Universidad Autónoma de Baja California

Aplicaciones web

JavaScript & browser

Profesor: MC. Itzel Barriba Cázares

JavaScript & el navegador

- Los navegadores web muestran documentos HTML. Si desea que un navegador web ejecute código JavaScript, debe incluir (o hacer referencia) a ese código de un documento HTML.
- La etiqueta utilizadas es <script>.
- El código de JavaScript se puede incrustar directamente, es mas común usar el atributo sec de la etiqueta script para especificar la URL de un archivo que contiene código JavaScript.

JavaScript & el navegador

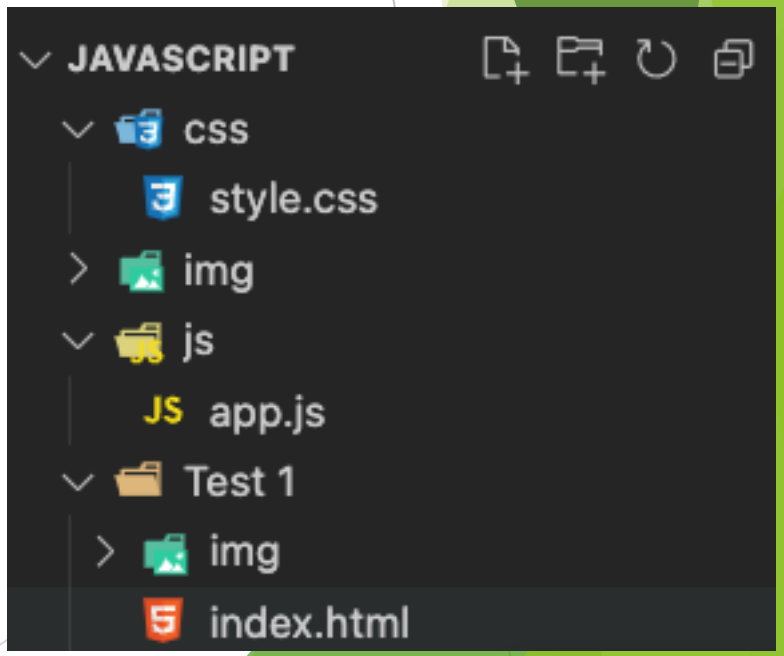
- JavaScript puede cambiar todos los elementos HTML de la página.
- JavaScript puede cambiar todos los atributos HTML de la página.
- JavaScript puede cambiar todos los estilos CSS de la página.
- JavaScript puede eliminar elementos y atributos HTML existentes
- JavaScript puede agregar nuevos elementos y atributos HTML
- JavaScript puede reaccionar a todos los eventos HTML existentes en la página.
- JavaScript puede crear nuevos eventos HTML en la página

JavaScript & el navegador

- Por lo tanto, se extrae el código JavaScript del archivo HTML y lo almacenamos en su propio archivo script siguiendo la siguiente estructura de archivos, entonces la etiqueta <script> debe hacer referencia a ese archivo de código de esta manera:

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

- Un archivo JavaScript contiene JavaScript puro, sin etiquetas <script> ni ningún otro HTML.
- Por convención, los archivos deben terminar en .js.



Hola JavaScript

- index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>JavaScript App</title>
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <h1>App Name</h1>
    <p>App Description</p>
    <script src="js/app.js"></script>
</body>
</html>
```

- app.js

```
console.log("Hola, JavaScript");
```

- style.css

```
*{
    margin: 0;
    padding: 0;
}
```

DOM

- La API para trabajar con documentos HTML se conoce como Modelo de objetos de documento o DOM.
- Los documentos HTML contienen elementos HTML anidados unos dentro de otros, formando un árbol.

