DOM-модель

JavaScript

Атрибуты

- Доступ к атрибутам осуществляется при помощи стандартных методов:
  - elem.hasAttribute(name) – проверяет наличие атрибута
  - elem.getAttribute(name) – получает значение атрибута
  - elem.setAttribute(name, value) – устанавливает атрибут
  - elem.removeAttribute(name) – удаляет атрибут
  - elem.attributes – получает все атрибуты элемента
- В отличие от свойств, атрибуты:
  - Всегда являются строками
  - Их имя *нечувствительно* к регистру
  - Видны в innerHTML

Пример

- <body>
  -
  - <script>
    - alert(elem.getAttribute('About'));
    - elem.setAttribute('Test', 123);
    - alert(document.body.innerHTML);
    - var attrs = elem.attributes;
    - for (var i = 0; i < attrs.length; i++) {
      - alert(attrs[i].name + " = " + attrs[i].value);
    - }
  - </script>
- </body>

Пример 2

- <input id="input" type="checkbox" checked>
- <script>
  - // работа с checked через атрибут
  - alert(input.getAttribute('checked'));
  - input.removeAttribute('checked');
  - // работа с checked через свойство
  - alert(input.checked);
  - input.checked = true;
- </script>

Пример 3

- <body>
  - <input id="input" type="text" value="markup">
  - <script>
    - input.setAttribute('value', 'new');
    - alert(input.value);
  - </script>
- </body>

Классы

- elem.className – возвращает классы элемента в виде строки
- elem.classList – возвращает классы элемента в виде объекта
  - elem.classList.contains("class") – возвращает true/false, в зависимости от того, есть ли у элемента класс class
  - elem.classList.add/remove("class") – добавляет/удаляет класс class
  - elem.classList.toggle("class") – если класса class нет, добавляет его, если есть — удаляет.

Пример

- <body class="main page">
  - <script>
    - var classList = document.body.classList;
    - classList.remove('page');
    - classList.add('post');
    - for (var i = 0; i < classList.length; i++) {
      - alert(classList[i]);
    - }
    - alert(classList.contains('post'));
    - alert(document.body.className);
  - </script>
- </body>

Нестандартные атрибуты

-
- <script>
  - alert(elem.id); // elem
  - alert(elem.about); // undefined
- </script>

Создание узлов

- document.createElement(tag) – создает новый элемент с указанным тегом
- document.createTextNode(text) – создает новый *текстовый* узел с данным текстом

- var div = document.createElement('div');
- div.className = "alert alert-success";
- div.innerHTML = "Важное сообщение!";

Добавление элемента

- parentEl.appendChild(elem) – добавляет elem в конец дочерних элементов parentEl
- parentEl.insertBefore(elem, nextSibling) – вставляет elem в коллекцию детей parentEl, перед элементом nextSibling

Пример

- <ol id="list">
  - 0
  - 1
  - 2
-
- <script>
  - var newLi = document.createElement('li');
  - newLi.innerHTML = 'Привет, мир!';
  - list.appendChild(newLi);
- </script>

Пример 2

- <ol id="list">
  - 0
  - 1
  - 2
-
- <script>
  - var newLi = document.createElement('li');
  - newLi.innerHTML = 'Привет, мир!';
  - list.insertBefore(newLi, list.children[1]);
- </script>

Клонирование узлов

- elem.cloneNode(true) – создает «глубокую» копию элемента вместе с атрибутами, включая подэлементы
- elem.cloneNode(false) – создает копию без дочерних элементов

Пример

- <script>
  - var div = document.createElement('div');
  - div.className = "alert alert-success";
  - div.innerHTML = "Важно";
  - document.body.appendChild(div);
  - var div2 = div.cloneNode(true);
  - div2.querySelector('strong').innerHTML = 'Супер!';
  - document.body.appendChild(div2);
- </script>

Удаление узлов

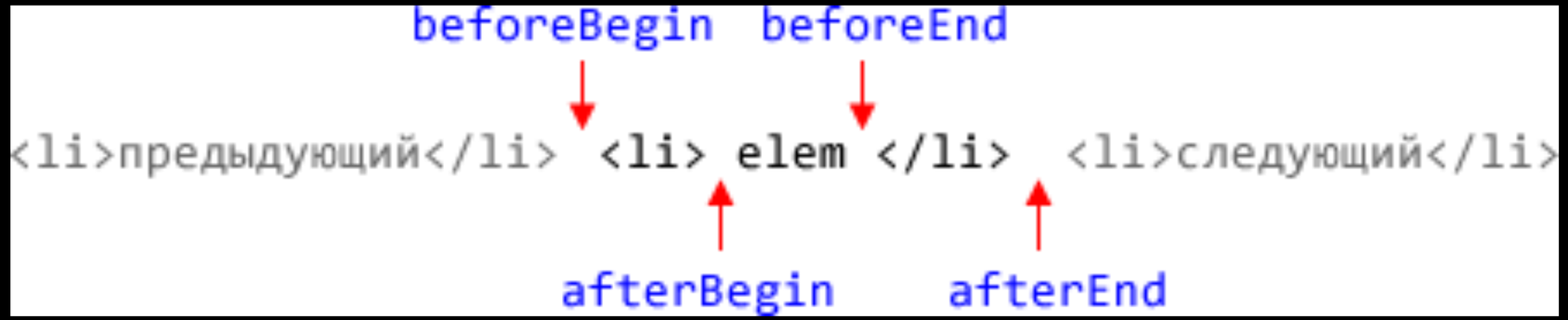
- parentElem.removeChild(elem) – удаляет elem из списка детей parentElem
- parentElem.replaceChild(newElem, elem) – среди детей parentElem удаляет elem и вставляет на его место newElem

