11 - Геометрия документа, элементов и прокрутка.

1. Введение

До сих пор мы рассматривали документы как некие абстрактные деревья элементов и текстовых узлов. Но когда броузер отображает документ в своем окне, он создает визуальное представление документа, в котором каждый элемент имеет определенную позицию и размеры. Часто веб-приложения могут интерпретировать документы как деревья элементов и никак не заботиться о том, как эти элементы отображаются на экране. Однако иногда бывает необходимо определить точные геометрические характеристики элемента. Если вам потребуется использовать CSS для динамического позиционирования элемента (такого как всплывающая подсказка или сноска) рядом с элементом, который позиционируется браузером, вам необходимо иметь возможность определять положение этого элемента. В этом лекции рассказывается, как можно переходить от абстрактной, древовидной модели документа к геометрическому, основанному на системе координат визуальному представлению документа в окне браузера, и обратно.

2. Координаты документа и видимой области

Позиция элемента измеряется в пикселах. Координата X растет слева направо, а координата Y – сверху вниз. Однако существуют две точки, которые мы можем считать началом координат: координаты X и Y элемента могут отсчитываться относительно верхнего левого угла документа или относительно верхнего левого угла видимой области. Для вкладок «видимой областью» является часть окна браузера, в которой фактически отображается содержимое документа: в нее не входит обрамление окна (меню, панели инструментов, вкладки).

Если документ меньше видимой области или если он еще не прокручивался, верхний левый угол документа находится в верхнем левом углу видимой области, и начала систем координат документа и видимой области совпадают. Однако в общем случае, чтобы перейти от одной системы координат к другой, необходимо добавлять или вычитать смещения прокрутки.

Система координат документа является более фундаментальной, чем система координат видимой области, и на нее не оказывает влияния величина прокрутки. Однако в клиентских сценариях довольно часто используются координаты видимой области. Система координат документа используется при позиционировании элементов с помощью CSS. Но проще всего получить координаты элемента в системе координат видимой области. Аналогично, когда регистрируется функция-обработчик событий от мыши, координаты указателя мыши передаются ей в системе координат видимой области.

Чтобы перейти от одной системы координат к другой, необходимо иметь возможность определять позиции полос прокрутки окна браузера, эти значения можно узнать с помощью свойств **pageXOffset и pageYOffset** объекта Window.

```
alert( 'Текущая прокрутка сверху: ' + window.pageYOffset );
alert( 'Текущая прокрутка слева: ' + window.pageXOffset );
```

Иногда бывает удобно иметь возможность определять размеры видимой области, например, чтобы определить, какая часть документа видима в настоящий момент. Самый простой способ узнать размеры видимой области использовать **innerWidth, innerHeight.**

Свойства **document.documentElement.clientWidth/**`Height` - если есть полоса прокрутки, возвращают именно ширину/высоту внутри неё, доступную для документа, а window.innerWidth/Height – игнорируют её наличие.

3. Ширина/высота страницы с учётом прокрутки

Теоретически, видимая часть страницы – это documentElement.clientWidth/Height, а полный размер с учётом прокрутки – по аналогии, **documentElement.scrollWidth/scrollHeight.** Это верно для обычных элементов. А вот для страницы с этими свойствами возникает проблема, когда прокрутка то есть, то нет. В этом случае они работают некорректно. В браузерах Chrome/Safari и Opera при отсутствии прокрутки значение documentElement.scrollHeight в этом случае может быть даже меньше, чем documentElement.clientHeight, что, конечно же, выглядит как совершеннейшая чепуха и нонсенс. Эта проблема возникает именно для documentElement, то есть для всей страницы. Надёжно определить размер страницы с учетом прокрутки можно, взяв максимум из нескольких свойств:

```
var scrollHeight = Math.max(
      document.body.scrollHeight,
      document.documentElement.scrollHeight,
      document.body.offsetHeight,
      document.documentElement.offsetHeight,
      document.body.clientHeight,
      document.documentElement.clientHeight
);
alert( 'Высота с учетом прокрутки: ' + scrollHeight
);
```

4. Определение геометрии элемента.

