```
<p>Esto es un párrafo</p>
<div class="layer">Esto es una capa</div>
```

```
<label for="nombre">Nombre</label>
<input id="nombre" placeholder="Inserte el nombre completo" />
```

HTML5

```
<!DOCTYPE html>
```

MathML 2.0

```
<!DOCTYPE math PUBLIC "-//W3C//DTD MathML 2.0//EN"
                      "http://www.w3.org/TR/MathML2/dtd/mathml2.dtd">
```

SVG 1.1 Full

```
<!DOCTYPE svg PUBLIC
          "-//W3C//DTD SVG 1.1//EN"
          "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

SVG 1.1 Básico

```
<!DOCTYPE svg PUBLIC
          "-//W3C//DTD SVG 1.1 Basic//EN"
          "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-basic.dtd">
```

SVG 1.1 Reducido

```
<!DOCTYPE svg PUBLIC
          "-//W3C//DTD SVG 1.1 Tiny//EN"
          "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-tiny.dtd">
```

XHTML1.1

```
<!DOCTYPE html PUBLIC
          "-//W3C//DTD XHTML 1.1//EN"
          "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<html lang="es">...</html>
```

1.1.1. Etiqueta head

```
<head>
    <!-- Información del documento -->
</head>
```

1.1.2. Etiqueta body

```
<body>
    <!-- Contenido del cuerpo de la página -->
</body>
<!-- Esto es un comentario de HTML -->
```

Elemento base

```
<head>
    <base href="https://www.ejemplo.com/" target="_blank" />
</head>
```

Elemento link

```
<link rel="stylesheet" type="text/css" href="custom.css" />
```

```
<meta name="author" content="Pablo E. Fernández Casado">
```

EL ATRIBUTO CONTENT

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

EL ATRIBUTO CHARSET

```
  <meta charset="UTF-8">
```

EL ATRIBUTO HTTP-EQUIV

```
<meta http-equiv="refresh" content="60">
```

EL ATRIBUTO SCHEME

```
<meta name="date" content="01-01-2020" scheme="DD-MM-YYYY">
<meta name="identifier" content="0-2345-6634-6" scheme="ISBN">
```

Elemento title

```
<title>Curso de creación de páginas web</title>
```

Elemento style

```
<style media="screen" type="text/css">
    html{
```

```
        font-family: Arial, sans-serif;
        font-size: 14px;
        font-style: normal;
    }
</style>
```

Elemento script

```
script src="./validarNIF.js"></script>
```

Principales metadatos a definir

```
<meta name="viewport" content="width=device-width, initial-scale=1 />
```

Atributo accesskey

```
<button accesskey="h">Ayuda</button>
```

Atributo autocapitalize

```
<label>
    Nombre:
    <input type="text" name="name" autocapitalize="words">
</label>
```

Atributo autofocus

```
<label for"name">Nombre:</label>
<input type="text" id="name" name="name" autofocus>
<p class="toLeft">
    Este párrafo tiene una clase que hace que se alineé a la izquierda
</p>

<p class="toRight">
    Este párrafo tiene una clase que hace que se alineé a la derecha
</p>
```

Atributo contenteditable

```
<p contentEditable="true">
    Este párrafo es modificable,
    <span>y esta etiqueta también.</span>
```

```
</p>
```

Atributo draggable

```
<p draggable="true">Este es un párrafo arrastable</p>
```

1.1.3. Atributo dir

```
<p dir="rtl">
    Texto legible de derecha a izquierda para idiomas como el árabe.
</p>
```

1.1.4. Atributo enterkeyhint

```
<input enterkeyhint="search">
```

1.1.5. Atributo hidden

```
<p hidden>Este párrafo permanecerá oculto hasta nueva orden.</p>
```

1.1.6. Atributo inputmode

```
<div contenteditable="" inputmode="search"></div>
```

```
<label>
    Nombre completo
    <input id="name" />
</label>
```

1.1.7. Atributo lang

```
<p lang="es">Mi nombre es Pablo</p>
<p lang="en">My name is Paul</p>
```

```
<link rel="search" href="booksearch.asp"
      nonce="bGlicm8tY29uc3J0dWNjacOzbi1kZS1wYWdpbmFzLXdlYg==">
```

```
Content-Security-Policy: script-src 'nonce-
bGlicm8tY29uc3J0dWNjacOzbi1kZS1wYWdpbmFzLXdlYg=='
```

1.1.8. Atributo slot

```
<span slot="book">Domine JavaScript 4ª Edición</span>
```

```
<h1 style="margin-top: 5px;">Título con margen superior</h1>
```

1.1.9. Atributo tabindex

```
<p tabindex="0">
    Este párrafo puede tomar el foco
</p>
```

1.1.10. Atributo title

```
<p title="Esto es un párrafo de prueba!">
    Esto es un párrafo con un mensaje emergente
</p>
```

1.1.11. Atributo translate

```
<p>Visita www.<span translate="no">Islavisual</span>.com</p>
```

1.1.12. Atributos personalizados

```
<p id="p1" data-id="002">
    Esto es un párrafo con un atributo personalizado
</p>
```

```
<script type="text/javascript">
    document.querySelector("#p1").dataset.id;
</stript>
```

1.1.13. Elemento article

```
<article>
    <header>
        <h2>HTML5</h2>
    </header>

    <div>
        <p> El lenguaje HTML5 (HyperText Markup Language Versión 5) es un
lenguaje de marcado de hipertexto que está vigente desde el año 2014 y
puede ser utilizado para...</p>
    </div>

    <footer>
        <a href="html-usos.html">Seguir leyendo</a>
    </footer>
</article>
```

1.1.14. Elemento aside

```
<aside>
    <h3>Artículos relaccionados</h3>
    <ul>
        <li><a href="#">Artículo 1</a></li>
        <li><a href="#">Artículo 2</a></li>
        <li><a href="#">Artículo 3</a></li>
    </ul>
</aside>
```

1.1.15. Elemento div

```
<div>
    <p>Esto puede ser un texto descriptivo sobre HTML5</p>
    <img src="./html5.png" />
</div>
```

1.1.16. Elemento footer

```
<footer>
    <ul>
        <li>Copyright ©2020</li>
        <li>Polícita de Privacidad</li>
        <li>Versión para móviles</li>
    </ul>
</footer>
```

1.1.17. Elementos h1..h6

```
<h1>Esto es un encabezado de nivel 1</h1>
<h2>Esto es un encabezado de nivel 2</h2>
<h3>Esto es un encabezado de nivel 3</h3>
<h4>Esto es un encabezado de nivel 4</h4>
<h5>Esto es un encabezado de nivel 5</h5>
<h6>Esto es un encabezado de nivel 6</h6>
```

1.1.18. Elemento header

```
<article>
    <header>
        <h2>HTML5</h2>
    </header>
    ...
</article>
```

1.1.19. Elemento hgroup

```
<hgroup>
    <h1>Fast and Furious 4</h1>
    <h2>Aún más rápido</h2>
</hgroup>
```

1.1.20. Elemento main

```
<main>
    <h1>CSS</h1>
    <p>CSS es un lenguaje de marcado para proveer estilos al contenido.</p>
```

```
    <!—Más contenidos -->

    <aside>Otros contenidos</aside>
</main>
```

1.1.21. Elemento nav

```
<nav>
    <ul class="nav navbar-nav navbar-right">
        <li><a href="#home">Inicio</a></li>
        <li><a href="#about">Acerca de Nosotros</a></li>
        <li><a href="#features">Servicios</a></li>
        <li><a href="#blog">Blog</a></li>
        <li><a href="#support">Contactar</a></li>
        <li><a href="javascript:showSearchLayer()">Buscar</a></li>
    </ul>
</nav>
```

1.1.22. Elemento section

```
<section>
    <article>
        <h3>Atmósfera de Mercurio</h3>
        <p>La atmósfera de Mercurio contiene un 31.7% de Potasio, un 24.9%
de Sodio, un 9.5% de Oxígeno atómico, un 7.0% de Argón, un 5.9% de Helio,
un 5.6% de Oxígeno molecular, un 5.2% de Nitrógeno, un 3.6% de Dióxido de
carbono, un 3.4% de Agua y un 3.2% de Hidrógeno.</p>
    </article>
</section>
```

1.2. FORMATEANDO EL TEXTO

1.2.1. Elemento abbr

```
<abbr title="Cascading Style Sheets">CSS</abbr> es un lenguaje de diseño
gráfico para definir y crear la presentación de un documento estructurado
escrito en un lenguaje de marcado.
```

1.2.2. Elemento address

```
<address>
    Escrito por Pablo Enrique Fernández Casado.
    Visita <a href="https://ejemplo.com">Ejemplo.com</a>
    Castellana 58, local
    28046 Madrid
```

```
    España
</address>
<address>
    Email de contacto:
    <a href="mailto:ejemplo@gmail.com">ejemplo@gmail.com</a><br>
    Teléfono: <a href="tel:+34999999999">(+34) 999.999.999</a>
</address>
```

1.2.3. Elemento bdo

```
<p dir="ltr">Esta palabara arábica <bdo dir="rtl">ARABIC PLACEHOLDER</bdo>,
está escrita de izquierda a derecha, pero se muestra al revés.</p>
```

1.2.4. Elementos blockquote y cite

```
<blockquote cite="https://blog.com/einstein">
    Hay dos cosas infinitas, el Universo y la estupidez humana
</blockquote>
<p>
    <cite>
        Hay dos cosas infinitas, el Universo y la estupidez humana
    </cite>, dicho por Albert Einstein
</p>
```

1.2.5. Elemento code

```
<code>
    <script type="text/javascript">
        document.querySelector("body").style.fontSize = "14px";
    </script>
</code>
```

1.2.6. Elemento data

Un ejemplo podría ser:

```
<ul>
    <li><data value="3967381398">Producto pequeño</data></li>
    <li><data value="3967381399">Producto mediano</data></li>
    <li><data value="3967381400">Producto grande</data></li>
</ul>
```

1.2.7. Elemento dfn

```
<p>
    El <dfn>HTML</dfn> es un lenguaje de marcado para hipertextos.
```

```
</p>
```

1.2.8. Elemento em

```
<p>
    Este texto no tiene énfasis,
    <em>pero este texto sí está con énfasis</em>
</p>
```

1.2.9. Elemento i

```
<p>
    Este texto no tiene énfasis,
    <i>pero este texto sí está con énfasis</i>
</p>
```

1.2.10. Elementos ins y del

```
<p>
    El cometa <del>C/2020 F3</del> <ins>Neowise</ins>, descubierto ...
</p>
```

1.2.11. Elemento kbd

```
<kbd>alt + S</kbd>
<p>
    Pulse <kbd><kbd style="border: 1px solid #000; border-radius: 4px;
padding: 2px;">Ctrl</kbd> + <kbd style="border: 1px solid #000; border-
radius: 4px; padding: 2px;">R</kbd></kbd> para recargar la página.
</p>
```

1.2.12. Elemento mark

```
<p>
    Los <mark>elementos P no deben contener etiquetas que no sean de
texto</mark>. Esto es, no es aconsejable introducir en una etiqueta de
párrafo un elemento DIV, SECTION, ARTICLE, ...
</p>
```

1.2.12.1. Ejemplo de superíndices

```
<math>
    <msup>
        <mi>n</mi>
        <mn>7</mn>
    </msup>
</math>
```

1.2.12.2. Ejemplo de subíndices

```
<math>
    <msub>
        <mi>n</mi>
        <mn>7</mn>
    </msub>
</math>
```

1.2.12.3. Ejemplo de fracciones

```
<math>
    <mfrac>
        <mn>1</mn>
        <mn>2</mn>
    </mfrac>
</math>
```

1.2.12.4. Ejemplo de raíces

```
<math>
    <mroot>
        <mn>-8</mn>
        <mn>3</mn>
    </mroot>
</math>
```

1.2.12.5. Ejemplo de sumatorios