- div.parentNode.removeChild(div);

insertAdjacent

- Метод insertAdjacentHTML позволяет вставлять произвольный HTML в любое место документа, в том числе и между узлами
- Поддерживается всеми браузерами, кроме Firefox меньше версии 8
- elem.insertAdjacentHTML(where, html);

where


beforeBegin
beforeEnd
предыдущий
 elem
следующий
afterBegin
afterEnd

Пример

-
  - 1
  - 2
  - 4
-
- <script>
  - var ul = document.body.children[0];
  - var li4 = ul.children[2];
  - li4.insertAdjacentHTML("beforeBegin", "
3
");
- </script>

Схожие методы

- elem.insertAdjacentElement(where, newElem) – вставляет в произвольное место элемент newElem
- elem.insertAdjacentText(where, text) – создаёт текстовый узел из строки text и вставляет его в указанное место относительно elem

Современные методы

- Методы:
  - node.append(...nodes) – вставляет nodes в конец node
  - node.prepend(...nodes) – вставляет nodes в начало node
  - node.after(...nodes) – вставляет nodes после узла node
  - node.before(...nodes) – вставляет nodes перед узлом node
  - node.replaceWith(...nodes) – вставляет nodes вместо node
- Методы ничего не возвращают
- nodes — DOM-узлы или строки
- Строки вставляются как текстовые узлы

document.write / writeln

- Метод пишет текст прямо в HTML
- Работает только из скриптов, выполняемых в процессе загрузки страницы (запуск после загрузки приведёт к очистке документа)
- Метод не изменяет существующий документ, а работает на стадии текста, до того как DOM-структура сформирована

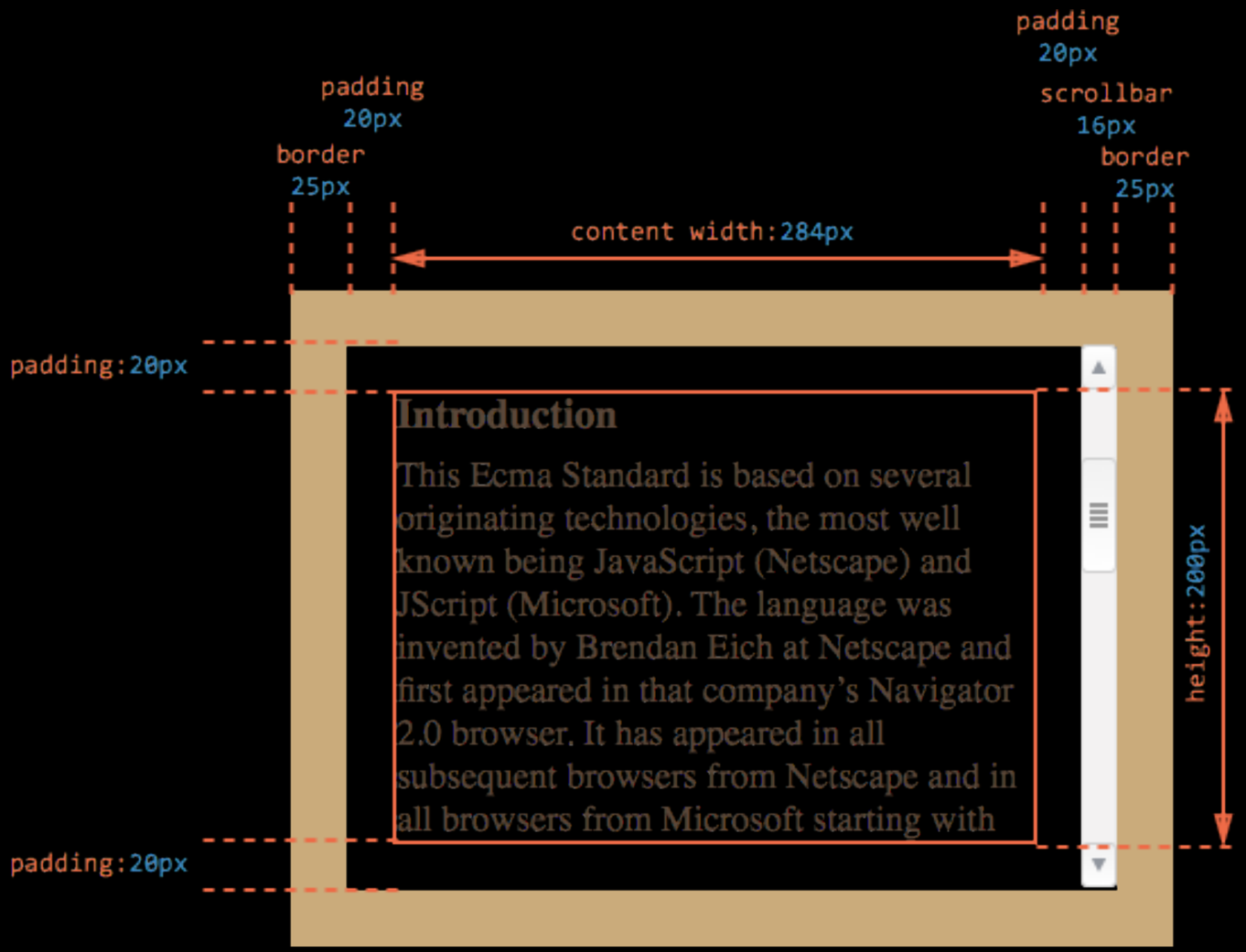
Полный стиль

- <style>
  - body { margin: 10px }
- </style>
- <body>
  - <script>
    - var style = getComputedStyle(document.body);
    - alert(style.marginTop);
    - alert(style.color);
  - </script>
- </body>

