c4 {

width: 33%;

}

.c5 {

width: 34%;

}

}

Aquí se ve cómo los distintos elementos cambian su disposición en pantallas más grandes, adaptándose de manera fluida al tamaño del dispositivo.

**CÓMO MANEJAR LOS ESTILOS EN ARCHIVOS SEPARADOS?**

Una práctica recomendada es separar los estilos de los media queries en diferentes archivos CSS para cada tipo de dispositivo:

Creación de archivos CSS específicos: Crea archivos como tablet.css y desktop.css.

Enlace de archivos CSS con media queries:

<link rel="stylesheet" href="tablet.css" media="screen and (min-width: 600px)">

<link rel="stylesheet" href="desktop.css" media="screen and (min-width: 800px)">

Al hacer esto, mantienes el código organizado y permites que el navegador decida qué estilos cargar basándose en el tamaño del dispositivo. Aunque se descargan todos, el navegador prioriza los del tamaño actual y carga los otros después.

El enfoque del proyecto mostly fluid permite que tu sitio sea adaptable, reorientando elementos según el dispositivo, pero permaneciendo limitado para un control mayor del diseño.

**¿QUÉ ES EL PATRÓN LAYOUT SHIFTER?**

El diseño web Mobile First es una metodología que prioriza la experiencia del usuario en dispositivos móviles. Uno de los patrones que podemos utilizar en este enfoque es el Layout Shifter. Este patrón permite que, al aumentar el tamaño de la pantalla, los bloques o contenedores se reposicionen, ofreciendo así un diseño adaptativo. A continuación, te explicaré cómo construirlo utilizando HTML y CSS.

**¿CÓMO CONSTRUIR UN LAYOUT CON HTML Y CSS?**

Para implementar un Layout Shifter, es esencial empezar con una estructura básica en HTML, donde los contenedores se organizan de forma jerárquica. Usaremos CSS para aplicar propiedades de flexbox que nos ayudarán a reposicionar elementos conforme el tamaño de pantalla cambia.

**CÓMO IMPLEMENTAR MEDIA QUERIES PARA CAMBIOS DE DISEÑO?**

Los Media Queries permiten que el diseño cambie cuando se detecta un tamaño de pantalla específico. Configura puntos de quiebre que se adapten al tipo de dispositivos en los que se visualizará tu diseño.

**RECOMENDACIONES**

* Agrega más contenedores y experimenta con la posición y el contenido según los breakpoints.
* Introduce texto, imágenes o videos dentro de tus contenedores para un diseño más realista.
* Usa esta técnica para crear layouts únicos que ofrezcan experiencias optimizadas tanto en móviles como en computadoras de escritorio.

**¿QUÉ ES EL PATRÓN DE DISEÑO COLUMN DROP?**

En el diseño web, es crucial que nuestras páginas se adapten correctamente a diferentes tamaños de pantalla. El patrón de diseño "column drop" es una técnica que nos ayuda a lograr una transición fluida desde un diseño móvil hasta uno de escritorio. Este enfoque se enfoca en reorganizar los contenedores al aumentar el tamaño del viewport, posicionando las columnas de manera vertical al principio, y luego distribuyéndolas horizontalmente según la necesidad del contenido.

**COMO SE IMPLEMENTA?**

La implementación del patrón column drop en CSS es sencilla y efectiva, utilizando contenedores flexibles para cambiar la orientación de sus elementos a medida que el tamaño de la pantalla cambia.

**CONSEJOS**

**Haz pruebas en múltiples dispositivos:** Es fundamental probar cómo se comporta tu diseño en diferentes dispositivos para asegurar que la adaptabilidad sea la correcta.

**Combina con otras tecnologías:** Aunque este patrón puede manejarse principalmente con CSS, considera utilizar JavaScript para manipular elementos más complejos o contenido dinámico.

**Considera los aspectos visuales**: Al reposicionar elementos, especialmente si contienen imágenes o multimedia, asegúrate de que su calidad visual se mantenga.

**¿CUÁLES SON LAS MEJORES PRACTICAS PARA LA CARGA RÁPIDA DE PROYECTOS EN MOBILE FIRST?**

Cuando se trata de optimizar un proyecto siguiendo el enfoque mobile first, es esencial garantizar que la carga de la página sea rápida y eficiente. Aquí te dejo algunos consejos valiosos para lograrlo:

**Divide el código CSS**: Evita cargar todos los media queries en un solo archivo CSS, ya que esto puede aumentar considerablemente el tamaño del archivo y, por ende, afectar la velocidad de carga. A medida que empieces a implementar breakpoints o media queries, divídelos en diferentes archivos según sea necesario y cárgalos de manera selectiva.

**Utiliza un archivo CSS dedicado al mobile**: Comienza con un archivo base como style.css o mobile.css, que contenga lo esencial para que el diseño móvil se visualice correctamente.

