Topic 1: Introduction to DOM Structure

مقدمة في هيكلية DOM

What is the DOM?

[ما هو DOM؟]

```
// DOM = Document Object Model
// كشجرة من العناصر والكائنات (HTML) يمثل المستند

// الوصول إلى العنصر الجذر للمستند
const rootElement = document.documentElement;

// الوصول إلى عنصر body
const bodyElement = document.body;

// عرض هيكل الصفحة في وحدة التحكم
console.log(document);
```

كشجرة من العقد، مما يسمح للبرامج بتغيير XMLو HTML أو نموذج كائن المستند هو واجهة برمجة تمثل صفحات DOM بنية المستند ومحتواه وأسلوبه

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart TD
    A[Document] --> B[HTML]
    B --> C[Head]
    B --> D[Body]
    C --> E[Title]
    C --> F[Meta]
    D --> G[Div]
    D --> H[Script]
    G --> I[Paragraph]
    G --> J[Button]

    style A fill:#f9f,stroke:#333,stroke-width:1px
    style B fill:#bbf,stroke:#333,stroke-width:1px
    style C fill:#bbf,stroke:#333,stroke-width:1px
    style D fill:#bbf,stroke:#333,stroke-width:1px
```

Topic 2: Basic DOM Selectors

محددات DOM الأساسية

getElementById

[الحصول على عنصر بواسطة المعرف]

```
// HTML: <div id="container">محتوى</div>

// الحصول على العنصر بواسطة المعرف
const container = document.getElementById('container');

// التحقق من وجود العنصر قبل استخدامه
if (container) {
    console.log(container.textContent); // طباعة محتوى النص
}
```

تستخدم طريقة getElementById للحصول على عنصر واحد فقط باستخدام قيمة خاصية id

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart LR
    A[document] --> B[getElementById]
    B --> C['container']
    C --> D[single element]

    style A fill:#f9f,stroke:#333,stroke-width:1px
    style B fill:#bbf,stroke:#333,stroke-width:1px
    style D fill:#afa,stroke:#333,stroke-width:1px
```

querySelector

[محدد الاستعلام]

```
// الحصول على أول عنصر يطابق المحدد CSS
const firstButton = document.querySelector('button');
const redElement = document.querySelector('.red-text');
const formInput = document.querySelector('#contact-form input');

// معقدة CSS يمكنك استخدام محددات
const nestedElement = document.querySelector('div.container > ul > li:first-
child');
```

تستخدم querySelector للحصول على أول عنصر يطابق محدد CSS. يمكن استخدام أي محدد CSS، مما يجعلها أكثر مرونة من getElementById

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart LR
    A[document] --> B[querySelector]
    B --> C['CSS selector']
    C --> D[first matching element]

    style A fill:#f9f,stroke:#333,stroke-width:1px
```

```
    style B fill:#bbf,stroke:#333,stroke-width:1px
    style D fill:#afa,stroke:#333,stroke-width:1px
```

querySelectorAll

[محدد الاستعلام للحصول على جميع العناصر]

```
// الحصول على جميع العناصر التي تطابق المحدد CSS
const allParagraphs = document.querySelectorAll('p');
const redElements = document.querySelectorAll('.red-text');

// استخدام الحلقة للتفاعل مع كل عنصر
allParagraphs.forEach(paragraph => {
    console.log(paragraph.textContent);
});

// يمكن أيضًا استخدام for...of
for (const element of redElements) {
    element.style.color = 'darkred'; // تغيير اللون
}
```

تستخدم querySelectorAll لتحديد جميع العناصر التي تطابق محدد CSS. تعيد NodeList التي يمكن التكرار عليها باستخدام forEach أو for...of

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart LR
    A[document] --> B[querySelectorAll]
    B --> C['CSS selector']
    C --> D[NodeList of elements]

    style A fill:#f9f,stroke:#333,stroke-width:1px
    style B fill:#bbf,stroke:#333,stroke-width:1px
    style D fill:#afa,stroke:#333,stroke-width:1px
```

Topic 3: Modifying Content and Styles

تعديل المحتوى والأنماط

Changing Text Content

[تغيير محتوى النص]

```
const paragraph = document.querySelector('p');

// استخدام طريقة 1: textContent (النص فقط)
paragraph.textContent = 'هذا نص جديد';
```

```
// استخدام 2: طريقة innerHTML (يمكن أن يتضمن HTML)
paragraph.innerHTML = 'هذا <strong>نص</strong> منسق';

// 3 طريقة: innerText (مشابهة لـ textContent لكن مع بعض الاختلافات)
paragraph.innerText = 'محتوى نصي جديد';
```

