

LEARN WEB

ale66

ADVANCED CSS

BASIC ORGANISATION

```
selector {  
  property1: value1;  
  property2: value2;  
  ...  
}
```

```
#id-value {  
  property1: value1;  
  property2: value2;  
  ...  
}
```

TOO MUCH?

WEB PAGE

Metaphors for New Words

"Because in our brief lives we catch so little of the vastness of history, we tend too much to think of language as being solid as a dictionary, with a granite-like permanence, rather than as the rampant restless sea of metaphor which it is."

—Julian Jaynes

id= "section-quote"

We make metaphors for many things, but when we make many metaphors for one thing, it says that thing is important to us. We make metaphors for new words almost as readily as we make new words.

id= "section-summary"

CLASS VS. ID

Best Practices: Classes Versus IDs

When should you use an ID selector versus a class selector? Ask yourself the following questions:

- Will the styles I want to use be applied to one and only one element?

If so, use an ID selector on that element.

- Will the styles I want to use be applied to multiple elements?

If so, use a class selector on each of those elements.

- Will the styles I want to use be applied to only one element now but could be applied to other elements in the future?

If so, use a class selector on that one element now. You can always apply the class selector to other elements as needed down the road.

CSS WITH DOM INHERITANCE

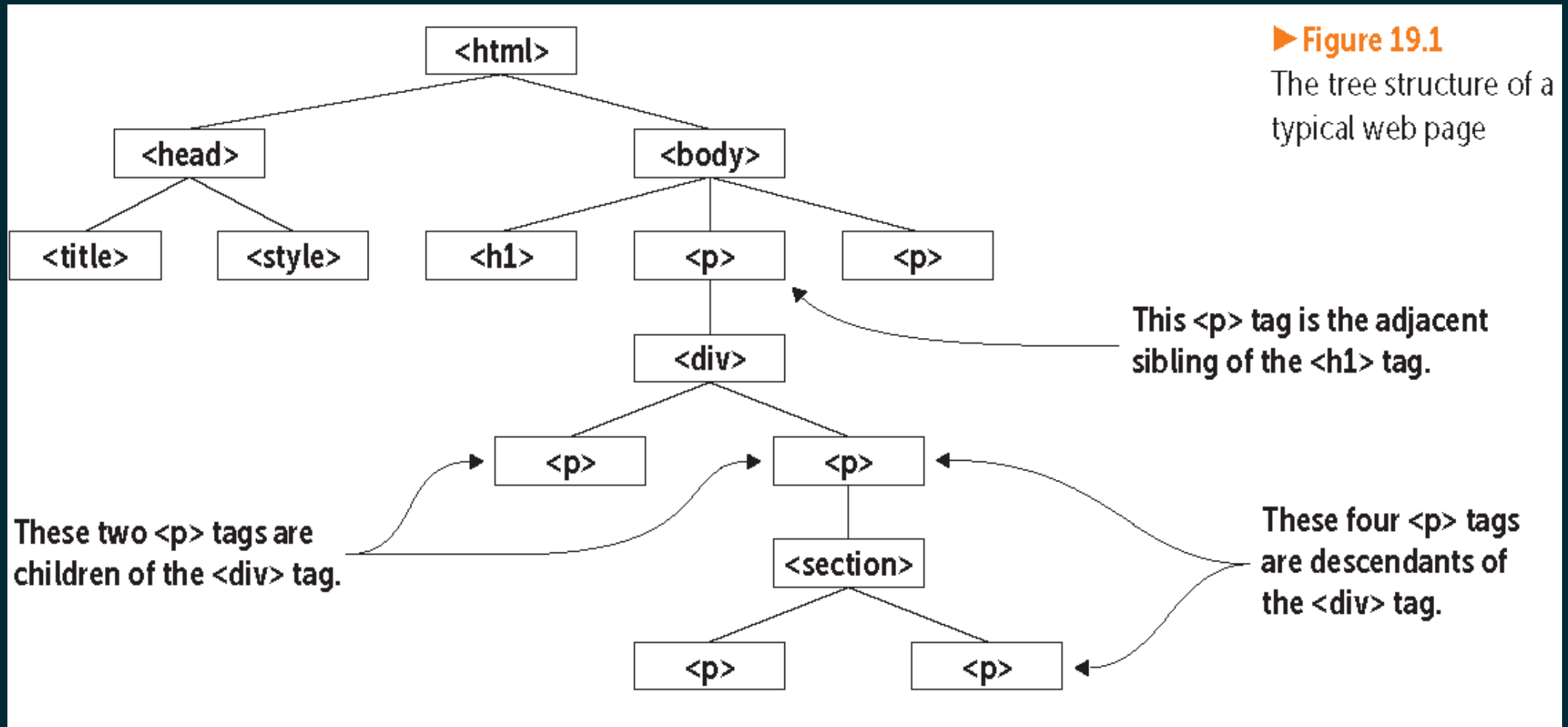
CSS works mostly by

- inheritance (from ancestor elements)
- exception / refinement
- cascading (from layers of styles)

