HTML DOM

FACULTY OF INFORMATION TECHNOLOGY

Universitas Advent Indonesia

Verse of the day

Colossians 3:23

Whatever you do, work at it with all your heart, as working for the Lord, not for human masters,

FACULTY OF INFORMATION TECHNOLOGY
Universitas Advent Indonesia

HTML DOM

- The HTML DOM (Document Object Model) is a standardized way to access and manipulate the elements of an HTML document.
- It represents the document as a tree-like structure, with each element in the document represented as a node in the tree.

FACULTY OF INFORMATION TECHNOLOGY
Universitas Advent Indonesia

Accessing HTML Element

- Using document.getElementById() to access an element by its id attribute
- Using document.getElementsByTagName() to access a group of elements by their tag name
- Using document.querySelector() and document.querySelectorAll() to access elements using CSS-style selectors

FACULTY OF INFORMATION TECHNOLOGY
Universitas Advent Indonesia

Accessing HTML Element

Using document.getElementById() to access an element by its id attribute

document.getElementById(): This method allows you to access an element by its id attribute. For example, given the following HTML:

```
<div id="myDiv">This is my div</div>
```

You can access the div element using the following JavaScript:

```
const myDiv = document.getElementById('myDiv');
```

FACULTY OF INFORMATION TECHNOLOGY
Universitas Advent Indonesia

Accessing HTML Element

Using document.getElementsByTagName() to access a group of elements by their tag name

document.getElementsByTagName(): This method allows you to access a group of elements by their tag name. For example, given the following HTML:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

You can access both paragraph elements using the following JavaScript:

```
const paragraphs = document.getElementsByTagName('p');
```

FACULTY OF INFORMATION TECHNOLOGY
Universitas Advent Indonesia

Using document.querySelector() and document.querySelectorAll() to access elements using CSS-style selectors

document.querySelector() and document.querySelectorAll(): These methods allow you to access elements using CSS-style selectors. For example, given the following HTML:

```
<div class="myClass">This is a div with the class "myClass"</div>
```

You can access the div element using the following JavaScript:

```
const myClass = document.querySelector('.myClass');
```

Modifying HTML Element

- Using the innerHTML property to change the content of an element
- Using the setAttribute() method to change the value of an attribute
- Using the style property to change the style of an element

FACULTY OF INFORMATION TECHNOLOGY
Universitas Advent Indonesia

Using the innerHTML property to change the content of an element

Changing the content of an element: You can use the innerHTML property to change the content of an element. The innerHTML property represents the HTML content inside an element. For example, given the following HTML:

```
<div id="myDiv">This is the original content</div>
```

You can change the content of the div element using the following JavaScript:

```
const myDiv = document.getElementById('myDiv');
myDiv.innerHTML = 'This is the new content';
```

The resulting HTML will be:

```
<div id="myDiv">This is the new content</div>
```

Using the setAttribute() method to change the value of an attribute

Changing the content of an element: You can use the setAttribute() method to change the value of an attribute. The setAttribute() method takes two arguments: the name of the attribute, and the new value for the attribute. For example, given the following HTML:

```
<a href="#">Click here</a>
```

You can change the href attribute of the a element using the following JavaScript:

```
const link = document.querySelector('a');
link.setAttribute('href', 'http://www.example.com');
```

The resulting HTML will be:

```
<a href="http://www.example.com">Click here</a>
```

Using the style property to change the style of an element

Changing the attribute values of an element: You can use the style property to change the style of an element. The style property is an object that represents the inline style of an element. You can access and set the values of specific style properties using dot notation. For example, given the following HTML:

```
<div id="myDiv">This is a div</div>
```

You can change the background color of the div element using the following JavaScript:

```
const myDiv = document.getElementById('myDiv');
myDiv.style.backgroundColor = 'red';
```

Using the style property to change the style of an element

The resulting HTML will be:

```
<div id="myDiv" style="background-color: red;">This is a div</div>
```

Note that the CSS property name is written in **camelCase** (e.g. backgroundColor) in JavaScript, but in hyphenated form (e.g. background-color) in the HTML style attribute.

