## **1. \***

|  |  |
| --- | --- |
| 1  2  3  4 | \* {  **margin**: 0;  **padding**: 0;  } |

Универсальный селектор соответствует любому элементу на странице. Часто разработчики используют его для обнуления свойств margin и padding. Это допустимо в тестовом режиме, но я не рекомендую использовать этот селектор в рабочих проектах. Он слишком нагружает браузер.

Универсальный селектор \* также можно использовать для стилизации всех потомков элемента.

|  |  |
| --- | --- |
| 1  2  3 | #container \* {  **border**: 1px solid black;  } |

В данном примере стиль применится ко всем потомкам элемента #container div. Но, повторюсь, не стоит использовать эту технику слишком часто, а лучше вообще её не использовать.

### **Совместимость**

* IE6+
* Firefox
* Chrome
* Safari
* Opera

Advertisement

## **2. #X**

|  |  |
| --- | --- |
| 1  2  3  4 | #container {  **width**: 960px;  **margin**: auto;  } |

Использование символа решётки позволяет обратиться к уникальному идентификатору элемента id. Но старайтесь не злоупотреблять использованием идентификаторов.

*Спросите себя: мне действительно нужно использовать id для этого элемента?*

Идентификатор id должен быть уникальным и может использоваться на странице только один раз. При возможности старайтесь обходиться без идентификаторов. Старайтесь использовать тэги или псевдоклассы.

### **Совместимость**

* IE6+
* Firefox
* Chrome
* Safari
* Opera

## **3. .X**

|  |  |
| --- | --- |
| 1  2  3 | .error {  **color**: red;  } |

class - это селектор класса. Отличие селектора class от id в том, что класс не должен быть уникальным и может использоваться для выбора нескольких элементов на странице. Использование id позволяет выбрать конкретный уникальный элемент.

### **Совместимость**

* IE6+
* Firefox
* Chrome
* Safari
* Opera

## **4. X Y**

|  |  |
| --- | --- |
| 1  2  3 | li a {  **text-decoration**: none;  } |

Ещё один из часто используемых типов селекторов - это контекстный селектор. Он используется для выбора элементов, удовлетворяющих определённому контексту. Например, вместо того, чтобы задавать стиль всем ссылкам, вам нужно изменить отображение только тех ссылок, которые расположены внутри несортированного списка.

***Совет****- Если ваш селектор выглядит так: X Y Z A B.error, то вы совершаете ошибку. Старайтесь избегать таких длинных вложений.*

### **Совместимость**

* IE6+
* Firefox
* Chrome
* Safari
* Opera

## **5. X**

|  |  |
| --- | --- |
| 1  2 | a { **color**: red; }  ul { **margin-left**: 0; } |

Что вы будете делать, если вам понадобится выбрать только элементы определённого типа? В этом случае вы не сможете использовать id или class. И вам на помощь придёт селектор по типу элемента. Если вам нужны все несортированные списки, просто используйте ul {}.

### **Совместимость**

* IE6+
* Firefox
* Chrome
* Safari
* Opera

## **6. X:visited и X:link**

|  |  |
| --- | --- |
| 1  2 | a:link { **color**: red; }  a:visted { **color**: purple; } |

Мы используем псевдокласс :link для стилизации ссылок, на которые пользователь ещё не нажимал.

Также нам доступен псевдокласс :visited для стилизации только тех ссылок, на которые пользователь нажал.

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **7. X + Y**

|  |  |
| --- | --- |
| 1  2  3 | ul + p {  **color**: red;  } |

Это соседний селектор. Он поможет нам выбрать только тот элемент, который следует сразу же за указанным элементом. В этом примере мы выберем только первый параграф текста, следующий сразу за тэгом ul.

