Cápsula 3: DOM

Hola, bienvenidxs a una cápsula del curso Visualización de Información. En esta hablaré sobre el DOM y como utilizar JavaScript para manipular una página web.

DOM es sigla para Document Object Model, y es una forma de representar el contenido de una página web como un objeto de programación. Con este modelo en mente, seremos capaces de utilizar JavaScript para afectar el contenido del documento mismo.

Como recordarás, el contenido de una página es un documento HTML, donde todo se escribe entre etiquetas. Todo se encontraba dentro de etiquetas “html”, y mediante composición de etiquetas, encontramos más y más definiciones y elementos.

Lo que define el DOM, es que toda etiqueta dentro del “html”, incluyendo “html”, es un nodo. Y si una etiqueta está directamente dentro de otro par de etiquetas, como “head” dentro de “html” y “body” dentro de “html”, entonces sus nodos asociados son hijos directos del nodo de “html”.

Mediante esta relación padre-hijo de nodos, es posible formar un árbol, donde en la raíz está el objeto que representa el documento completo “document”, y bajando por las ramas, podemos encontrar el contenido completo. Incluso el texto y atributos de etiquetas HTML. Mediante esta representación y definiciones de objetos y funciones de programación, podemos afectar el contenido del documento mediante JavaScript.

Por ejemplo, el código en pantalla usa el objeto “document” y accede a una propiedad llamada URL, y la imprime. Esto imprimiría la dirección del documento. Luego, accede a la propiedad children, que entrega la colección de elementos que son hijos directos de ese nodo, que en este caso es el nodo del elemento “html”.

Podríamos acceder a todo el contenido así, pero también es posible acceder a ellos mediante funciones de “document”. El método “getElementById” busca un elemento en todo el documento que tenga el identificador proporcionado como atributo, y lo retorna.

También es posible crear nuevos elementos. El método “creteElement” crea un nuevo elemento utilizando el tipo indicado, en el ejemplo crea un elemento de párrafo. Podemos agregarle contenido a este párrafo, creando un nodo de texto con “createTextNode”. Para enlazar el nodo de texto al párrafo, hay que indicar que es hijo suyo, así siguiendo el modelo DOM. Podemos hacer eso mediante “appendChild” (o agregar hijo).

Ahora, de la misma forma que con el texto, este párrafo aún no es parte del documento a menos de que lo indiquemos. Una forma es agregando como hijo de un elemento que ya es parte del documento. En el código mostrado, se crea el elemento de párrafo, y va a buscar un elemento cuyo identificador es “raiz”, y agrega el párrafo como hijo directo.

Veamos el código en la práctica. En pantalla está el programa de ejemplo, y el documento HTML que lo importa. Si te fijas, no hay contenido dentro de body, solo se especifica su identificador mediante la palabra “raiz”. Al abrir el documento en un navegador, vemos el resultado del párrafo dentro del documento, realizado mediante programación.

Este tipo de programación permite de todo. Es posible alterar el nombre de las clases de elementos, de tal forma de que sean afectados por reglas de CSS o no. También se pueden asignar identificadores y otras funcionalidades.

Otra funcionalidad importante es la capacidad de agregar interacción mediante eventos. El método addEventListener permite agregar comportamiento que se gatilla cuando ocurre algo. Si ocurre el evento indicado sobre el elemento, entonces se llama la función asignada.

En este ejemplo, si se hace clic sobre el elemento principal, llama a esta función que crea un párrafo que indica un contador del número de clics realizados, y luego agrega el párrafo como hijo de un elemento “raiz” en el documento. Podemos verlo en funcionamiento aquí. Donde al cliquear efectivamente aparecen párrafos en el documento.

Con eso termina el contenido de esta cápsula. Recuerda que si tienes preguntas, puedes dejarlas en los comentarios del video para responderlas en la sesión en vivo de esta temática. ¡Chao!