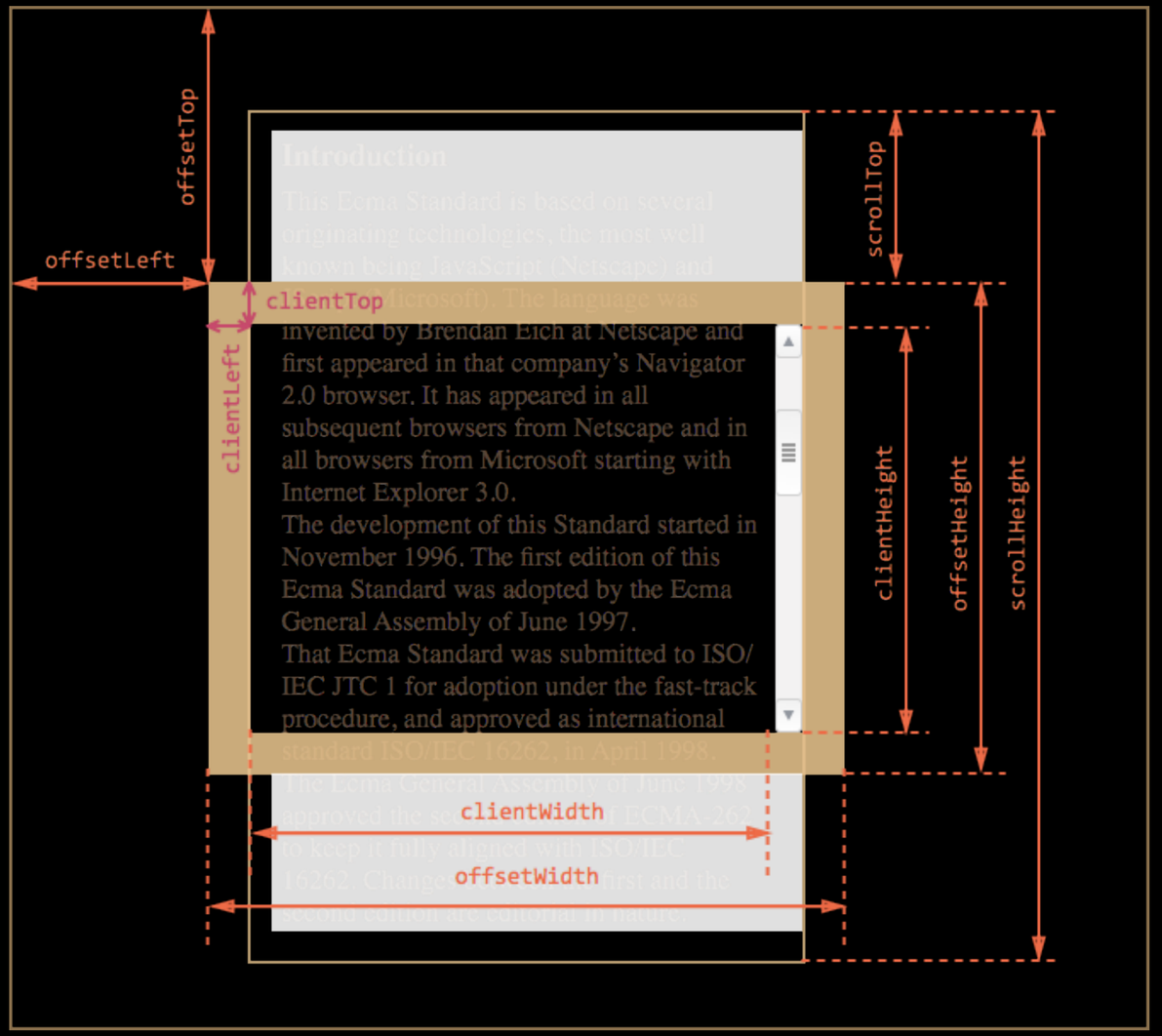
Образец документа

- <style>
  - #example {
    - width: 300px;
    - height: 200px;
    - border: 25px solid #E8C48F;
    - padding: 20px;
    - overflow: auto;
  - }
- </style>
- ...Текст...

