**Координаты в документе**

*Система координат относительно страницы* или, иначе говоря, *относительно документа*, начинается в левом-верхнем углу, но не окна, а именно страницы.

И координаты в ней означают позицию по отношению не к окну браузера, а к документу в целом.

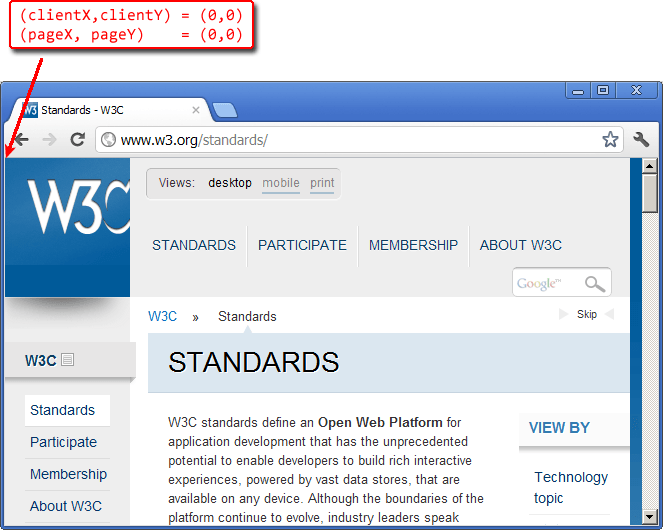
Если провести аналогию с CSS, то координаты относительно окна – это position:fixed, а относительно документа – position:absolute (при позиционировании вне других элементов, естественно).

Мы будем называть координаты в ней pageX/pageY.

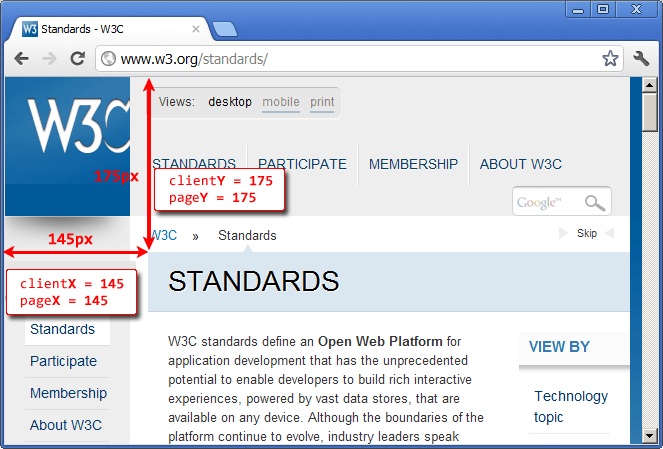
Они нужны в первую очередь для того, чтобы показывать элемент в определённом месте страницы, а не окна.

**[Сравнение систем координат](https://learn.javascript.ru/coordinates-document" \l "%D1%81%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC-%D0%BA%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82)**

Когда страница не прокручена, точки начала координат относительно окна (clientX,clientY) и документа (pageX,pageY) совпадают:



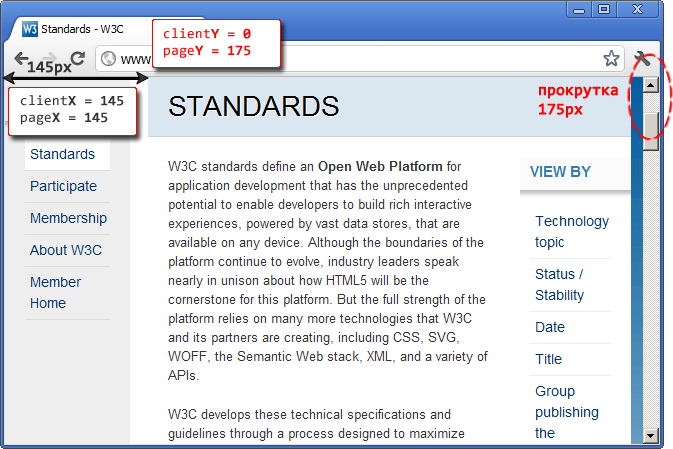
Например, координаты элемента с надписью «STANDARDS» равны расстоянию от верхней/левой границы окна:



**Прокрутим страницу, чтобы элемент был на самом верху:**

Посмотрите на рисунок ниже, на нём – та же страница, только прокрученная, и тот же элемент «STANDARDS».

