CSS

SELECTORS

What are selectors?

Selectors are very important. They allow authors to **target specific HTML elements** so that they can be styled.

In CSS 2.1, there are **22 different types** of selectors. We will talk about 11 of them here.

Let’s see the different types of **selectors in action**…

1. Type selectors

The **type selector** is written using the *element type* only.

Element type

```
a { color: blue; }
img { color: blue; }
h1 { color: blue; }
p { color: blue; }
form { color: blue; }
```

A type selector selects **every instance** of the element type…

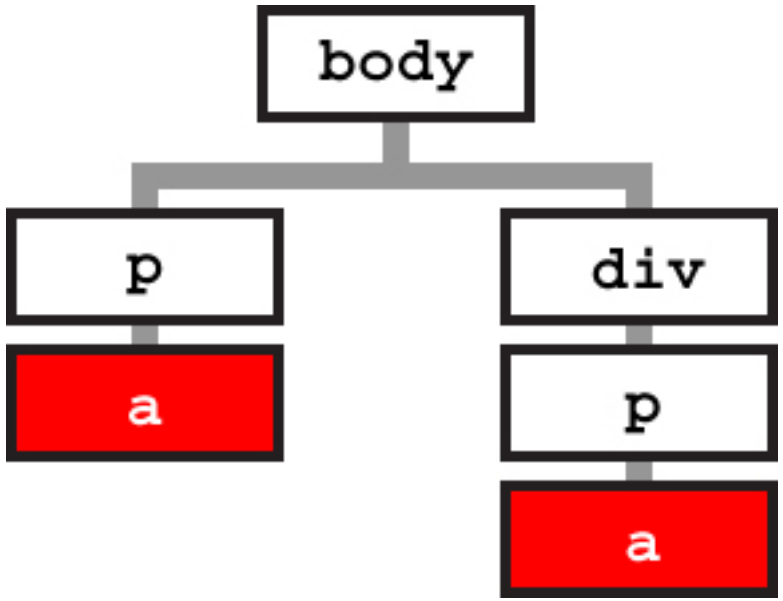
```
a { color: blue; }
```

CSS

```
<p>
    <a href="#">This is a li
```

HTML

…**regardless of their position** in the document tree.



Support for **type selectors**:

IE6	Partial*
IE7	Full
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

* Lack of support for abbr element

2. Class selectors

The **class selector** is written using a “.” followed by the class value.

Class value

```
.intro { color: blue; }
.news { color: blue; }
.help { color: blue; }
.search { color: blue; }
.topical { color: blue; }
```

Class values are **case-sensitive**. Browsers will interpret “a” and “A” differently.

```
.Intro { color: blue; }
.intro { color: blue; }

.NewsStory { color: blue; }
.newsstory { color: blue; }
```

The class selector selects any HTML element that has the **relevant class value**...

