**Cliente-servidor**

Cliente hace pedidos

Servidor responde

Backend: Js, node, express, apache, MySGL, MongoDB, PHP secuelize

**Web e internet**

**Web:**es un sistema de distribución que puede interconectar documentos que contengan multimedia (esta adentro de la compu). Se trasmiten a agtraves de HTTP (hipertex tranfer protocol ) con el lenguaje HTML (hipertext mark languaje)

**Internet**: es la estructura que nos permite mantener conectada la computadora a una red internacional, es descentralizada que conecta computadoras y servidores entre si. Utiliza protocolos TPC/IP.

**HTML**

**Listas**

**Ordenadas:**

<ol **type**=”1/A/a/”(otra posibilidad es **start**=”20”>

<li> primero</li>

<li>segundo</li>

</ol>

**Desordenandas:**

<ul type=”none/disc/square/circle”>

<li> primero</li>

<li>segundo</li>

</ul>

**Anidadas**

<ul type=”none/disc/square/circle”>

<li> Recordar para el viaje

<ol **type**=”1/A/a/”(otra posibilidad es **start**=”20”>

<li> primero</li>

<li>segundo</li>

</ol>

</li>

<li>llamar por tel </li>

<li>segundo</li>

</ul>

**Listas detalladas**

<dl>

<dt>Beast of Bodmin</dt>

<dd>A large feline inhabiting Bodmin Moor.</dd>

<dt>Morgawr</dt>

<dd>A sea serpent.</dd>

<dt>Owlman</dt>

<dd>A giant owl-like creature.</dd>

</dl>

**Enlaces**

<a href=<https://www.google.com>( otra posibilidad es “inicio.html”)> Vamos a Google</a>

**Imágenes**

<img width=”320” heigth=”500” src=”img/fotoperfil.jpg” alt=”foto perfil”>

**CSS**

<link rel=”stylecheet” href=”css/style.css”>

**Selector ID**

<h3 id=”saludo”>Hola</h3>

#saludo {color: blue;}

**Selector de clase**

<h3 class=”noticia”>Hola</h3>

.noticia {color: blue;}

**Selectores combinados**

<h3 class=”noticia”>Hola</h3>

h3.noticia {color: blue;}

**Selectores descendentes: atrapan los li que esten dentro de ul con el id lista**

<ul id =”lista”>

<li> primer item</li>

</ul>

ul#lista li {text-align:center;}

**Formato de texto**

Font-family/Font-size/Font-style/Font-weigth/text-align/text-decoration/line-heigth

@import url(‘https://fonts.googleapis.com/css2?family=Roboto&display=swap’)

**Color**

Hexadecimal:#f05331

RGB: rgb(255, 100, 50)

RGBA:rgba(122,50,125, 0.5)

Background-color

Opacity: 0.5

Background-image: url(‘../img/bibi.jpg’)}

Background-repeat: repeat-5

Background-position: right top

Background-attachement: fixed, scroll, inherit, inicial

Bacckground-size: 500px

**Iconos**

<link rel=”stylesheet” href=<https://use.fontwasome.com/releases/v5.15.2/css.all.css>>

<i class=”fas fa-paw”></i>

**Cajas**

Contenido

Padding: espacio entre el contenido y el borde

Border: solid/dotted/dashes/double 3px yellow} (entre el contenido y el margen)

Margin: separa una caja de otra

Box-sizing: content-box (solo el contenido)

\*{Box-sizing: border-box;} (el contenido, padding y borde)

{Position: sticky; top:15px}

Position:absolute (se mueve en relación al body, cuando lo movamos los otros elementos van a ocupar su lugar)

Position: relative ( se mueve en relación a su posición original, cuando lo movamos el resto de los elementos no se va a mover)

Position: fixed ( se mueve en relación a los costados del navegador, aunque hagamos scroll siempre queda en el mismo lugar)

z-index: para poner imagen delante de otra

**FlexBox**

Contenedor padre { display: flex;

* Flex wrap: wrap/nowrap/wrap reverse; (permite que los elementos mantengan el tamanio original o no y pasen a la línea de abajo)}
* Flex direction: row/columna
* Main axis: justify content: flex start/flex end/center/space between/space around
* Cross Axis: align ítems: strech, flex end/flex start/center
* Align-content: cuando el contenedor tiene multiples líneas, los alinea a lo largo del cross axis

**Items dentro de flex**

.caja-d {order: -1 (pasa al principio de la fila); 1 (pasa al final de la fila)

Flex-grow: 1;

Align-self: permite alinear algun item en particular a lo largo del cross axis (flex start/flex end/center/stretch)

**Diseño adaptativo**

**Viewport**: parte visible del navegador (etiqueta <meta name=”viewport” content=”width=device-width, initial-scale=1”>

**Medidas relativas**:

Porcentajes: siempre relacionada con la medida del elemento padre que la contiene (no usar para heigth)

ems: relativos al font-size del elemento padre, tomaran valor de referencia al **Font-size** (16px=1 em)

rems: siempre toman de base el tamanio de Font-size del elemento html. No se afecta por el elemento padre. Siempre mejor usar rem que em

**vw**: relativo al ancho del Viewport, son porcentajes

**vh** relativo al alto total del viewport, son porcentajes

**Media queries:** permiten que nuestros documentos html se conviertan en estructuras adaptativas o responsive

Recomendado:mobile first. Breakpoints medida de quiebre a cada salto de tamanio (0-480, 481-768, 769-1279, +1280)