REMEMBER DOM?

- The `html` element is the root of the structure.
- The `html` element has two main branches: `head` and `body`.
- The `head` element has two branches: `title` and `style`.
- The `body` element has three branches: an `h1` element and two `p` elements.
- The first of the `p` elements has a `div` branch.
- That `div` branch has two `p` branches.
- The second of those `p` branches has a `section` branch.
- The `section` branch has two `p` branches.

CLASS VS. ID: GUIDELINES



SUBTREE STYLES

The diagram illustrates the concept of subtree styles in CSS. It features a code block with the following structure:

```
ancestor descendant {  
    property1: value1;  
    property2: value2;  
    ...  
}
```

Handwritten red annotations explain the components:

- An arrow points from the text "The element's parent element" to the ancestor selector.
- An arrow points from the text "The element you want to style" to the descendant selector.
- An arrow points from the text "The styles you want to apply" to the curly braces of the style block.

SUBTREE STYLES, 2

WEB PAGE	<h2>Weird Word Origins</h2> <p>Welcome to the always wonderful, sometimes weird, and often downright wacky world of word histories</p> <p>Never thought you'd hear adjectives such as <i>wacky</i> and <i>weird</i> associated with the history of words? Think again, oh soon-to-be-even-wiser-than-you-are-now reader! The study of word origins isn't about memorizing technical terms or resurrecting dead languages or puzzling over parts of speech. Instead, it's all about telling stories.</p> <p>The history of a word is a narrative, plain and simple: where the word began, how it changed over time, and how it got where it is today. Delightfully, these narratives are often full of plot twists, turning points, heroes and villains, and surprise endings.</p>
CSS	<pre>body { color: blue; font-family: Verdana, sans-serif; font-size: 1.2em; } div p { color: #444; font-family: Georgia, serif; font-size: 0.75em; }</pre> <p>Styles applied to all text</p> <p>Styles applied only to p elements that are descendants of a div element</p>

An inverted/perverse logic applies...

THE CHILD-ONLY SELECTOR

The diagram illustrates the child-only selector syntax with handwritten red annotations. The selector is `parent > child {`. A red arrow points from the text "The element's parent element" to the `parent` part of the selector. Another red arrow points from the text "The element you want to style" to the `child` part of the selector. The styles are listed inside the curly braces: `property1: value1;`, `property2: value2;`, and `...`. A red arrow points from the text "The styles you want to apply" to the opening curly brace of the style block.

```
parent > child {  
    property1: value1;  
    property2: value2;  
    ...  
}
```

The element's parent element

The element you want to style

The styles you want to apply

THE SIBLINGS SELECTOR

The diagram illustrates the Siblings Selector syntax. It shows the selector `element ~ sibling {` where `element` and `sibling` are underlined. A red arrow points from the text "The reference element" to the underlined `element`. Another red arrow points from the text "The element you want to style" to the underlined `sibling`. Below the opening curly brace, the styles are listed: `property1: value1;`, `property2: value2;`, and `...`. A red bracket on the right side of these styles points to the text "The styles you want to apply". The closing curly brace `}` is on the same line as the ellipsis.

```
element ~ sibling {  
    property1: value1;  
    property2: value2;  
    ...  
}
```

The reference element

The element you want to style

The styles you want to apply

THE SIBLINGS SELECTOR

WEB PAGE	<p>A Smart Vocabulary—Contents</p> <p>Chapter 1: Names of Things You Didn't Know Had Names</p> <p>From the indentation on your upper lip to the indentation on the bottom of a wine bottle.</p> <p>Chapter 2: Making Word Whoopee</p> <p>Codswallop, nincompoop, willy-nilly, and other words that will bring a smile to your face.</p>
CSS	<pre>div { font-family: Georgia, serif; font-weight: normal; } h1 ~ div { font-family: Verdana, sans-serif; font-weight: bold; }</pre>

The <h1> tag

A Smart Vocabulary—Contents

Chapter 1: Names of Things You Didn't Know Had Names

From the indentation on your upper lip to the indentation on the bottom of a wine bottle.

