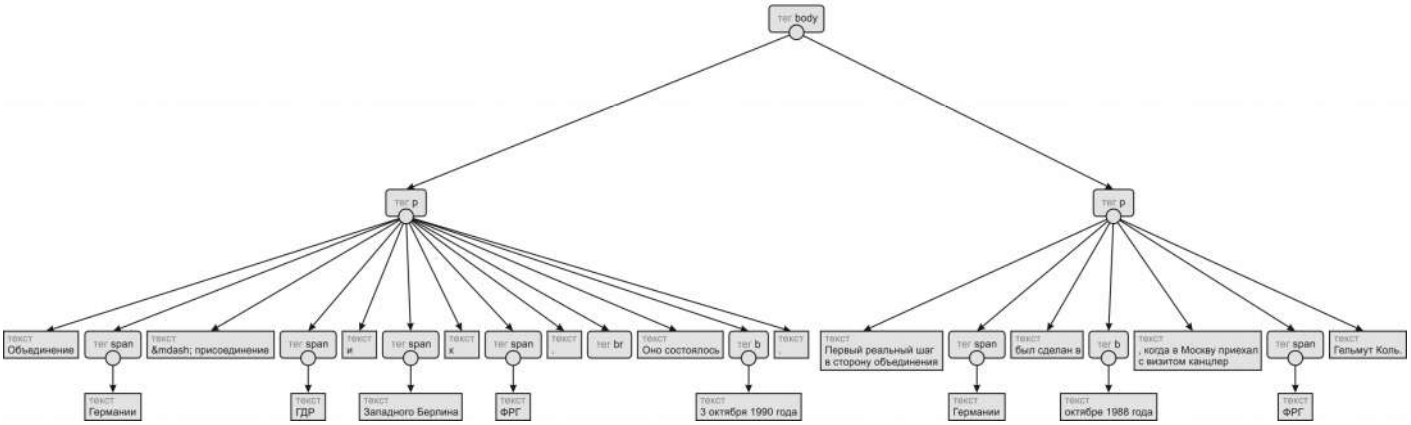
(3.62) Document Object Model

DOM — Document Object Model (объектная модель документа) — программный интерфейс, позволяющий программам получить доступ к содержимому HTML-документов (XHTML-документов, XML-документов), а также изменять содержимое, структуру и оформление таких документов.

DHTML — Dynamic HTML — это способ создания интерактивного веб-сайта, при котором JavaScript используется для управления веб-страницей через интерфейс DOM.

```
<html>
  <head>
    <title>Объединение Германии</title>
  </head>
  <body>
    <p>
      Объединение <span class='Country'>Германии</span>
      &mdash; присоединение <span class='Country'>ГДР</span>
      и <span class='Country'>Западного Берлина</span>
      к <span class='Country'>ФРГ</span>.<br>
      Оно состоялось <b>3 октября 1990 года</b>.
    </p>
    <p>
      Первый реальный шаг в сторону объединения
      <span class='Country'>Германии</span> был сделан
      в <b>октябре 1988 года</b>, когда в Москву приехал
      с визитом канцлер <span class='Country'>ФРГ</span> Гельмут Коль.
    </p>
  </body>
</html>
```



(3.64) Поиск элементов в дереве

В дереве элементов можно выполнять поиск элементов.
Элементы могут быть найдены по их идентификаторам (т.е. по значению атрибута id), по классу (т.е. по значению атрибута class), по имени тега, а также по произвольному CSS-селектору.

```
document.getElementById(идентификатор)
```

Ищет в документе элемент с указанным идентификатором.
Возвращает **ссылку** на найденный элемент (объект класса HTMLElement), или null, если такой элемент не найден.

```
document.getElementsByClassName(имя_класса)
```

Ищет во всём документе элементы с указанным классом.
Возвращает **псевдо-массив**, содержащий ссылки на найденные элементы.

```
document.getElementsByTagName(имя_тега)
```

Ищет во всём документе элементы с указанным именем тега.
Возвращает **псевдо-массив**, содержащий ссылки на найденные элементы.

```
document.querySelectorAll(CSS-селектор)
```

Ищет во всём документе элементы, соответствующие указанному CSS-селектору.
Возвращает **псевдо-массив**, содержащий ссылки на найденные элементы.

```
document.querySelector(CSS-селектор)
```

Ищет во всём документе первый попавшийся элемент, соответствующий указанному CSS-селектору.
Возвращает **ссылку** на найденный элемент, или null, если такой элемент не найден.