```
<math>
    <mrow>
        <munderover>
            <mo>∑</mo>
            <mrow>
                <mi>n</mi>
                <mo>=</mo>
                <mn>1</mn>
            </mrow>
            <mrow>
                <mo>+</mo>
                <mn>∞</mn>
            </mrow>
        </munderover>
        <mfrac>
```

```
            <mn>1</mn>
            <msup>
                <mi>n</mi>
                <mn>2</mn>
            </msup>
        </mfrac>
    </mrow>
</math>
```

1.2.12.6. Ejemplo de matrices

```
<math>
    <mrow>
        <mo>[</mo>
        <mtable>
            <mtr>
                <mtd> <mn style="color: var(--color2-bg);">x</mn> </mtd>
                <mtd> <mn>1</mn> </mtd>
            </mtr>
            <mtr>
                <mtd> <mn>2</mn> </mtd>
                <mtd> <mn>3</mn> </mtd>
            </mtr>
        </mtable>
        <mo>]</mo>
    </mrow>
</math>
```

1.2.12.7. Ejemplo de integrales

```
<math>
    <munderover>
        <mo>∫</mo>
        <mi>a</mi>
        <mi>b</mi>
    </munderover>
    <mrow>
        <mo>(</mo>
        <mn>5</mn>
        <mi>x</mi>
        <mo>+</mo>
        <mn>2</mn>
        <mi>cos</mi>
        <mrow>
            <mo>(</mo>
            <mi>x</mi>
            <mo>)</mo>
        </mrow>
        <mo>)</mo>
```

```
    </mrow>
    <mi>dx</mi>
</math>
```

1.2.13. Elemento pre

```
<pre>
    <p>
        Los  espacios repetidos y
        Saltos de línea de    este   elemento se muestran tal cuál!
    </p>
</pre>
```

1.2.14. Elementos sub y sup

```
<p>La fórmula del agua es H<sub>2</sub>O</p>
<p>E = MC<sup>2</sup></p>
```

1.2.15. Elemento var

```
<var>x</var> = Millones de personas;
```

1.3. LISTAS

1.3.1. Elemento ul

```
<ul style="list-style-type: disc">
    <li>
        Planetas del Sistema Solar
        <ul>
            <li>Mercurio</li>
            <li>Venus</li>
            <li>La Tierra</li>
            <li>Marte</li>
            <li>Júpiter</li>
            <li>Saturno</li>
            <li>Urano</li>
            <li>Neptuno</li>
            <li>Phattie (hipotético planeta helado)</li>
        </ul>
    </li>
    <li>
        Planetas enanos del Sistema Solar
        <ul>
            <li>Eris</li>
```

```
            <li>Plutón</li>
            <li>Makemake</li>
            <li>Haumea</li>
            <li>Ceres</li>
        </ul>
    </li>
</ul>
```

1.3.2. Elemento ol

```
<ol style="list-style-type: decimal">
    <li>...</li>
    <li>
        Satélites de Plutón
        <ol>
            <li>Nix</li>
            <li>Caronte</li>
            <li>Hydra</li>
        </ol>
    <li>...</li>
</ol>
```

1.3.3. Elemento dl

```
<h4>Términos de astronomía</h4>
<dl>
    <dt>Planeta</dt>
    <dd>Es aquel cuerpo celeste que orbita alrededor del Sol, posee una
masa como para que su propia gravedad domine las fuerzas presentes como
cuerpo rígido, lo que implica una forma eférica determinada por el
equilibrio hidrostático y es claramente dominante en su vecindad, habiendo
limpiado su órbita de cuerpos similares a él.</dd>

    <dt>Planeta enano</dt>
    <dd> Es aquel cuerpo celeste que orbita alrededor del Sol, posee una
masa como para que su propia gravedad domine las fuerzas presentes como
cuerpo rígido, lo que implica una forma eférica determinada por el
equilibrio hidrostático y no es dominante en su vecindad y no es un
satélite de otro planeta o cuerpo estelar.</dd>
</dl>
```

1.4. ENLACES

1.4.1. Elemento a

```
<a href="https://www.google.es">Visitar Google España</a>
```

1.4.1.1. Propiedad list-style

```
ul { list-style: circle inside url("circle.png"); }
```

1.4.2. Márgenes internos y externos

```
li  { margin: 15px 15px 15px 15px; }
```

```
p   { padding: 1em 0.5em 1em; }
```

```
div { margin: 0 auto; }
```

```
li  { padding: 15px; }
```

```
ul  { margin-top: 2ch; }
li  { margin-right: auto; }
p   { padding-bottom: 1vw; }
div { padding-left: 1rem; }
```

1.4.2.1. Propiedades border y outline

```
div { border: 2px solid #f0f0f0; }
div { outline: 2px solid #f0f0f0; }
```

1.4.2.2. Propiedades border-color y outline-color

```
li  { border-color: red #F00 #FF0000 rgba(255, 0, 0); }
li  { outline-color: red #F00 #FF0000 rgba(255, 0, 0); }
```

```
p   { border-color: lightblue darkblue lightblue; }
```

```
div { border-color: black gray; }
```

```
li  { border-color: orange; }
```

.

```
div { border-top-color: rgb(0, 0, 0); }
div { border-right-color: rgba(0, 0, 0, 1); }
```

```
div { border-bottom-color: hsl(0, 0%, 0%); }
div { border-left-color: hsla(0, 0%, 0%, 1); }
```

1.4.2.3. Propiedad border-style y outline-style

```
li  { border-style: solid solid solid solid ; }
```

```
p   { border-style: solid dotted solid; }
```

```
div { border-style: dashed dotted; }
```

```
li  { border-style: ridge; }
```

```
div { border-top-style: solid; }
div { border-right-style: solid; }
div { border-bottom-style: solid; }
div { border-left-style: solid; }
li  { border-width: 2px 2px 2px 2px; }
```

```
p   { border-width: 2px 1px 2px; }
```

```
div { border-width: 2px 4px; }
```

```
li  { border-width: 2px; }
```

```
div { border-top-width: 2px; }
div { border-right-width: 1px; }
div { border-bottom-width: 1px; }
div { border-left-width: 2px; }
li  { border-radius: 15px 15px 15px 15px; }
```

```
p   { border-radius: 15px 0px 15px; }
```

```
div { border-width: 15px 0px; }
```

```
li  { border-width: 15px; }
RGB(0, 0, 255);       /* Color azul al 50% de transparencia */
RGBA(0, 0, 255, 0.5); /* Color azul al 50% de transparencia */
```

1.4.2.3.1 PROPIEDAD BACKGROUND

```
div { background: purple; }
div { background: url(./imagen-fondo.png) repeat-x; }
div { background: padding-box black; }
div { background: no-repeat centrer/cover url(./imagen-fondo.png); }
```

```
li  { background: #FFF url("plus.png") no-repeat fixed left center; }
```

1.4.2.3.2 PROPIEDAD BACKGROUND-COLOR

```
div { background-color: rgba(128, 0, 128, 1); }
```

1.4.2.3.3 PROPIEDADES TEXT-SHADOW Y BOX-SHADOW

```
text-shadow: PosX PosY radio_desenfoque color;
box-shadow:  inset PosX PosY radio_desenfoque radio_propagación color;
```

1.4.3. Posicionamiento

1.4.3.1. Propiedad clear

```
aside { clear: right; }
div   { clear: both; }
p     { clear: left; }
```

1.4.3.2. Propiedad float

```
aside { float: right; }
div   { float: left; }
p     { float: none; }
```

1.4.3.3. Propiedad position

1.4.3.4. Regla import

Ejemplo:

```
@import "custom-styles.css";
```

Ejemplo:

```
@import "mobile.css" screen and (max-width: 768px);
```

1.4.3.5. Regla keyframes

Ejemplo:

```
@keyframes ejemplo1 { from {top: 0px;} to {top: 200px;} }
```

```
@keyframes ejemplo2 {
    0%   {top: 0px; left 0; }
    25%  {top: 0px; left: 50px; }
    50%  {top: 50px; left: 50px}
    75%  {top: 50px; left: 0}
    100% {top: 0; left 0; }
}
```

1.4.3.6. Regla media

```
@media screen and (min-width: 768px) and (max-width: 1024px){
    /* Reglas CSS aplicables para estas circunstancias */
}
```

```
@media (any-hover: hover){
    /* Reglas CSS aplicables cuando puede pasar por los elementos */
}

@media (any-hover: none){
    /* Reglas CSS aplicables cuando NO puede pasar por los elementos */
}
@media (any-pointer: coarse){
    /* Reglas CSS aplicables cuando el dispositivo es de baja precisión */
}

@media (any-pointer: fine){
    /* Reglas CSS aplicables cuando el dispositivo es de alta precisión */
}

@media (any-pointer: none){
    /* Reglas CSS aplicables cuando no hay dispositivo señalador */
}
```

```
@media (aspect-ratio: 16/9){
 /* Reglas CSS aplicables cuando el dispositivo es 16 a 9 */
}

@media (min-aspect-ratio: 1/1){
/* Reglas CSS aplicables cuando el dispositivo es, como máximo, 1 a 1 */
}

@media (max-aspect-ratio: 11/16){
 /* Reglas CSS aplicables cuando el dispositivo es, como máximo, 11 a 16 */
}
```

```
@media (color){
    /* Reglas CSS aplicables cuando el dispositivo es color */
}

@media (min-color: 8){
    /* Reglas CSS aplicables cuando el dispositivo admite 8 bits */
```

```
}

@media (max-color: 16){
    /* Reglas CSS aplicables cuando el dispositivo admite 16 bits */
}
```

```
@media (color-gamut: srgb){
    /* Reglas CSS aplicables para estas circunstancias */
}

@media (color-gamut: p3){
    /* Reglas CSS aplicables para estas circunstancias */
}
@media (color-gamut: srgb){
    /* Reglas CSS aplicables para estas circunstancias */
}
```

```
@media (color-index: 0){
    /* Reglas CSS aplicables para estas circunstancias */
}

@media (min-color-index: 10){
    /* Reglas CSS aplicables cuando permite, como mínimo, 10 colores */
}

@media (max-color-index: 256){
    /* Reglas CSS aplicables cuando permite, como máximo, 256 colores */
}
@media (grid: 0){
    /* Reglas CSS aplicables cuando NO está basado en rejilla */
}
```

```
@media (height: 610px){
    /* Reglas CSS aplicables para estas circunstancias */
}

@media (min-height: 40vh){
    /* Reglas CSS aplicables para estas circunstancias */
}

@media (max-height: 80vh){
    /* Reglas CSS aplicables para estas circunstancias */
}
```

```
@media (hover: hover){
    /* Reglas CSS aplicables cuando puede desplazarse */
}

@media (hover: none){
```

```
    /* Reglas CSS aplicables cuando NO puede desplazarse */
}
```

```
@media (inverted-colors: inverted){
    /* Reglas CSS aplicables cuando los colores están invertidos */
}

@media (inverted-colors: none){
    /* Reglas CSS aplicables cuando los colores NO están invertidos */
}
```

```
@media (monochrome: 0){
    /* Reglas CSS aplicables cuando no es monocromo */
}

@media (monochrome){
    /* Reglas CSS aplicables cuando es monocromo */
}
```

```
    /* Reglas CSS aplicables cuando el dispositivo está en horizontal */
}

@media (orientation: portrait){
    /* Reglas CSS aplicables cuando el dispositivo está en vertical */
}
```

```
@media (pointer: coarse){
    /* Reglas CSS aplicables cuando el dispositivo es de baja precisión */
}

@media (pointer: fine){
    /* Reglas CSS aplicables cuando el dispositivo es de alta precisión */
}

@media (pointer: none){
    /* Reglas CSS aplicables cuando no hay dispositivo señalador */
}
```

```
@media (resolution: 100dpi){
/* Reglas CSS aplicables cuando la densidad es 100 píxeles */
}