Результат


padding
20px
padding
20px
scrollbar
16px
border
25px
border
25px
content width:284px
padding:20px
Introduction
This Ecma Standard is based on several originating technologies, the most well known being JavaScript (Netscape) and JScript (Microsoft). The language was invented by Brendan Eich at Netscape and first appeared in that company's Navigator 2.0 browser. It has appeared in all subsequent browsers from Netscape and in all browsers from Microsoft starting with
height:200px
padding:20px

Метрики


offsetTop
offsetLeft
clientTop
clientLeft
scrollTop
clientHeight
offsetHeight
scrollHeight
clientWidth
offsetWidth
Introduction
This Ecma Standard is based on several originating technologies, the most well known being JavaScript (Netscape) and JScript (Microsoft). The language was invented by Brendan Eich at Netscape and first appeared in that company's Navigator 2.0 browser. It has appeared in all subsequent browsers from Netscape and in all browsers from Microsoft starting with Internet Explorer 3.0.
The development of this Standard started in November 1996. The first edition of this Ecma Standard was adopted by the Ecma General Assembly of June 1997.
That Ecma Standard was submitted to ISO/IEC JTC 1 for adoption under the fast-track procedure, and approved as international standard ISO/IEC 16262, in April 1998.
The Ecma General Assembly of June 1998
approved the second edition of ECMA-262
to keep it fully aligned with ISO/IEC
16262. Changes between the first and the
second edition are editorial in nature.

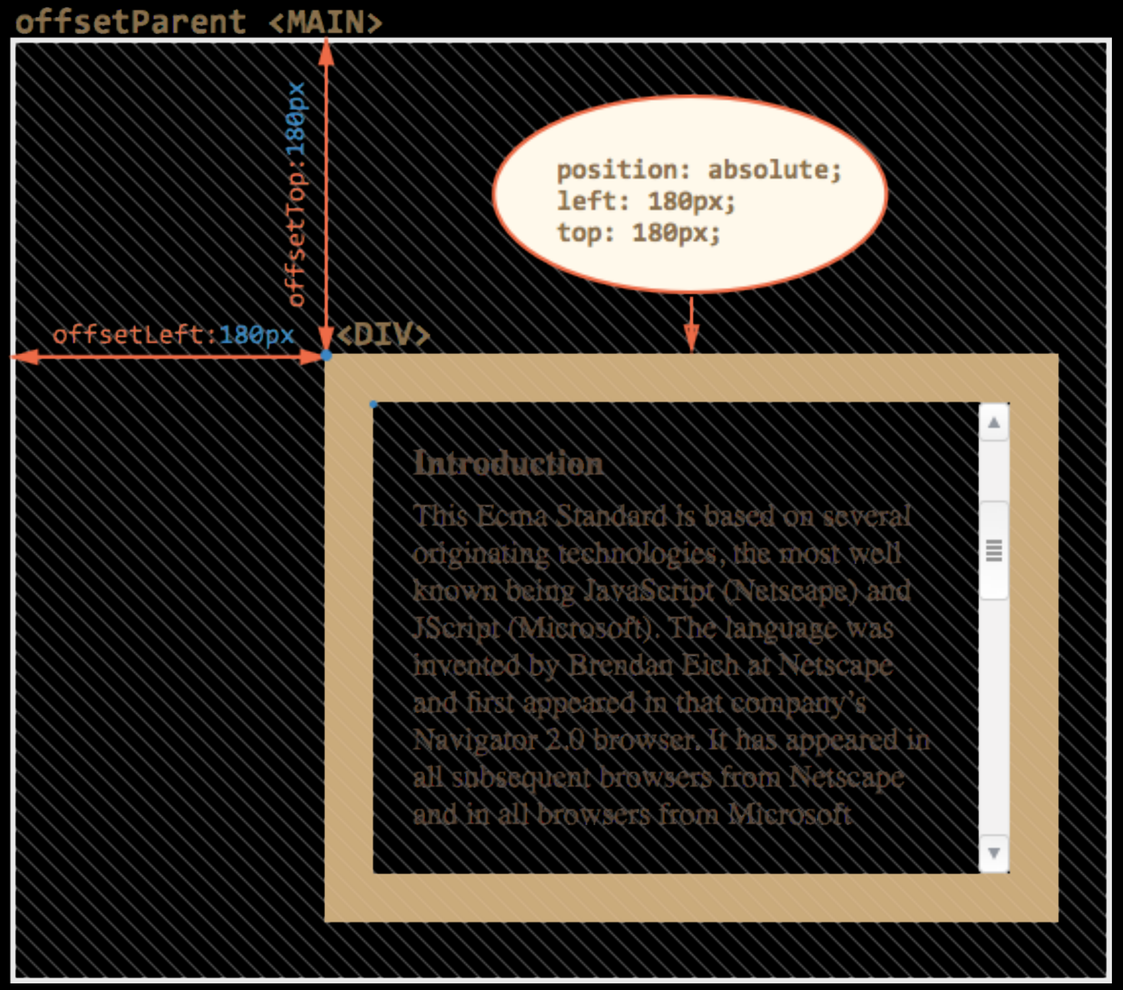
offsetParent

- В offsetParent находится ссылка на родительский элемент отображения
- Браузер высчитывает «дерево рендеринга», содержащее всю информацию о размерах
- Элементы рисуются один внутри другого
- При position: absolute, расположение вычисляется относительно ближайшего позиционированного элемента или BODY