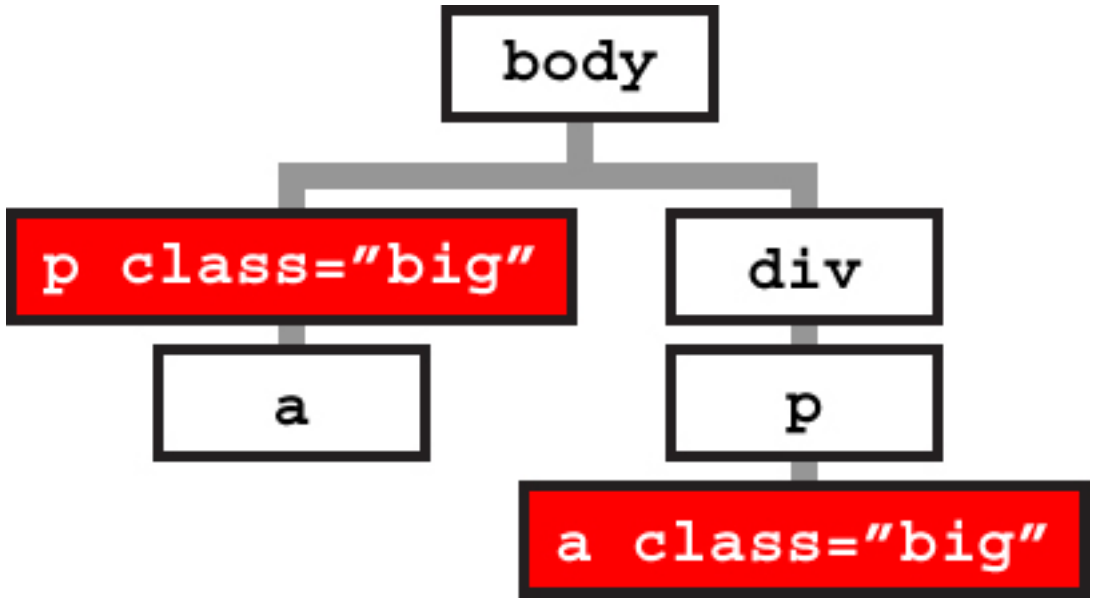
CSS

```
.big { color: blue; }
```

HTML

```
<p class="big">
    This is a paragraph of t
```

…**regardless of their position** in the document tree.



Support for **class selectors**:

IE6	Partial*
IE7	Full
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

* Lack of support for class names starting with hyphens or underscore

2a. Combination class selectors

You can also combine type and class selectors to **isolate** specific instances of an element.

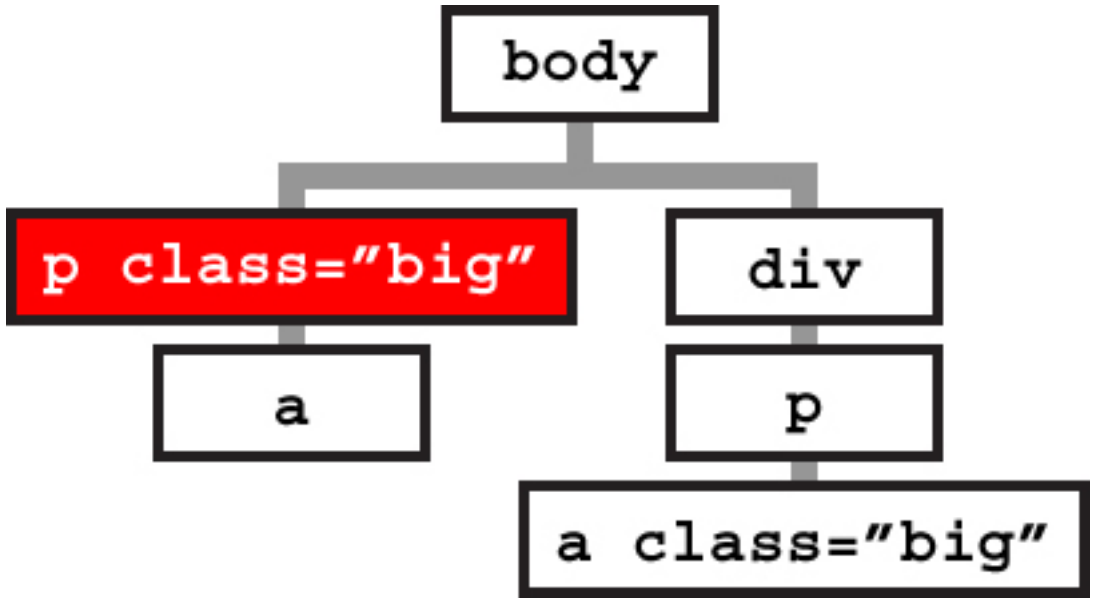
CSS

```
p.big { color: blue; }
```

HTML

```
<p class="big">
    This is a paragraph of t
```

This means that **only paragraphs** with a class of “big” will be selected.



3. ID selectors

The **ID selector** is written using a "#" followed by the ID value.

ID value

```
#intro { color: blue; }
#news { color: blue; }
#help { color: blue; }
#search { color: blue; }
#topical { color: blue; }
```

ID values are **case-sensitive**. Browsers will interpret “a” and “A” differently.

```
#Intro { color: blue; }
#intro { color: blue; }

#NewsStory { color: blue; }
#newsstory { color: blue; }
```

The ID selector selects any HTML element that has the **relevant ID value**...

