Repaso para el Parcial

📚 2026-01 — ⏱️ 45 min.

Introducción al Frontend

Front-end vs Back-end

Front-end (Client-side):

- Aquello que el usuario **ve** y manipula
- Se ejecuta en el navegador del usuario
- "El cliente"

Back-end (Server-side):

- Aquello que **NO** está bajo control del usuario
- Maneja la lógica de negocio y los datos
- "El servidor"

El front-end interactúa con el back-end para obtener y enviar datos.

Front-end Moderno

Tecnologías fundamentales:

📄 **HTML** → Estructura

🎨 **CSS** → Estilo

⚡ **JavaScript** → Interactividad

Librería vs Framework

Librería:

- Solo proporciona herramientas específicas (UI, routing, estado, etc.)
- Tú eliges las herramientas adicionales
- Mayor flexibilidad
- Ejemplos: React, Vue, Svelte

Framework:

- Incluye todo lo necesario
- Estructura predefinida
- Más opinado sobre arquitectura
- Ejemplos: Angular, Next.js, Nuxt, SvelteKit

HTML - Conceptos Fundamentales

¿Qué es HTML?

- Lenguaje de **marcado** estándar para crear páginas web
- Define la **estructura** y el **contenido**
- **NO** es un lenguaje de programación
- Versión actual: **HTML5** (2014)

Estructura Básica de HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Página</title>
</head>
<body>
    <!-- Contenido aquí -->
</body>
</html>
```

Elementos HTML

Un elemento HTML consta de:

- **Etiqueta de apertura**: `<p>`
- **Contenido**: El texto o elementos internos
- **Etiqueta de cierre**: `</p>`

```
<p>Este es un párrafo</p>
```

Elementos auto-cerrados:

```
<img src="imagen.jpg" alt="Descripción">
<br>
<input type="text">
```

Atributos HTML Comunes

```
<a href="https://example.com" target="_blank" class="link">
    Enlace
</a>
```

Atributos importantes:

- `id` : Identificador único
- `class` : Clase para estilos
- `style` : Estilos en línea
- `title` : Texto de ayuda
- `data-*` : Atributos personalizados

Elementos HTML Importantes

Encabezados: `<h1>` a `<h6>` (solo un `<h1>` por página)

Texto: `<p>`, `<strong>`, `<em>`, `<mark>`, `<small>`

Enlaces: `<a href="url">texto</a>`

Imágenes: `<img src="ruta" alt="descripción">`

Listas: `<ul>`, `<ol>`, `<li>`

Contenedores: `<div>` (bloque), `<span>` (en línea)

HTML Semántico

HTML semántico usa etiquetas que **describen su significado**.

```
<header>...</header>    <!-- Encabezado -->
<nav>...</nav>          <!-- Navegación -->
<main>...</main>        <!-- Contenido principal -->
<article>...</article>  <!-- Contenido independiente -->
<section>...</section>  <!-- Sección temática -->
<aside>...</aside>      <!-- Contenido relacionado -->
<footer>...</footer>    <!-- Pie de página -->
```

Beneficios: Mejor accesibilidad, SEO, legibilidad y mantenimiento.

No Semántico vs Semántico

❌ **No semántico (malo):**

```
<div id="header">
    <div id="nav">...</div>
</div>
<div id="content">...</div>
<div id="footer">...</div>
```

✅ **Semántico (bueno):**

```
<header>
    <nav>...</nav>
</header>
<main>...</main>
<footer>...</footer>
```

Formularios Básicos

```html
<form action="/submit" method="POST">
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" name="nombre" required>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>

    <button type="submit">Enviar</button>
</form>
```

Atributos: `action` (URL destino), `method` (GET o POST)

Siempre usar `<label>` para accesibilidad.

Tipos de Input Comunes

```
<input type="text">      <!-- Texto simple -->
<input type="email">     <!-- Email con validación -->
<input type="password">  <!-- Contraseña oculta -->
<input type="number">    <!-- Número -->
<input type="date">      <!-- Fecha -->
<input type="checkbox">  <!-- Casilla de verificación -->
<input type="radio">     <!-- Botón de opción -->
<input type="file">      <!-- Subir archivo -->

<textarea></textarea>    <!-- Texto multilínea -->
<select>...</select>     <!-- Lista desplegable -->
```

Validación HTML5

