styleSheet Objekt des Internet Explorer

ab IE 4.x

Ansatz:

Dieses Objekt dient der Verwaltung von Styles (StyleSheets) im Dokument, die auf ganz bestimmte Arten in das Dokument implementiert wurden:

Beispiele für Arten:

```
<HEAD>
<STYLE>
        @import url("MeineStyleSheetDatei.css");
</STYLE>
<HEAD>

<LINK REL=stylesheet HREF="styles.css" type="text/css">
```

per Pseudoklasse @page (Objekt page)

```
<HEAD>
<STYLE>
        P {color:green}
</STYLE>
</HEAD>
```

per document.createStyleSheet('styles.css');

Diese o g. Arten von Implementationen können als Block von Styles angesehen werden, mit anderen Worten: Als Menge von Style-Regeln.

Besonders für den Programmierer ist die vereinfachte Kodierung von Styles in externen CSS-Dateien und die Verwendung der browsereigenenen Pseudoklassen wie @page und @import interessant: Anhand dieser sind komplette Style-Regeln-Manipulationen zur Laufzeit des HTML-Dokumentes möglich. So kann z.B. das Layout des ganzen HTML-Dokumentes nach seinem kompletten Laden auf einen Schlag geändert werden.

Allen Arten von Style-Implementationen haben eines gemeinsam: Es werden einzelne Styles im Browser verwaltet, die auch alle einzeln durch die übliche Style-Eigenschaften und Style-Methoden verwaltet werden könnten (siehe style Objekt).

Hinweis zur Pars-Reihenfolge des Internet Explorer innerhalb der HTML-Kodierung bezüglich dem
Tag-Name und den Element-Attributen CLASS, ID, STYLE

1. Element-Bezeichner
2. CLASS-Attribut mit Bezeichner aus Klassendeklaration im HEAD
3. ID-Attribut
4. STYLE-Attribut mit Style-Werten (nicht mit Bezeichner aus Style-Deklaration im HEAD)

Es gilt: Wenn gleiche Bezeichner verwendet, so nur Werte des **zuletzt** geparsten Bezeichners verwendet !!!

Die CLASS-Deklaration aus dem HEAD des Dokumentes wird wertmäßig durch die Style-Deklaration per Attribut des HTML-Elementes überschrieben, wenn gleiche Style-Eigenschaften betroffen sind (ansonsten hinzufügen).

Hinweis zu dynamischen Eigenschaftenveränderung zur Laufzeit:
Die zuletzt während der Laufzeit getätigte Defintion ersetzt wertmäßig den aktuellen Attributwert wenn gleiche Attributnamen/Eigenschaften betroffen sind (sonst hinzufügen).

Verwaltung des styleSheet-Objektes in der Collection document.styleSheets:

Ein styleSheet-Objekt ist ein Element der Collection document.styleSheets, die in der Reihenfolge der Style-Implemenationen im Quellcode der HTML-Seite beim Laden des Dokumentes gefüllt, also initialsiert wird. Jede o.g. Implementation erzeugt je ein styleSheet-Objekt als Element der Collection dodument.styleSheets.

Beispiel:

```
<HEAD>
<STYLE>
        .StyleRegel1 {color:"red"}
        .StyleRegel2 {color:"blue"}
</STYLE>
<HEADY
```

Es wird ein styleSheet-Objekt erzeugt ,das 2 Regeln besitzt.

Verwaltung der Implementationsarten im styleSheet-Objekt:

Im styleSheet-Objekt wird jede o.g. Art der Style-Implementation einzeln verwaltet und zwar mit je einer eigenen Collection des styleSheet-Objektes.Ein styleSheet-Objekt referenziert selbst diese Collectionen, die je eine vordefinierte Art der Style-Implementation in das Dokument verwalten. Ob diese Collectionen auch reale Elemente besitzen, hängt davon ab, ob Styles in o.g. Arten implementiert wurden oder noch werden.