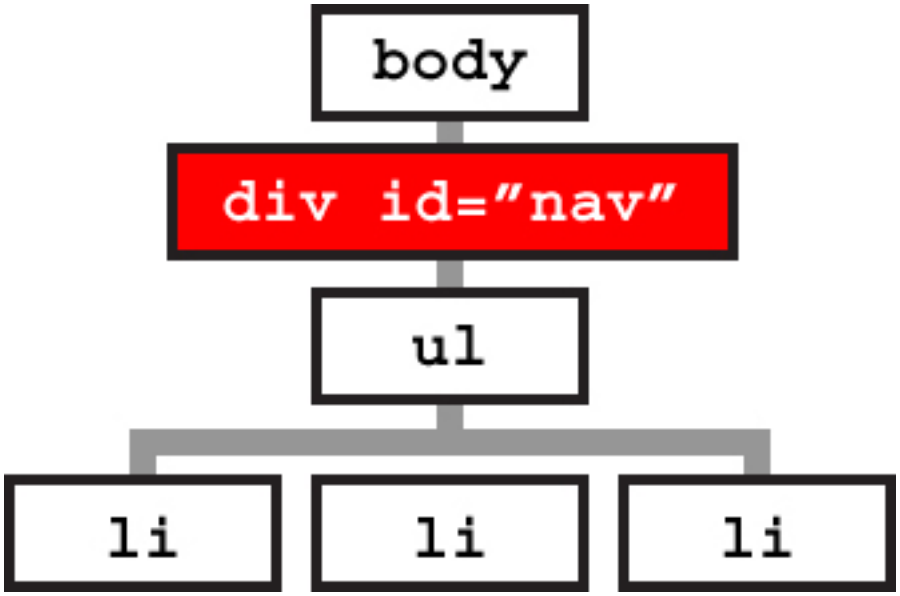
CSS

```
#nav { color: blue; }
```

HTML

```
<div id="nav">
    <ul>
```

…**regardless of its position** in the document tree.



Support for **ID selectors**:

IE6	Partial*
IE7	Full
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

* ID selector ignored unless the last ID selector in the simple selector.

3a. Combination ID selectors

You can also combine type and ID selectors to **isolate** specific instances of an element.

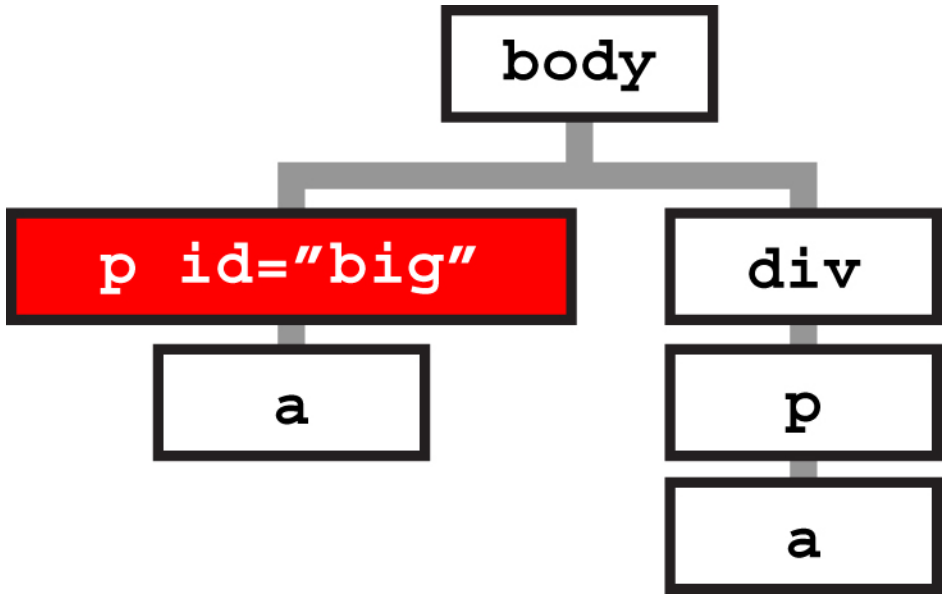
CSS

```
p#big { color: blue; }
```

HTML

```
<p id="big">
    This is a paragraph of t
```

This means that **only paragraphs** with a ID of “big” will be selected.



4. Descendant selectors

The **descendant selector** is written using two or more selectors separated by white space.

```
p a { color: blue; }
table th { color: blue; }
form label { color: blue; }
div#nav ul li a
{ color: blue; }
```

This rule will select only <a> elements that are **descendants of a <p>** element.