* Координата clientY изменилась. Она была 175, а стала 0, так как элемент находится вверху окна.
* Координата pageY осталась такой же, так как отсчитывается от левого-верхнего угла *документа*.



Итак, координаты pageX/pageY не меняются при прокрутке, в отличие от clientX/clientY.

**[Получение координат](https://learn.javascript.ru/coordinates-document" \l "getCoords)**

К сожалению, готовой функции для получения координат элемента относительно страницы нет. Но её можно легко написать самим.

Эти две системы координат жёстко связаны: pageY = clientY + текущая вертикальная прокрутка.

Наша функция getCoords(elem) будет брать результат elem.getBoundingClientRect() и прибавлять текущую прокрутку документа.

Результат getCoords: объект с координатами {left: .., top: ..}

function getCoords(elem) { // кроме IE8-

var box = elem.getBoundingClientRect();

return {

top: box.top + pageYOffset,

left: box.left + pageXOffset

};

}

Если нужно поддерживать более старые IE, то вот альтернативный, самый кросс-браузерный вариант:

function getCoords(elem) {

// (1)

var box = elem.getBoundingClientRect();

var body = document.body;

var docEl = document.documentElement;

// (2)

var scrollTop = window.pageYOffset || docEl.scrollTop || body.scrollTop;

var scrollLeft = window.pageXOffset || docEl.scrollLeft || body.scrollLeft;

// (3)

var clientTop = docEl.clientTop || body.clientTop || 0;

var clientLeft = docEl.clientLeft || body.clientLeft || 0;

// (4)

var top = box.top + scrollTop - clientTop;

var left = box.left + scrollLeft - clientLeft;

return {

top: top,

left: left

};

}

Разберем что и зачем, по шагам:

1. Получаем прямоугольник.
2. Считаем прокрутку страницы. Все браузеры, кроме IE8- поддерживают свойство pageXOffset/pageYOffset. В более старых IE, когда установлен DOCTYPE, прокрутку можно получить из documentElement, ну и наконец если DOCTYPE некорректен – использовать body.
3. В IE документ может быть смещен относительно левого верхнего угла. Получим это смещение.
4. Добавим прокрутку к координатам окна и вычтем смещение html/body, чтобы получить координаты нужного нам элемента.

**[Устаревший метод: offset\*](https://learn.javascript.ru/coordinates-document" \l "%D1%83%D1%81%D1%82%D0%B0%D1%80%D0%B5%D0%B2%D1%88%D0%B8%D0%B9-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4-offset)**

Есть альтернативный способ нахождения координат – это пройти всю цепочку offsetParent от элемента вверх и сложить отступы offsetLeft/offsetTop.

Мы разбираем его здесь с учебной целью, так как он используется лишь в старых браузерах.

Вот функция, реализующая такой подход.

function getOffsetSum(elem) {

var top = 0,

left = 0;

while (elem) {

top = top + parseInt(elem.offsetTop);

left = left + parseInt(elem.offsetLeft);

elem = elem.offsetParent;

}

return {

top: top,

left: left

};

}

Казалось бы, код нормальный. И он как-то работает, но разные браузеры преподносят «сюрпризы», включая или выключая размер рамок и прокруток из offsetTop/Left, некорректно учитывая позиционирование. В итоге результат не всегда верен. Можно, конечно, разобрать эти проблемы и посчитать действительно аккуратно и правильно этим способом, но зачем? Ведь есть getBoundingClientRect.

Вы можете увидеть разницу между вычислением координат через offset\* и getBoundingClientRectна примере.

В прямоугольнике ниже есть 3 вложенных DIV. Все они имеют border, кое-кто из них имеет position/margin/padding.