Traversing the HTML DOM tree

The HTML DOM tree consists of parent nodes, child nodes, and sibling nodes. You can use the following properties to navigate the tree:

- Using parentNode
- Using childNodes
- Using firstChild and lastChild
- Using nextSibling and previousSibling

FACULTY OF INFORMATION TECHNOLOGY

Universitas Advent Indonesia

Using parentNode

parentNode: The parentNode property returns the parent node of an element. For example, given the following HTML:

```
<div id="parent">
    <p id="child">This is a child element</p>
</div>
```

You can access the parent element of the p element using the following JavaScript:

```
const child = document.getElementById('child');
const parent = child.parentNode;
```

The parent node will be the div element with the id "parent".

Traversing the HTML DOM tree

Using childNodes

childNodes: The childNodes property returns a list of child nodes of an element. The list includes both element nodes and text nodes. For example, given the following HTML:

```
<div id="parent">
    <p>This is the first child</p>
    This is a text node
    <p>This is the second child</p>
</div>
```

Traversing the HTML DOM tree

Using childNodes

You can access the child nodes of the div element using the following JavaScript:

```
const parent = document.getElementById('parent');
const childNodes = parent.childNodes;
```

The childNodes list will contain three nodes: a p element, a text node, and another p element.

Traversing the HTML DOM tree

Using firstChild and lastChild

firstChild and lastChild: The firstChild and lastChild properties return the first and last child nodes of an element, respectively. For example, given the same HTML as above:

```
const firstChild = parent.firstChild;
const lastChild = parent.lastChild;
```

The firstChild will be the p element containing "This is the first child", and the lastChild will be the p element containing "This is the second child".

Using nextSibling and previousSibling

nextSibling and previousSibling: The nextSibling and previousSibling properties return the next and previous sibling nodes of an element, respectively. A sibling node is a node that has the same parent as the element. For example, given the following HTML:

```
<p id="first">This is the first element</p>
<p id="second">This is the second element</p>
```

Using nextSibling and previousSibling

You can access the next and previous sibling nodes of the p element with the id "second" using the following JavaScript:

```
const second = document.getElementById('second');
const nextSibling = second.nextSibling;
const previousSibling = second.previousSibling;
```

The nextSibling will be null, because there are no more elements after the second element. The previousSibling will be the p element containing "This is the first element".

Event handling in the HTML DOM

Events are actions that can be detected by the HTML DOM, such as a mouse click or a key press. You can use event listeners to specify a function to be executed when a particular event occurs.

Here's an example of how to add an event listener to an element:

```
const element = document.getElementById('myElement');
element.addEventListener('click', function() {
  console.log('The element was clicked!');
});
```

The **addEventListener()** method takes two arguments: the name of the event to listen for, and a function to be executed when the event occurs.

Event handling in the HTML DOM

Here are some common events that you can listen for:

- **onclick**: The onclick event is triggered when the element is clicked with the mouse.
- **onmouseover**: The onmouseover event is triggered when the mouse pointer moves over the element.
- **onkeydown**: The onkeydown event is triggered when a key is pressed down.
- **onsubmit**: The onsubmit event is triggered when a form is submitted.

Event handling in the HTML DOM

Here's an example of how to use the onsubmit event to validate a form before it is submitted:

```
<form id="myForm" onsubmit="return validateForm()">
  <input type="text" id="name">
  <input type="submit" value="Submit">
</form>
```

```
<script>
  function validateForm() {
    const name = document.getElementById('name').value;
    if (name === '') {
      alert('Please enter a name');
      return false;
    }
    return true;
  }
</script>
```

If the user tries to submit the form without entering a name, the validateForm() function will display an alert and return false, preventing the form from being submitted.