Collectionen des styleSheet Objektes	**Beispiele zu den Arten der verwalteteten Style-Implementationen**
styleSheet.imports	<HEAD> <STYLE>

```
                                        @import url("MeineStyleSheetDatei.css");
                                </STYLE>
                                <HEAD>

                                <LINK REL=stylesheet HREF="styles.css" type="text/css">

                                document.createStyleSheet('styles.css');

                                Methode .addImport()

styleSheet.pages                per Pseudoklasse @page (Objekt page)

styleSheet.rules                <HEAD>
                                <STYLE>
                                        P {color:green}
                                </STYLE>
                                </HEAD>
```

Damit weist ein Eintrag in der Collection document.styleSheets auf ein styleSheet Objekt, das wiederum genau die zur Art der Implementation passende Collection referenziert, die wiederum konkrete Style-Objekte referenziert, welche eben nur auf eine von o.g. Arten der Implementationen in das Dokument eingebettet wurden und sich ansonsten **nicht** z.B. von einem inline-Style (per STYLE-Attribut im HTML-Tag eingebetter Style) unterscheiden.

Beispiel:

```
<HEAD>
<STYLE>
        .StyleRegel1 {color:"red"}
        .StyleRegel2 {color:"blue"}
</STYLE>
<HEADY

Implementation füllt Collection styleSheet.rules mit 2 Elementen (2 Regeln)

Wenn obige Implementation die ERSTE im Dokument ist, so wird das ERSTE styleSheet-Objekt erzeugt, also
        das Element document.styleSheets[0], das auf die Collection styleSheet.rules verweist.
```

Zugriff auf styleSheet-Objekt:
nur per Collection document.styleSheets

Hinweis: Style, der per Link oder @import anhand einer CSS-Datei eingebunden, kann zur Laufzeit **nur** verändert werden, wenn document.designMode auf "On" gesetzt ist

Beispiel 1:

```
<HEAD>
<STYLE>
        BODY {background-color: #CFCFCF;}
        @import url("MeineStyleSheetDatei.css");
</STYLE>
<SCRIPT>
        window.onload=fnInit;

        function fnInit()
        {
                // Style des BODY laut Klassendefinition
                var StyleSheetObjekt =document.styleSheets[0];

                // erste Regel zum Style des BODY holen
                var RegelObjekt = StyleSheetObjekt.rules[0];

                // und Style ändern
                RegelObjekt.style.backgroundColor="#0000FF";

                // neue Regel für BODY hinzufügen
                StyleSheetObjekt.addRule("BODY"," border-color: #FFFF00;");

                // und weitere neue Regel für BODY
                StyleSheetObjekt.imports[0].color="#000000";
        }
</SCRIPT>
</HEAD>
```

Beispiel 2:

```
<HTML>
<HEAD>
```


```
<SCRIPT>
        function Anzeige(RegelIndex)
        {
                alert(          "Regel Nr " + RegelIndex
                        + " hat style.color =  "
                        + document.styleSheets[0].rules.item(RegelIndex).style.color
                        + "."
                    );
        }
</SCRIPT>
<STYLE>
        .StyleRegel1 {color:"red"}
        .StyleRegel2 {color:"blue"}
</STYLE>
</HEAD>
<BODY>
        <P CLASS=" StyleRegel1">
                StyleRegel1
        </P>
        <P CLASS=" StyleRegel2">
                StyleRegel2
        </P>

        <BUTTON onclick="Anzeige(0)">StyleRegel1</BUTTON>
        <BUTTON onclick="Anzeige(1)"> StyleRegel2</BUTTON>
</BODY>
</HTML>
```

Zugriff auf Collectionen des styleSheet-Objektes:

siehe Beschreibung der Collectionen

Eigenschaften des styleSheet-Objektes:

.href	Url einer externen Style-Sheet-Ressource-Datei (externe Style-Datei *.css)
.owningElement	Referenz auf den im StyleSheet implementierten Style als Kindobjekt Hinweis: Es sind die Eigenschaften und Methoden des style Objektes referenzierbar

Beispiel:

```
for ( var i = 0; i < document.styleSheets.length; i++ )
{
        if ( document.styleSheets(i).owningElement.tagName == "STYLE" )
        {
                for ( var j = 0; j < document.styleSheets(i).imports.length; j++ )
                {
                        alert(          "Importierter StyleSheet (Style-Regel) Nr " + j
                                + " hat HREF = " + document.styleSheets(i).imports(j).href
                            );
                }
        }
}
```