```
<input type="email" required>
<input type="number" min="1" max="10" required>
<input type="text" pattern="[A-Za-z]{3,}" required>
<input type="text" minlength="3" maxlength="20">
```

Atributos de validación:

- `required` : Campo obligatorio
- `pattern` : Expresión regular
- `min/max` : Valores mínimo/máximo
- `minlength/maxlength` : Longitud

Mejores Prácticas HTML

- ✅ Usar **HTML semántico** (evitar `<div>` genéricos)
- ✅ Solo **un** `<h1>` **por página**
- ✅ Usar **atributo** `alt` en todas las imágenes
- ✅ Usar `<label>` **con todos los inputs**
- ✅ Estructura jerárquica de encabezados (h1 → h2 → h3)
- ✅ **Indentar correctamente** para legibilidad
- ✅ **Cerrar todas las etiquetas** en el orden correcto

CSS - Conceptos Fundamentales

¿Qué es CSS?

CSS = Cascading Style Sheets

- Define el **estilo visual** de las páginas web
- Controla colores, fuentes, espaciado, layout, animaciones
- **Separa** contenido (HTML) de presentación (CSS)

Beneficios:

- ✅ Separación de responsabilidades
- ✅ Reutilización de estilos
- ✅ Mantenimiento más fácil
- ✅ Diseño responsivo

Sintaxis Básica

```
selector {
    propiedad: valor;
    otra-propiedad: otro-valor;
}
```

Ejemplo:

```
p {
    color: blue;
    font-size: 16px;
    margin: 10px;
}
```

Selectores Básicos

```css
/* Selector de elemento */
p { color: blue; }

/* Selector de clase */
.mi-clase { color: red; }

/* Selector de ID */
#mi-id { color: green; }

/* Selector universal */
* { margin: 0; }

/* Pseudo-clases */
a:hover { color: red; }
input:focus { border-color: blue; }
```

La Cascada y Especificidad

La **cascada** determina qué estilos se aplican cuando hay conflictos.

Orden de prioridad:

1. **Importancia**: `!important` (⚠️ evitar)
2. **Especificidad**: IDs > Clases > Elementos
3. **Orden de código**: El último estilo gana

Especificidad:

```
p { color: blue; }              /* 0,0,1 */
.texto { color: red; }          /* 0,1,0 - GANA */
#principal { color: green; }    /* 1,0,0 - GANA */
```

Propiedades CSS Comunes

Colores:

```
color: red;                    /* Nombre */
color: #ff0000;                /* Hexadecimal */
color: rgb(255, 0, 0);         /* RGB */
color: rgba(255, 0, 0, 0.5);   /* RGBA con transparencia */
```

Tipografía:

```
font-family: Arial, sans-serif;
font-size: 16px;
font-weight: bold;             /* normal, bold, 700 */
line-height: 1.5;
text-align: center;
```

Box Model

El Modelo de Caja

Cada elemento HTML es una caja con:

1. **Content**: El contenido (texto, imagen, etc.)
2. **Padding**: Espacio interno (entre contenido y borde)
3. **Border**: El borde de la caja
4. **Margin**: Espacio externo (entre cajas)

```
div {
    width: 300px;
    padding: 20px;
    border: 2px solid black;
    margin: 10px;
}
```

Ancho total = width + padding + border + margin

Box Sizing

```css
/* Por defecto (content-box) */
div {
    box-sizing: content-box;
    width: 300px;      /* Solo el contenido */
    padding: 20px;     /* Se suma al width */
}

/* Border-box (recomendado) */
div {
    box-sizing: border-box;
    width: 300px;      /* Incluye padding y border */
    padding: 20px;     /* No se suma al width */
}
```

Mejor práctica: `* { box-sizing: border-box; }`

Padding y Margin - Formas Cortas

```css
/* 4 valores (arriba, derecha, abajo, izquierda) */
padding: 10px 20px 10px 20px;

/* 2 valores (vertical, horizontal) */
padding: 10px 20px;

/* 1 valor (todos los lados) */
padding: 10px;

/* Individual */
padding-top: 10px;
padding-right: 20px;
```

Lo mismo aplica para `margin`.

Position

```css
/* Relativo (relativo a posición original) */
position: relative;
top: 10px;

/* Absoluto (relativo al ancestro posicionado) */
position: absolute;

/* Fijo (relativo al viewport) */
position: fixed;

/* Sticky (híbrido relativo-fijo) */
position: sticky;
top: 0;
```

Z-Index: Controla el orden de apilamiento (valores más altos = más cerca).

Flexbox

Flexbox - Introducción

Flexbox es un modelo de layout **unidimensional** (fila o columna).

Perfecto para alinear elementos y distribuir espacio.

```
.container {
    display: flex;
}
```

- El contenedor → **flex container**
- Los hijos directos → **flex items**

Flexbox - Propiedades del Container

```
.container {
    display: flex;

    flex-direction: row | column;

    /* Justificación */
    justify-content: flex-start | center | flex-end |
                     space-between | space-around;

    /* Alineación */
    align-items: flex-start | center | flex-end | stretch;

    /* Permitir salto de línea */
    flex-wrap: nowrap | wrap;
}
```

Flexbox - Propiedades de Items

```css
.item {
    /* Crecer para llenar espacio */
    flex-grow: 1;

    /* Encogerse si es necesario */
    flex-shrink: 1;

    /* Tamaño base */
    flex-basis: 200px;

    /* grow shrink basis */
    flex: 1 1 200px;
}
```

Flexbox - Ejemplos Comunes

Centrar elemento:

```
.container {
    display: flex;
    justify-content: center;
    align-items: center;
}
```

Distribución uniforme:

```
.container {
    display: flex;
    justify-content: space-between;
}
```

Items de igual tamaño:

```
.item { flex: 1; }
```

CSS Grid

Grid - Introducción

Grid es un modelo de layout **bidimensional** (filas y columnas).

Perfecto para layouts complejos y estructuras de página.