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **8. X > Y**

|  |  |
| --- | --- |
| 1  2  3 | div#container > ul {  **border**: 1px solid black;  } |

Разница между селекторами X Y и X > Y в том, что в последнем примере мы выберем только прямого потомка. Например, у нас есть такой код.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | <**div** id="container">     <**ul**>        <**li**> List Item          <**ul**>             <**li**> Child </**li**>          </**ul**>        </**li**>        <**li**> List Item </**li**>        <**li**> List Item </**li**>        <**li**> List Item </**li**>     </**ul**>  </**div**> |

Селектор #container > ul выберет только тот элемент ul, который является прямым потомком элемента div с id равным container. Он не выберет, например, элемент ul , являющийся потомком первого элемента li.

Данный способ выбора элементов является очень эффективным. Рекомендую вам использовать его, особенно при работе с JavaScript.

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **9. X ~ Y**

|  |  |
| --- | --- |
| 1  2  3 | ul ~ p {  **color**: red;  } |

Этот селектор похож на X + Y, но он менее строгий. Соседний селектор (ul + p) выберет только первый элемент, следующий сразу же за указанным элементом. Селектор X ~ Y выберет все элементы p, расположенные после элемента ul.

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **10. X[title]**

|  |  |
| --- | --- |
| 1  2  3 | a[title] {  **color**: green;  } |

Это селектор атрибутов. Он выберет только те ссылки, у которых имеется указанный атрибут title. Но что, если вам нужно что-то более конкретное? Что же...

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **11. X[href="foo"]**

|  |  |
| --- | --- |
| 1  2  3 | a[href="[http://net.tutsplus.com](http://net.tutsplus.com/)"] {  **color**: #1f6053; /\* nettuts green \*/  } |

Указанный выше код выберет все ссылки, указывающие на http://net.tutsplus.com. Они окрасятся в зелёный цвет. Стиль остальных ссылок останется без изменений.

*Не забудьте, что значение атрибута нужно заключать в кавычки. Особенно внимательны будьте при использовании JavaScript.*

Этот метод отлично работает, но он слишком строгий. Что, если ссылка указывает на Nettuts+, но путь прописан как nettuts.com? В этом случае мы можем использовать синтаксис регулярных выражений.

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **12. X[href\*="nettuts"]**

|  |  |
| --- | --- |
| 1  2  3 | a[href\*="tuts"] {  **color**: #1f6053; /\* nettuts green \*/  } |

Вот то, что нам нужно. Звёздочка означает, что указанное значение должно быть частью указанного атрибута. Поэтому этот селектор выберет и nettuts.com, и net.tutsplus.com,, и даже tutsplus.com.

Но помните, что этот способ слишком обширен. Что, если у вас будет ссылка, не указывающая на tuts+, но имеющая в адресе слово tuts? Для составления более строго правила используйте ^ и $, указывающие на начало и конец строки соответственно.

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **13. X[href^="http"]**

|  |  |
| --- | --- |
| 1  2  3  4 | a[href^="http"] {  **background**: url(path/to/external/icon.png) no-repeat;  **padding-left**: 10px;  } |

Вам было интересно как на некоторых сайтах реализован показ иконки возле внешних ссылок? Уверен, вы встречали подобное. Иконка напоминает, что это внешняя ссылка, которая переведёт вас на другой сайт.

Это как раз случай использования символа ^. В регулярных выражениях он используется для указания начала строки. Если мы хотим выбрать все ссылки, начинающиеся с http, то мы можем использовать код из примера сверху.

*Учтите, что мы не ищем http://. В этом нет необходимости.*

А что, если мы хотим стилизовать все ссылки, указывающие, например, на фото? В этом случае нас интересует конец строки.

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **14. X[href$=".jpg"]**

|  |  |
| --- | --- |
| 1  2  3 | a[href$=".jpg"] {  **color**: red;  } |

Мы используем символ из регулярных выражений $, который указывает на конец строки. В этом случае мы ищем все ссылки на фото, т.е. путь в них должен заканчиваться на .jpg. Учтите, что это не сработает для файлов gif и png.