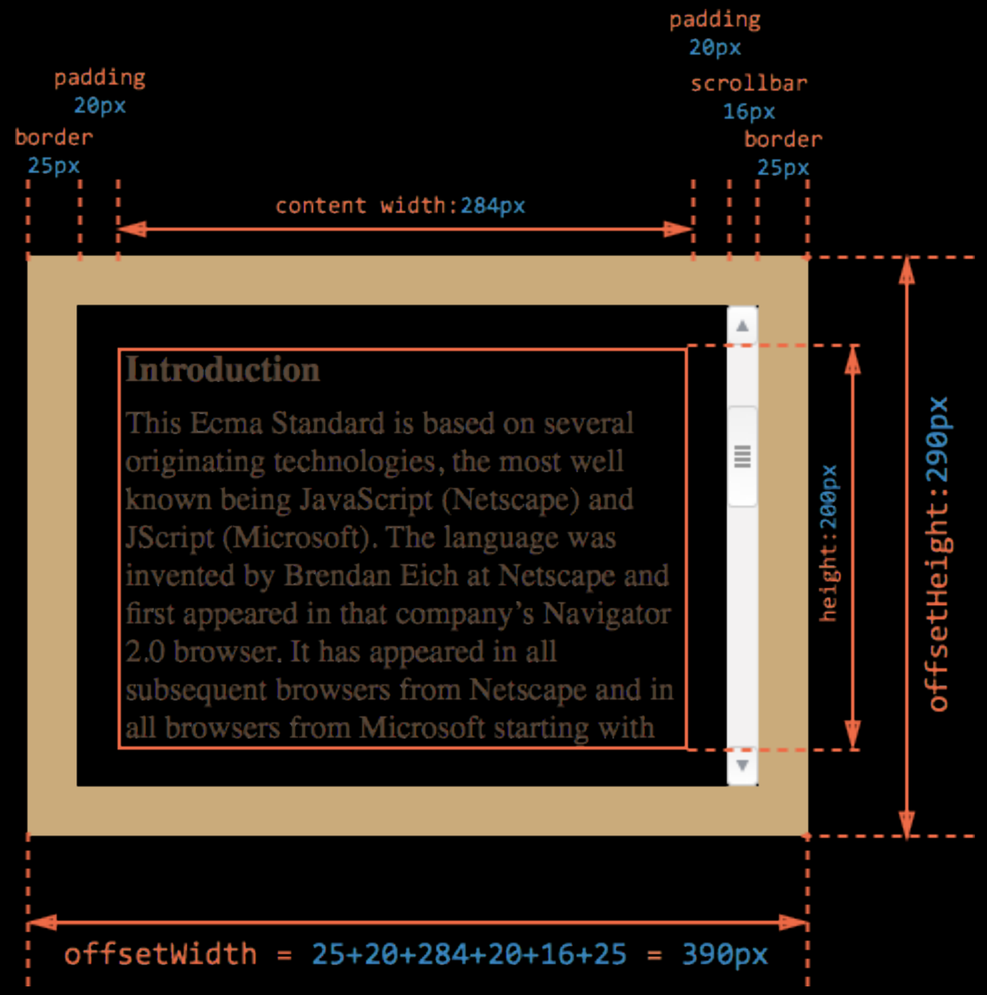
offsetLeft/Top

- Свойства offsetLeft/Top задают смещение относительно offsetParent
- <main style="position: relative">
  - <form>
    -
      - ...
    -
  - </form>
- </main>