.parentStyleSheet	Referenz auf das den StyleSheet implementierende Objekt (Elternobjekt)
.readOnly	Art der Implementation des StyleSheets im Dokument ermitteln
.title	Titel des StyleSheets
.type	Mimetyp des StyleSheets muss "text/css" sein

Methoden des styleSheet-Objektes:

.addImport()	StyleSheet aus externer CSS-Datei importieren und als Element der Collection styleSheet.imports erzeugen (entspricht @import url() innerhalb STYLE im HEAD)
.addPageRule()	StyleSheet importieren, als ob er wie ein in das Dokument direkt kodierter Style implementiert wurde, und als Element der Collection styleSheet.pages erzeugen (entspricht @page vom Objekt page)

Beispiel:

```
function TextFaerben ()
{document.styleSheets[0].addPageRule("DIV B", "color:blue", 0);}

<DIV onmouseover="TextFaerben ();">
        <B>dieser Text wird per CSS mit blauer Fabrbe angezeigt</B>
</DIV>
```

.addRule()	StyleSheet importieren, als ob er wie ein in das Dokument direkt kodierter Style implementiert wurde, und als Element der Collection styleSheet.rules erzeugen (entspricht xx { ...} innerhalb STYLE im HEAD)

Beispiel:

```
function TextFaerben ()
{document.styleSheets[0].addRule("DIV B", "color:blue", 0);}

<DIV onmouseover="TextFaerben ();">
        <B>dieser Text wird per CSS mit blauer Fabrbe angezeigt</B>
```


```
                </DIV>
.fireEvent()            ein Event auslösen
                        true    Event erfolgreich ausgelöst
                        false   Event nicht ausgelöst
.removeRule()           StyleSheet aus der Collection styleSheet.rules entfernen
                        Achtung: Um Style auch sichtbar zu entfernen, muss
                                        Dokument muss neu geladen werden
                                oder    alle betroffene Style-Elemente neu mit Wert belegen
                                                durch Zuweisung auf sich selbst (siehe Beispiel)
        Beispiel:
                <STYLE>
                        P {color:green}
                </STYLE>
                 <SCRIPT>
                        function StyleEntfernen()
                        {
                                var StyleSheetsObjekt = document.styleSheets;
                                var AnzahlStyleSheets = StyleSheetsObjekt.length;

                                if (AnzahlStyleSheets > 0)
                                {

                                        // StyleSheet mit Index 0 bearbeiten also P {color:green}
                                        var StyleSheetAnIndex0 = StyleSheetsObjekt[0];

                                        //  wobei P {color:green} eine Regel ist, also Collection rules verwenden
                                        var StyleSheetsRegelCollection = StyleSheetAnIndex0.rules;

                                        var AnzahlRegeln = StyleSheetsRegelCollection.length;

                                        if (AnzahlRegeln > 0)
                                        {
                                                // Regel P {color:green} enfernen
                                                StyleSheetAnIndex0.removeRule(0);

                                                // visuell auch die Farbe entfernen durch Zuweisung auf sich selbst also
                                                //              nun ohne die Regel P {color:green}
                                                ID_P.innerHTML= ID_P.innerHTML;
                                        }
                                }
                        }
                </SCRIPT>

                <P ID="ID_P" >Test</P>
                <BUTTON onclick="StyleEntfernen()">Text im P-Tag entfaerben.</BUTTON>
```

styleSheet.imports Collection des Internet Explorer

verwaltet die per

Link auf eine externe Style-Sheet-Ressource-Datei (externe Style-Datei *.css)
@import einer externen Style-Sheet-Ressource-Datei (externe Style-Datei *.css)
Methode .addImport()

importierten Style-Regeln

z.B.
```
<HEAD>
<STYLE>
        @import url("MeineStyleSheetDatei.css");
</STYLE>
<HEAD>
```

z.B.
```
<LINK REL=stylesheet HREF="styles.css" type="text/css">
```

z.B.
```
document.createStyleSheet('styles.css');
```

Hinweis: Für ein Element der Collection sind die Eigenschaften und Methoden des style Objektes referenzierbar