```
<html>
  <head>
    <title>Sample Document</title>
  </head>
  <body>

  <h1>An HTML Document</h1>
  <p>This is a <i>simple</i> document.
</body>
</html>
```

- Para cada etiqueta HTML en el documento, hay un objeto elemento JavaScript correspondiente y para cada ejecución de texto en el documento, hay un objeto Texto correspondiente.

DOM

- La representación de DOM de este documento es el árbol

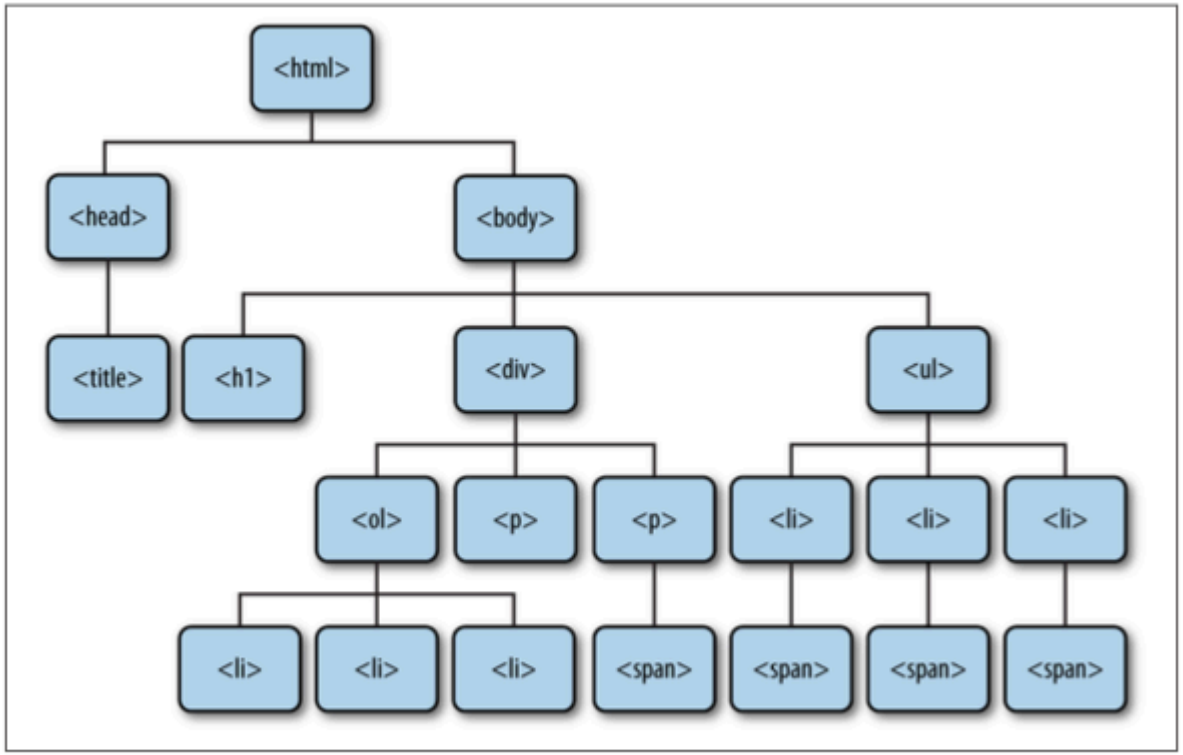
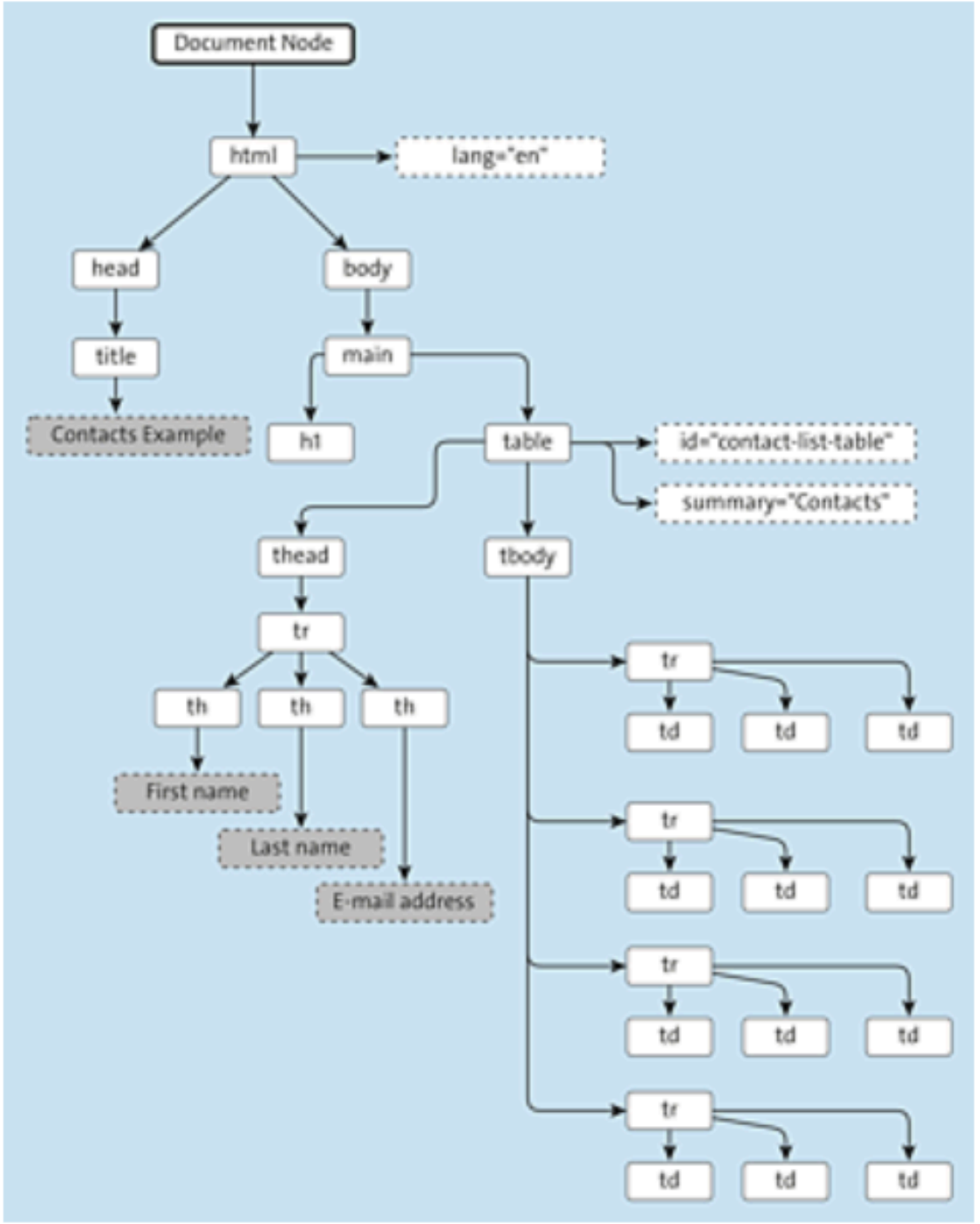


Figure 2-5. A tree representation of the preceding DOM

DOM