**Selecciona los breakpoints adecuados**: No es necesario adaptar tu diseño a todos los dispositivos existentes, sino a aquellos más comunes. Por ejemplo, las dimensiones comunes para móviles empiezan en 320 píxeles de ancho. Más allá de eso, 375 y 414 píxeles suelen ser buenos puntos de inicio para móviles.

**CÓMO IDENTIFICAR DIMENSIONES IMPORTANTES PARA BREAKPOINTS?**

La elección adecuada de breakpoints es crucial para un diseño eficiente y funcional en diversos dispositivos. Una herramienta útil es el sitio web "my device io", donde puedes encontrar datos sobre los viewports de diferentes dispositivos:

**Ancho mínimo móvil:** Los dispositivos móviles más pequeños generalmente tienen un ancho de 320 píxeles, por lo que este debería ser tu punto de inicio.

**Variedad en las dimensiones:** Además de 320 píxeles para móviles, considera 768 y 800 píxeles para tablets. No te preocupes por demasiados breakpoints, entre 3 y 4 será suficiente para gestionar un código eficiente y fácil de mantener.

**PATRON DE DISEÑO FLUID LAYOUT: DROPBOX**

Dropbox utiliza un patrón fluido donde todo el contenido se presenta inicialmente en un 100% en sentido vertical. A medida que el viewport se adapta, ciertos contenedores limitan su crecimiento ajustando el contenido vertical automáticamente.

**PATRÓN DE DISEÑO LAYOUT SHIFTER: GITHUB**

GitHub emplea el patrón de layout shifter. En dispositivos móviles, el menú cambia drásticamente en su UI, moviendo elementos clave, como el formulario de registro, según el tamaño del viewport. Este tipo de patrón permite reorganizar el contenido para que se adapte óptimamente en cada tamaño de pantalla.

**PATRON DE DISEÑO COLUMN DROP: SMASHING MAGAZINE**

Smashing Magazine utiliza el patrón column drop, donde el contenido se ajusta de tres columnas a dos, y finalmente a una sola columna conforme se reduce el viewport. Este patrón es útil para manejar layouts con diferentes niveles de contenido, utilizando CSS Grid Layout para la implementación.

**SEMANTICA**

La semántica está relacionada con las etiquetas contenedoras en HTML5, por ejemplo: header, main, sidebar y footer. Estas agregan información importante para aquellos que tengan problemas con la visualización de la página. Les permite a estos usuarios orientarse en qué sección de la página se encuentran.

Por eso, es importante utilizar las diferentes etiquetas que HTML5 ofrece para tener la mejor semántica posible y la accesibilidad.

**PORQUE ES IMPORTANTE LA ACCESIBILIDAD EN EL TEXTO?**

Garantizar la accesibilidad en el contenido textual de una página web es esencial para asegurar que todas las personas, independientemente de sus capacidades visuales, puedan interactuar y consumir el contenido de manera efectiva. La accesibilidad no depende únicamente de elegir fuentes atractivas o sencillas, sino de cómo se mide y se ajusta ese texto para que sea escalable según las necesidades del usuario.

**DESVENTAJAS DE MEDIDAS ABSOLUTAS COMO PIXELES**

Cuando se establecen tamaños de fuente en píxeles, se está utilizando una medida absoluta. Esto significa que, independientemente de la configuración del usuario o de las herramientas del navegador para aumentar el tamaño del texto, el contenido no se ajustará. Esta rigidez puede afectar enormemente a las personas con problemas visuales, ya que no podrán ampliar el texto para facilitar su lectura.

**QUÉ PROBLEMAS ENFRENTA UN USUARIO CON LIMITACIONES VISUALES?**

**Imposibilidad de ajuste:** Con medidas en píxeles, el texto permanece estático y no responde a las configuraciones de accesibilidad del usuario.

**Experiencia deficiente:** Los usuarios deben encontrar alternativas, como acercar el contenido con la función de zoom, lo que puede resultar incómodo.

**Inclusión limitada:** No todos los usuarios pueden interactuar equitativamente, reduciendo la accesibilidad.

**CÓMO PUEDEN MEJORAR LA ACCESIBILIDAD LAS MEDIDAS RELATIVAS COMO REM?**

El uso de medidas relativas como rem resuelve las limitaciones de los píxeles ofreciendo flexibilidad en el texto. Rem es una unidad que se basa en el tamaño de la raíz html. Esto permite que el texto se escale de acuerdo con las preferencias del usuario, haciendo las interfaces más inclusivas.

**BENEFICIOS DE UTILIZAR REM**

**Escalabilidad:** Permite que el texto se ajuste dinámicamente con las configuraciones del navegador.

**Accesibilidad universal:** Ofrece a las personas con problemas visuales la capacidad de adaptar el texto a sus necesidades específicas.

**Flexibilidad de diseño**: Permite crear interfaces que mantienen la consistencia visual sin sacrificar la accesibilidad.