Методы getElementsByClassName, getElementsByTagName, querySelectorAll, querySelector могут быть также вызваны не для объекта document, а для любого элемента (т.е. объекта класса HTMLElement). В этом случае поиск элементов выполняется не во всём документе, а среди дочерних элементов данного объекта.

```
элемент.getElementsByClassName(имя_класса)
элемент.getElementsByTagName(имя_тега)
элемент.querySelectorAll(CSS-селектор)
элемент.querySelector(CSS-селектор)
```

С псевдо-массивами, возвращаемыми некоторыми из упомянутых методов, в некоторых отношениях можно обращаться как с обычными массивами — у них есть свойство *массив*.length, можно обращаться к их элементам через *массив*[*индекс*], однако, у них нет методов, присущих объектам класса Array.

Содержимое псевдо-массивов, возвращаемых методами document.getElementsBy*, **живое** — если дерево DOM изменяется, при следующем обращении к псевдо-массиву автоматически обновляется его содержимое; т.е. эти псевдо-массивы всегда содержат актуальные результаты поиска элементов дерева.

Т.к. все вышеперечисленные методы ищут и возвращают элементы (т.е. теги), этими методами нельзя найти что-либо по CSS-селекторам, заканчивающимся на псевдоэлемент (::first-line и т.д.).

(3.66) Установка и чтение стилевых свойств элемента

Для элемента можно устанавливать любые стилевые свойства CSS через свойство style:

```
элемент.style.имя_стилевого_свойства=значение
```

Сделанные на уровне DOM стилевые установки имеют приоритет над установками, сделанными на уровне CSS-правил.

Например, установим у элемента с идентификатором EEE красный цвет текста:

```
var EL=document.getElementById('EEE');
EL.style.color='red';
```

При установке на уровне DOM стилевого свойства в пустую строку, значение данного стилевого свойста будет определяться на уровне CSS-правил:

```
элемент.style.имя_стилевого_свойства=''
```

Например, найдём все изображения и установим для них рамку синего цвета:

```
var ELS=document.getElementsByTagName('img');
for ( var E=0; E<ELS.length; E++ )
{
  var EL=ELS[E];
  EL.style.borderStyle='solid';
  EL.style.borderColor='blue';
  EL.style.borderWidth='1px';
}
```

Названия стилевых свойств в CSS не всегда совпадают с названиями стилевых свойств DOM:

свойство CSS	свойство DOM
background	background
background-color	backgroundColor
background-image	backgroundImage
background-position	backgroundPosition
background-repeat	backgroundRepeat
border	border
border-bottom	borderBottom
border-bottom-color	borderBottomColor
border-bottom-style	borderBottomStyle
border-bottom-width	borderBottomWidth
border-collapse	borderCollapse
border-color	borderColor
border-left	borderLeft
border-left-color	borderLeftColor
border-left-style	borderLeftStyle
border-left-width	borderLeftWidth
border-right	borderRight
border-right-color	borderRightColor
border-right-style	borderRightStyle
border-right-width	borderRightWidth
border-style	borderStyle
border-top	borderTop
border-top-color	borderTopColor
border-top-style	borderTopStyle
border-top-width	borderTopWidth
border-width	borderWidth