```
.container {
    display: grid;
}
```

- El contenedor → **grid container**
- Los hijos directos → **grid items**

Grid - Definir Estructura

```css
.container {
    display: grid;

    /* 3 columnas */
    grid-template-columns: 200px 200px 200px;

    /* Usando repeat() */
    grid-template-columns: repeat(3, 200px);

    /* Fracciones (fr) */
    grid-template-columns: 1fr 2fr 1fr;

    /* Espaciado */
    gap: 20px;
    column-gap: 20px;
    row-gap: 20px;
}
```

Grid - Colocar Items

```css
.item {
    /* Empezar en columna 1, terminar en 3 */
    grid-column: 1 / 3;

    /* Ocupar 2 columnas */
    grid-column: 1 / span 2;

    /* Ocupar toda la fila */
    grid-column: 1 / -1;

    /* Filas */
    grid-row: 1 / 3;
}
```

Grid - Template Areas

```css
.container {
    display: grid;
    grid-template-areas:
        "header header header"
        "sidebar main main"
        "footer footer footer";
    grid-template-columns: 200px 1fr 1fr;
}

.header  { grid-area: header; }
.sidebar { grid-area: sidebar; }
.main    { grid-area: main; }
.footer  { grid-area: footer; }
```

Forma visual e intuitiva de definir layouts.

Diseño Responsivo

Media Queries

Aplican estilos basados en características del dispositivo:

```
/* Para pantallas de 768px o menos */
@media (max-width: 768px) {
    .container {
        flex-direction: column;
    }
}

/* Para pantallas de 769px o más */
@media (min-width: 769px) {
    .container {
        flex-direction: row;
    }
}
```

Breakpoints Comunes - Mobile First

```css
/* Mobile (por defecto) */
.container { padding: 10px; }

/* Tablet */
@media (min-width: 768px) {
    .container { padding: 20px; }
}

/* Desktop */
@media (min-width: 1024px) {
    .container { padding: 30px; }
}

/* Large Desktop */
@media (min-width: 1440px) {
    .container { padding: 40px; }
}
```

Unidades Responsivas

```
/* Porcentaje (relativo al padre) */
width: 50%;

/* Viewport */
width: 50vw;   /* 50% del ancho del viewport */
height: 100vh; /* 100% de la altura del viewport */

/* REM (relativo al root - html) */
font-size: 1.5rem;

/* EM (relativo al padre) */
padding: 2em;

/* Funciones modernas */
width: min(500px, 100%);
width: clamp(300px, 50%, 800px);
```

Imágenes Responsivas

```css
/* Básico */
img {
    max-width: 100%;
    height: auto;
}

/* Control de ajuste */
img {
    width: 300px;
    height: 200px;
    object-fit: cover | contain;
}
```

CSS Variables y BEM

CSS Variables (Custom Properties)

```css
/* Definir variables */
:root {
    --primary-color: #007bff;
    --spacing-unit: 8px;
}

/* Usar variables */
.button {
    background-color: var(--primary-color);
    padding: var(--spacing-unit);
}

/* Con valor por defecto */
.element {
    color: var(--undefined-color, black);
}
```

Ventaja: Cambiar valores en un solo lugar.

BEM - Block Element Modifier

BEM es una metodología de nombrado de clases CSS.

Estructura:

- **Block**: Componente independiente (`.card`)
- **Element**: Parte del bloque (`.card__title`)
- **Modifier**: Variación (`.card--featured`)

```
/* Block */
.card { }

/* Elements */
.card__header { }
.card__title { }

/* Modifiers */
.card--featured { }
.card__title--large { }
```

BEM - Ejemplo en HTML

```html
<div class="card card--featured">
    <div class="card__header">
        <h2 class="card__title card__title--large">Título</h2>
    </div>
    <div class="card__body">
        <p class="card__text">Contenido</p>
    </div>
    <button class="card__button card__button--primary">
        Acción
    </button>
</div>
```

Ventajas: Nombres descriptivos, evita conflictos, código escalable.

Transiciones y Animaciones

Transiciones

Animan cambios de propiedades CSS:

```css
.button {
    background-color: blue;
    transition: background-color 0.3s ease;
}

.button:hover {
    background-color: red;
}

/* Múltiples propiedades */
.element {
    transition:
        opacity 0.3s ease,
        transform 0.5s ease;
}
```

Transform

```css
/* Trasladar */
transform: translate(50px, 100px);
transform: translateX(50px);

/* Escalar */
transform: scale(1.5);

/* Rotar */
transform: rotate(45deg);

/* Combinar */
transform: translate(50px) rotate(45deg) scale(1.2);
```

Animaciones con Keyframes

```css
/* Definir animación */
@keyframes slide-in {
    0% {
        transform: translateX(-100%);
        opacity: 0;
    }
    100% {
        transform: translateX(0);
        opacity: 1;
    }
}

/* Usar animación */
.element {
    animation: slide-in 0.5s ease-out;
}
```

Mejores Prácticas - Resumen

HTML - Mejores Prácticas

- ✅ Usar HTML **semántico** (`<header>` , `<nav>` , `<main>` , etc.)
- ✅ Solo un `<h1>` **por página**
- ✅ Siempre usar `alt` **en imágenes**
- ✅ Siempre usar `<label>` **con inputs**
- ✅ **Estructura jerárquica** de encabezados
- ✅ **Validar** el HTML (W3C Validator)

CSS - Mejores Prácticas

- ✅ Usar **nombres de clase descriptivos** (BEM)
- ✅ Usar **variables CSS** para valores reutilizables
- ✅ **Evitar** `!important`
- ✅ **Mobile First** con media queries
- ✅ Preferir **Flexbox y Grid** para layouts
- ✅ Usar `box-sizing: border-box` globalmente
- ✅ Mantener **especificidad baja**

Conceptos Clave para el Examen

Checklist de Conceptos - Frontend

- 📌 **Diferencia entre Front-end y Back-end**
- 📌 **Qué es una librería vs un framework**
- 📌 **React es una librería**
- 📌 **Tecnologías del frontend moderno**: HTML, CSS, JavaScript
- 📌 **Modelo MVC**: Vista (Frontend), Modelo (Backend), Controlador

Checklist de Conceptos - HTML

📌 **Estructura básica** de un documento HTML

📌 **Elementos HTML**: sintaxis, atributos, anidamiento

📌 **HTML semántico**: `<header>`, `<nav>`, `<main>`, `<article>`, etc.

📌 **Formularios**: estructura, tipos de input, validación

📌 **Diferencia entre** `<div>` **y** `<span>` (bloque vs en línea)

📌 **Mejores prácticas**: accesibilidad, SEO, semántica

Checklist de Conceptos - CSS Básico

📌 **Sintaxis CSS**: selector, propiedad, valor

📌 **Selectores**: elemento, clase, ID, pseudo-clases

📌 **Cascada y especificidad**: orden de prioridad

📌 **Box Model**: content, padding, border, margin

📌 **Box-sizing**: `content-box` vs `border-box`

📌 **Display**: block, inline, inline-block, none

📌 **Position**: static, relative, absolute, fixed, sticky

Checklist de Conceptos - CSS Avanzado

- 📌 **Flexbox**: container y item properties, cuándo usar
- 📌 **Grid**: estructura bidimensional, template areas
- 📌 **Diseño responsivo**: media queries, breakpoints
- 📌 **Unidades responsivas**: %, vw, vh, rem, em
- 📌 **CSS Variables**: definición y uso
- 📌 **BEM**: metodología de nombrado
- 📌 **Transiciones y transforms**

Flexbox vs Grid - ¿Cuándo usar cada uno?

Flexbox (unidimensional):

- ✅ Alinear elementos en una dirección (fila o columna)
- ✅ Distribución de espacio entre items
- ✅ Navegación, botones, componentes pequeños

Grid (bidimensional):

- ✅ Layouts complejos con filas y columnas
- ✅ Estructuras de página completas
- ✅ Galerías, dashboards

Ambos se pueden combinar: Grid para el layout general, Flexbox dentro de grid items.

Consejos para el Examen

- 💡 **Practica escribir código** sin autocomplete
- 💡 **Entiende la diferencia** entre conceptos clave (librería vs framework, flexbox vs grid)
- 💡 **Recuerda la sintaxis básica** de HTML y CSS
- 💡 **Conoce las propiedades comunes** de Flexbox y Grid
- 💡 **Entiende HTML semántico** y por qué es importante
- 💡 **Repasa especificidad** y la cascada de CSS

Recursos Adicionales

Documentación:

- MDN Web Docs: https://developer.mozilla.org
- HTML Standard: https://html.spec.whatwg.org
- CSS Tricks: https://css-tricks.com

Práctica Interactiva:

- Flexbox Froggy: https://flexboxfroggy.com
- Grid Garden: https://cssgridgarden.com

Validación:

- W3C Validator: https://validator.w3.org
- Can I Use: https://caniuse.com