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **15. X[data-\*="foo"]**

|  |  |
| --- | --- |
| 1  2  3 | a[data-filetype="image"] {  **color**: red;  } |

Как же нам выбрать все типы изображений, включая png, jpeg,jpg, gif? Мы можем создать такой селектор:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | a[href$=".jpg"],  a[href$=".jpeg"],  a[href$=".png"],  a[href$=".gif"] {  **color**: red;  } |

Но это очень громоздкая и неэффективная конструкция. Другой вариант - использовать пользовательские атрибуты. Мы можем добавить атрибут data-filetype к каждой ссылке, указывающей на изображение.

|  |  |
| --- | --- |
| 1 | <**a** href="path/to/image.jpg" data-filetype="image"> Image Link </**a**> |

Теперь мы можем использовать селектор атрибутов для выбора только этих ссылок.

|  |  |
| --- | --- |
| 1  2  3 | a[data-filetype="image"] {  **color**: red;  } |

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **16. X[foo~="bar"]**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | a[data-info~="external"] {  **color**: red;  }    a[data-info~="image"] {  **border**: 1px solid black;  } |

Вот пример, который поразит ваших коллег. Этот трюк известен очень немногим. Символ тильда (~) позволяет выбрать атрибут, в котором есть значения, разделённые пробелом.

Теперь мы можем создать атрибут data-info и использовать в нём любые значения, разделяя их пробелами. Для примера укажем, что ссылка у нас внешняя и ведёт на изображение.

|  |  |
| --- | --- |
| 1 | "<**a** href="path/to/image.jpg" data-info="external image"> Нажми на меня, дурашка </**a**> |

Этот код позволит нам выбрать любой элемент, имеющий в качестве значения атрибута любое нужное нам слово.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | /\* Выбираем атрибут data-info со значением "external" \*/  a[data-info~="external"] {  **color**: red;  }    /\* Выбираем атрибут со значчением "image" \*/  a[data-info~="image"] {  **border**: 1px solid black;  } |

Прикольно, правда?

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **17. X:checked**

|  |  |
| --- | --- |
| 1  2  3 | input[type=radio]:checked {  **border**: 1px solid black;  } |

Данный псевдокласс выберет только отмеченные элементы пользовательского интерфейса: радиокнопки или чекбоксы. Вот так просто.

### **Совместимость**

* IE9+
* Firefox
* Chrome
* Safari
* Opera

## **18. X:after**

Псевдоклассы before и after сейчас очень популярны. Каждый день разработчики находят новые оригинальные способы их использования. Эти псевдоклассы позволяют нам сгенерировать контент вокруг указанного элемента.

Многие впервые знакомятся с этими псевдоклассами при изучении хака clear-fix.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13 | .clearfix:after {  **content**: "";  **display**: block;  **clear**: both;  **visibility**: hidden;  **font-size**: 0;  **height**: 0;    }    .clearfix {   \***display**: inline-block;  **\_height**: 1%;  } |

Этот хак использует псевдокласс :after для добавления пробела после элемента. Обязательно занесите этот трюк в свою копилку знаний. Он особенно полезен там, где нельзя использовать overflow: hidden;.

Ещё один интересный пример использования этих псевдоклассов: [советы по созданию теней](https://net.tutsplus.com/tutorials/html-css-techniques/quick-tip-getting-clever-with-css3-shadows/).

*Согласно спецификации CSS3, псевдоклассы должны отделяться двойным двоеточим ::. Но современные браузеры прекрасно поймут ваш код и с одним двоеточим. На самом деле я рекомендую вам использовать в своих проектах одно двоеточие.*

### **Совместимость**

* IE8+
* Firefox
* Chrome
* Safari
* Opera

## **19. X:hover**

|  |  |
| --- | --- |
| 1  2  3 | div:hover {  **background**: #e3e3e3;  } |

Ой, ну ладно. Вы, конечно же, знаете этот селектор. Официальное его название псевдокласс пользовательского действия. Звучит немного странно. На практике это означает, что вы сможете изменить стиль элемента в момент наведения на него кусора мыши.