Кликните по внутреннему (жёлтому) элементу, чтобы увидеть результаты обоих методов: getOffsetSum и getCoords, а также реальные координаты курсора – event.pageX/pageY (мы обсудим их позже в статье [Мышь: IE8-, исправление события](https://learn.javascript.ru/fixevent)).

Кликните, чтобы получить координаты getOffsetSum и getCoords

**getOffsetSum**:значение getOffsetSum()

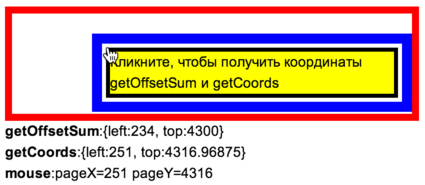
**getCoords**:значение getCoords()

**mouse**:координаты мыши

**При клике на любом месте желтого блока вы легко увидите разницу между getOffsetSum(elem) и getCoords(elem).**

Для того, чтобы узнать, какой же результат верный, кликните в левом-верхнем углу жёлтого блока, причём обратите внимание – кликать нужно не на жёлтом, а на чёрном, это рамка, она тоже входит в элемент. Будут видны точные координаты мыши, так что вы можете сравнить их с getOffsetSum/getCoords.

Пример клика в правильном месте (обратите внимание на разницу координат):

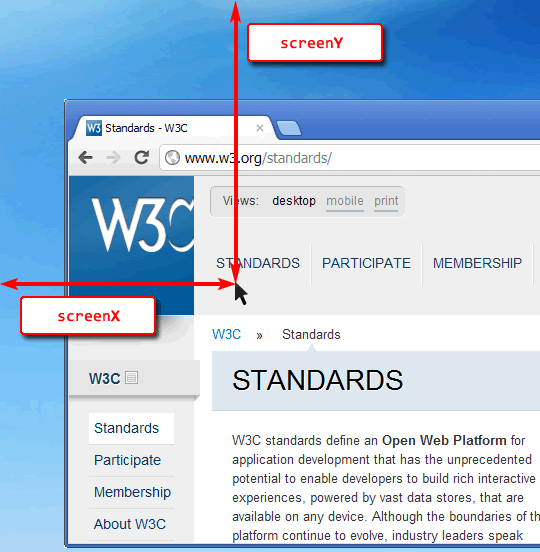


Именно getCoords всегда возвращает верное значение.

**[Координаты на экране screenX/screenY](https://learn.javascript.ru/coordinates-document" \l "%D0%BA%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82%D1%8B-%D0%BD%D0%B0-%D1%8D%D0%BA%D1%80%D0%B0%D0%BD%D0%B5-screenx-screeny)**

Есть ещё одна система координат, которая используется очень редко, но для полноты картины необходимо её упомянуть.

*Координаты относительно экрана* screenX/screenY отсчитываются от его левого-верхнего угла. Имеется в виду именно *весь экран*, а не окно браузера.



Такие координаты могут быть полезны, например, при работе с мобильными устройствами или для открытия нового окна посередине экрана вызовом [window.open](https://developer.mozilla.org/en-US/docs/DOM/window.open).

* Размеры экрана хранятся в глобальной переменной [screen](https://developer.mozilla.org/en/DOM/window.screen):
* // общая ширина/высота
* alert( screen.width + ' x ' + screen.height );
* // доступная ширина/высота (за вычетом таскбара и т.п.)
* alert( screen.availWidth + ' x ' + screen.availHeight );

// есть и ряд других свойств screen (см. документацию)

* Координаты левого-верхнего угла браузера на экране хранятся в window.screenX, window.screenY(не поддерживаются IE8-):

alert( "Браузер находится на " + window.screenX + "," + window.screenY );

Они могут быть и меньше нуля, если окно частично вне экрана.

Заметим, что общую информацию об экране и браузере получить можно, а вот координаты конкретного элемента на экране – нельзя, нет аналога getBoundingClientRect или иного метода для этого.

**[Итого](https://learn.javascript.ru/coordinates-document" \l "%D0%B8%D1%82%D0%BE%D0%B3%D0%BE)**

У любой точки в браузере есть координаты:

1. Относительно окна window – elem.getBoundingClientRect().
2. Относительно документа document – добавляем прокрутку, во всех фреймворках есть готовая функция.
3. Относительно экрана screen – можно узнать координаты браузера, но не элемента.