@media (min-width o max-width:460 o 748px) and (orientation:landscape/portrait) {

Body {background:red;

}

}

**Formularios**

Permiten la entrada de datos por parte del cliente. Capturan: html, validan Js+node.js, procesa Node.js

Etiquetas de formularios: form tiene action(ruta donde se van a guardar los datos) y method (define como se envia la información)

<form action=”/registro” method=”GET o POST”>

<label for=”debe ir lo mismo que en id de input”>Nombre </label> (asocia texto descriptivo al formulario)

<input type=”text/password/email/tel/number” name=”nombre de la variable” placeholder=”pone el campo con letra gris” ”id=” idem for de label” required oninvalid=”setCustomValidity(‘por favor llenar con opción correcta’)”>

(ej en type=”tel” se puede usar pattern=” [0-9]{3}-[0-9]{3})

<textarea (acepta multiples líneas) name=”cometario” id=”comentario”>Deje aquí su comentario</textarea>

<select name=”color”> (permite elegir opciones)

<option value=”#codigo color” selected (predefine la opción)> Rojo</option>

<option value=”#codigo colot”> Verde</option>

</select>

<button type=”reset o submit”>Cancelar o enviar </button>

</form>

**Radio-button**

<label> Medio de pago</label>

<input type=”radio” name=”mediodepago” value=”efectivo”>

<input type=”radio” name=”mediodepago” value=”tarjeta de credito”>

<input type=”radio” name=”mediodepago” value=”tarjeta de debito” checked (opción preseleccionada)>

**Checkbox** : se pueden elegir varias opciones/también sirve para aceptar una condición (ej terminos)

<input type=”checkbox” name=”hobbie” value=”musica”>

<input type=”checkbox” name=”hobbie” value=”patines” >

**Fechas**

<input type=”date” name=”fecha” value=”2021-07-22” min=”2018-01-01” max=”2022-12-31”>

**Archivos**

<input type=”file” name=”avatar” accept=”.jpg” multiple (permite agregar mas archivos)>

**Fieldset:** nos permite agrupar varios elementos del formulario

<fieldset>

<legend>Tipo de equipo </legend> <br>

<input type=”radio” name=”cualquiera” id=”pc” value=”radiobutton>

<label id=”pc”>PC</label>

<input type=”radio” name=”cualquiera” id=”linux” value=”radiobutton>

<label id=”linux”>Linux</label>

</fieldset>

**Optgroup**: agrupa las opciones de selección

<select>

<optgroup label=”Ferrari”>

<option> spider</option>

<option> Berlineta</option>

<option> California</option>

</optgroup>

<optgroup label=”Porche”>

<option> macam</option>

<option> spy</option>

</optgroup>

</select>

**Datalist:** permite elegir entre las opciones o poner una propia

<label for=”mascota”>Seleccione mascota</label>

<input type=”text/range/color/date” id=”mascota” name=”masc” list=”mascotas”>

<datalist id=”mascotas”>

<option value=”perro”>

<option value=”gato”>

<option value=”hamster”>

</datalist>

**Pseudoselectores**

**Pseudoclases:**Nos permiten crear estilos en function de la estructura de HTML. Primero el selector luego : luego la pseudoclase

:Link (cualquier <a> que aun no se ha visitado)

:Hover (cuando el usuario interactúa con un elemento con el mouse, pero no necesariamente lo active)

:Visited (cuando ya se visito un sitio)

:Active (para <a> y <button>, realza el commando cuando se lo clickea)

<input> :Focus (Ej cuando se esta completando un formulario se marca)

:Enable

:Disable

:target

**Pseudoelementos:** Nos permite crear nuevos elementos a partir de CSS. Primero el selector :: luego el pseudoelemento (before, after, first-letter, first-line)

::before o after{

Content: “contenido a agregar”;

Color:blue }

**Animaciones**

* Captar la atención de los usuarios,
* Presentar los productos y/o servicios,
* Ofrecer un descuento especial,
* Distraer al visitante mientras se carga la web,
* Abrir nuevosbloques de información, etc…

@keyframe fantasma { 0% {opacity:0%}

25% {Transform:translatey 25px; opacity 100%}

100% {opacity:0%} }

**Forma abreviada**

.fantasma { animation: fantasma 3s infinite;}

Circulo que se agranda y achica

mi-elemento {

width: 200px;

height: 200px;

position: relative;

border-radius: 100%;

animation-iteration-count: infinite;

animation-direction: alternate;

animation-name: achicando; (igual nombre que el @keyframe)

animation-duration: 1000ms;}

@keyframes achicando {

0% { background-color: salmon; }

100% { background-color: peachpuff;

height: 100px;

width: 100px;

opacity: 0.5; }

}

**Transform**

Permite transformar un elemento de CSS en bidimensional o tridimensional

Transform: scale(2)/skew(15deg)/rotate(180deg)/translate(50px)/transform-origin(modifica del origen de las transformaciones)

**Transition**

.mi-elemento {

Transition: border-radius 450ms;}

.mi-elemento::hover {border-radius:50%;}

Transition-property: propiedades donde se va a aplicar la transición

Transition-duration: tiempo que dura la transicion