```
<!DOCTYPE html>
<html>
    <head lang="en">
        <title>Contacts Example</title>
    </head>
    <body>
        <main id="main">
            <h1>Contacts</h1>
            <table id="contact-list-table" summary="Contacts">
                <thead>
                    <tr>
                        <th id="table-header-first-name">First name</th>
                        <th id="table-header-last-name">Last name</th>
                        <th id="table-header-email">E-mail address</th>
                    </tr>
                </thead>
                <tbody>
                    <tr class="row odd">
                        <td>John</td>
                        <td>Doe</td>
                        <td>john.doe@javascripthandbuch.de</td>
                    </tr>
                    <tr class="row even">
                        <td>James</td>
                        <td>Doe</td>
                        <td>james.doe@javascripthandbuch.de</td>
                    </tr>
                    <tr class="row odd">
                        <td>Peter</td>
                        <td>Doe</td>
                        <td>peter.doe@javascripthandbuch.de</td>
                    </tr>
                    <tr class="row even">
                        <td>Paul</td>
                        <td>Doe</td>
                        <td>paul.doe@javascripthandbuch.de</td>
                    </tr>
                </tbody>
            </table>
        </main>
    </body>
</html>
```



DOM

- El nodo de documento, es el punto de entrada para el DOM y está representado por el objeto global del documento, que tiene varias propiedades y métodos.

Property	Description
`document.title`	Contains the title of the current document
`document.lastModified`	Contains the date when the document was last modified
`document.URL`	Contains a URL of the current document
`document.domain`	Contains the domain of the current document
`document.cookie`	Contains a list of all cookies for the document
`document.forms`	Contains a list of all forms in the document
`document.images`	Contains a list of all images in the document
`document.links`	Contains a list of all links in the document

Table 5.1 Selected Properties of the document Object

Seleccionar elementos

- Ya sea que desee cambiar información existente en una pagina web o agregar nueva información, primero debe seleccionar un elemento en la página web que desea cambiar o al que desea adjuntar la nueva información.

Property/Method	Description	Return Code
`getElementById()`	Selects an element based on an ID	Single element
`getElementsBy-ClassName()`	Selects elements based on a class name	List of elements
`getElementsBy-TagName()`	Selects all elements with the specified element name	List of elements
`getElementsBy-Name()`	Selects elements by their name	List of elements
`querySelector()`	Returns the first element that matches a given CSS selector	Single element

Seleccionar elementos

Property/Method	Description	Return Code
`lastElementChild`	Returns the last child element for a node	Single element
`lastChild`	Returns the last child node for a node	Single node
`childNodes`	Returns all child nodes for a node	List of nodes
`children`	Returns all child elements for a node	List of elements

Property/Method	Description	Return Code
`querySelectorAll()`	Returns all elements that match a given CSS selector	List of elements
`parentElement`	Returns the parent element for a node	Single element
`parentNode`	Returns the parent node for a node	Single node
`previousElement-Sibling`	Returns the previous sibling element for a node	Single element
`previousSibling`	Returns the previous sibling node for a node	Single node
`nextElementSibling`	Returns the next sibling element for a node	Single element
`nextSibling`	Returns the next sibling node for a node	Single node
`firstElementChild`	Returns the first child element for a node	Single element
`firstChild`	Returns the first child node for a node	Single node

Metodos y propiedades

- Los métodos HTML DOM son acciones que puede realizar (en elementos HTML)
- Las propiedades HTML DOM son valores (de elementos HTML) que puede configurar o cambiar.
- En el DOM todos los elementos HTML se definen como objetos
- Una propiedad es un valor que puede obtener o establecer (como cambiar el contenido de un elemento HTML).
- Un método es una acción que puedes realizar (como agregar o eliminar un elemento HTML)

Seleccionando Elementos by ID

- A los elementos de una página web se les puede asignar una identificación (que es única en esa página web) a través del atributo id.
- Este ID se puede usar en reglas CSS y puede seleccionar el elemento correspondiente a través de JavaScript con el método getElementById() del objeto del documento.

Eventos

- Un evento en JavaScript es una acción realizada por un visitante.
- Un visitante puede pasar el mouse sobre una imagen, ingresar datos en un formulario o hacer clic en un enlace.
- Un controlador de eventos es un script que se escribe para reaccionar ante un evento.
- Los controladores de eventos para eventos particulares siempre inician con on, seguido del nombre del evento.
- Los controladores de eventos especializados son específicos de elementos específicos. El elemento FORM, tiene controladores de formulario onSubmit, y onReset. Esto significa que alguien ha hecho clic en los botones Enviar y Restablecer.

Eventos del Mouse

Alphabetical List of Event Handlers

Event	Meaning
onAbort	A user clicks the Stop button while an image is loading.
onBlur	An element loses focus.
onChange	An element changes.
onClick	The mouse button is pressed and released once.
onDblClick	The mouse button is pressed and released twice in rapid succession.
onError	An error occurs.
onFocus	An element receives focus.
onKeyDown	A key is pressed.
onKeyPress	A key is pressed and released.
onKeyUp	A pressed key is released.
onLoad	The Web page is loaded into the browser.
onMouseDown	The mouse button is pressed.
onMouseMove	The mouse pointer moves.
onMouseOut	The mouse pointer moves off the object.
onMouseOver	The mouse pointer moves over the object.
onMouseUp	The mouse button is released.
onMove	A window is moved.
onReset	A form is reset.
onResize	A window is resized.
onSelect	Text is selected in a text box or text area.
onSubmit	A form is submitted.
onUnload	The browser moves on to another page.

Ejemplo(1)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript App</title>
    <script>
        function showOver() {
            document.outputForm.readItHere.value = "Over";
        }
        function showOut() {
            document.outputForm.readItHere.value = "Out";
        }
    </script>
</head>
<body>
    <h1>App Name</h1>
    <p>App Description</p>
    <img src="img/imagen1.jpeg" alt="JavaScriptTest"  onMouseOver="showOver()"
onMouseOut="showOut()">
    <form id="outputForm" name="outputForm" >
        <input type="text" id="readItHere" name="readItHere" value="Nada">
    </form>
</body>
</html>
```