هناك عدة طرق لتغيير محتوى العناصر، منها textContent (للنص العادي)، innerHTML (للنص مع HTML)، وinnerText (وسط بينهما)

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart TB
    A[element] --> B[textContent]
    A --> C[innerHTML]
    A --> D[innerText]
    B --> E[Plain text only]
    C --> F[HTML content]
    D --> G[Visible text]

    style A fill:#f9f,stroke:#333,stroke-width:1px
    style B fill:#bbf,stroke:#333,stroke-width:1px
    style C fill:#bbf,stroke:#333,stroke-width:1px
    style D fill:#bbf,stroke:#333,stroke-width:1px
```

Modifying Styles

[تعديل الأنماط]

```
const div = document.querySelector('.content');

// تعديل نمط مباشرة
div.style.backgroundColor = 'lightblue';
div.style.color = '#333';
div.style.padding = '10px';
div.style.borderRadius = '5px';

// ملاحظة: خصائص CSS ذات الواصلات تستخدم camelCase في JavaScript
// مثال: background-color تصبح backgroundColor
```

يمكن تعديل أنماط CSS للعناصر باستخدام خاصية style. لاحظ أن أسماء خصائص CSS التي تحتوي على واصلات (-) تكتب في JavaScript بنمط camelCase

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart LR
    A[element] --> B[style]
    B --> C[backgroundColor]
    B --> D[fontSize]
    B --> E[marginTop]
```

```
    style A fill:#f9f,stroke:#333,stroke-width:1px
    style B fill:#bbf,stroke:#333,stroke-width:1px
```

CSS Classes Manipulation

[التعامل مع فئات CSS]

```
const element = document.querySelector('.item');

// إضافة فئة
element.classList.add('highlight');

// إزالة فئة
element.classList.remove('inactive');

// تبديل فئة (إضافة إذا لم تكن موجودة، وإزالة إذا كانت موجودة)
element.classList.toggle('selected');

// التحقق من وجود فئة
if (element.classList.contains('active')) {
    console.log('العنصر نشط');
}

// استبدال فئة بأخرى
element.classList.replace('old-class', 'new-class');
```

توفر واجهة classList طرقًا سهلة للتعامل مع فئات CSS للعناصر، مثل إضافة وإزالة وتبديل والتحقق من وجود فئات

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart TB
    A[element.classList] --> B[add]
    A --> C[remove]
    A --> D[toggle]
    A --> E[contains]
    A --> F[replace]

    style A fill:#f9f,stroke:#333,stroke-width:1px
    style B fill:#bbf,stroke:#333,stroke-width:1px
    style C fill:#bbf,stroke:#333,stroke-width:1px
    style D fill:#bbf,stroke:#333,stroke-width:1px
    style E fill:#bbf,stroke:#333,stroke-width:1px
    style F fill:#bbf,stroke:#333,stroke-width:1px
```

Topic 4: Basic Event Handling

التعامل مع الأحداث الأساسية

Click Events

[أحداث النقر]

```
const button = document.querySelector('#submitBtn');

// طريقة addEventListener :1 استخدام
button.addEventListener('click', function() {
    console.log('تم النقر على الزر!');
    alert('تم النقر على الزر!');
});

// طريقة onclick :2 استخدام خاصية
button.onclick = function() {
    console.log('تم النقر على الزر باستخدام onclick!');
};

// دالة منفصلة للتعامل مع الحدث
function handleButtonClick(event) {
    console.log('تم النقر!', event);
    // الوصول إلى العنصر الذي أثار الحدث
    console.log('العنصر:', event.target);
}

// إضافة المعالج
button.addEventListener('click', handleButtonClick);
```

يمكن التعامل مع أحداث النقر وغيرها من أحداث المستخدم باستخدام addEventListener أو خصائص مثل onclick. توفر addEventListener مرونة أكبر وتسمح بإضافة معالجات متعددة للحدث نفسه

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart TB
    A[element] --> B[addEventListener]
    A --> C[onclick property]
    B --> D['click', function]
    C --> E[function]

    style A fill:#f9f,stroke:#333,stroke-width:1px
    style B fill:#bbf,stroke:#333,stroke-width:1px
    style C fill:#bbf,stroke:#333,stroke-width:1px
```