Chapter 2: Making Word Whoopee

Codswallop, nincompoop, willy-nilly, and other words that will bring a smile to your face.

These <div> tags are siblings of the <h1>.

These <div> tags are not siblings of the <h1>.

Styles for all div text

Styles for div elements that are siblings of h1

EXERCISES

- Visit the wdpg.io playground
- do the first three exercises for Ch. 19.
- cover the second part of this presentation
- do 19.5 and 19.7

COMBINING, I

Challenge: can you tell what is going to happen when we use these complex conditions?

Example

```
<div class="sidebar alert">
```

```
p.footnote {styles}
```

```
p.footnote > a {styles}
```

```
p.footnote a.external {styles}
```

```
#payables-table li:nth-child(even)  
{styles}
```

COMBINING, II

Example	Description
<code><div class="sidebar alert"></code>	Applies both the class named <code>sidebar</code> and the class named <code>alert</code> to the <code>div</code> element
<code>p.footnote {styles}</code>	Applies a rule to those <code>p</code> elements that have been assigned the class named <code>footnote</code>
<code>p.footnote > a {styles}</code>	Applies a rule to <code>a</code> elements that are the children of those <code>p</code> elements that have been assigned the class named <code>footnote</code>
<code>p.footnote a.external {styles}</code>	Applies a rule to <code>a</code> elements that have been assigned the class named <code>external</code> and that are the descendants of those <code>p</code> elements that have been assigned the class named <code>footnote</code>
<code>#payables-table li:nth-child(even) {styles}</code>	Applies a rule to the even numbered <code>li</code> elements in the list that has been assigned the ID <code>payables-table</code>

AFTER/BEFORE BY EXAMPLE

the **before** and **after** condition rewrite the page to change the HTML around the element that is being styled:

WEB PAGE	<p>Here are some interesting characters to use in place of the standard bullets:</p> <ul style="list-style-type: none">☞ Circled bullet: ☹☹ Circled white bullet: ☹☞ Rightwards arrow with loop: ☞☞ Black star: ★☹ White star: ☆☹ Triangle bullet: ▶ <p>☞ Pointing finger character as a custom bullet</p>
CSS	<pre>ul { list-style-type: none; margin-left: 0; padding-left: 1em; text-indent: -1em; } li::before { content: '\261e\00a0'; color: red; font-size: 1.1em; }</pre> <p>Removes the default bullet</p> <p>Ensures that bullet text wraps correctly</p> <p>Adds a pointing finger and space</p> <p>Styles the custom bullet</p> <p><i>continued</i></p>

Good for styling, bad for readability!

SIMPLE TOP-DOWN INHERITANCE, I

WEB PAGE	<p>A child <code></code></p> <p>The parent <code><div></code></p> <p>Why don't <i>all</i> CSS properties inherit their parent's styles? * Because in some cases it would lead to weird or nonsensical results. For example, if you apply a border around, say, a <code>div</code> element, it would look odd indeed to apply the same border to a child <code>span</code> or <code>strong</code> element. Similarly, applying, say, a <code>p</code> element's <code>width</code> value to a child <code>em</code> element doesn't make sense.</p> <p>* See www.w3.org/TR/REC-CSS2/propidx.html</p> <p>A child <code><code></code></p> <p>A child <code><div></code></p>
CSS	<pre>.intro { color: saddlebrown; font-size: 1.1em; line-height: 1.4; }</pre> <p>Styles for the intro class</p>

SIMPLE TOP-DOWN INHERITANCE, II

WEB PAGE	<p>A child <code></code></p> <p>The parent <code><div></code></p> <p>Why don't <i>all</i> CSS properties inherit their parent's styles?* Because in some cases it would lead to weird or nonsensical results. For example, if you apply a border around, say, a <code>div</code> element, it would look odd indeed to apply the same border to a child <code>span</code> or <code>strong</code> element. Similarly, applying, say, a <code>p</code> element's <code>width</code> value to a child <code>em</code> element doesn't make sense.</p> <p>* See www.w3.org/TR/REC-CSS2/propidx.html</p> <p>A child <code><code></code></p> <p>A child <code><div></code></p>
CSS	<pre>.intro { color: saddlebrown; font-size: 1.1em; line-height: 1.4; }</pre> <p>Styles for the <code>intro</code> class</p>

continue..