@media (min-resolution: 72dpi){
/* Reglas CSS aplicables cuando la densidad es, como minimo, 72 píxeles */
}

@media (max-resolution: 300dpi){
/* Reglas CSS aplicables cuando la densidad es, como minimo, 300 píxeles */
}
```

```
@media (width: 32vw){
    /* Reglas CSS aplicables para estas circunstancias */
}

@media (min-width: 64vw){
    /* Reglas CSS aplicables para estas circunstancias */
}

@media (max-width: 96vw){
    /* Reglas CSS aplicables para estas circunstancias */
}
```

1.5. FUNCIONES

.

1.5.1. Funciones de pseudo-elementos

1.5.1.1. Función attr

```
div::before { content: attr(data-value); }
```

1.5.1.2. Función counter

```
ul.falso-ol    { counter-reset: indice;}
ul.falso-ol li { counter-increment: indice; }
ul.falso-ol li::before { content: counter(indice); }
```

1.5.2. Funciones de cálculo

1.5.2.1. Función calc

Ejemplos:

```
div     { width: calc(100% - 20px); }
aside   { right: calc(5% - 0.5em); }
nav     { border-width: calc(100% - 2rem); }
```

1.5.3. Funciones gráficas

1.5.3.1. Función linear-gradient

```
div { background-image: linear-gradient(to right, black, transparent); }
```

```
div { border-image: linear-gradient(45deg, black, transparent); }
div { border-image: linear-gradient(50grad, black, transparent); }
div { border-image: linear-gradient(0.125turn, black, transparent); }
div { border-image: linear-gradient(0.7854rad, black, transparent); }
```

1.5.3.2. Función repeating-linear-gradient

```
div {
    background-image: repeating-linear-gradient(
          45deg,
          black 45px, transparent 47px, gray 60px);
}

/* El resultado debería ser algo como: */
```

1.5.3.3. Función radial-gradient

```
div { background: radial-gradient(
                  #FFFFFF 5px, #CCCCCC 50px,
                  #AAAAAA 100px, #000000 200px); }

/* El resultado debería ser algo como: */
```

```
div { background: radial-gradient(
                  ellipse at 0 0,
                  #FFFFFF 5px,
                  #CCCCCC 50px,
                  #AAAAAA 100px,
                  #000000 200px); }

/* El resultado debería ser algo como: */
```

```
div { background: radial-gradient(
                  ellipse farthest-corner at 200px 40px,
                  #FFFFFF 5px,
                  #CCCCCC 50px,
                  #AAAAAA 100px,
```

```
                    #000000 200px); }

/* El resultado debería ser algo como: */
```

1.5.3.4. Función repeating-radial-gradient

```
div {
    background-image: repeating-radial-gradient(
                      closest-side,
                      #000000 5px, #888888 15px, #888000 25px);
}

/* El resultado debería ser algo como: */
```

1.5.3.5. Función conic-gradient

```
div {
    background: conic-gradient(black, darkgray, gray, lightgray, white);
}

/* El resultado debería ser algo como: */
```

```
div { background: conic-gradient( black 0deg, #333 0deg 10deg, #666 10deg
20deg, #999 40deg 120deg, #ccc 120deg 200deg, white 400deg); }

/* El resultado debería ser algo como: */
```

1.6. VARIABLES

```
:root { --background: whitesmoke; }
```

```
div {
    background: var(--backcolor);
    --forecolor: yellow;
    --backcolor: black;
}
```

```
div::before{
    content:"Texto en amarillo";
    color: var(--backcolor);
}
```

```
document.documentElement.style.setProperty('--background', 'whitesmoke');
```

```
:root {
    --backcolor: #f0f0f0;
    --forecolor: #003366;
    --fontFamily: Arial, sans-serif;
    --fontSize: 15px;
}
var http = new XMLHttpRequest()

http.open("GET", './getCustomPersonalization.php')
http.onreadystatechange = function(){
    if(this.readyState == 4 && this.status == 200){
        var resultado = JSON.parse(this.responseText)
        override(resultado);
    }
}

http.send();

function override(customJSON){
    var style = document.documentElement.style;
    for(key in customJSON){
        style.setProperty('--' + key, customJSON[key]);
    }
}
```

1.7. PRACTICA Y JUEGA

5

IMÁGENES Y MULTIMEDIA

2.1. TIPOS DE IMÁGENES

2.2. ELEMENTOS DISPONIBLES EN HTML5

2.2.1. Elemento audio

```
<audio controls>
    <source src="colibri.mp3" type="audio/mpeg">

    El navegador no soporta la etiqueta audio.
</audio>
```

2.2.2. Elementos figure y figcaption

```
<figure>
    <img src="./images/gantt.jpg" alt="diagrama-de-gantt" />
    <figcaption>Ejemplo de diagrama de Gantt</figcaption>
</figure>
```

```
<img src="./images/gantt.jpg" alt="diagrama-de-gantt" />
```

2.2.3. Elemento picture

```
<picture>
    <source srcset="./img/land-desktop.png" media="(min-width: 1680px)" />
    <source srcset="./img/land-laptop.png" media="(min-width: 1366px)" />
    <source srcset="./img/land-tablet.png" media="(min-width: 640px)" />
    <source srcset="./img/land-mobile.png" media="(min-width: 360px)" />

    <img src="./img/land-laptop.png" />
</picture>
```

2.2.4. Elemento source

```
<picture>
    <source srcset="./img/land-laptop.png" media="(min-width: 1366px)" />
    <source srcset="./img/land-tablet.png" media="(min-width: 900px)" />
    <source srcset="./img/land-mobile.png" media="(min-width: 768px)" />
</picture>
```

2.2.4.1. SRCSET

2.2.4.2. MEDIA

2.2.4.3. TYPE

2.2.5. Elemento video

```
<video controls>
    <source src=" el-quinto-elemento.mp4" type="audio/mp4">

    El navegador no soporta la etiqueta video.
</video>
```

2.3. PROPIEDADES DISPONIBLES EN CSS

2.3.1. Propiedad background-attachment

2.3.2. Propiedad background-clip

2.3.3. Propiedad background-image

2.3.4. Propiedad background-origin

2.3.5. Propiedad background-position

2.3.6. Propiedad background-repeat

2.3.7. Propiedad background-size

2.3.8. Propiedad object-position

2.4. IMÁGENES RECEPTIVAS Y ADAPTATIVAS

```
<style>
    div { display: block; height: 250px; position: relative; width: 100%; }
    img { height: auto; object-fit: none; width: 100%; }
</style>

<div>
    <img src="nebulosa-aguila.jpg" alt="M-16 - Nebulosa del águila" />
<div>
```

2.4.1. Adaptación de imágenes mediante consultas de medios

```
<style>
    .banner {
        background-image: url(imagen-640x360.jpg);
        border-bottom: 1px solid rgba(0,0,0,0.1);
        color: #fff;
        display: block;
        height: 100vh;
        position: relative;
        text-align: center;
        width: 100%;
    }

    @media (min-width: 800px) {
        .banner { background-image: url(imagen-1280x720.jpg); }
    }

    @media (min-width: 1400px) {
        .banner { background-image: url(imagen-1920x1080.jpg); }
    }

    @media (min-width: 2000px) {
        .banner { background-image: url(imagen-2560x1440.jpg); }
    }
</style>
```

```
<div class="banner"><div>
```

2.4.2. Selección de imágenes mediante IMG y SRCSET

```
<img src="imagen-640x360.jpg" alt="texto-alternativo-para-imagen"
     srcset="imagen-1280x720.jpg 800w"
     srcset="imagen-1920x1080.jpg 1400w"
     srcset="imagen-2560x1440.jpg 2000w"
 />
```

```
<img src="imagen-640x360.jpg" alt="texto-alternativo-para-imagen"
     srcset="imagen-1280x720.jpg 800w"
     srcset="imagen-1280x720-2.jpg 800w 2x"
     srcset="imagen-1920x1080.jpg 1400w"
     srcset="imagen-2560x1440.jpg 2000w"
 />
```

```
<img sizes="(max-width: 800px) 100vw,
           (min-width: 1280px) 50vw,
           100vw"
     srcset="imagen-640x360.jpg 800w,
             imagen-1280x720.jpg 1280w,
             imagen-1920x1080.jpg 1360w,
             imagen-2560x1440.jpg 2000w"
     src="imagen-2560x1440.jpg"
     alt="texto-alternativo-imagen" />
```

2.4.3. Selección de imágenes mediante PICTURE y SOURCE

```
<picture>
    <source srcset="imagen-2560x1440.jpg" media="(min-width: 2000px" />
    <source srcset="imagen-1920x1080.jpg" media="(min-width: 1360px" />
    <source srcset="imagen-1280x720.jpg" media="(min-width: 768px" />

    <img src="imagen-640x360.jpg" alt="texto-alternativo-imagen" />
</picture>
```

```
<picture>
    <source srcset="imagen-640x360.jpg" media="(max-width: 768px" />
    <source srcset="imagen-1280x720.jpg" media="(max-width: 1360px" />
    <source srcset="imagen-1920x1080.jpg" media="(max-width: 2000px" />

    <img src="imagen-2560x1440.jpg" alt="texto-alternativo-imagen" />
```

```
</picture>
<img src="imagen-640x360.jpg" alt="texto-alternativo-para-imagen"
     srcset="imagen-1280x720.jpg 800w"
     srcset="imagen-1280x720-2.jpg 800w 2x"
     srcset="imagen-1920x1080.jpg 1400w"
     srcset="imagen-2560x1440.jpg 2000w"
 />
```

2.5. VIDEOS RECEPTIVOS

```
video {
    width: 100%;
    display: block;
    object-fit: cover;
    height: 100vh;
}

<video src="./4703.mp4"
       type="video/mp4"
       loop=""
       muted=""
       controls=""
       autoplay="">
    Tu navegador no admite el elemento <code>video</code>. Puedes descargar
el video desde la URLhttps://www.pexels.com/video/beautiful-timelapse-of-
the-night-sky-with-reflections-in-a-lake-857251/
</video>
```

2.6. PRACTICA Y JUEGA

6

DISEÑOS BASADOS EN DIMENSIONES

3.1. TABLAS

3.1.1. Elementos disponibles en HTML5

3.1.1.1. Elemento caption

3.1.1.2. Elemento table

3.1.1.3. Elemento colgroup

```
<colgroup>
    <col style="background: whitesmoke;"></col>
    <col span="2" style="background: lavender;"></col>
</colgroup>
```

3.1.1.4. Elementos thead y tfoot

3.1.1.5. Elemento tbody

3.1.1.6. Elemento tr

3.1.1.7. Elemento th

3.1.1.8. Elemento td

3.1.2. Elementos disponibles en CSS

3.1.2.1. Propiedades border-collapse

3.1.2.2. Propiedad border-spacing

3.1.2.3. Propiedad caption-side

3.1.2.4. Propiedad empty-cells

3.1.2.5. Propiedad table-layout

3.1.3. Creación de tablas responsive

3.1.3.1. Mediante barras de desplazamiento

Código CSS

```
@media screen and (max-width: 620px) {
    table { display: block; overflow-x: auto; width: 100%; }
}
```

3.1.3.2. Mediante consultas de medios

Código CSS

```
html,
body { margin: 0; padding: 0; font-family: 'Roboto', sans-serif;
       display: block; font-size: 14px;
}

table         { border: 1px solid rgba(0,0,0,0.2); border-spacing: 2px;
                margin: 10px 0; }

table caption { background: #000000; color: #fff; font-size: 1.0rem;
                font-weight: bold; line-height: 1.5; padding: 0;
```

```
                text-transform: uppercase; }

table td,
table th      { border: 1px solid rgba(0,0,0,0.2); border-spacing: 0;
                font-size: 1rem; padding: 5px; text-align: left;
                margin: 2px 0; white-space: nowrap; }

table thead th:nth-child(6),
table tbody td:nth-child(6){ text-align: right; }

@media screen and (max-width: 620px) {
    table { width: 100%; }

    thead { display: none; }

    tr td:first-child { background: #f0f0f0; font-weight: bold; }

    tbody td { display: block; text-align: right; }

    tbody td:before { content: attr(data-field); display: block;
                      float: left; font-weight: bold; padding: 0 10px 0 0;
                      text-align: left; width: auto; }
}
```

3.2. DISEÑO BASADO EN CAJAS FLEXIBLES (FLEXBOX)

3.2.1. Principales elementos disponibles en CSS

3.2.1.1. Propiedad align-content

3.2.1.2. Propiedad align-items

3.2.1.3. Propiedad allign-self

3.2.1.4. Propiedad flex

.