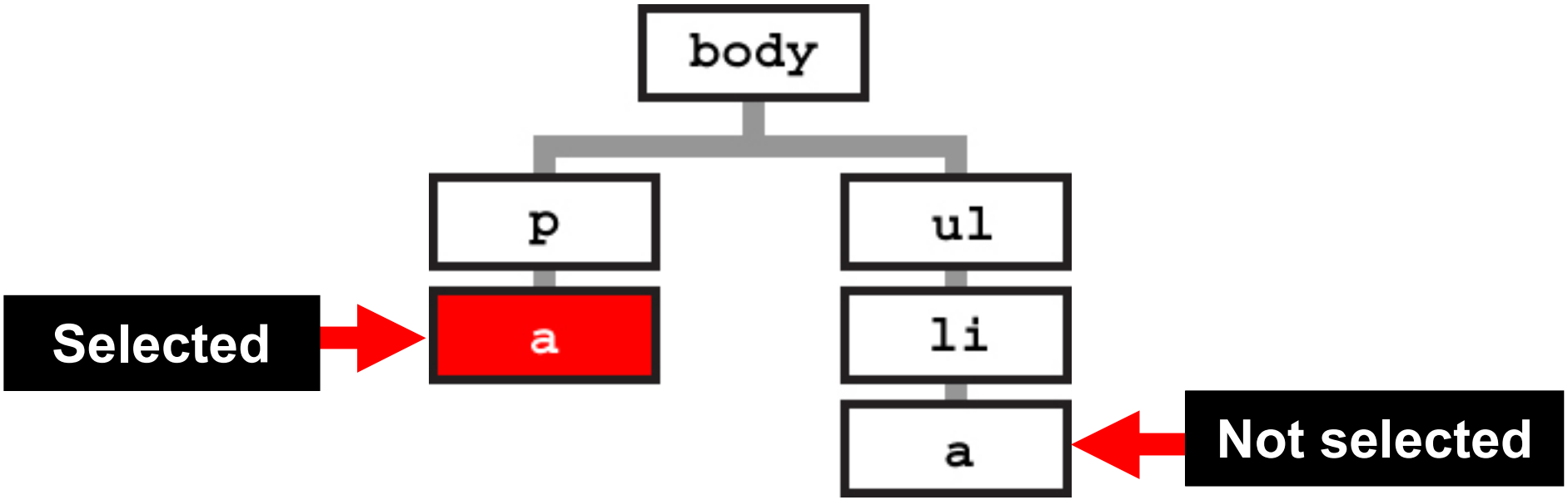
CSS

```
p a { color: blue; }
```

HTML

```
<p>
    <a href="#">A link is a
```

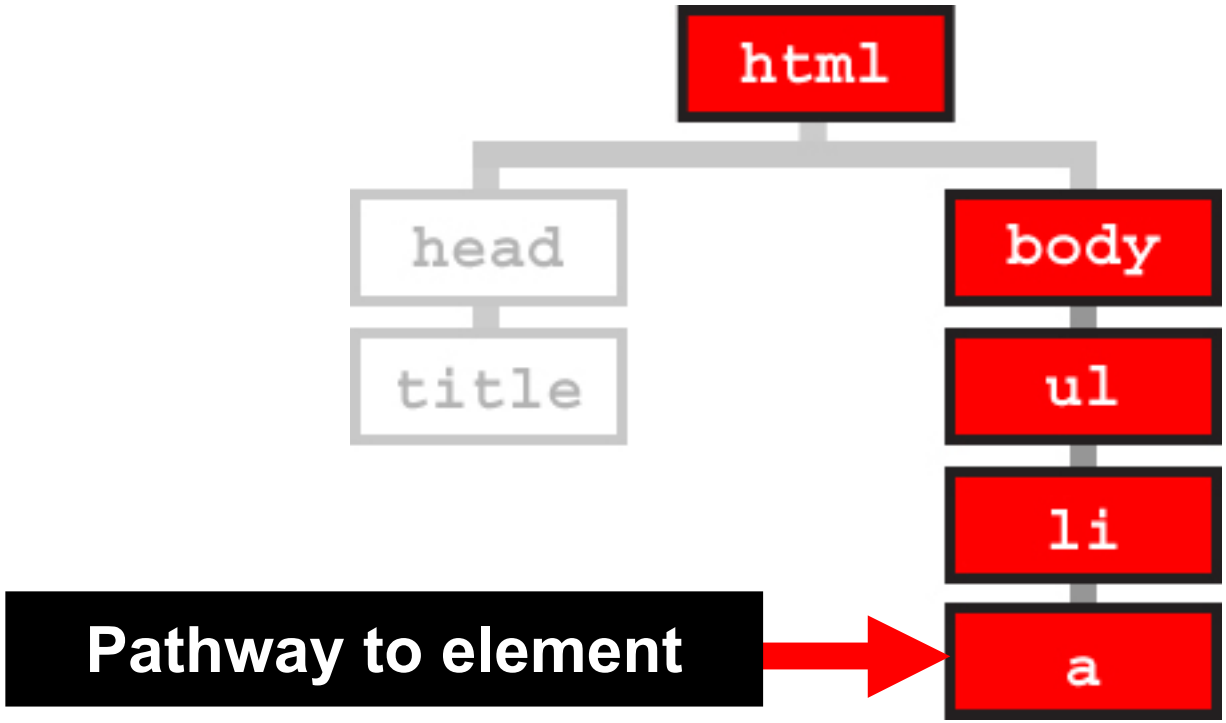
So, <a> elements **without <p> ancestors** will not be selected.