The parent div element

A child em element

`<div class="intro">`

Why don't all CSS properties inherit their parent's styles?^{*} Because in some cases it would lead to weird or nonsensical results. For example, if you apply a border around, say, a `<div>` element, it would look odd indeed to apply the same border to a child `` or `` `<code>` element. Similarly, applying, say, a `<p>` element's `<code>width</code> value to a child element doesn't make sense.`

A child code element

`<div>`

`[*] See www.w3.org/TR/REC-CSS2/propidx.html`

`</div>`

`</div>`

A child div element


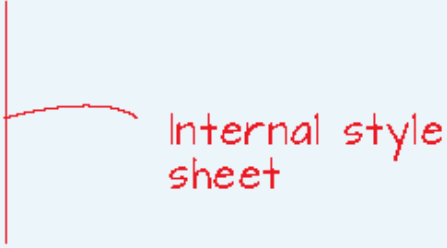

THE CASCADE

The cascade organizes these five sources of style data into the following hierarchy:

- User agent style sheet
- User style sheet
- External style sheets
- Internal style sheets
- Inline styles


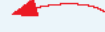
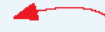


SPECIFICITY, I

The in-line setting is applied last...

WEB PAGE	<p>What is the color of this text?</p> 
HTML	<pre><style> div { color: red; } </style></pre>  <pre><div style="color: blue;"> What is the color of this text? </div></pre> 

SPECIFICITY, II

This is harder to determine

WEB PAGE	What is the color of this text?  The <p> tag
HTML	<pre><style> p.colored-text { color: purple; } .colored-text { color: blue; } div p { color: green; } p { color: red; } </style> <div> <p class="colored-text">What is the color of this text?</p> </div></pre> <p> Specificity = 11 points</p> <p> Specificity = 10 points</p> <p> Specificity = 2 points</p> <p> Specificity = 1 point</p>

SPECIFICITY, III


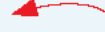
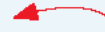


- 1 Count the number of elements (such as `p` or `div`) and pseudo-elements (such as `::before`), and assign 1 point to each.
- 2 Count the number of classes and pseudo-classes (such as `:hover`), and assign 10 points to each.
- 3 Count the number of IDs, and assign 100 points to each.
- 4 If the selector is part of an inline style sheet, assign 1,000 points.

The points assigned are indicative of the weight each selector carries. Returning to the example, count the points:

- `p.colored-text`—This selector contains one element and one class, for a total of 11 points.
- `.colored-text`—This selector contains one class, for a total of 10 points.
- `div p`—This selector contains two elements, for a total of 2 points.
- `p`—This selector contains one element, for a total of 1 point.

SPECIFICITY, IV

`p.colored-text` is the most specific (higher points)

WEB PAGE	What is the color of this text?  The <p> tag
HTML	<pre><style> p.colored-text { color: purple; } .colored-text { color: blue; } div p { color: green; } p { color: red; } </style> <div> <p class="colored-text">What is the color of this text?</p> </div></pre> <p> Specificity = 11 points</p> <p> Specificity = 10 points</p> <p> Specificity = 2 points</p> <p> Specificity = 1 point</p>

SUMMARY, I

- An *ID selector* applies CSS rules to any element that uses the specified ID value.
- To target all the elements contained within a parent element, use the *descendant selector*, which is the parent and descendant element names separated by a space.
- To target all the child elements contained within a parent element, use the *child selector*, which is the parent and child element names separated by a greater-than sign (>).
- To target all the elements that are siblings of some other element, use the *sibling selector*, which is the names of the two elements separated by a tilde (~).

SUMMARY, II

- Many CSS properties are inherited from the element's parent.
- Inheritance occurs via the cascade, which assigns greater importance to declarations whose sources are closer to the element. In ascending order, these sources are browser default styles, user custom styles, external style sheets, internal style sheets, and inline styles.
- For declarations from the same source, specificity tells the browser to render the styles from the more specific of the selectors. In ascending order, these selectors are elements and pseudo-elements, classes and pseudo-classes, IDs, inline styles, and the `!important` keyword.

EXERCISES

- Fun ::before and ::after selectors
- Understanding inheritance
- Learning about the cascade

(mind the **!important** selector)

- Introducing specificity