Иногда в старом коде можно встретить использование offsetTop/Left для подсчёта координат. Это очень старый и неправильный способ, не стоит его использовать.

Координаты будут нужны нам далее, при работе с событиями мыши (координаты клика) и элементами (перемещение).

[**Задачи**](https://learn.javascript.ru/coordinates-document#tasks)

### О[бласть видимости для документа](https://learn.javascript.ru/coordinates-document" \l "%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C-%D0%B2%D0%B8%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8-%D0%B4%D0%BB%D1%8F-%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0)

важность: 5

Напишите функцию getDocumentScroll(), которая возвращает объект с информацией о текущей прокрутке и области видимости.

Свойства объекта-результата:

* top – координата верхней границы видимой части (относительно документа).
* bottom – координата нижней границы видимой части (относительно документа).
* height – полная высота документа, включая прокрутку.

В этой задаче учитываем только вертикальную прокрутку: горизонтальная делается аналогично, а нужна сильно реже.

решение

* top – это pageYOffset.
* bottom – это pageYOffset плюс высота видимой части documentElement.clientHeight.
* height – полная высота документа, её вычисление дано в главе [Размеры и прокрутка страницы](https://learn.javascript.ru/metrics-window).

Итого:

function getDocumentScroll() {

var scrollHeight = Math.max(

document.body.scrollHeight, document.documentElement.scrollHeight,

document.body.offsetHeight, document.documentElement.offsetHeight,

document.body.clientHeight, document.documentElement.clientHeight

);

return {

top: pageYOffset,

bottom: pageYOffset + document.documentElement.clientHeight,

height: scrollHeight

};

}

**[Разместить заметку рядом с элементом (absolute)](https://learn.javascript.ru/coordinates-document" \l "%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%81%D1%82%D0%B8%D1%82%D1%8C-%D0%B7%D0%B0%D0%BC%D0%B5%D1%82%D0%BA%D1%83-%D1%80%D1%8F%D0%B4%D0%BE%D0%BC-%D1%81-%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BC-absolute)**

важность: 5

Модифицируйте решение задачи [Разместить заметку рядом с элементом](https://learn.javascript.ru/task/position-at), чтобы при прокрутке страницы заметка не убегала от элемента.

Используйте для этого координаты относительно документа и position:absolute вместо position:fixed.

В качестве исходного документа используйте решение задачи [Разместить заметку рядом с элементом](https://learn.javascript.ru/task/position-at), для тестирования прокрутки добавьте стиль <body style="height: 2000px">.

решение

/\*\*

\* Вспомогательная функция для координат относительно документа

\*/

function getCoords(elem) {

var box = elem.getBoundingClientRect();

var body = document.body;

var docEl = document.documentElement;

var scrollTop = window.pageYOffset || docEl.scrollTop || body.scrollTop;

var scrollLeft = window.pageXOffset || docEl.scrollLeft || body.scrollLeft;

var clientTop = docEl.clientTop || body.clientTop || 0;

var clientLeft = docEl.clientLeft || body.clientLeft || 0;

var top = box.top + scrollTop - clientTop;

var left = box.left + scrollLeft - clientLeft;

return {

top: top,

left: left

};

}

function positionAt(anchor, position, elem) {

var anchorCoords = getCoords(anchor);

switch (position) {

case "top":

elem.style.left = anchorCoords.left + "px";

elem.style.top = anchorCoords.top - elem.offsetHeight + "px";

break;

case "right":

elem.style.left = anchorCoords.left + anchor.offsetWidth + "px";

elem.style.top = anchorCoords.top + "px";

break;

case "bottom":

elem.style.left = anchorCoords.left + "px";

elem.style.top = anchorCoords.top + anchor.offsetHeight + "px";

break;

}

}

/\*\*

\* Показывает заметку с текстом html на позиции position

\* относительно элемента anchor

\*/

function showNote(anchor, position, html) {

var note = document.createElement('div');

note.className = "note";

note.innerHTML = html;

document.body.appendChild(note);

positionAt(anchor, position, note);