Descendant selectors are powerful. They allow us to **isolate and target** specific elements using the document tree.

The key to descendant selectors is understanding **pathways to elements**…

Every element has a **pathway**, starting with the HTML element.



With descendant selectors, paths can be **written in full or in part**, depending on your need.

```
html body ul li a { }
body ul li a { }
ul li a { }
li a { }
a { }
```

Paths can also skip a level. As long as the path is **correct**, the selector will work.

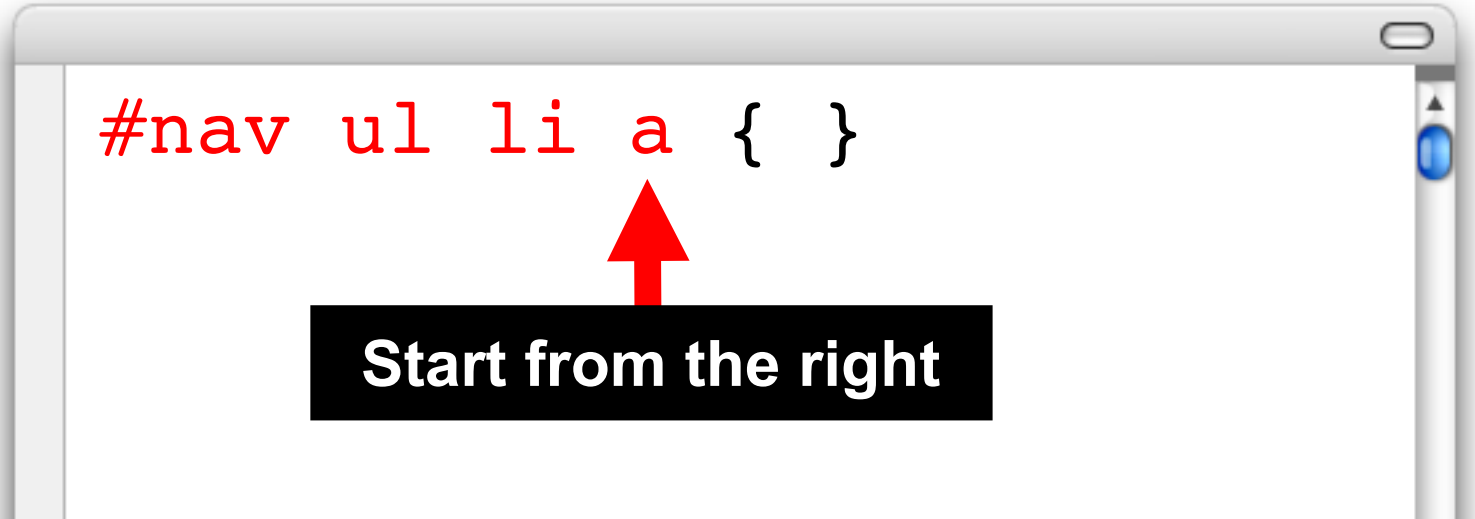
```
body a { }
ul a { }
```

In many cases, descendant selectors can be used to **replace unnecessary classes or ids**.