```
li  { flex: 1 1 auto; } /* FLEX-GROW FLEX-SHRINK FLEX BASIS */
```

```
p   { flex: 1 100%; }  /* FLEX-GROW FLEX-BASIS */
p   { flex: 1 1; }     /* FLEX-GROW FLEX-SHRINK */
```

```
p   { flex: 1; }      /* FLEX-GROW */
p   { flex: 100%; }   /* FLEX-BASIS */
```

3.2.1.5. Propiedad flex-basis

3.2.1.6. Propiedad flex-direction

3.2.1.7. Propiedad flex-grow

3.2.1.8. Propiedad flex-shrink

3.2.1.9. Propiedad flex-wrap

3.2.1.10. Propiedad justify-content

3.2.1.11. Propiedad order

3.2.2. Creación de flexbox responsive

```
<div class="flexbox">
    <div class="caption">EJEMEPLO DE TABLA CON FLEXBOX CSS</div>

    <div class="row header">
        <div class="col">ID</div>
        <div class="col">Empresa</div>
        <div class="col">F. Movimiento</div>
        <div class="col">Tipo</div>
        <div class="col">Concepto</div>
        <div class="col">Importe</div>
        <div class="col">Estado</div>
    </div>

    <div class="row">
        <div class="col">1</div>
        <div class="col">Consultores SA</div>
        <div class="col">30-12-2019</div>
        <div class="col">Ingreso</div>
        <div class="col">Nómina</div>
        <div class="col">+1268.00 €</div>
        <div class="col">Efectuado</div>
    </div>

    <div class="row">...</div>
    ...
</div>
```

Código CSS

```
.flexbox .caption { display: block; text-align: center; font-weight: 600;
                    background: #000; color: #fff; }
.flexbox          { border: 1px solid #ccc; }
.flexbox .row     { display: flex; width: 100%; max-width: 100%;margin: 0;}
.flexbox .col     { border: 1px solid #ccc; display: flex;
                    flex-flow: column nowrap; justify-content: flex-start;
                    align-items: flex-start; flex: 1 1 100%; margin: 1px;
                    padding: 0 5px; max-width: calc(100% / 7); }
.flexbox .header .col { font-weight: 600; }
```

3.3. DISEÑO BASADO EN CUADRÍCULAS (GRID LAYOUT)

3.3.1.1. Propiedades grid-template-rows y grid-template-columns

```
.grid {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr;
    grid-template-rows: 1fr 1fr 1fr 1fr;
    gap: 5px;
}
```

3.3.1.2. Propiedades grid-row-start y grid-row-end , grid-column-start, grid-column-end

```
.large-item {
  grid-column-start: 2;
  grid-column-end: five;
  grid-row-start: row1-start;
  grid-row-end: 3;
}
```

3.3.1.3. Propiedad align-items y justify-items

```
.grid {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr;
    grid-template-rows: 1fr 1fr 1fr 1fr;
    gap: 5px;
    align-items: center;
    justify-items: center;
}
```

3.3.1.1. Función repeat y las palabras clave

```
.grid {
    display: grid;
    grid-template-columns: repeat(7, 1fr);
    grid-template-rows: repeat(4, 1fr);
    gap: 5px;
}
.grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
    grid-template-rows: minmax(max-content, 1fr);
    gap: 5px;
}
```

3.3.2. Creación de grid responsive

```
<div class="grid">
    <div class="caption">EJEMEPLO DE TABLA CON FLEXBOX CSS</div>

    <div class="row">
        <div class="col">ID</div>
        <div class="col">Empresa</div>
        <div class="col">F. Movimiento</div>
        <div class="col">Tipo</div>
        <div class="col">Concepto</div>
        <div class="col">Importe</div>
        <div class="col">Estado</div>

        <div class="col">1</div>
        <div class="col">Consultores SA</div>
        <div class="col">30-12-2019</div>
        <div class="col">Ingreso</div>
        <div class="col">Nómina</div>
        <div class="col">+1268.00 €</div>
        <div class="col">Efectuado</div>

        <div class="col">2</div>
        ...
    </div>
</div>
```

Código CSS

```
.grid .row     { display: grid; grid-template-columns: repeat(7, 1fr);
                 grid-template-rows: minmax(max-content, 1fr);
                 padding: 1px 1px; border: 1px solid #ccc; }
.grid .caption { display: block; text-align: center; font-weight: 600;
                 background: #000; color: #fff; }
.grid .col     { border: 1px solid #ccc; margin: 1px; padding: 0 5px; }
```

```
.grid .row .col:nth-child(-n+7) { font-weight: 600; }
```

3.4. PRACTICA Y JUEGA

7

FORMULARIOS

4.1. TIPOS DE FORMULARIO

4.1.1. Formularios de contacto

4.1.2. Formularios de acceso

4.1.3. Formularios de registro

4.1.4. Formularios de entrada general

4.2. ELEMENTOS DISPONIBLES EN HTML5

4.2.1. Elemento form

```
<form action="/action_page.php" method="get" name="frm">
    <!-- ... -->
</form>
```

4.2.2. Elemento button

```
<button type="submit"
```

```
        formaction="./pages/login-test-2.php"
        formmethod="POST"
        formenctype="multipart/form-data"
        formtarget="_blank">
        value="Probar test 2"
</button>
```

4.2.3. Elemento datalist

```
<input list="technologies">

<datalist id="technologies">
  <option value="HTML5">
  <option value="JavaScript">
  <option value="CSS3">
  <option value="SVG">
</datalist>
```

4.2.4. Elemento fieldset

```
<form action="./login.php">
    <fieldset>
        <legend>Acceso Privado</legend>

        <!-- ... -->
    </fieldset>
</form>
```

4.2.5. Elemento input

```
<form action="/action_page.php" method="get" name="frm">
    <label for="field1">Campo de texto:</label>
    <input type="text" id="field1" name="field1">

    <label for="field2">Campo sólo números:</label>
    <input type="number" id="field2" name="field2">

    <label for="field3">Campo fecha:</label>
    <input type="date" id="field3" name="field3">

    <label for="field4">Campo URL:</label>
    <input type="url" id="field4" name="field4">

    <label for="field5">Campo para emails:</label>
    <input type="email" id="field5" name="field5">

    <label for="field6">Campo para teléfonos:</label>
    <input type="tel" id="field6" name="field6">
```

```
    <label for="field7">Campo para imágenes:</label>
    <input type="image" id="field7" name="field7">

    <label for="field8">Elemento para adjuntar archivos:</label>
    <input type="file" id="field8" name="field8">

    <label for="field9">Casilla de verificación:</label>
    <input type="checkbox" id="field9" name="field9">

    <label for="field10">Campo tipo radio:</label>
    <input type="radio" id="field10" name="field10">

    <label for="field11">Campo oculto:</label>
    <input type="hidden" id="field11" name="field11">

    <label for="submit">Botón enviar:</label>
    <input type="submit" id="submit" name="submit">

    <button type="submit">
        Guardar datos
    </button>
</form>
```

4.2.6. Elemento label

```
<label for="name">Nombre:</label>
<input type="text" id="name" name="name">
```

4.2.7. Elemento legend

```
<form action="./login.php">
    <fieldset>
        <legend>Acceso a ejemplo.com</legend>

        <!-- ... -->
    </fieldset>
</form>
```

4.2.8. Elemento meter

```
    <meter id="available-space" value="324" min="0" max="1024">
        324MB libres de 1024 MB
    </meter>
```

4.2.9. Elemento optgroup

```
<select id="motorcycles">
    <optgroup label="Harley Davidson">
```

```
        <option value="01">Sportster</option>
        <option value="02">Softail</option>
        <option value="03">Touring</option>
    </optgroup>
    <optgroup label="Indian">
        <option value="04">Chief</option>
        <option value="05">Springfield</option>
        <option value="06">Scout Sixty</option>
    </optgroup>
</select>
```

4.2.10. Elemento option

```
<select id="rate">
    <option value="01">Mala</option>
    <option value="02">Media</option>
    <option value="03">Buena</option>
</select>
```

4.2.11. Elemento output

```
<form oninput="res.value=parseInt(op1.value) + parseInt(op2.value)">
    <label>Suma de dos operandos</label>

    <input type="range" id="op1" value="50">100
    +
    <input type="range" id="op2" value="50">
    =
    <output name="res" for="op1 op2">100</output>
</form>
```

4.2.12. Elemento progress

```
<form oninput="res.value=parseInt(op1.value) + parseInt(op2.value)">0
    <label>Progreso de instalación</label>
    <progress id="progress" value="54" max="100"> 54% </progress>
</form>
```

4.2.13. Elemento select

```
<select id="rate">
    <option value="01">Mala</option>
    <option value="02">Media</option>
    <option value="03">Buena</option>
</select>
```

4.2.14. Elemento textarea

```
<textarea id="desc" rows="24" cols="80"></textarea>
```

4.3. ELEMENTOS DISPONIBLES EN CSS

4.3.1. Propiedad box-sizing

4.3.2. Propiedad resize

```
textarea { resize: none; }
```

4.4. PRACTICA Y JUEGA

8

INTRODUCCIÓN A JAVASCRIPT

5.1. VARIABLES Y ÁMBITOS

5.1.1. Declaración de variables

```
var fechaActual = new Date();
let fechaActual = new Date();
```

```
const fechaActual = new Date();
```

5.1.2. Ámbito de las variables

```
let total = 0;

function suma(a, b) {
    let aux = a + b;
    total = aux;
}

function resta(a, b) {
    let aux = a - b;
    total = aux;
}
```

5.2. TIPOS DE DATOS

5.2.1. Tipo String

```
String(4);            // Devuelve "4"
String(true);         // Devuelve "true"
String(String(5));    // Devuelve "5"
String(var);          // Devuelve error de sintaxis
```

5.2.1.1. Propiedades

5.2.1.2. Métodos

5.2.1.3. Conversión de Strings

```
parseInt("4")               // Devuelve 4
parseInt("hola")            // Devuelve NaN (no es un número)
parseInt("21 calles")       // Devuelve 21
parseInt("1e3")             // Devuelve 1
parseFloat("1.5")           // Devuelve 1.5
parseFloat("1,5")           // Devuelve 1
String(new Date())          // Devuelve la fecha actual en formato GMT
```

5.2.1.4. Formateado de Strings

```
/* Literales de cadena */
'Esto es un literal de cadena'
"Esto es otro literal de cadena"

/* Secuencia escapada en hexadecimal */
"\x41"      // Devuelve "A"

/* Secuencia escapada en Unicode */
"\u0041"    // Devuelve "A"
```

.