}

// проверка работоспособности

var blockquote = document.querySelector('blockquote');

showNote(blockquote, "top", "заметка сверху");

showNote(blockquote, "right", "заметка справа");

showNote(blockquote, "bottom", "заметка снизу");

[**Разместить заметку внутри элемента**](https://learn.javascript.ru/coordinates-document#%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%81%D1%82%D0%B8%D1%82%D1%8C-%D0%B7%D0%B0%D0%BC%D0%B5%D1%82%D0%BA%D1%83-%D0%B2%D0%BD%D1%83%D1%82%D1%80%D0%B8-%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0)

важность: 5

Расширьте предыдущую задачу [Разместить заметку рядом с элементом (absolute)](https://learn.javascript.ru/task/position-at-absolute): научите функцию positionAt(anchor, position, elem) вставлять elem внутрь anchor.

Новые значения position:

* top-out, right-out, bottom-out – работают так же, как раньше, то есть вставляют elem над/справа/под anchor.
* top-in, right-in, bottom-in – вставляют elem внутрь anchor: к верхней границе/правой/нижней.

Например:

// покажет note сверху blockquote

positionAt(blockquote, "top-out", note);

// покажет note сверху-внутри blockquote

positionAt(blockquote, "top-in", note);

Пример результата:

В качестве исходного документа возьмите решение задачи [Разместить заметку рядом с элементом (absolute)](https://learn.javascript.ru/task/position-at-absolute).

решение

function getCoords(elem) {

var box = elem.getBoundingClientRect();

var body = document.body;

var docEl = document.documentElement;

var scrollTop = window.pageYOffset || docEl.scrollTop || body.scrollTop;

var scrollLeft = window.pageXOffset || docEl.scrollLeft || body.scrollLeft;

var clientTop = docEl.clientTop || body.clientTop || 0;

var clientLeft = docEl.clientLeft || body.clientLeft || 0;

var top = box.top + scrollTop - clientTop;

var left = box.left + scrollLeft - clientLeft;

return {

top: top,

left: left

};

}

function showNote(anchor, position, html) {

var note = document.createElement('div');

note.className = "note";

note.innerHTML = html;

document.body.appendChild(note);

positionAt(anchor, position, note);

}

/\*\*

\* Позиционирует элемент elem относительно элемента anchor, как указано в

\* в position.

\*

\* @param {Node} anchor Элемент-якорь, относительно которого задана позиция

\* @param {string} position вне элемента top-out/right-out/bottom-out, внутри элемента \*-in

\* @param {Node} elem Элемент, который будет позиционирован

\*

\* Оба элемента elem && anchor должны быть видимы в документе.

\*/

function positionAt(anchor, position, elem) {

var anchorCoords = getCoords(anchor);

switch (position) {

case "top-out":

elem.style.left = anchorCoords.left + "px";

elem.style.top = anchorCoords.top - elem.offsetHeight + "px";

break;

case "right-out":

elem.style.left = anchorCoords.left + anchor.offsetWidth + "px";

elem.style.top = anchorCoords.top + "px";

break;

case "bottom-out":

elem.style.left = anchorCoords.left + "px";

elem.style.top = anchorCoords.top + anchor.offsetHeight + "px";

break;

case "top-in":

elem.style.left = anchorCoords.left + "px";

elem.style.top = anchorCoords.top + "px";

break;

case "right-in":

elem.style.width = '150px';

elem.style.left = anchorCoords.left + anchor.offsetWidth - elem.offsetWidth + "px";

elem.style.top = anchorCoords.top + "px";

break;

case "bottom-in":

elem.style.left = anchorCoords.left + "px";

elem.style.top = anchorCoords.top + anchor.offsetHeight - elem.offsetHeight + "px";

break;

}

}

var blockquote = document.querySelector('blockquote');

showNote(blockquote, "top-in", "заметка top-in");

showNote(blockquote, "top-out", "заметка top-out");

showNote(blockquote, "right-out", "заметка right-out");

showNote(blockquote, "bottom-in", "заметка bottom-in");