For example, using one ID and descendant selectors, we can **target every element below**.

```
#nav { }
#nav ul { }
#nav ul li { }
#nav ul li a { }
#nav form { }
#nav fieldset { }
#nav legend { }
```

Do you find descendant selectors confusing? Just **start from the right** (the element that is being selected) and work backwards to the left.



This selector will select any **<a> element** that is inside an <li> that is inside a <ul> that is inside an element with an ID value of “nav”.

```
#nav ul li a { }
```

The element being selected

Support for **descendant selectors**:

IE6	Partial*
IE7	Full
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

* ID selector ignored unless the last ID selector in the simple selector.

5. Child selectors

The **child selector** is written using two selectors separated by a “>”. White space between selectors is optional.

```
p > a { color: blue; }
div > p { color: blue; }

p>a { color: blue; }
div>a { color: blue; }
```

The **child selector** selects any element that is a direct child of another HTML element.

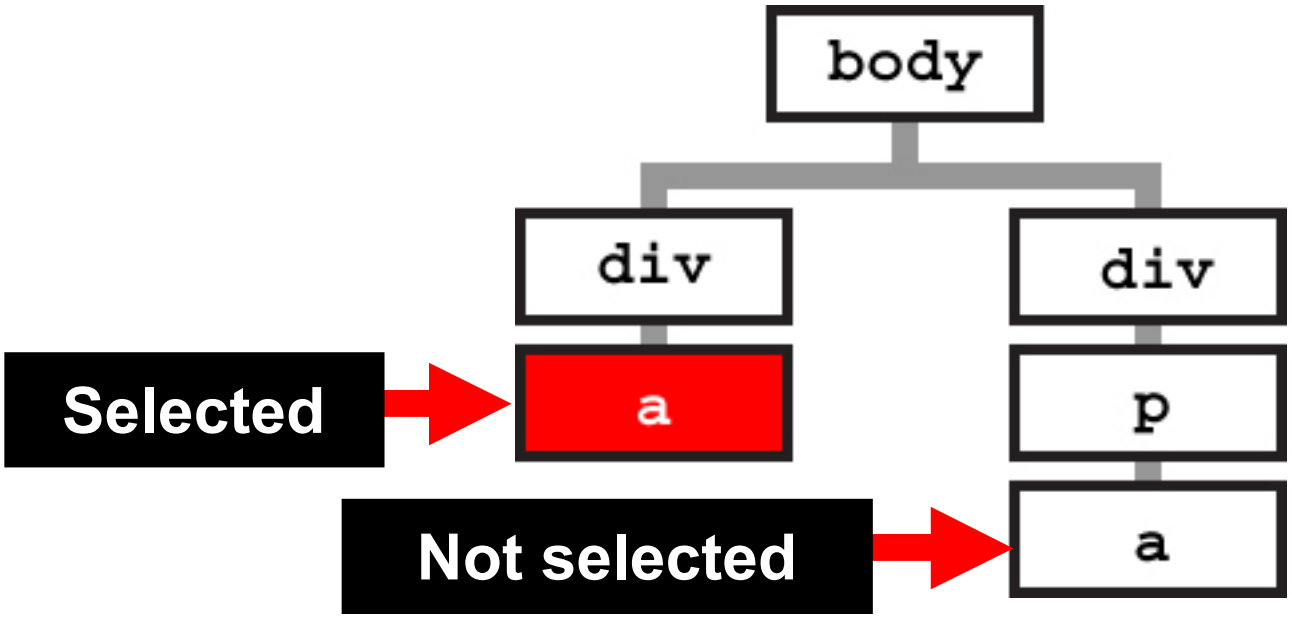
CSS

```
div > a { color: blue; }
```

HTML

```
<div>
    <a href="#">links</a>
```

Only child elements, rather than descendant elements, will be selected.