```
console.log('\u{1F440}', "\uD83D\uDC40");
```

```
/* Literales de cadena multilínea */
console.log('Nombre: Pablo\n\
Apellidos: Fernández');
```

```
/* Literales de plantilla (sólo con ES6 y superiores) */
console.log(`Nombre: Pablo
Apellidos: Fernández`);
let nombre = 'Pablo';

console.log('Nombre:\t\t' + nombre + '\n\
Apellidos\t: Fernández');
```

```
let nombre = 'Pablo';

console.log(`Nombre:\t\t${nombre}
Apellidos:\tFernández`);

// Ambos fragmentos de código deberían mostrar algo como:
Nombre:      Pablo
Apellidos:   Fernández
```

5.2.2. Tipo Number

```
Number("4");                // Devolverá 4
Number("Hola")              // Devolverá NaN porque no es un número
Number("21 calles")         // Devolverá NaN porque no es un número
Number(1e3)                 // Devolverá 1000
Number(true);               // Devolverá 1
Number(false);              // Devolverá 0
```

5.2.2.1. Propiedades

5.2.2.2. Métodos

5.2.2.2.1 CONVERSIÓN DE NÚMEROS

```
 (2.0).toString();          // Devuelve "2"
(2).toString();             // Devuelve "2"
2.toString();               // Error de sintaxis
Number(new Date())          // Devuelve un timestamp como 1567845361045
```

5.2.2.3. Formateado de números

```
 (123456.123).toLocaleString();          // Devuelve "123.456,123"
```

5.2.2.4. Operaciones con números

5.2.2.4.1 EL OBJETO MATH

5.2.3. Tipo BigInt

```
BigInt(9007199254740991);                 // Devuelve 9007199254740991n
BigInt("0x1fffffffffffff")                // Devuelve 9007199254740991n
BigInt("21 calles")                       // Devuelve Error de sintaxis
BigInt(true);                             // Devuelve 1n
BigInt(false);                            // Devuelve 0n
```

5.2.3.1. Propiedades

5.2.3.2. Métodos

5.2.4. Tipo Boolean

```
Boolean(0);                         // Devuelve false
Boolean("");                        // Devuelve false
Boolean(null);                      // Devuelve false
Boolean(",");                       // Devuelve true
Boolean(4);                         // Devuelve true
Boolean(1) - 1 == false             // Devuelve true
```

5.2.4.1. Propiedades

5.2.4.2. Métodos

5.2.5. Tipo Symbol

Para crear un símbolo sólo hay que hacer:

```
let s1 = Symbol("Hola Pablo");
let s2 = Symbol("Hola Pablo");
console.log(s1 == s2);                          // Devuelve false
```

5.2.6. Literal null

```
this == null                                    // Devolverá false
```

```
document.getElementById("elementoNoexistente") == null
```

5.2.7. Literales undefined y typeof

```
typeof x == "undefined"
```

5.3. OPERADORES Y EXPRESIONES

```
3 + 5  * 2;                          // Devolverá 13
(3 + 5) * 2;                         // Devolverá 16
```

5.3.1. Operadores generales

5.3.2. Operadores bit a bit

.

5.3.2.1. Operador &

```
let b1 = 12;         // Equivale a 1100
let b2 = 4;          // Equivale a 0100
————————————————————————————————
let b3 = b1 & b2;    // Resultado: 0100                 En decimal: 4
```

5.3.2.2. Operador |

```
let b1 = 12;         // Equivale a 1100
let b2 = 4;          // Equivale a 0100
————————————————————————————————
let b3 = b1 | b2;    // Resultado: 1100                 En decimal: 12
```

5.3.2.3. Operador ^

```
let b1 = 12;         // Equivale a 1100
let b2 = 4;          // Equivale a 0100
————————————————————————————————
let b3 = b1 ^ b2;    // Resultado: 1000                 En decimal: 8
```

5.3.2.4. Operador ~

```
let b1 = 38;         // Equivale a 00100110
————————————————————————————
let b2 = ~ b1;       // Resultado: 11011001              En decimal: -39
```

5.3.2.5. Operadores de desplazamiento

```
4 >> 2                              // Devuelve 1
4 << 2                              // Devuelve 16
```

5.4. CONTROL DE FLUJO Y GESTIÓN DE ERRORES

5.4.1. Estructura if

```
if(fecha == '20-02-2002' ){
    console.log('Es 20 de febrero de 2002');
}
```

5.4.2. Estructura if...else

```
if(fecha == '20-02-2002' ){
    console.log('Es 20 de febrero de 2002')
} else {
    console.log('No es 20 de febrero de 2002')
}
```

5.4.3. Estructura switch

```
switch(marca) {
    case 'ford':
        console.log('El coche es de la marca Ford');
        break;
    case 'seat':
        console.log('El coche es de la marca Seat');
        break;
    default:
        console.log('El coche es de otra marca');
}
```

.

5.4.4. Control de errores por tipo de dato

```
let fecha = '';
if(typeof arguments[0] == 'object' ){
    fecha = new Date(aux.anio + "-" + aux.mes + "-" + aux.dia);
} else {
```

```
    fecha = new Date(aux);
}
```

5.4.5. Control de errores por presencia

```
console.log('insertRule' in document.styleSheets[0]);
```

```
let json = { nombre: 'Pablo', edad: 18 };
console.log(json.hasOwnProperty("apellidos"));   // Devolverá false
console.log(json.hasOwnProperty("nombre"));      // Devolverá true
```

5.4.6. Manejo de excepciones

5.4.6.1. Sentencia Try...catch

```
allert("Hola mundo!");
console.log("Todo OK!");
```

```
try {
    allert("Hola mundo!");
} catch(err) {
    console.log(err.message);
}

console.log("¡Todo OK!");
```

5.4.6.2. Sentencia finally

```
try {
    allert("Hola mundo!");
} catch(err) {
    console.log(err.message);
} finally {
    console.log("Todo OK!");
}
```

5.4.6.3. Sentencia throw

```
throw "Error";                    // Devuelve "Uncaught Error"
throw 18;                         // Devuelve "Uncaught 18"
```

```
throw false;                          // Devuelve "Uncaught false"
```

```
function excepcionPersonalizada(mensaje) {
    let error = new Error(mensaje);

    error.code = "Error cero";
    return error;
}

excepcionPersonalizada.prototype = Object.create(Error.prototype);

throw excepcionPersonalizada("Error de entrada")
```

```
Uncaught Error: Error de entrada
    at excepcionPersonalizada (main.js:134)
    at main.js:142
```

5.5. BUCLES Y LA ITERACIÓN

5.5.1. Estructura for

```
let chars = new Array();
for(let x = 0; x < 100000; x++){
    chars[x] = {code: x, char: String.fromCharCode(x)};
}
```

```
for (let i = 0, j = 10; i <= j; i++, j--){
    console.log(i, j)
}
```

5.5.2. Estructura for...in

```
let arr = [1, 1, 2, 3, 5, 8];
for (let x in arr){
    console.log("X: ", x);  // Devuelve "X: 0" hasta "X: 9"
}
```

```
console.time();
let arrFor = new Array();
for(let x = 0; x < chars.length; x++){
    arrFor[x] = chars[x];
}
console.timeEnd();
```

```
console.time();
let arrForIn = new Array();
for(let key in chars){
    arrForIn[key] = chars[key]
}
console.timeEnd();
```

5.6. ESTRUCTURA FOR...OF

```
let arr = [1, 1, 2, 3, 5, 8];
for (let x of arr){
    console.log("X: ", x);  // Devuelve 1,1,2,3,5,8
}
```

```
console.time();
let arrForOf = new Array();
for(let [key, value] of chars.entries()){
    arrForOf[key] = value
}
console.timeEnd();
```

5.7. ESTRUCTURA FOREACH

```
console.time();
let arrForEach = new Array();
source.forEach(function(v, i){
    arrForEach[i] = v
});
console.timeEnd();
```

La función de **callback** puede recibir tres parámetros, aunque lo normal es que reciba sólo dos.

```
["A", "B", "C", "D"].forEach(function(valor, indice){
    console.log("Índice:", indice, "Valor:", valor);
});
// Devuelve Índice: 0 Valor: A
// Devuelve Índice: 1 Valor: B,...
```

```
let arr = ["A", "B", "C", "D"];
arr.forEach(function(val, idx, arr){
    arr[idx] = val.charCodeAt();
```

```
});

console.log(arr);                          // Devuelve [65, 66, 67, 68]
```

```
function calculadora() {
    this.suma = 0;
    this.producto = 1;
}

calculadora.prototype.sumar = function(array) {
  array.forEach(function(valor) {
    this.suma += valor;
  }, this);
};

calculadora.prototype.multiplicar = function(array) {
  array.forEach(function(valor) {
    this.producto *= valor;
  }, this);
};

let arr = [1,1,2,3,5,8];
let s = new calculadora();
s.sumar(arr);
s.multiplicar(arr);

console.log(s.suma, s.producto)                  // Devuelve 20 240
```

5.8. ESTRUCTURA DO...WHILE

```
let x = 0;
do {
    x += 1;                 // Forma abreviada de hacer x = x + 1;
    console.log("X: ", x);
} while (x < 10);
```

5.9. ESTRUCTURA WHILE

```
let x = 0;
while (x < 10){
    x += 1;                 // Forma abreviada de hacer x = x + 1;
    console.log("X: ", x)
};
```

5.10. SENTENCIA BREAK

```
for (let x = 0; x < 10; x++){
    if (x == 5) break;
    console.log("X: ", x);
}
```

5.11. SENTENCIA CONTINUE

```
for (let x = 0; x < 10; x++){
    if (x == 5) continue;
    console.log("X: ", x);
}
```

5.12. PRACTICA Y JUEGA

9

OBJETOS DE JAVASCRIPT

6.1. TIPOS DE OBJETO

6.2. PROPIEDADES

6.3. MÉTODOS

6.4. ARRAYS

6.4.1. Creación de arrays

```
let arr = new Array();
let arr = [];
```

6.4.2. Acceso a elementos de un array

```
let arr = [1, 1, 2, 3, 5, 8];
console.log(arr[0]);                                    // Devuelve 1
```

6.4.3. Inserción y almacenamiento de elementos en un array

```
let arr = [1, 1, 2, 3, 5, 8];
arr[6] = 13;
arr.push(21);
console.log(arr[7]);                                    // Devuelve 21
```

```
let arr = new Array();
arr[0][0] = 1;                              // Devuelve un error de tipo

let arr[0] = new Array();
arr[0][0] = 1;                              // Ahora sí lo inserta
```

6.4.4. Eliminación de elementos de un array

```
let arr = [1, 1, 2, 3, 5, 8];
delete(arr[4]);

console.log(arr);                  // Devolverá [1, 1, 2, 3, empty, 8]
```

```
let arr = [1, 1, 2, 3, 5, 8];
arr.splice(4, 1);

console.log(arr);                           // Devolverá [1, 1, 2, 3, 8]
```

6.4.5. Propiedades

6.4.6. Métodos

6.5. JSON

6.5.1. Sintaxis

```
 [
   { "abbreviation": "Ene", "name": "Enero", "days": 31 },
   { "abbreviation": "Feb", "name": "Febrero", "days": 28 },
   { "abbreviation": "Mar", "name": "Marzo", "days": 31 },
   { "abbreviation": "Abr", "name": "Abril", "days": 30 },
   { "abbreviation": "May", "name": "Mayo", "days": 31 },
   { "abbreviation": "Jun", "name": "Junio", "days": 30 },
   { "abbreviation": "Jul", "name": "Julio", "days": 31 },
   { "abbreviation": "Ago", "name": "Agosto", "days": 31 },
   { "abbreviation": "Sep", "name": "Septiembre", "days": 30 },
   { "abbreviation": "Oct", "name": "Octubre", "days": 31 },
   { "abbreviation": "Nov", "name": "Noviembre", "days": 30 },
   { "abbreviation": "Dic", "name": "Diciembre", "days": 31 },
]
```

6.5.2. Creación de JSON

```
// Declaración de un JSON vacío
let persona = {};

// Declaración de un JSON con datos de una persona
let persona = {
    nombre: 'Pablo',
    apellido: 'Fernández',
```

```
    estatura: 1.60,
    edad: 18,
    trabaja: true
}
```

Sin embargo, también es posible crearlo a través de su constructor:

```
let persona = JSON.constructor();
persona.nombre = 'Pablo';
persona.apellido = 'Fernández';
persona.estatura = 1.60;
persona.edad = 18;
```

6.5.3. Acceso a elementos de un JSON