*Помните, что старые версии Internet Explorer не распознают :hover, если он используется не на ссылках.*

Чаще всего вы будете использовать этот селектор для стилизации ссылок (border-bottom) при наведении кусора.

|  |  |
| --- | --- |
| 1  2  3 | a:hover {  **border-bottom**: 1px solid black;  } |

***Совет****- border-bottom: 1px solid black; выглядит гораздо лучше чем text-decoration: underline;.*

### **Совместимость**

* IE6+ (в IE6, :hover может применяться только к ссылкам)
* Firefox
* Chrome
* Safari
* Opera

## **20. X:not(selector)**

|  |  |
| --- | --- |
| 1  2  3 | div:not(#container) {  **color**: blue;  } |

Псевдокласс отрицания бывает очень полезен. Представьте, что вам нужно выбрать все тэги div, кроме одного с id равным container. Код выше сделает именно это.

Или, например, если я хочу выбрать каждый элемент на странице (не делайте так), кроме тэгов p.

|  |  |
| --- | --- |
| 1  2  3 | \*:not(p) {  **color**: green;  } |

### **Совместимость**

* IE9+
* Firefox
* Chrome
* Safari
* Opera

## **21. X::pseudoElement**

|  |  |
| --- | --- |
| 1  2  3  4 | p::first-line {  **font-weight**: bold;  **font-size**: 1.2em;  } |

Мы можем использовать псевдоэлементы (отделённые символом ::) для стилизации части элемента: первой строки параграфа или первой буквы. Помните, что псевдоэлементы будут работать только с блоковыми элементами.

*Псевдоэлемент отделяется символом ::*

#### **Выберем первую букву параграфа**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | p::first-letter {  **float**: left;  **font-size**: 2em;  **font-weight**: bold;  **font-family**: cursive;  **padding-right**: 2px;  } |

Этот код найдёт все параграфы на странице и выберет из них только первые буквы.

Такой метод обычно используется для придания странице газетного стиля, когда первая буква в статье заметно больше остальных.

#### **Выберем первую строку параграфа**

|  |  |
| --- | --- |
| 1  2  3  4 | p::first-line {  **font-weight**: bold;  **font-size**: 1.2em;  } |

Псеводэлемент ::first-line, как вы уже догадались, выберет только первую строку указанного элемента.

*"Для совместимости с уже существующими стилями, браузеры поддерживают старый синтаксис написания псевдоэлементов, задекларированный в спецификациях CSS 1 и 2 ( :first-line, :first-letter, :before и :after). Но для новых псевдоэлементов такой синтаксис не поддерживается." - [Источник](https://www.w3.org/TR/css3-selectors/)*

### **Совместимость**

* IE6+
* Firefox
* Chrome
* Safari
* Opera

## **22. X:nth-child(n)**

|  |  |
| --- | --- |
| 1  2  3 | li:nth-child(3) {  **color**: red;  } |

Помните деньки, когда у нас не было возможности выбрать конкретный элемент в стеке? Псевдокласс nth-child решает эту проблему!

Помните, что nth-child принимает в качестве параметра число, но отсчёт ведётся не от нуля. Если вам нужен второй по счёту элемент, используйте li:nth-child(2).

мы можем использовать этот псевдокласс для выбора нескольких элементов. Например, li:nth-child(4n) позволит нам выбрать каждый четвертый элемент списка.

### **Совместимость**

* IE9+
* Firefox 3.5+
* Chrome
* Safari

## **23. X:nth-last-child(n)**

|  |  |
| --- | --- |
| 1  2  3 | li:nth-last-child(2) {  **color**: red;  } |

Представьте, что у вас есть огромный список ul, а вам нужно выбрать только третий элемент с конца? Вместо того, чтобы делать li:nth-child(397), вы можете использовать псевдокласс nth-last-child.

Этот метод работает аналогично nth-child, Отличие в том, что отсчёт элементов ведётся с конца.

### **Совместимость**

* IE9+
* Firefox 3.5+
* Chrome
* Safari
* Opera

## **24. X:nth-of-type(n)**

|  |  |
| --- | --- |
| 1  2  3 | ul:nth-of-type(3) {  **border**: 1px solid black;  } |

Бывает, что вместо наследников child вам нужно выбрать элементы по их типу.