JavaScript asincrónico

- Los programas JavaScript en un navegador web suelen estar controlados por eventos, lo que significa que esperan a que el usuario haga clic o toque antes de hacer algo. Y los servidores basados en JavaScript normalmente esperan a que lleguen las solicitudes de los clientes a través de la red antes de hacer algo.
- Los promises, son objetos que representan el resultado aún no disponible de una operación asincrónica.

Callbacks

- La programación asincrónica en JavaScript se realiza con callbacks. Un Callback es una función que escribes y luego pasas a alguna otra función. Esa otra función luego invoca (“callbacks”) a su función cuando se cumple alguna condición o ocurre algún evento (asincónico).
- La invocación del la función callback que tu proporciona le notifica la condición o algún evento (asincrónico), la invocación incluirá argumentos de función que proporcionan detalles adicionales.

Timers

- Uno de los tipos mas simples de asincronia es cuando se desea ejecutar algún código después de que haya transcurrido un cierto periodo de tiempo.

```
setTimeout(checkForUpdates, 60000);
```

- El primer argumento es una función y el segundo es un intervalo de tiempo en milisegundos
- setTimeout() llama a la función de devolución de llamada especificada una vez, sin pasar ningún argumento, y luego se olvida.

Eventos

- Los programas JavaScript del lado del cliente están controlado por eventos.
- Normalmente esperan a que el usuario haga algo y luego respondan a sus acciones.
- El navegador web genera un evento cuando el usuario presiona una tecla en el teclado, mueve el mouse, hace clic en un botón del mouse o toca un dispositivo con pantalla táctil.
- Los programas JavaScript controlados por eventos registran funciones de devolución de llamada para tipos específicos de eventos en contextos específicos, y el navegador web invoca esas funciones cada vez que ocurren los eventos especificados.
- Estas funciones de devolución de llamada se denominan controladores de eventos o detectores de eventos y se registran con addEventListener():