```
console.log(persona['nombre']);                    // Devuelve 'Pablo'
console.log(persona.edad);                         // Devuelve 18
```

6.5.4. Inserción y almacenamiento de elementos en un JSON

```
persona.talla = "S";
persona['nombre'] = "Elena";
```

6.5.5. Eliminación de elementos de un JSON

```
delete(persona.talla);
console.log(persona);

// Devuelve lo siguiente:
{
    apellido: "Fernández"
    edad: 18
    estatura: 1.6
    nombre: "Elena"
    trabaja: true
    ► __proto__: Object
}
```

6.5.6. Envío y recepción de JSON

6.5.6.1. Método stringify

```
let objeto = {
    texto: 'valor',
    digito: 1
}
```

```
JSON.stringify(objeto);     // Devolverá '{"texto":"valor","digito":1}'
```

6.5.6.2. Método parse

```
let cadena = '{"texto":"valor","digito":1}';

let objeto = JSON.parse(cadena);
// Devuelve un String como:
{
    texto: "valor"
    digito: 1
    ► __proto__: Object
}

console.log(objeto.texto);  // Devolverá "valor"
```

6.6. ESPECIALES

6.6.1. El objeto window

6.6.1.1. Métodos

6.6.1.2. Propiedades

6.6.2. El objeto document

6.6.2.1. Métodos

6.6.2.2. Propiedades

6.6.3. El objeto Screen

6.6.4. La interfaz Navigator

6.6.5. La interfaz Location

6.6.6. La interfaz HTMLElement

6.6.6.1. Propiedades

6.6.6.2. Métodos más importantes

6.6.7. El objeto History

6.6.7.1. Método back

```
history.back();
```

6.6.7.2. Método forward

```
history.forward();
```

6.6.7.3. Método go

```
history.go(-2);
```

6.6.7.4. Método pushState

```
history.pushState(null, 'Listado de Productos', "./catalogo.html");
history.pushState(null, 'mosquis!', "http://google.es");
```

6.6.7.5. Método replaceState

```
history.replaceState(null, 'Mi Web', "/Productos/Catalogo");
```

6.6.7.6. Evento onpopstate

```
window.onpopstate = function(e){
    console.log(e.state)
}
```

6.6.7.7. Ejemplo de anulación del botón volver del navegador

```
let title = document.head.querySelector("title").innerHTML;

history.pushState(null, title, location.href);

window.onpopstate = function(e){
```

```
    history.forward();
}
```

6.6.8. El objeto this

```
function producto(a, b){
    let p = a * b;
    this.p = p;
}
producto.p = 0;

// Ejecutamos la función y mostramos su propiedad "p"
producto(2,3);
console.log(producto.p);                                    // Devolverá 0
```

```
let prod = new producto(2,3);
console.log(prod.p);                                        // Devolverá 6
```

```
function producto(a, b){
    let p = a * b;Ç
    producto.p = p;
}
producto.p = 0;

// Ejecutamos la función y mostramos su propiedad "p"
producto(2,3);
console.log(producto.p);                                    // Devolverá 6
```

```
function valorThis(){
    console.log(this == window);
}
valorThis();                                                // Devolverá true
```

```
function valorThis(){
    'use strict'
    console.log(this == window)
}
valorThis();                                               // Devolverá false
```

6.6.9. El objeto globalThis

```
let it = {name: 'IT', version: '1.0'}
it.imprimirContextos = function(){
    console.log("this: ", this);
    console.log("globalThis: ", globalThis);
}
```

```
let it = function(){ console.log('IT inicializado'); }
it.imprimirContextos = function(){
    console.log("this: ", this);
    console.log("globalThis: ", globalThis);
}
```

6.6.10. El objeto prototype

```
Date.prototype.littleEndianFormat = function () {
    const local = new Date(this);

    // Se calcula el diferencial GMT
    local.setMinutes(this.getMinutes()-this.getTimezoneOffset());

    let aux = local.toJSON().slice(0, 10);
    aux = aux.split('-')[2] + "-" +
          aux.split('-')[1] + "-" +
          aux.split('-')[0];

    return aux;
};

new Date().littleEndianFormat();
```

6.7. OTRAS COSAS QUE SABER SOBRE LOS OBJETOS DE JAVASCRIPT

6.7.1. La herencia

```
function Persona(nombre, apellidos, edad) {
    this.nombre = nombre + " " + apellidos;
    this.edad = edad;
};
```

```
// Método para recuperar la edad de la persona
Persona.prototype.getEdad = function() {
    alert(this.nombre + ' tiene ' + this.edad + ' años!');
};

// Método para recuperar el nombre de la persona
Persona.prototype.getNombre = function() {
    alert('Mi nombre es ' + this.nombre);
};
```

```
function Alumno(persona, curso, asignaturas) {
    for(let key in persona){
```

```
        this[key] = persona[key];
    }

    this.curso = curso;
    this.asignaturas = asignaturas;
};

function Profesor(curso) {
    for(let key in persona){
        this[key] = persona[key];
    }
    this.curso = curso;
};
```

```
Alumno.prototype = new Persona();
Alumno.prototype.constructor = Alumno;

Profesor.prototype = new Persona();
Profesor.prototype.constructor = Alumno;
```

```
let pablo = new Persona('Pablo', 'Fernández',18);
let alumno = new Alumno(pablo, '1º FP', '...');
```

```
console.log(pablo);
// devuelve lo siguiente:
{
    edad: 18
    nombre: "Pablo Fernández"
    ► __proto__: Object
}

console.log(alumno);

// Devuelve lo siguiente:
{
    asignaturas: "..."
    curso: "1º FP"
    edad: 18
    ► getEdad: f{}
    ► getNombre: f{}
    nombre: "Pablo Fernández"
    ► __proto__: Persona
}
```

6.7.2. Sentencias get y set

```
get ultimo() {
    if (this.log.length > 0) {
        return this.log[this.log.length - 1];
    } else {
        return "";
    }
}
```

```
set mensaje(mensaje) {
    this.log.mensaje(mensaje);
}
```

6.7.2.1. Ejemplo completo de get y set

```
let Historico = {
    get ultimo() {
        if (this.log.length > 0) {
            return this.log[this.log.length - 1];
        } else {
            return "";
        }
    },
    set mensaje(mensaje) {
        this.log.push(mensaje);
    }, log: []
}

// Imprimimos el ultimo valor
console.log(Historico.ultimo);                 // Devuelve ""

// Añadimos un mensaje
Historico.mensaje = "hola que tal";            // Devuelve "Hola que tal"

// Imprimimos el ultimo valor
console.log(Historico.ultimo);                // Devuelve "Hola que tal"
```

6.8. PRACTICA Y JUEGA

10

FUNCIONES EN JAVASCRIPT

7.1. CREACIÓN DE FUNCIONES

```
let fn = function(){
    console.log("Función definida como expresión")
}
```

```
function fn(){
    console.log("Función definida como expresión")
}
```

7.1.1. Diferencia entre modo estricto o modo no estricto

```
let mostrarContenidoThis = function(){
    console.log(this)
}

mostrarContenidoThis();             //       Devolverá       ►       Window
{postMessage:...}
```

```
let mostrarContenidoThis = function(){
    'use strict'
    console.log(this)
}

mostrarContenidoThis();                              // Devolverá undefined
```

7.2. PASO DE PARÁMETROS

7.2.1. Por asignación directa

```
alert("Página cargada");              // "Página cargada" es el argumento
```

7.2.2. El objeto arguments

7.2.2.1. Propiedad callee

7.2.2.2. Propiedad caller

7.2.2.3. Propiedad name

7.2.2.4. Propiedad length

```
let square = function(a){
    return a ** 2;
}

let op = function(op, v1, v2){
    console.log(arguments);
    if(op == "Square") return square(v1);
}

op("Square", 2);
```

```
Arguments(2) ["Square", 2, callee: ƒ, Symbol(Symbol.iterator): ƒ]
0: "Square"
1: 2
► callee: f (op, v1, v2)
    arguments: null
    caller: null
    length: 3
    name: "op"
    prototype: {constructor: ƒ}
    __proto__: f ()
length: 2
Symbol(Symbol.iterator): f values()
__proto__: Object
```

7.3. FUNCIONES ANÓNIMAS

```
function(){
    console.log("Página cargada");
}
(function(){
    console.log("Página cargada");
})();
```

7.3.1. Ventajas e inconvenientes

```
setTimeout(function(){
    console.log('Este mensaje está dentro de una función anónima')
}, 1000);
let x = 0;
let interval = setInterval (function(){
    console.log('Este mensaje está dentro de una función anónima')

    x++;

    if(x == 10) clearInterval(interval)
}, 1000);
```

7.4. FUNCIONES CLAUSURA

.

```
let Persona = function() {
    var _fname = new Array();

    function setName(val) {
      _fname['name'] = val;
    }

    function setSurname(val) {
      _fname['surname'] = val;
    }

    return {
        asignarApellidos: function(t) {
            setSurname(t);
        },
        asignarNombre: function(t) {
            setName(t);
        },
        mostrar: function() {
            return _fname['name'] + " " + _fname['surname'];
        }
    }
}

// Para utilizar esta función
let p = Persona();
    p.asignarNombre("Pablo");
    p.asignarApellidos("Fernández");

p.mostrar();                                  // Devuelve "Pablo Fernández"
```

7.4.1. Ventajas e inconvenientes

```
console.log(Persona())
```

```
console.log(p)

// Ambos devolverán lo mismo:

{asignarApellidos: ƒ, asignarNombre: ƒ, mostrar: ƒ}
    ► asignarApellidos: f (t)
    ► asignarNombre: f (t)
    ► mostrar: f (t)
    ► __proto__: Object
```

7.5. FUNCIONES FLECHA

```
let getName = () => "Pablo Fernández";
```

```
let getName = function(){
    return "Pablo Fernández"
}
let a = new Array(3, 40, 200, 5, 1);
a.sort((a, b) => {
    return a - b;
});
let ArrayOperations = function() {
    return Calculadora
}

ArrayOperations.sort = (arr) => arr.sort((a, b) => a - b );

ArrayOperations.max = (arr) => {
    return arr.reduce((a,b) => {
        return a >= b ? a : b;
    });
}

ArrayOperations.min = (arr) => {
    return arr.reduce((a,b) => {
        return a <= b ? a : b;
    });
}

ArrayOperations.addIVA = (arr, iva) => {
    return arr.map((a) => {
        return a * (1 + iva);
    });
}

let a = new Array(3, 40, 200, 5, 1);
console.log(ArrayOperations.sort(a));
console.log(ArrayOperations.max(a));
```

```
console.log(ArrayOperations.min(a));
console.log(ArrayOperations.addIVA(a, 0.21));
leerFichero().then(() => analizarFichero()).then(() =>
devolverResumen());
```

7.5.1. Ventajas e inconvenientes

7.6. FUNCIONES ESPECIALES

7.6.1. Función de prototipo bind

```
funcion.bind(nuevoThis, arg1, arg2, ..., argN);
```

7.6.1.1. Ejemplo

```
// Definimos la función saludar
let saludar = function(){
    alert("Hola " + this.nombre)
}

// Definimos el objeto Persona
function Persona(nombre, apellidos, edad) {
    this.nombre = nombre + " " + apellidos;
    this.edad = edad;
};

// Llamaaos a la función
saludar.bind(new Persona("Pablo", "Fernández", 18));
saludar.bind(new Persona("Pablo", "Fernández", 18))();
```

7.6.2. Función de prototipo call

```
funcion.call(nuevoThis, arg1, arg2, ..., argN);
```

7.6.2.1. Ejemplo

```
// Definimos la función saludar
let saludar = function(){
    alert("Hola " + this.nombre)
}

// Definimos el objeto Persona
function Persona(nombre, apellidos, edad) {
    this.nombre = nombre + " " + apellidos;
    this.edad = edad;
};
```

