DOM

Was ist das DOM?

JavaScript-Objekte sind äußerst flexibel und können verschiedene Felder enthalten – auch wenn diese Felder selbst Objekte sind

Das `document`-Objekt nutzt dieses Paradigma, um den gesamten Inhalt einer Webseite zu organisieren

Durch die Organisation einer kompletten Seite als JavaScript-Objekt können wir die Seitenelemente programmatisch manipulieren

Beispiel eines HTML-Dokuments

```
<html>
<head>
    <title>Hello, world</title>
</head>
<body>
    <h2>Here's my page</h2>
    <p>World, hello</p>
    <a href="test.html">Link</a>
</body>
</html>
```

DOM-Properties

`innerHTML`

Enthält den HTML-Inhalt innerhalb eines Elements

`innerText`

Enthält den Text-Inhalt innerhalb eines Elements

`nodeName`

Name eines HTML-Elements oder Attributs

`id`

Das "id"-Attribut eines HTML-Elements

DOM-Properties

`parentNode`

Referenz zum übergeordneten Knoten im DOM

`childNodes`

Liste von Referenzen zu untergeordneten Knoten

`attributes`

Liste von Attributen eines HTML-Elements

`style`

Objekt für CSS/HTML-Styling eines Elements

DOM-Properties

Für

```
<p>Dies ist <strong>wichtiger</strong> Text</p>
```

liefert:

```
innerHTML: "Dies ist <strong>wichtiger</strong> Text"
innerText: "Dies ist wichtiger Text"
```

DOM-Methoden

`getElementById(id)`

Findet ein Element mit der angegebenen ID

`getElementsByTagName(tag)`

Findet alle Elemente mit dem angegebenen Tag

`appendChild(node)`

Fügt einen neuen Knoten als Kind hinzu

`removeChild(node)`

Entfernt den angegebenen Kindknoten

DOM-Methoden

`querySelector / querySelectorAll:`

Elemente per CSS-Selektor finden

```
document.querySelector('.highlight') // erstes Element mit class="highlight"
document.querySelector('nav > ul')   // erstes ul direkt in nav
```

DOM-Manipulation: Praktisches Beispiel

```
// HTML-Element finden und modifizieren
let colorDiv = document.getElementById('colorDiv');
colorDiv.style.backgroundColor = 'green';

// Neues Element erstellen und einfügen
let newParagraph = document.createElement('p');
newParagraph.innerHTML = 'Dynamisch hinzugefügt';
document.body.appendChild(newParagraph);
```

DOM-Collections

Häufig verwendete DOM-Collections:

```
document.forms     // alle Formulare
document.images    // alle Bilder
document.links     // alle Links
document.anchors   // alle Anker (<a name="...">)
```

DOM-Collections

```
// Beispiele:
let erstesFormular = document.forms[0];     // erstes Formular
let formularName = document.forms['login']; // Formular mit name="login"
let alleFormulare = document.forms;         // HTMLCollection aller Formulare
```

DOM-Manipulation mit jQuery

```
// Klassisches JavaScript
document.getElementById('colorDiv').style.backgroundColor = 'green';

// Mit jQuery (nur als Beispiel)
$('#colorDiv').css('background-color', 'green');
```

DOMContentLoaded Event

```
document.addEventListener('DOMContentLoaded', function() {
    // DOM ist jetzt bereit für Manipulation
    document.querySelector('form').addEventListener('submit',
        function(e) {
            console.log('Formular wurde abgeschickt');
            e.preventDefault();
    });
});
```

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

Übungsaufgabe

```
// Vervollständigen Sie die Funktion:
// - Finden Sie alle Paragraphen
// - Fügen Sie jedem einen roten Rahmen hinzu
// - Fügen Sie einen Click-Event-Handler hinzu,
//   der den Rahmen grün färbt

function markiereParagraphen() {
    // Ihre Lösung hier
}
```

Übungsaufgabe

```
function markiereParagraphen() {
    let paragraphs = document.getElementsByTagName('p');
    for(let p of paragraphs) {
        p.style.border = '1px solid red';
        p.addEventListener('click', function() {
            this.style.border = '1px solid green';
        });
    }
}
```

Fragen zum Nachbereiten

Was ist der Unterschied zwischen `innerHTML` und `innerText`?

Warum ist die Reihenfolge von Script-Ausführung und DOM-Aufbau wichtig?

Können Sie die Analogie zwischen dem DOM und einer Baumstruktur erklären?

Das DOM bildet eine HTML-Seite als hierarchische Objekt-Baumstruktur ab

An der Wurzel steht das document-Objekt; jedes Element ist ein Objekt

Properties wie innerHTML, innerText und style geben Zugriff auf Elemente

Methoden wie getElementById, querySelector und appendChild verändern den Baum

Events und Event-Handler ermöglichen Reaktionen auf Benutzerinteraktionen

DOMContentLoaded stellt sicher, dass der DOM vor der Manipulation geladen ist