Действительность



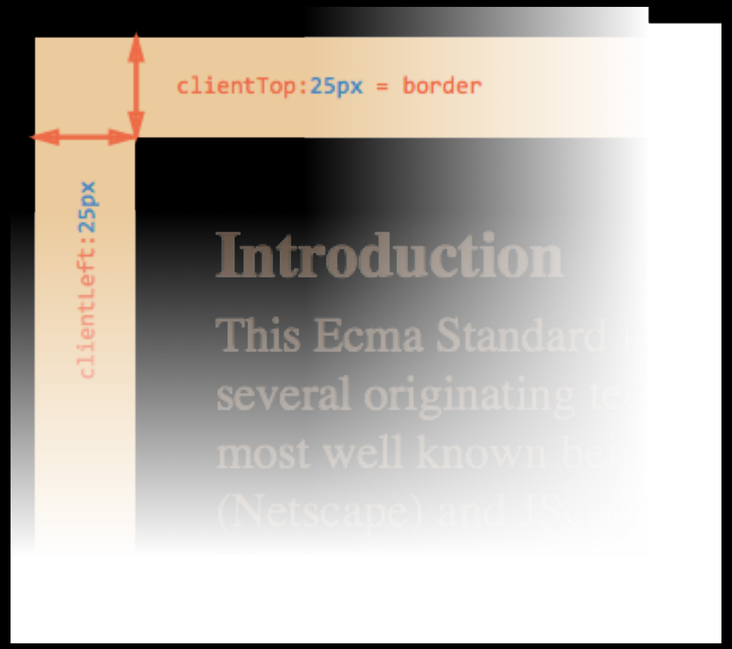
offsetWidth/Height



clientTop/Left

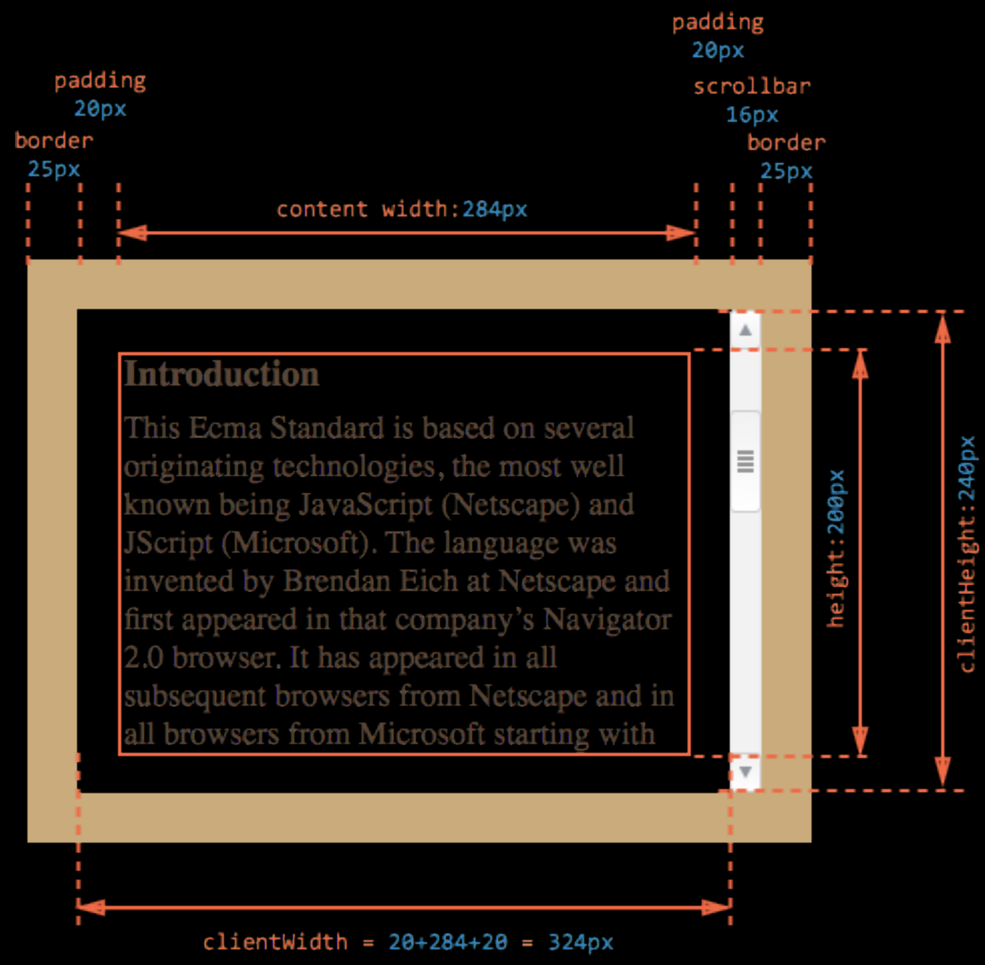
- clientLeft/clientTop – отступ области содержимого от левого-верхнего угла элемента
- Если операционная система располагает вертикальную прокрутку справа, то равны ширинам левой/верхней рамки
- Если слева (ОС на иврите, арабском), то clientLeft включает в себя еще и прокрутку

Пример

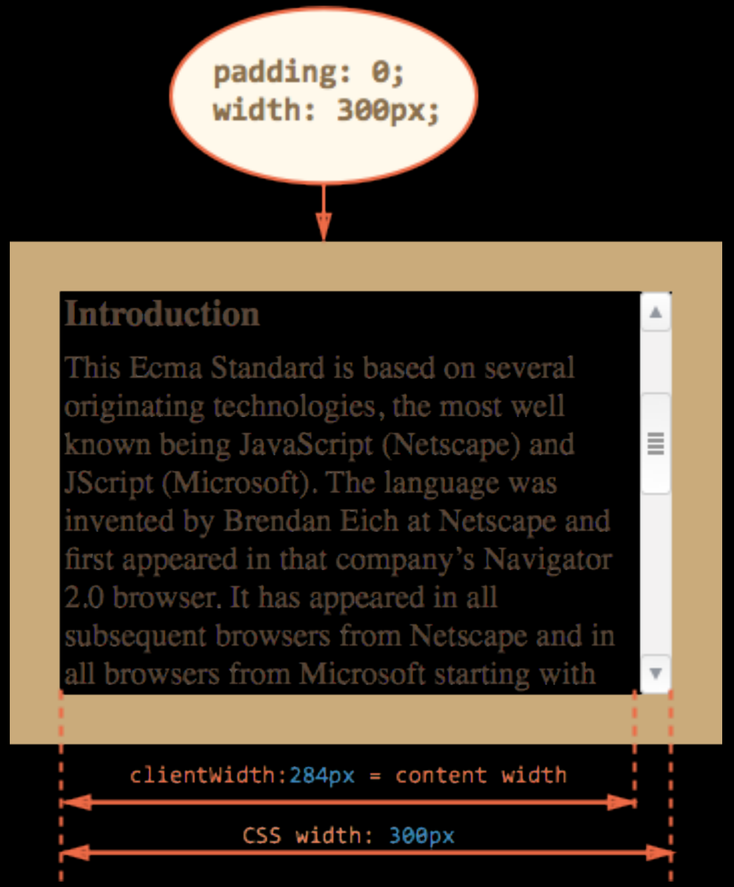

clientTop:25px = border
clientLeft:25px
Introduction
This Ecma Standard
several originating
most well known
(Netscape) and