```
// Llamaaos a la función
saludar.call(new Persona("Pablo", "Fernández", 18));
```

7.6.3. Función de prototipo apply

7.6.3.1. Ejemplo

```
let saludar = function(){
    alert("Hola " + this.nombre)
}
function Persona(nombre, apellidos, edad) {
    this.nombre = nombre + " " + apellidos;
    this.edad = edad;
};

saludar.apply(new Persona("Pablo", "Fernández", 18));
```

7.6.4. Diferencias entre call y apply

```
function Empresa(nombre) {
    this.nombre = nombre || 'una empresa desconocida';
};

let saludar = function(nombre, apellidos){
    if(typeof apellidos == "undefined") apellidos = "";

    const msg1 = "Acabas de acceder a " + this.nombre;
    const msg2 = "Bienvenid@ " + nombre + ' ' + apellidos;

    alert(msg1 + '.<br/>' + msg2);
}
let empresa = new Empresa("Mi empresa");
saludar.call(empresa, 'Pablo', 'Fernández');
let empresa = new Empresa("Mi empresa");
saludar.apply(empresa, ['Pablo', 'Fernández']);
```

7.7. CONTEXTOS Y ENCAPSULAMIENTO

```
function sumaResta(a, b){
    let s = suma();
    let r = resta();

    function suma(){ let c = a + b; return c }
    function resta(){ let c = a - b; return c }
```

```
    return [s, r];
}

console.log(sumaResta(3, 2));
```

11

CLASES EN JAVASCRIPT

8.1. CREACIÓN DE CLASES

```
class Persona {
    constructor(nombre, apellidos) {
        this.nombre = nombre;
        this.apellidos = apellidos;
    }
}
```

```
let Persona = class {
    constructor(nombre, apellidos) {
        this.nombre = nombre;
        this.apellidos = apellidos;
    }
}
```

8.2. INSERCIÓN DE MÉTODOS

```
getNombre(){
    return this.nombre + " " + this.apellidos
}
```

```
static getList(){
    let personasList = globalThis.personas;
    for(let x in personasList){
        console.log(personasList [x]);
    }
}
```

8.2.1. Sentencias get y set

```
class Persona {
    get age(){
        return this.age;
    }

    set age(n){
        this.age = n;
    }
}

let p1 = new Persona("Pablo", "Fernández");
p1.age = 18;
```

```
class Persona {
    get age(){
        return this._age;
    }

    set age(n){
        this._age = n;
    }
}

let p1 = new Persona("Pablo", "Fernández");
p1.age = 18;
```

```
class Persona {
    constructor(nombre, apellidos) {
        this.nombre = nombre;
        this.apellidos = apellidos;
    }
```

```
    nombreCompleto(){ return this.nombre + " " + this.apellidos; }

    get edad(){ return this._age; }

    set edad(n){ this._age = n; }
}

// Creamos una nueva instancia
let p1 = new Persona("Pablo", "Fernández");

// Le asignamos la edad
p1.edad = 18;

// Recuperamos el nombre completo como propiedad
p1.nombreCompleto();
```

8.3. EXTENSIÓN DE CLASES

```
class Persona {
    constructor(nombre, apellidos) {
        this.nombre = nombre;
        this.apellidos = apellidos;
    }

    saluda(){
        return "Hola soy " + this.nombre + " " + this.apellidos;
    }
}

class Estudiante extends Persona {
    saluda(){
        return "Hola soy " + this.nombre + " " + this.apellidos +
        " y soy estudiante";
    }
}

// Creamos una nueva instancia
let e1 = new Estudiante("Pablo", "Fernández");

// Le pedimos que salude
e1.saluda();
```

8.3.1. Extensión a través de species

```
class String2 extends String {
    static get [Symbol.species]() { return String; }
}

let s = new String2("Esto es una prueba");
let r = s.toString();
```

```
console.log(s);                                    // Devuelve String2
console.log(r);                                    // Devuelve un String
console.log(s instanceof String);                  // Devuelve true
```

.

8.3.2. Extensión a través de super

```
class CustomArray extends Array{
    constructor(){
        super();
    }

    static get [Symbol.species]() { return Array; }

    sumarTodo(){
        return this.reduce(function(a, b){ return a + b });
    }
}

let ca = new CustomArray();
ca.push(1,1,2,3,5,8);
ca.sumarTodo();
console.log(a instanceof MyArray);                 // Devolverá true
console.log(a instanceof Array);                   // Devolverá true
class Persona {
    constructor(nombre, apellidos) {
        this.nombre = nombre;
        this.apellidos = apellidos;
    }

    saluda(){
        return "Hola soy " + this.nombre + " " + this.apellidos;
    }
}

class Estudiante extends Persona {
    saluda(){
        return super.saluda();
    }
}

// Creamos una nueva instancia
let e1 = new Estudiante("Pablo", "Fernández");

// Le pedimos que salude
e1.saluda();
```

8.4. CLASES ABSTRACTAS Y MIXINS

8.5. PRACTICA Y JUEGA

12

EVENTOS EN JAVASCRIPT

9.1. PRINCIPIO FUNDAMENTAL DE PROPAGACIÓN

```
event.stopPropagation();
```

9.2. EL OBJETO EVENT

9.2.1. Propiedades más frecuentes

9.2.1.1. Propiedad target

```
input.addEventListener('change', function (e){
    // Mostramos por consola el valor del input cuando cambia
    console.log(e.target.value);
});
```

9.2.1.2. Propiedad type

```
input.addEventListener('change', function (e){
    console.log(e.type);          // Devuelve 'change'
});
```

9.2.1.3. Propiedad bubbles

```
elemento.addEventListener('keydown', function (e){
    if(e.bubbles) console.log("El burbujeo está habilitado!");
});
```

9.2.1.4. Propiedad cancelable

```
// Mostrar si es cancelable en el momento de hacer click
elemento.onclick = function(e){ console.log(e.cancelable); });
```

9.2.1.5. Propiedad defaultPrevented

```
console.log(event.defaultPrevented);
```

9.2.1.6. Propiedad path

```
console.log(event.path);
```

```
KeyboardEvent {
    ...
    ► path: Array(5)
    0: input#fecha
    1: body.chrome
    2: html
    3: document
    4: window
}
```

9.2.1.7. Propiedades clientX, clientY, pageX y pageY

```
// Mostrar la posición del puntero en el momento de hacer click
elemento.onclick = function(event){
    console.log(event.clientX, event.clientY);
});
```

9.2.1.8. Propiedades keyCode y which

```
input.addEventListener('keydown', function (e){
    var code = e.keyCode | e.which;
    console.log(code); // Si pulsamos la tecla 'a' devuelve 65
});
```

9.2.1.9. Propiedades altKey, ctrlKey y shiftKey

```
input.addEventListener('keydown', function (e){
    var code = e.keyCode | e.which;
    console.log('Ctrl presionada:', e.ctrlKey, ', Código:', code);
```

```
});
```

9.3. LA INTERFAZ TOUCHEVENT

9.3.1. El objeto Touch

9.3.1.1. Propiedades más importantes

9.4. LA INTERFAZ KEYBOARDEVENT

9.4.1. Propiedades más importantes

9.5. LA INTERFAZ MOUSEEVENT

9.5.1. Propiedades más importantes

9.6. PRINCIPALES MANEJADORES DE EVENTOS

9.6.1. Eventos de ratón

.

9.6.2. Eventos de formulario

9.6.3. Eventos de HTML

9.6.4. Eventos de tratamiento táctil

9.7. OYENTES O LISTENERS

9.7.1. Método addEventListener

```
document.addEventListener('scroll', function (e){
    console.log(e)
});
```

9.7.2. Método removeEventListener

```
function addEvent(){
    ...
}

document.removeEventListener('scroll', addEvent);
```

9.7.3. Otras formas de establecer listeners

9.7.3.1. A través de HTML

```
<input type="number" onclick="console.log(Event)/>
```

9.7.3.2. A través de un objeto o elemento

```
input.onclick = function (e){ console.log(e) });
```

9.8. PRINCIPALES EVENTOS DEL DOM

9.8.1. Document DOMContentLoaded

```
function allReady(){
    console.log("El DOM está cargado!");
}

document.addEventListener("DOMContentLoaded", ready);
```

9.8.2. Window load

.

```
window.onload = function (e){
    console.log("Página cargada");
});
```

9.8.3. Window resize

```
window.onresize = function (e){
```

```
    console.log("El tamaño de la ventana cambió");
});
```

9.8.4. El evento scroll

.

```
window.onscroll = function (){
    console.log(document.body.scrollTop);
});
```

9.9. PRACTICA Y JUEGA

13

EL DOM DE JAVASCRIPT

10.1. PROCESO DE CARGA

10.2. LOS NODOS Y SUS TIPOS

10.3. SELECCIÓN DE ELEMENTOS

```
// Encontrar todos los elementos tipo tabla
document.getElementsByTagName("table");

// Encontrar todos los elementos que contengan la clase "card"
document. getElementsByClassName("card");

// Encontrar el elemento con id="cabecera"
document.getElementById("cabecera");
```

10.3.1. Interfaz NodeList

```
document.querySelectorAll("body");
```

10.3.1.1. Método item

```
document.querySelectorAll("body").item(0)
```

10.3.1.2. Método forEach

```
var list = document.querySelectorAll('body > *');
Array.prototype.forEach.call(list, function (element) {
    console.log(element)
});

// Otra manera, un poco más clara, de hacer lo mismo sería:
var list = document.querySelectorAll('body > *');
list.forEach(function (element) {
    console.log(element)
});
```

10.3.2. Los selectores

```
// Encontrar el primer elemento que contenga la clase button
document.querySelector(".button");

// Encontrar todos los elementos que contengan la clase button
document.querySelectorAll(".button");
```

10.3.3. Métodos para acceder a los nodos y elementos

10.3.3.1. Método querySelector

.

```
document.querySelector("table");
document.querySelector(".tarjeta");
document.querySelector("#cabecera");
```

10.3.3.2. Método querySelectorAll

```
document.querySelectorAll("input");
document.querySelectorAll(".button");
document.querySelectorAll(":checked");
```

.

10.4. MANUPULACIÓN DE NODOS Y ELEMENTOS

10.4.1. Interfaz DOMTokenList

.

10.4.2. Método createElement

```
var label = document.createElement("label");
```

10.4.3. Propiedad id

```
label.id = "nombre-label";
```

10.4.4. Propiedad innerHTML

```
console.log(document.body.innerHTML);
label.innerHTML = '<label for="name">Nombre de Usuario</label>';
```

10.4.5. Propiedad value

```
// Establecemos un valor en el elemento con ID "name"
document.querySelector("#nombre").value = "Pablo";

// Recuperamos su valor y lo mostramos por consola
console.log(document.querySelector("#nombre").value);
```

10.4.6. Método setAttribute

```
label.setAttribute("class", "clase-de-prueba");
```

10.4.7. Propiedad classList

```
var body = document.body;

// Añadimos la clase "Chrome" al body
body.classList.add("Chrome");

// Añadimos la clase "Firefox" al body, porque no la tiene
body.classList.toggle("Firefox");

// Eliminamos la clase "Chrome" de la etiqueta body
body.classList.remove("Chrome");

// Reemplazamos la clase "Firefox" por "IE" en la etiqueta body
body.classList.replace("Firefox", "IE");

// Si contiene la clase IE, la eliminamos, si no, no hacemos nada
body.classList.contains("IE") ? body.classList.remove("IE")  : ''
```

10.4.8. Propiedad previousElementSibling

```
var elem = document.querySelector("#cabecera");
```

```
console.log(elem.previousElementSibling);
```

10.4.9. Propiedad nextElementSibling

```
var elem = document.querySelector("#cabecera");
console.log(elem.nextElementSibling);
```

10.4.10. Propiedad parentElement

```
var elem = document.querySelector("#cabecera");
console.log(elem.parentElement);
```

10.4.11. Método appendChild