Support for **child selectors**:

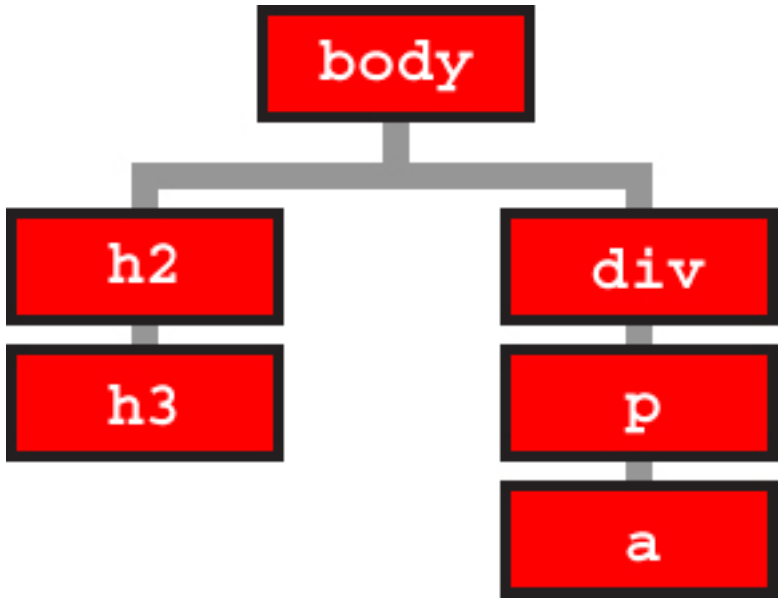
IE6	None
IE7	Almost full
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

6. Universal selectors

The **universal selector** is written using a "*".

```
* { color: blue; }
```

This will **select all elements** within the document.



Support for **universal selectors**:

IE6	None
IE7	Partial
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

7. :link pseudo-class

The **:link** pseudo-class will select any link that is defined as "unvisited"

```
a:link { color: blue; }
```

Support for **:link selectors**:

IE6	Partial
IE7	Full
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

* Does not support chaining of pseudo-classes

8. :visited pseudo-class

The **:visited** pseudo-class will select any link that is defined as "visited"

```
a:visited { color: blue; }
```

Support for **:visited selectors**:

IE6	Partial
IE7	Full
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

* Does not support chaining of pseudo-classes

9. :focus pseudo-class

The **:focus** pseudo-class selects elements that have keyboard input focus.

```
a:focus { color: blue; }
```

Support for **:focus selectors**:

IE6	None
IE7	None
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

* Does not support chaining of pseudo-classes

10. :hover pseudo-class

The **:hover** pseudo-class selects any element when the user's cursor is over the element.

```
a:hover { color: blue; }
```

Support for **:hover selectors**:

IE6	Partial
IE7	Partial
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

* Lack of support except for elements with href attribute

11. :active pseudo-class

The **:active** pseudo-class selects any element that is currently being activated

```
a:active { color: blue; }
```

Support for **:active selectors**:

IE6	Partial
IE7	Partial
IE8	Full
Firefox 3	Full
Firefox 3.5	Full
Safari 3	Full
Safari 4	Full
Opera 9.5	Full
Opera 10	Full
Chrome 2	Full

* Lack of support except for elements with href attribute, sticky active state

A note on pseudo-class order for links

Ideally, you should define
all five link states:

:link

:visited

:focus

:hover

:active

But what **order** should you place them in?

As all five selectors have the same weight, the **ones written last** will win.

For this reason, “:visited” should always be **written after** “:link”. Otherwise this state will not be seen.

```
a:link { color: blue; }
a:visited { color: blue; }
```

Then, ":focus" should come before ":hover", and ":active" should **always be last**.

```
a:focus { color: blue; }
a:hover { color: blue; }
a:active { color: blue; }
```

So, the five link states should be written in the **following order**:

```
a:link { color: blue; }
a:visited { color: blue; }
a:focus { color: blue; }
a:hover { color: blue; }
a:active { color: blue; }
```

Combining selectors

Authors can combine **different types of selectors** in order to target elements more accurately.

For example, combining **type and class** selectors:

```
h1.first { color: blue; }
a.more { color: blue; }
td.alt { color: blue; }
p.intro { color: blue; }
```

Or, combining **type and ID** selectors:

```
p#first { color: blue; }
a#intro { color: blue; }
td#alt { color: blue; }
p#intro { color: blue; }
```

You can also combining selectors into **descendant selectors**:

```
#new .nav { color: blue; }
.nav #intro { color: red; }
```

You can even combine **type, class & pseudo-class** selectors:

```
a.intro:visited
{ color: blue; }

td.alternate:hover
{ background: yellow; }
```

Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley