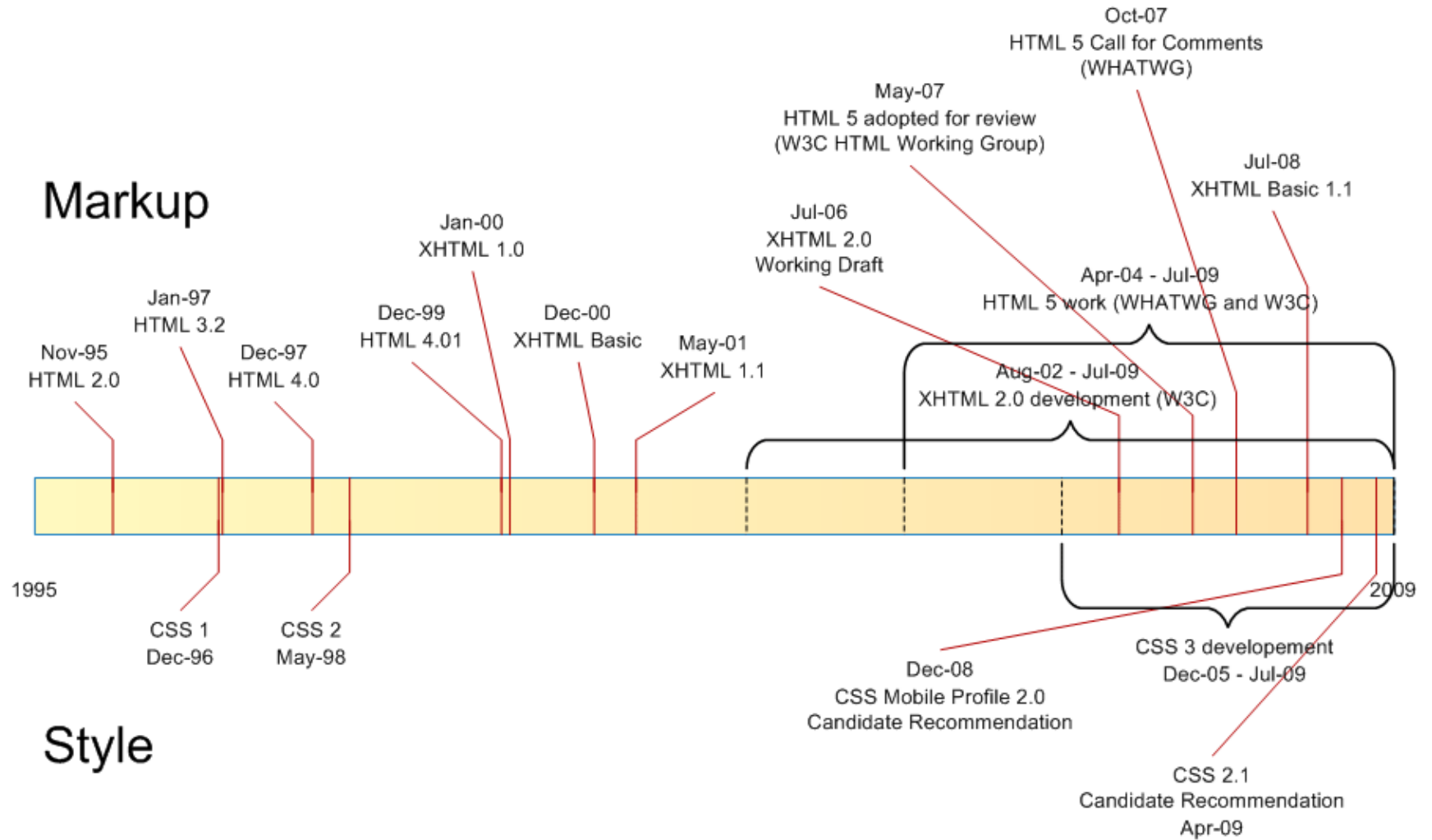


IT2805 - Web Technologies

Cascading Style Sheets (CSS)

Kshitij Sharma

Department of Computer and Information Science
Faculty of Information Technology and Electrical Engineering



The Problems with HTML – Poor Coding

- No consistency in the way a document is developed
 - The same markup can often be written a number of ways.
- One can omit attributes, or include irrelevant or illegal attributes (browsers usually ignore such problems), this encourages poorly written code.
 - `<table border>`
 - `<table border="1">`
 - `<table>`
 - `<table bdr>` (wrong syntax)
 - `<table border="0">`

Solution: XHTML



XHTML	HTML
All elements explicitly closed eg: <code>
</code>	Not necessary eg: <code>
</code>
All elements/attributes must be <i>lower case</i>	Case insensitive
Must have quotes for attributes eg: <code><body lang="en"></code>	Quotes not necessary eg: <code><body lang=en></code>
Necessary to have: <code><html>, <head>, or <body></code>	Not necessary

XHTML is stricter than HTML and is recommended for accuracy reasons

The Problems with HTML – Coding Design

- As appearance and design became a central issue new design tags were created.
- `` is the classic example.
 - It defines typographic presentation. This tag is not easy to use, e.g., every time you change an attribute of the font, you must close and re-open the tag.

```
<font size="3" color="red">This is some text!</font>
```