Представьте, что у вас на странице есть пять списков. И вам нужно стилизовать только третий список. И у него нет уникального id. Тут вам и пригодится псевдокласс nth-of-type(n). В примере выше только у третьего списка будет рамка.

### **Совместимость**

* IE9+
* Firefox 3.5+
* Chrome
* Safari

## **25. X:nth-last-of-type(n)**

|  |  |
| --- | --- |
| 1  2  3 | ul:nth-last-of-type(3) {  **border**: 1px solid black;  } |

Да, как в предыдущих примерах, мы можем использовать nth-last-of-type для отсчёта элементов с конца.

### **Совместимость**

* IE9+
* Firefox 3.5+
* Chrome
* Safari
* Opera

## **26. X:first-child**

|  |  |
| --- | --- |
| 1  2  3 | ul li:first-child {  **border-top**: none;  } |

Этот псевдокласс позволит нам выбрать только первый дочерний элемент. Чаще всего он используется для устранения рамок вокруг первого и последнего элемента списка.

Например, у вас есть какое-то количество строк, каждая из которых имеет border-top и border-bottom. В этом случае первая и последняя строка будут выглядеть не очень красиво.

Многие дизайнеры используют классы first и last для устранения дублирующихся рамок. Но лучше использовать эти псевдоклассы.

### **Совместимость**

* IE7+
* Firefox
* Chrome
* Safari
* Opera

## **27. X:last-child**

|  |  |
| --- | --- |
| 1  2  3 | ul > li:last-child {  **color**: green;  } |

В отличие от first-child, last-child выберет последний дочерний элемент.

### **Пример**

Давайте рассмотрим пример использования этих псевдоклассов. Создадим список.

#### **Markup**

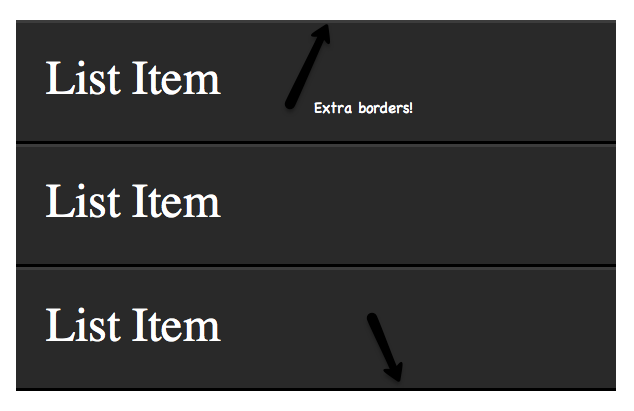
|  |  |
| --- | --- |
| 1  2  3  4  5 | <**ul**>     <**li**> List Item </**li**>     <**li**> List Item </**li**>     <**li**> List Item </**li**>  </**ul**> |

Ничего особенного, просто список.

#### **CSS**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13 | ul {  **width**: 200px;  **background**: #292929;  **color**: white;  **list-style**: none;  **padding-left**: 0;  }    li {  **padding**: 10px;  **border-bottom**: 1px solid black;  **border-top**: 1px solid #3c3c3c;  } |

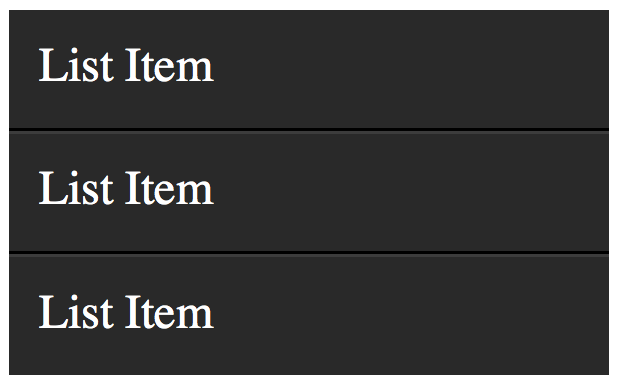
Мы задали фон, убрали дефолтные отступы, добавляемые браузером, и добавили верхний и нижний бордер для каждого элемента li.