Syntax:

[var FeldZeiger =] document.styleSheets**[**Index1**]**.imports
[var FeldElementZeiger =] document.styleSheets**[**Index1**]**.imports**[**Index2**]**

Index1: Integer und ab 0
muss in **[ ]** kodiert sein

Index2: Integer und ab 0
muss in **[ ]** kodiert sein

Beispiel:

```
for ( var i = 0; i < document.styleSheets.length; i++ )
{
    if ( document.styleSheets(i).owningElement.tagName == "STYLE" )
    {
        for ( var j = 0; j < document.styleSheets(i).imports.length; j++ )
        {
            alert(      "Importierter StyleSheet (Style-Regel) Nr " + j
                    + " hat HREF = " + document.styleSheets(i).imports(j).href
                );
        }
    }
}
```

Eigenschaften:

.length Anzahl der Feldelemente also Feldlänge z.B. bei Collection

Methoden:

.item() Referenz auf Feldelement anhand des Integer-Indexes oder des Attributnamen (analog zu ID oder NAME-Attribut) liefern
außer bei Formular mit <INPUT TYPE=image ...>
da dafür die children-Collection verwendet werden muss !!!

.namedItem() Referenz auf Eintrag (FeldElement) anhand des Namen (analog zu ID oder NAME-Attribut) liefern

styleSheet.pages Collection des Internet Explorer

verwaltet die per Pseudoklasse @page (Objekt page) importierten Style-Regeln

Syntax:

[var FeldZeiger =] document.styleSheets**[**Index1**]**.pages
[var FeldElementZeiger =] document.styleSheets**[**Index1**]**.pages**[**Index2**]**

Index1: Integer und ab 0
muss in **[ ]** kodiert sein

Index2: Integer und ab 0
muss in **[ ]** kodiert sein

Hinweis: Für ein Element der Collection sind die Eigenschaften und Methoden des style Objektes referenzierbar

Eigenschaften:

.length Anzahl der Feldelemente also Feldlänge z.B. bei Collection

Methoden:

.item() Referenz auf Feldelement anhand des Integer-Indexes oder des Attributnamen (analog zu ID oder NAME-Attribut) liefern
außer bei Formular mit <INPUT TYPE=image ...>

styleSheet.rules Collection des Internet Explorer

verwalten die per

xxx { ...} innerhalb von STYLE im HEAD
oder per .addRule()

importierten Style-Regeln

z.B.

```
<HEAD>
<STYLE>
    P {color:green}
</STYLE>
</HEAD>
```

Hinweis: Für ein Element der Collection sind die Eigenschaften und Methoden des style Objektes referenzierbar

Syntax:

[var FeldZeiger =] document.styleSheets**[**Index1**]**.rules
[var FeldElementZeiger =] document.styleSheets**[**Index1**]**.rules**[**Index2**]**

Index1: Integer und ab 0
muss in **[ ]** kodiert sein

Index2: Integer und ab 0
muss in **[ ]** kodiert sein

Beispiel:

```
<HTML>
<HEAD>
<SCRIPT>
    function Anzeige(RegelIndex)
    {
        alert(      "Regel Nr " + RegelIndex
                + " hat style.color =  "
```


```
                                        + document.styleSheets[0].rules.item(RegelIndex).style.color
                                        + "."
                                );
                }
        </SCRIPT>
        <STYLE>
                .StyleRegel1 {color:"red"}
                .StyleRegel2 {color:"blue"}
        </STYLE>
        </HEAD>
        <BODY>
                <P CLASS=" StyleRegel1">
                        StyleRegel1
                </P>
                <P CLASS=" StyleRegel2">
                        StyleRegel2
                </P>

                <BUTTON onclick=" Anzeige(0)">StyleRegel1</BUTTON>
                <BUTTON onclick=" Anzeige(1)">StyleRegel2</BUTTON>
        </BODY>
        </HTML>
```

Eigenschaften:

.length Anzahl der Feldelemente also Feldlänge z.B. bei Collection

Methoden:

.item() Referenz auf Feldelement anhand des Integer-Indexes oder des