Самый простой способ определить размеры и координаты элемента – обратиться к его методу getBoundingClientRect(). Он не принимает аргументов и возвращает объект со свойствами left, right, top и bottom. Свойства left и top возвращают координаты X и Y верхнего левого угла элемента, а свойства right и bottom возвращают координаты правого нижнего угла. Этот метод возвращает позицию элемента в системе координат видимой области. Чтобы перейти к координатам относительно начала документа, которые не изменяются после прокрутки окна браузера пользователем, нужно добавить смещения прокрутки:

```
var box = elem.getBoundingClientRect(); // Координаты в видимой области
var x = box.left + window.pageXOffset; // Перейти к координатам документа
var y = box.top + window.pageYOffset;
```

Кроме того, во многих браузерах (и в стандарте W3C) объект, возвращаемый методом **getBoundingClientRect(),** имеет свойства width и height. Как Вы знаете, содержимое элемента окружается необязательной пустой областью, которая называется отступом (padding). Отступы окружаются необязательной рамкой (border), а рамка

окружается необязательными полями (margins). Координаты, возвращаемые методом getBoundingClientRect(), включают рамку и отступы элемента, но не включают поля.

Для определения координат и размеров отдельных прямоугольников, занимаемых строчными элементами, можно воспользоваться методом **getClientRects()**, который возвращает объект, подобный массиву, доступный только для чтения, чьи элементы представляют объекты прямоугольных областей, подобные тем, что возвращаются методом getBoundingClientRect().

5. Определение элемента в указанной точке.

Метод getBoundingClientRect() позволяет узнать текущую позицию элемента в видимой области. Но иногда бывает необходимо решить обратную задачу – узнать, какой элемент находится в заданной точке внутри видимой области. Сделать это можно с помощью метода elementFromPoint() объекта Document. Он принимает координаты X и Y (относительно начала координат видимой области, а не документа) и возвращает объект Element, находящийся в этой позиции. Если передать ему координаты точки, находящейся за пределами видимой области, метод **elementFromPoint()** вернет null, даже если после преобразования координат в систему координат документа получится вполне допустимая точка.

Метод **elementFromPoint()** выглядит весьма практичным, и наиболее очевидный случай его использования – определение элемента, находящегося под указателем мыши по его координатам. Однако, объект события от мыши уже содержит эту информацию в своем свойстве target. Именно поэтому на практике метод elementFromPoint() почти не используется.

6. Прокрутка.

Чтобы заставить броузер прокрутить документ, можно присваивать значение свойствам scrollLeft и scrollTop, но существует более простой путь, поддерживаемый с самых ранних дней развития языка JavaScript.

Метод scrollTo() объекта Window (и его синоним scroll()) принимает координаты X и Y точки (относительно начала координат документа) и устанавливает их в качестве величин смещения полос прокрутки. То есть он прокручивает окно так, что точка с указанными координатами оказывается в верхнем левом углу видимой области. Если указать точку, расположенную слишком близко к нижней или к правой границе документа, браузер попытается поместить эту точку как можно ближе к верхнему левому углу видимой области, но не сможет обеспечить точное их совпадение. Следующий пример прокручивает окно браузера так, что видимой оказывается самая нижняя часть документа:

```
// Получить высоту документа и видимой области. Метод offsetHeight описывается ниже.
var documentHeight = document.documentElement.offsetHeight;
var viewportHeight = window.innerHeight;

// И прокрутить окно так, чтобы переместить последнюю "страницу" в видимую область
window.scrollTo(0, documentHeight - viewportHeight);
```

Метод scrollBy() объекта Window похож на методы scroll() и scrollTo(), но их аргументы определяют относительное смещение и добавляются к текущим позициям полос прокрутки. Тем, кто умеет быстро читать, мог бы понравиться следующий код:

```
// Прокручивает документ на 10 пикселов каждые 200 мсек.
setInterval(function() {scrollBy(0,10)}, 200);
```

Часто требуется прокрутить документ не до определенных числовых координат, а до элемента в документе, который нужно сделать его видимым. В этом случае можно определить координаты элемента с помощью метода getBoundingClientRect(), преобразовать их в координаты относительно начала документа и передать их методу scrollTo(), но гораздо проще воспользоваться методом **scrollIntoView()** требуемого HTML-элемента. Этот метод гарантирует, что элемент, относительно которого он будет вызван, окажется в видимой области. По умолчанию он старается прокрутить документ так, чтобы верхняя граница элемента оказалась как можно ближе к верхней границе видимой области. Если в единственном аргументе передать методу значение false, он попытается прокрутить документ так, чтобы нижняя граница элемента совпала с нижней границей видимой области. Кроме того, браузер выполнит прокрутку по горизонтали, если это потребуется, чтобы сделать элемент видимым.

ДЗ:
1. Постраничный scroll
2. Плавный возврат в начало
3. Режим чтения с настройкой