свойство CSS	свойство DOM
clear	clear
color	color
cursor	cursor
display	display
float	**cssFloat / styleFloat**
font	font
font-family	fontFamily
font-size	fontSize
font-style	fontStyle
font-variant	fontVariant
font-weight	fontWeight
height	height
left	left
letter-spacing	letterSpacing
line-height	lineHeight
list-style	listStyle
list-style-image	listStyleImage
list-style-position	listStylePosition
list-style-type	listStyleType
margin	margin
margin-bottom	marginBottom
margin-left	marginLeft
margin-right	marginRight
margin-top	marginTop

свойство CSS	свойство DOM
max-height	maxHeight
max-width	maxWidth
min-height	minHeight
min-width	minWidth
overflow	overflow
padding	padding
padding-bottom	paddingBottom
padding-left	paddingLeft
padding-right	paddingRight
padding-top	paddingTop
position	position
right	right
table-layout	tableLayout
text-align	textAlign
text-decoration	textDecoration
text-indent	textIndent
text-transform	textTransform
top	top
vertical-align	verticalAlign
visibility	visibility
white-space	whiteSpace
width	width
word-spacing	wordSpacing
z-index	zIndex

(здесь перечислены стилевые свойства из CSS2, в CSS3 появляется множество новых)

Для элемента можно устанавливать сразу несколько стилевых свойств:

```
элемент.style.cssText=любой_код_CSS
```

что аналогично установке атрибута style элемента:

```
элемент.setAttribute('style',любой_код_CSS)
```

например:

```
var EL=document.getElementById('EEE');
EL.style.cssText='border: solid red 1px; background-color: blue';
```

Для элемента можно устанавливать (или читать) имя CSS-класса через свойство className:

```
элемент.className=класс
```

Например, установим для элемента с идентификатором EEE CSS-классы SLeft и SGood:

```
var EL=document.getElementById('EEE');
EL.className='SLeft SGood';
```

Можно получить полный перечень всех фактически применённых к элементу (любым способом, в т.ч. с учётом каскадирования) стилевых свойств специальным методом window.getComputedStyle:

```
var Elem=document.getElementById('EEE');
var StyleH=window.getComputedStyle(Elem);
```

Метод window.getComputedStyle вернёт хэш, ключами которого являются названия **всех** стилевых свойств, которые браузер может применить к элементу (включая устаревшие свойства, свойства с вендорными префиксами и т.д.), а значениями — фактические значения стилевых свойств, часто унифицированные по способу задания (цвет — к формату rgb(*R*,*G*,*B*) или rgba(*R*,*G*,*B*,*A*)), единицам измерения (все размеры — к пикселям) и т.д.

Можно также вторым аргументом методу getComputedStyle передать имя псевдокласса CSS, например ':hover', если нужно узнать стилевые свойства, которые **будут** применены к элементу, если к нему будет поднесена мышь.

Некоторые особенности:

- в хэше гарантируется наличие элементов только для точных стилевых свойств (таких как borderBottomStyle), а групповых (таких как borderStyle, borderBottom, border) может не быть;
- в Internet Explorer до 9 версии не поддерживается; кроссбраузерно можно получить стили элемента следующим образом:

  ```
  var ElemStyleH=
    window.getComputedStyle?
    getComputedStyle(Elem,''):
    Elem.currentStyle;
  ```

  однако метод *элемент*.currentStyle() не все значения стилевых свойств унифицирует.

Также, есть возможность отключать и подключать целые стилевые таблицы. Стилевые таблицы — теги <style> и <link>, содержащие наборы стилевых правил. Они могут быть найдены в DOM, как и любые другие элементы (например, через document.getElementById), а также могут быть найдены в document.styleSheets (это объект, подобный массиву, т.е. содержит свойство length и индексы от 0 до length-1; кроме того, если в теге <style> или <link> задан атрибут title, соответствующая стилевая таблица может быть найдена в document.styleSheets по данному ключу).

Например, если стилевые таблицы описаны на уровне HTML следующим образом:

```
<link rel=stylesheet href='style.css' type='text/css' title='sheet1'>

<style title='sheet2'>
  h1 { color: red }
</style>
```

первую стилевую таблицу можно
получить через document.styleSheets[0] или через document.styleSheets.sheet1,
вторую — через document.styleSheets[1] или через document.styleSheets.sheet2.