clientTop:25px = border
clientLeft:41px

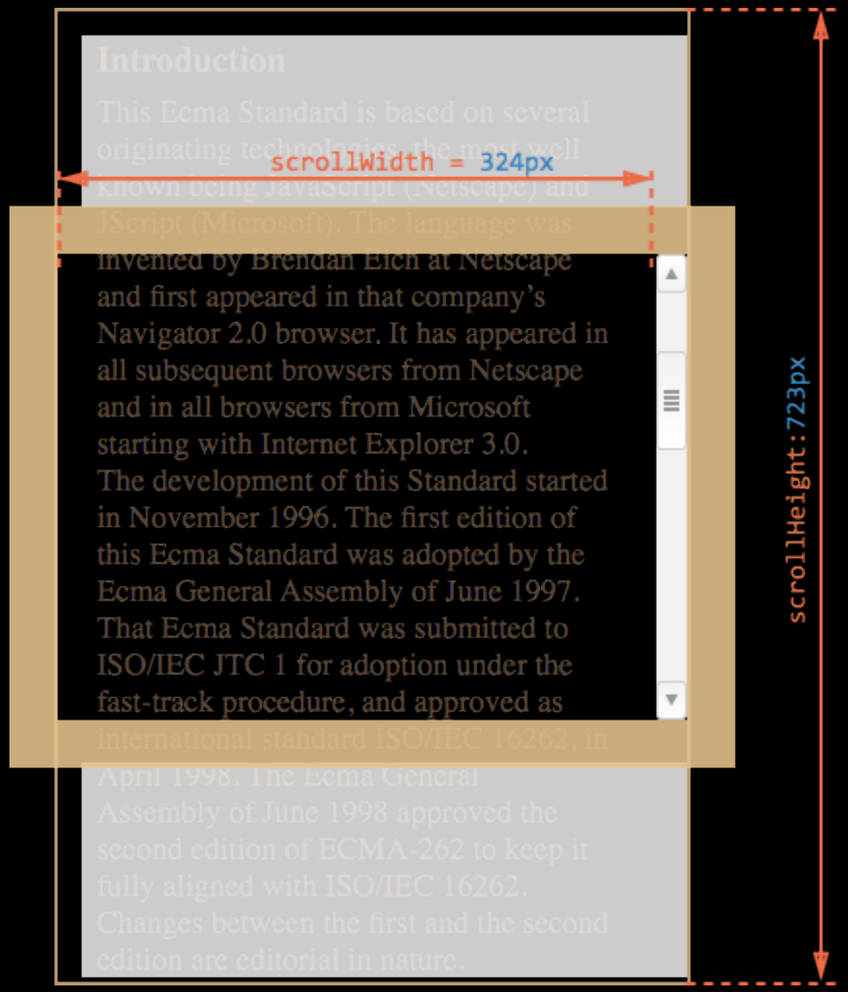
clientWidth/Height



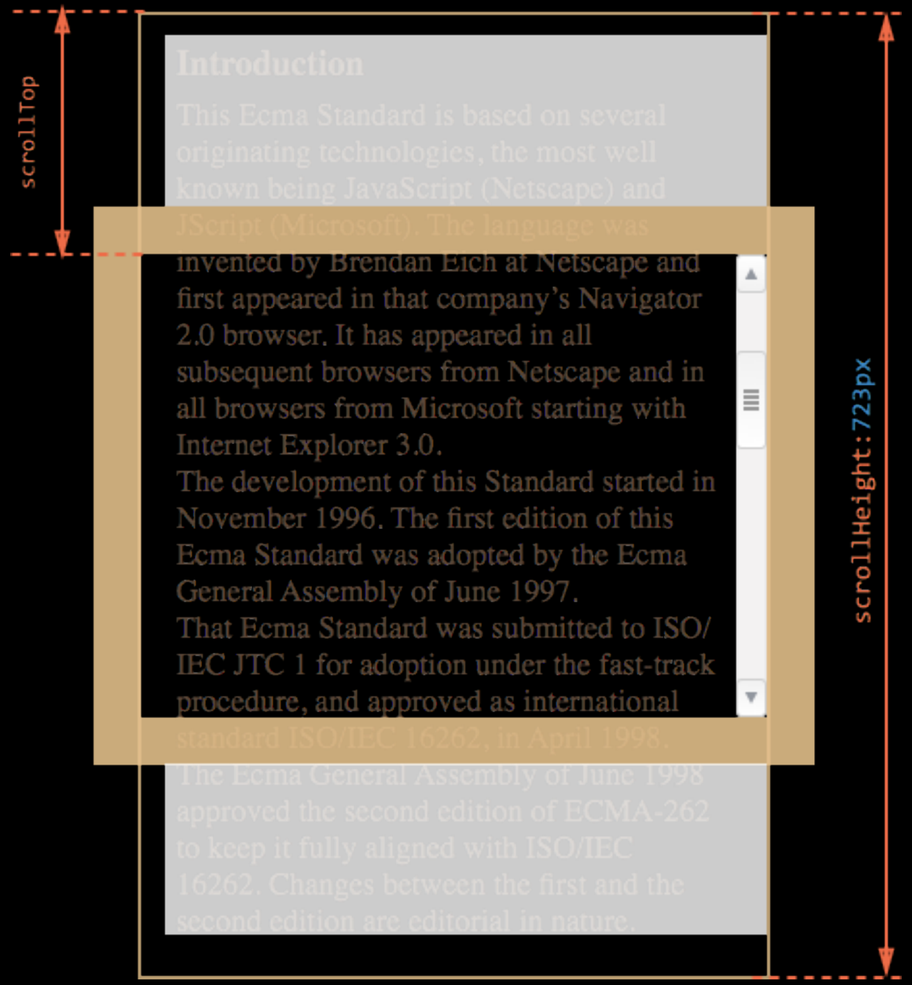
Размер без padding



scrollWidth/Height



scrollLeft/scrollTop



Видимая часть окна

- Размеров видимой части окна:
  - document.documentElement.clientWidth
  - document.documentElement.clientHeight
- Размер страницы с учётом прокрутки:
  - var scrollHeight = Math.max(document.body.scrollHeight, document.documentElement.scrollHeight, document.body.offsetHeight, document.documentElement.offsetHeight, document.body.clientHeight, document.documentElement.clientHeight);