```
// Creamos un nuevo elemento
var label = document.createElement("label");

// Le asignamos un ID
label.id = "nombre-label";

// Le añadimos el atributo for
body.setAttribute("for", "nombre");

// Le añadimos al documento
document.body.appendChild(label);
```

10.4.12. Método insertBefore

```
// Imaginemos un documento con la siguiente estructura:
► body
  └ span

var nuevo = document.createElement("label");
var span = document.querySelector("#span");
span.parentElement.insertBefore(nuevo, span);

// la estructura quedaría:
► body
  ├ label
  └ span
var nuevo = document.createElement("label");
var span = document.querySelector("#span");
span.parentElement.insertBefore(nuevo, span.nextElementSibling);
// la estructura quedaría:
► body
```

```
  ├ span
  └ label
var nuevo = document.createElement("label");
var span = document.querySelector("#span");
span.insertBefore(nuevo, span.childNodes[0]);
```

```
// la estructura quedaría:
► body
  └ span
    └ label
```

10.5. ELIMINACIÓN DE NODOS Y ELEMENTOS

10.5.1. Método remove

```
label.remove();
HTMLElement.prototype.remove=function(){
    try{ this.parentElement.removeChild(this); } catch(e){}
}
```

10.5.2. Método removeChild

```
// Añadimos un elemento label
var label = document.createElement("label");
document.body.appendChild(label);

// Ahora eliminamos el elemento label
var parentNode = document.body;
parentNode.removeChild(label);
// Ahora eliminamos el elemento label
var parentNode = document.body;
var oldElement = parentNode.removeChild(label);
```

10.6. DEFINICIÓN DE ESTILOS

10.6.1. La interfaz CSSStyleDeclaration

```
$0.style = "width: 360px; height: 100vh;";
```

10.6.1.1. Propiedades

10.6.1.1. Métodos

10.6.2. La interfaz CSSStyleSheet

```
console.log(document.styleSheets);

// Devolverá algo parecido a:
► 0: CSSStyleSheet {ownerRule: null, cssRules: CSSRuleList, ...}
► 1: CSSStyleSheet {ownerRule: null, type: "text/css", ...}
► 2: CSSStyleSheet {ownerRule: null, type: "text/css", ...}
```

10.6.2.1. Propiedades

10.6.2.1. Métodos

10.6.3. Propiedad style

```
document.getElementById("id").style.display = "block";
console.log(document.getElementById("id").style.display);
document.getElementById("id").style.display = "";
```

10.6.4. Método insertRule

```
// Recuperamos la hoja de estilo
var s = document.styleSheets[0];

s.insertRule("label { color: red; } ", s.cssRules.length);
```

10.6.5. Método deleteRule

```
// Recuperamos la hoja de estilo
var s = document.styleSheets[0];

s.deleteRule(0);
```

10.7. PRACTICA Y JUEGA

Juego: Juego de Simon

14

JAVASCRIPT ASÍNCRONO

11.1. EL ESTÁNDAR CORS

11.1.1. Encabezados de solicitud HTTP

11.1.2. Encabezados de respuesta HTTP

11.2. CONEXIONES HTTP

11.2.1. Objeto XMLHttpRequest

11.2.1.1. Propiedades

11.2.1.2. Métodos

Método open

```
http.open("GET", "http://ejemplo.org/doc");
```

Método setRequestHeader

:

```
http.setRequestHeader("Content-type", " text/html; charset=utf-8");
```

Método send

```
http.send();
```

11.2.2. Eventos

11.2.3. Ejemplo sencillo de XMLHttpRequest

```
var http = new XMLHttpRequest();
http.open('GET', 'http://ejemplo.org/doc', true);
http.setRequestHeader("Content-type", " text/html; charset=utf-8");

http.onreadystatechange = function (aEvt) {
  if (http.readyState == 4) {
     if(http.status == 200)
      console.log(http.responseText);
     else
     console.log("Error al realizar la petición");
  }
};

http.send(null);
```

11.3. PROMESAS

11.3.1. Objeto Promise

```
if(!self.Promise) {
    console.log("El objeto Promise no está disponible");

    let f = document.createElement('script');
    f.src =
"//cdn.jsdelivr.net/npm/bluebird@3.7.2/js/browser/bluebird.min.js";

    f.onload = function() { continuar_la_carga(); };
```

```
}
```

```
var promise = new Promise(function(resolve, reject) {
    setTimeout(() => resolve((function(){
        alert("Promesa resuelta");
    })()), 1000);
});
var promise = new Promise(function(resolve, reject) {
    setTimeout(() => reject(new Error("Promesa Rechazada")), 1000);
});
```

```
▾ Promise {<resolved>}
    ► __proto__: Promise
      [[PromiseStatus]]: "resolved"
      [[PromiseValue]]: "Promesa resuelta"

▾ Promise {<rejected>: "Promesa Rechazada"}
    ► __proto__: Promise
      [[PromiseStatus]]: "rejected"
      [[PromiseValue]]: "Promesa Rechazada"
```

11.3.1.1. Consumidor then

```
var nuevaPromesa = new Promise((resolve, reject) => {
    setTimeout(function(){
        try {
            allert("¡Hola mundo!");
            resolve("¡Todo bien!");
        } catch(err) {
            reject(err.message);
        }
    }, 1000);
});

nuevaPromesa.then(
    (result) => { console.log(result); },
    (error) => { console.log(error); },
);
```

11.3.1.2. Consumidor catch

```
var nuevaPromesa = new Promise((resolve, reject) => {
    setTimeout(function(){
        try {
            allert("¡Hola mundo!");
            resolve("¡Todo bien!");
        } catch(err) {
            reject(err.message);
        }
    }, 1000);
});

nuevaPromesa.then((result) => { console.log(result); });
nuevaPromesa.catch((error) => { console.log(error); });
```

11.3.1.3. Consumidor finally

.

```
new Promise((resolve, reject) => {
  throw new Error("Se lanza excepción");
})
.finally(() => alert("Se ejecutó la promesa"))
.catch(err => alert(err));
```

11.3.2. La API fetch

```
if(!self.fetch) {
    console.log("La API fetch no está disponible");

    var f = document.createElement('script');
    f.src = "//raw.githubusercontent.com/github/fetch/master/fetch.js";
    f.onload = function() {
        continuar_la_carga();
    };

    document.head.appendChild(f);
}
fetch('http://www.islavisual.com/rss')
```

```
.then(function(response) {
    return response.text();
})
.then(function(myRSS) {
    console.log(myRSS);
});
```

11.3.2.1. Propiedad body

11.3.2.2. Propiedad cache

11.3.2.3. Propiedad credentials

11.3.2.4. Propiedad headers

```
headers:{ 'Content-Type': 'application/json' }
```

11.3.2.5. Propiedad mode

11.3.2.6. Propiedad method

11.3.2.7. Ejemplo completo de solicitud en modo texto plano

```
var request = new Request("https://www.islavisual.com/rss", {
    method: 'POST',
    mode: 'cors',
    credentials: 'omit',
    cache: 'no-cache',
    referrerPolicy: 'no-referrer'
});

fetch(request)
.then(function(response) { return response.text(); })
.then(function(data) { console.log(data); })
.catch(function(err) { console.error(err); });
```

11.3.2.8. Ejemplo completo de solicitud con JSON

```
var request = new Request("http://ejemplo.com/municipios.json", {
```

```
    method: 'GET',
    mode: 'cors',
    credentials: 'include',
    cache: 'default',
    referrerPolicy: 'no-referrer'
});

fetch(request)
.then(function(response) {
    console.log('response.body =', response.body);
    console.log('response.bodyUsed =', response.bodyUsed);
    console.log('response.headers =', response.headers);
    console.log('response.ok =', response.ok);
    console.log('response.redirect =', response.redirect);
    console.log('response.status =', response.status);
    console.log('response.statusText =', response.statusText);
    console.log('response.type =', response.type);
    console.log('response.url =', response.url);
    return response.json();
});
.then(function(json) { console.log('Municipio', json[0].name); })
.catch(function(err) { console.error(err); });
response.body = ReadableStream {locked: false}
response.bodyUsed = false
response.headers = Headers {}
response.ok = true
response.redirect = undefined
response.status = 200
response.statusText = OK
response.type = cors
response.url = http://datosejemplo.com/municipios.json
Municipio "Amurrio"
```

11.4. PRACTICA Y JUEGA

15

VALIDACIÓN DE FORMULARIOS CON JAVASCRIPT

12.1. PROPIEDADES DE LOS FORMULARIOS

```
for(var i = 0; i < document.forms.length; i++){
    console.log(document.forms[i]);
}
document.forms[0].elements[0];
var formName = document.forms.nombreFormulario;
var formID = document.forms.idFormulario;
```

```
var formName = document.querySelector("[name='nombreFormulario']");
var formID = document.getElementById("idFormulario");
```

12.2. PROPIEDADES DE LOS ELEMENTOS DE FORMULARIO

12.3. CREACIÓN Y ENVÍO DE FORMULARIOS

Primero definimos el elemento que actuará como formulario y sus propiedades:

```
// Creamos el elemento de formulario
var f = document.createElement("form");

// Establecemos el método de envío y su URL destino
f.setAttribute('method',"post");
f.setAttribute('action',"cambiarcontraseña.html");
// Añadimos un elemento de formulario
var i = document.createElement("input");
i.setAttribute('type',"email");
i.setAttribute('name',"email");

// Añadimos el botón para envío
var b = document.createElement("button");
b.setAttribute('type',"submit");
b.setAttribute('value',"Enviar");
// Añadimos el input y el button al formulario
f.appendChild(i);
f.appendChild(b);

// Finalmente, lo añadimos al body
document.body.appendChild(f);
// Enviamos el formulario ahora
f.submit();
```

12.4. VALIDACIÓN DE FORMULARIOS

12.4.1. La interfaz ValidityState

12.4.2. Propiedades y métodos

12.4.3. Eventos

12.4.4. Ejemplo de validación

.

```
<h1>Prueba de validación</h2>
<form name="frm" enctype="multipart/form-data" method="get">
    <label>
```

```
        Estado:
        <input id="status" type="text" oninput="check(this)" />
    </label>

    <h2>Mensaje tras validación</h2>
    <div id="validateMsg"></div>

    <h2>Listado de errores de validity</h2>
    <div id="validity"></div>
</form>
function check(input) {
    var val = input.value;
    var vm  = document.getElementById("validateMsg");
    var va  = document.getElementById("validity");

    if (input.value != "Asignado" &&
        input.value != "En Progreso" &&
        input.value != "Finalizado") {
        var msg = '"' + val + '" no es un estado.';
        input.setCustomValidity(msg);
        vm.innerHTML = input.validationMessage;
        va.innerHTML = '';
        for(var key in input.validity){
            var status = input.validity[key];
            if(status){
                va.innerHTML += key + ": " + status.toString();
                va.innerHTML += "<br/>";
            }
        }

    } else {
        // Todo correcto.
        // Eliminamos el mensaje y el listado.
        input.setCustomValidity('');
        vm.innerHTML = "";
        va.innerHTML = "";
    }
}

var s = document.getElementById("status");
```

```
s.addEventListener("invalid", check, true);
```

12.5. PRACTICA Y JUEGA

REFERENCIAS

Casado, P. E. (2020). *Diseño y Construcción de Páginas Web.* Ra-Ma.

Casado, P. E. (2020). *Domine JavaScript 4ª Edición.* Ra-Ma.

Mozilla.org. (Marzo de 2023). *https://developer.mozilla.org/es/docs/Web/*. Obtenido de https://developer.mozilla.org/es/docs/Web/.

W3C Schools. (2023, Febrero). *HTML The language for building web pages*. Retrieved from HTML The language for building web pages: https://www.w3schools.com/

World Wide Web Consortium. (2023, Febrero). *W3C*. Retrieved from https://www.w3.org