Common Events

[الأحداث الشائعة]

```
const input = document.querySelector('input');
const form = document.querySelector('form');
const div = document.querySelector('div');
```

```
// أحداث النموذج
form.addEventListener('submit', function(event) {
    event.preventDefault(); // منع السلوك الافتراضي (إرسال النموذج)
    console.log('تم إرسال النموذج!');
});

// أحداث الإدخال
input.addEventListener('input', function() {
    console.log('قيمة الإدخال:', this.value);
});

// أحداث الفأرة
div.addEventListener('mouseover', function() {
    this.style.backgroundColor = 'yellow';
});

div.addEventListener('mouseout', function() {
    this.style.backgroundColor = '';
});

// أحداث لوحة المفاتيح
document.addEventListener('keydown', function(event) {
    console.log('تم الضغط على المفتاح:', event.key);
});
```

هناك العديد من أنواع الأحداث التي يمكن التعامل معها، مثل أحداث النموذج (submit)، وأحداث الإدخال (input)، وأحداث الفأرة (mouseover, mouseout)، وأحداث لوحة المفاتيح (keydown)

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart TB
    A[Events] --> B[Mouse Events]
    A --> C[Keyboard Events]
    A --> D[Form Events]
    A --> E[Window Events]
    B --> F[click]
    B --> G[mouseover]
    B --> H[mouseout]
    C --> I[keydown]
    C --> J[keyup]
    D --> K[submit]
    D --> L[input]
    E --> M[load]
    E --> N[resize]

    style A fill:#f9f,stroke:#333,stroke-width:1px
    style B fill:#bbf,stroke:#333,stroke-width:1px
    style C fill:#bbf,stroke:#333,stroke-width:1px
    style D fill:#bbf,stroke:#333,stroke-width:1px
    style E fill:#bbf,stroke:#333,stroke-width:1px
```

Topic 5: Simple Animations

الرسوم المتحركة البسيطة

CSS Transitions

[انتقالات CSS]

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

// إضافة انتقال CSS
box.style.transition = 'all 0.5s ease-in-out';

// تغيير الخصائص للتحريك
box.addEventListener('click', function() {
    if (box.style.transform === 'scale(1.5)') {
        box.style.transform = 'scale(1)';
        box.style.backgroundColor = 'blue';
    } else {
        box.style.transform = 'scale(1.5)';
        box.style.backgroundColor = 'red';
    }
});
```

يمكن إنشاء رسوم متحركة بسيطة باستخدام انتقالات CSS وتغيير الخصائص باستخدام JavaScript. تسمح transition خاصية بتحريك التغييرات بسلاسة

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart TB
    A[element] --> B[style.transition = 'all 0.5s']
    A --> C[Change properties]
    C --> D[transform]
    C --> E[backgroundColor]

    style A fill:#f9f,stroke:#333,stroke-width:1px
    style B fill:#bbf,stroke:#333,stroke-width:1px
    style C fill:#bbf,stroke:#333,stroke-width:1px
```

Animations with setInterval

[الرسوم المتحركة باستخدام setInterval]

```
const movingBox = document.querySelector('.moving-box');
let position = 0;
let direction = 1;

// إنشاء رسم متحرك باستخدام setInterval
const animation = setInterval(function() {
```

```
    position += 5 * direction;

    // عكس الاتجاه عند الوصول إلى الحدود
    if (position >= 300 || position <= 0) {
        direction *= -1;
    }

    // تحديث موضع العنصر
    movingBox.style.left = position + 'px';
}, 50); // تحديث كل 50 مللي ثانية

// إيقاف الرسم المتحرك بعد النقر
movingBox.addEventListener('click', function() {
    clearInterval(animation);
    this.style.backgroundColor = 'green';
});
```

الذي ينفذ شيفرة على فترات زمنية محددة. هذا يسمح بالتحكم setInterval، يمكن إنشاء رسوم متحركة مخصصة باستخدام الدقيق في الرسوم المتحركة

```
%%{init: {'theme': 'base', 'themeVariables': { 'fontSize': '10px'}}}%%
flowchart TB
    A[setInterval] --> B[Update position]
    B --> C[Check boundaries]
    C --> D[Update style]
    D --> E[Repeat]
    F[clearInterval] --> G[Stop animation]

    style A fill:#f9f,stroke:#333,stroke-width:1px
    style B fill:#bbf,stroke:#333,stroke-width:1px
    style F fill:#f96,stroke:#333,stroke-width:1px
    style G fill:#f96,stroke:#333,stroke-width:1px
```

Practical Exercise 1: Color-Changing Background

تمرين عملي 1: خلفية متغيرة اللون

```
// إنشاء أزرار للألوان
const colors = ['red', 'green', 'blue', 'purple', 'orange'];
const container = document.querySelector('.container');