Прокрутка окна

- window.pageYOffset – вертикальная прокрутка окна
- window.pageXOffset – горизонтальная прокрутка окна
- Работает везде кроме IE8-

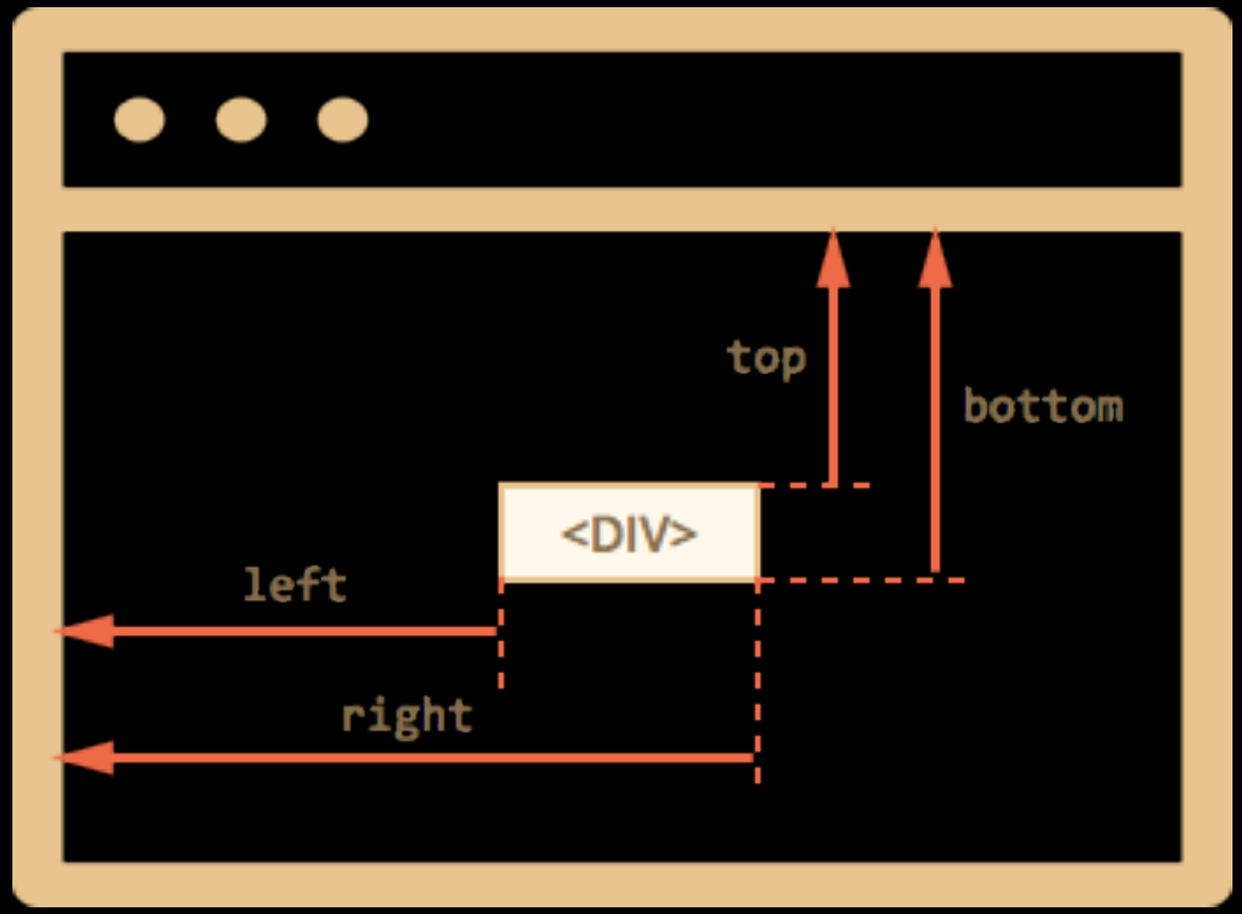
Установка прокрутки

- Установить прокрутку можно при помощи специальных методов:
  - window.scrollTo(pageX,pageY) – абсолютные координаты
  - window.scrollBy(x,y) – прокрутить относительно текущего места
  - elem.scrollIntoView(top) – прокрутить, чтобы элемент elem стал виден

getBoundingClientRect()

- elem.getBoundingClientRect() – возвращает координаты элемента
- Координаты возвращаются в виде объекта со свойствами:
  - top – Y-координата верхней границы элемента
  - left – X-координата левой границы
  - right – X-координата правой границы
  - bottom – Y-координата нижней границы

Прмиер



Нюансы

- Координаты могут быть дробными
- Координаты могут быть отрицательными, если верх элемента выходит за верхнюю границу окна
- Некоторые современные браузеры добавляют к результату свойства для ширины и высоты: width/height

elementFromPoint(x, y)

- elementFromPoint(x, y) – возвращает элемент, который находится на координатах (x, y) относительно окна
- <script>
  - var el = document.elementFromPoint(10,100);
  - el.style.background = "red";
  - alert(el.tagName);
  - el.style.background = "";
- </script>

Координаты в документе