*Для добавления глубины списку используйте border-bottom для каждого элемента li на пару оттенков темнее, чем цвет фона. Далее добавьте border-top на пару оттенков светлее.*

Есть одна проблема: первый и последний элементы списка выглядят не очень. Давайте исправим это с помощью :first-child и :last-child.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | li:first-child {  **border-top**: none;  }    li:last-child {  **border-bottom**: none;  } |



Готово!

### **Совместимость**

* IE9+
* Firefox
* Chrome
* Safari
* Opera

Да, IE8 поддерживает :first-child, но не поддерживает :last-child.

## **28. X:only-child**

|  |  |
| --- | --- |
| 1  2  3 | div p:only-child {  **color**: red;  } |

Честно говоря, вряд ли вы будете использовать псеводкласс only-child очень часто. Но он существует, и знание о нём вам точно не повредит.

Он позволяет выбрать элемент, являющийся единственным наследником своего родителя. В примере выше абзац, являющийся единственным наследником элемента div, получит красный цвет шрифта.

Давайте рассмотрим такой пример.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <**div**><**p**> My paragraph here. </**p**></**div**>    <**div**>   <**p**> Two paragraphs total. </**p**>   <**p**> Two paragraphs total. </**p**>  </**div**> |

В этом случае текст во втором элементе div останется без изменений. Только текст в первом div станет красным. Если у элемента более одного наследника, псевдокласс only-child не действует.

### **Совместимость**

* IE9+
* Firefox
* Chrome
* Safari
* Opera

## **29. X:only-of-type**

|  |  |
| --- | --- |
| 1  2  3 | li:only-of-type {  **font-weight**: bold;  } |

Этот псевдокласс можно использовать довольно оригинальными способами. Он выберет дочерний элемент, только если он единственный у родителя. Например, давайте выберем все списки ul, имеющие только один элемент списка.

Как мы можем решить эту задачу? Можно использовать ul li, но тогда мы выберем все элементы списка. Единственным решением будет использовать only-of-type.

|  |  |
| --- | --- |
| 1  2  3 | ul > li:only-of-type {  **font-weight**: bold;  } |

### **Совместимость**

* IE9+
* Firefox 3.5+
* Chrome
* Safari
* Opera

## **30. X:first-of-type**

Псевдокласс first-of-type выбирает первого наследника указанного типа.

#### **Тест**

Для лучшего усвоения давайте пройдём тест. Скопируйте этот код в свой текстовый редактор:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | <**div**>   <**p**> My paragraph here. </**p**>   <**ul**>      <**li**> List Item 1 </**li**>      <**li**> List Item 2 </**li**>   </**ul**>     <**ul**>      <**li**> List Item 3 </**li**>      <**li**> List Item 4 </**li**>   </**ul**>  </**div**> |

Теперь, не читая дальше, попробуйте выбрать только элемент "List Item 2". Читайте дальше после того, как решите задачу (или сдадитесь).

#### **Решение 1**

Есть несколько способов решения этой задачи. Начнём с использования first-of-type.

|  |  |
| --- | --- |
| 1  2  3 | ul:first-of-type > li:nth-child(2) {  **font-weight**: bold;  } |

Этот код говорит «найди первый несортированный список, затем найди его прямого наследника, который является элементом списка. Далее выбери второй элемент».

#### **Решение 2**

Другой способ - использование соседних селекторов.

|  |  |
| --- | --- |
| 1  2  3 | p + ul li:last-child {  **font-weight**: bold;  } |

Здесь мы находим список ul, следующий сразу же за тэгом p, а потом находим его последний дочерний элемент.

#### **Решение 3**

С этими селекторами мы можем делать практически что угодно.

|  |  |
| --- | --- |
| 1  2  3 | ul:first-of-type li:nth-last-child(1) {  **font-weight**: bold;  } |

Здесь мы выбираем первый список ul, затем находим первый элемент списка с конца! :)

### **Совместимость**

* IE9+
* Firefox 3.5+
* Chrome
* Safari
* Opera