Средствами DOM (document.createElement() и т.д.) можно создавать и удалять стилевые таблицы.

Стилевая таблица имеет логическое свойство disabled, установив disabled в true можно отключить стилевую таблицу (и тогда все стилевые описания, указанные в стилевой таблице, будут игнорироваться браузером):

```
document.styleSheets[0].disabled=true;
```

К правилам, содержащимся в стилевой таблице, можно обратиться через свойство стилевой таблицы cssRules, которое является подобным массиву (свойства length, 0, 1 и т.д.), в Internet Explorer оно называется rules. Правила можно менять программно, но формат хранения правила зависит от браузера. В cssRules (rules) также есть методы для добавления и удаления правил, также некроссбраузерные. Например, для кроссбраузерного добавления стилевого правила в стилевую таблицу можно использовать следующую функцию:

```
function AddCSSRule(sheet, selector, rules, index)
{
  if ( 'insertRule' in sheet )
    sheet.insertRule(selector+'{'+rules+'}',index);
  else if( 'addRule' in sheet )
    sheet.addRule(selector,rules,index);
}
```

Пример вызова этой функции:

```
AddCSSRule( document.styleSheets[0],
  'div.SSS1', 'float: left; margin: 0;');
```

(3.67) Работа с положением и размером элементов

Каждый элемент имеет свойства, позволяющие узнавать его положение и размер на странице. Положение элемента есть координаты левого верхнего угла border-box элемента относительно левого верхнего угла content-box его родителя (или относительно документа, окна браузера, ближайшего позиционированного предка — в зависимости от значения стилевого свойства position). Border-box — это прямоугольник элемента, включающий padding и border элемента; content-box — это прямоугольник элемента, не включающий ни padding, ни border элемента. **Изменить положение** или **размер** элемента в некоторых случаях можно, устанавливая стилевые свойства style.left, style.top, style.right, style.bottom, style.width, style.height, однако **читать положение** или **размер** элемента следует иначе:

Обращение	Возвращаемое значение
элемент.offsetWidth	Возвращает ширину элемента на экране, в пикселях.
элемент.offsetHeight	Возвращает высоту элемента на экране, в пикселях.
элемент.offsetLeft	Возвращает расстояние на экране от левого края элемента до левого края его контейнера/предка/страницы/окна, в пикселях.
элемент.offsetTop	Возвращает расстояние на экране от верхнего края элемента до верхнего края его контейнера/предка/страницы/окна, в пикселях.

Схема, указывающая точку отсчёта позиции элемента при различных вариантах позиционирования:

Для вычисления положения элемента относительно левого верхнего угла окна браузера (не относительно начала документа!) можно использовать метод элемента getBoundingClientRect, который возвращает хэш со свойствами top, left, right, bottom, а в современных браузерах — также и свойствами width и height.

```
var E=document.getElementById('proba');
var PosH=E.getBoundingClientRect();
// теперь PosH - хэш вида {left:XXX, top:XXX, right:XXX, bottom:XXX}
```

Позицию элемента относительно левого верхнего угла страницы можно получить следующей функцией:

```
function GetElementPos(EL)
{
  var bbox=EL.getBoundingClientRect();
  return {
    left: bbox.left+window.pageXOffset,
    top: bbox.top+window.pageYOffset
  };
}
```

Пример использования:

```
var E=document.getElementById('proba');
// теперь E - span со словом 'использования' из текста чуть выше
console.log( E.offsetLeft );
57
console.log( E.offsetTop );
337
var Pos=GetElementPos(E);
console.log( Pos );
{left:75.890625, top:21944.21875}
```

Также, если известны координаты относительно левого верхнего угла страницы, можно получить ссылку на элемент, расположенный в этих координатах:

```
var E=document.elementFromPoint(100,100);
// теперь E - элемент из точки 100,100; если его нет - null
```