// إنشاء الأزرار ديناميكيًا
colors.forEach(color => {
    const button = document.createElement('button');
    button.textContent = color;
    button.style.backgroundColor = color;
    button.style.color = 'white';
    button.style.padding = '10px';
    button.style.margin = '5px';
```

```
    // إضافة معالج النقر
    button.addEventListener('click', function() {
        document.body.style.backgroundColor = color;
    });

    container.appendChild(button);
});

// زر عشوائي
const randomButton = document.createElement('button');
randomButton.textContent = 'لون عشوائي';
randomButton.style.padding = '10px';
randomButton.style.margin = '5px';

randomButton.addEventListener('click', function() {
    // إنشاء لون عشوائي
    const randomColor = '#' + Math.floor(Math.random()*16777215).toString(16);
    document.body.style.backgroundColor = randomColor;
    this.style.backgroundColor = randomColor;
});

container.appendChild(randomButton);
```

Practical Exercise 2: Simple Drawing Board

تمرين عملي 2: لوحة رسم بسيطة

```
// إنشاء لوحة رسم بسيطة
const canvas = document.createElement('div');
canvas.className = 'drawing-canvas';
document.body.appendChild(canvas);

// تنسيق لوحة الرسم
canvas.style.width = '500px';
canvas.style.height = '400px';
canvas.style.border = '2px solid black';
canvas.style.position = 'relative';
canvas.style.backgroundColor = 'white';

// متغيرات للرسم
let isDrawing = false;
let currentColor = 'black';

// وظيفة إنشاء نقطة
function createDot(x, y) {
    const dot = document.createElement('div');
    dot.className = 'dot';
    dot.style.width = '5px';
    dot.style.height = '5px';
    dot.style.backgroundColor = currentColor;
```

```
    dot.style.position = 'absolute';
    dot.style.left = x + 'px';
    dot.style.top = y + 'px';
    dot.style.borderRadius = '50%';
    canvas.appendChild(dot);
}

// معالجات الأحداث للرسم
canvas.addEventListener('mousedown', function(e) {
    isDrawing = true;
    const rect = canvas.getBoundingClientRect();
    const x = e.clientX - rect.left;
    const y = e.clientY - rect.top;
    createDot(x, y);
});

canvas.addEventListener('mousemove', function(e) {
    if (isDrawing) {
        const rect = canvas.getBoundingClientRect();
        const x = e.clientX - rect.left;
        const y = e.clientY - rect.top;
        createDot(x, y);
    }
});

canvas.addEventListener('mouseup', function() {
    isDrawing = false;
});

// إنشاء قائمة ألوان
const colors = ['black', 'red', 'green', 'blue', 'yellow', 'purple'];
const colorPalette = document.createElement('div');
colorPalette.className = 'color-palette';
document.body.appendChild(colorPalette);

colors.forEach(color => {
    const colorButton = document.createElement('div');
    colorButton.className = 'color-option';
    colorButton.style.backgroundColor = color;
    colorButton.style.width = '30px';
    colorButton.style.height = '30px';
    colorButton.style.display = 'inline-block';
    colorButton.style.margin = '5px';
    colorButton.style.cursor = 'pointer';
    colorButton.style.border = '2px solid #ccc';

    colorButton.addEventListener('click', function() {
        currentColor = color;
        // إظهار اللون المحدد
        document.querySelectorAll('.color-option').forEach(opt => {
            opt.style.border = '2px solid #ccc';
        });
        this.style.border = '2px solid black';
    });
```

```
    colorPalette.appendChild(colorButton);
});

// زر المسح
const clearButton = document.createElement('button');
clearButton.textContent = 'مسح';
clearButton.style.margin = '10px';
clearButton.addEventListener('click', function() {
    const dots = document.querySelectorAll('.dot');
    dots.forEach(dot => dot.remove());
});
document.body.appendChild(clearButton);
```

Key Concepts Covered

المفاهيم الرئيسية التي تم تغطيتها

1. **هيكل DOM**: فهم كيفية تمثيل مستندات HTML كشجرة من العناصر
2. **محددات DOM**: استخدام getElementById وquerySelector وquerySelectorAll للوصول إلى العناصر
3. **تعديل المحتوى**: تغيير محتوى العناصر باستخدام innerHTML وtextContent
4. **تعديل الأنماط**: تغيير أنماط CSS وإضافة وإزالة فئات CSS
5. **التعامل مع الأحداث**: استخدام addEventListener للتفاعل مع أحداث المستخدم
6. **الرسوم المتحركة البسيطة**: إنشاء حركات باستخدام انتقالات CSS وsetInterval
7. **إنشاء عناصر**: إنشاء عناصر DOM ديناميكيًا وإضافتها إلى HTML