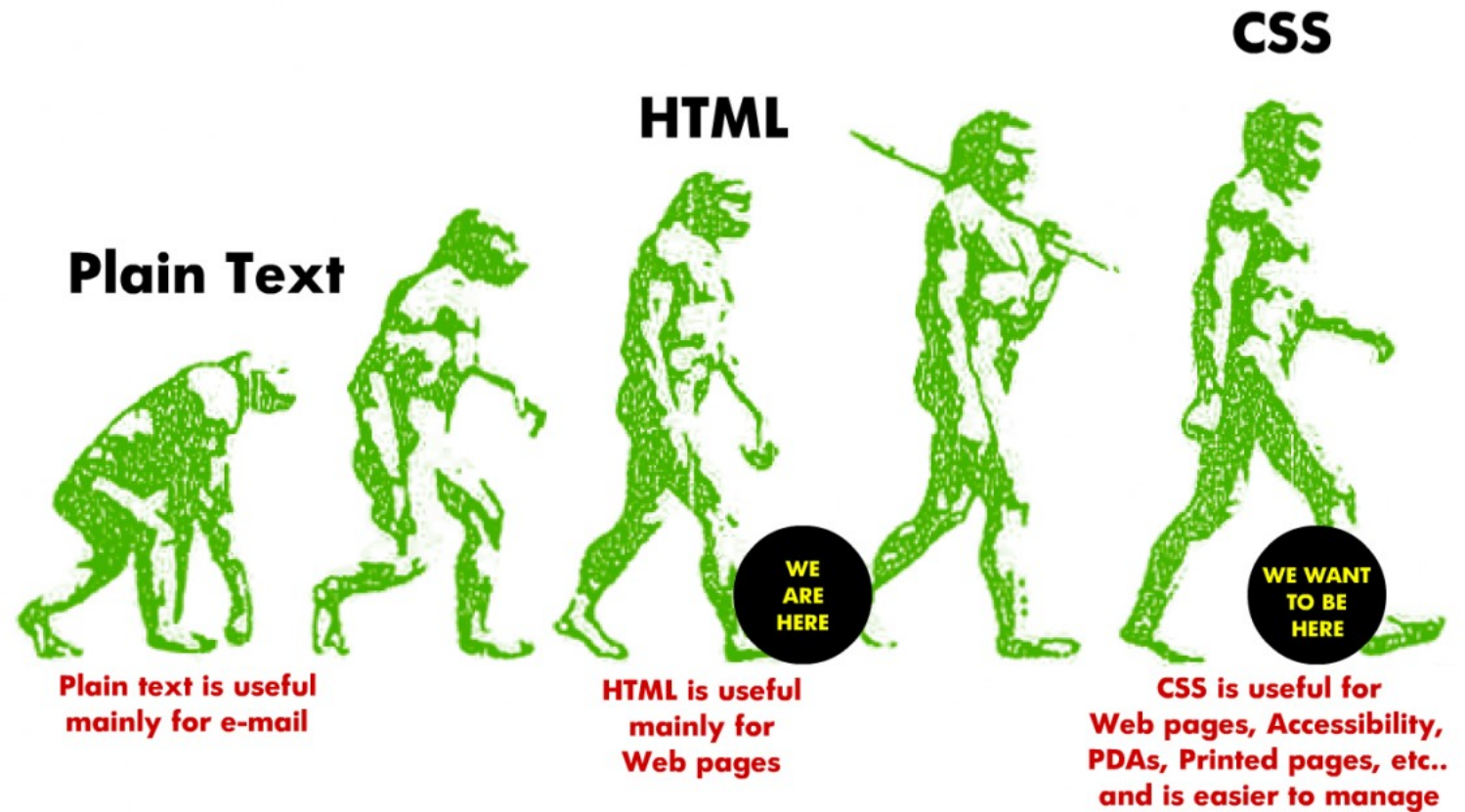
HTML `` Tag.
Not Supported in
HTML5.

The Problems with HTML - Compatibility

- The initial goal of HTML was across platforms. Unfortunately this goal has been degraded.
- Distinct differences between the main browsers, and how they interpret HTML.
- Proprietary tags introduced due to browser war. For example:
 - `<blink>` only works for Netscape
 - `<marquee>` only works for Internet Explorer

Solution: Separate Structure from Appearance

The standard language for doing this is “Cascading Style Sheets” (CSS), like HTML 5, CSS is an official W3C recommendation.



Facebook Login

You must log in to see this page.

Email:

Password:

☐ Keep me logged in

Log In

or [Sign up for Facebook](#) [Forgot your password?](#)

- [English \(US\)](#)
- [Afrikaans](#)
- [Español](#)
- [Português \(Brasil\)](#)
- [Français \(France\)](#)
- [Deutsch](#)
- [Italiano](#)



Evolution of CSS

CSS1 – DEC 1996

- TypeFace
- Colours of Text and Background
- Spacing between words, letters and lines
- Alignment
- Margin, border, padding
- Unique identification
- Classification of attributes

CSS2 – DEC 1998

- All of CSS1
- Absolute relative and fixed positioning of elements
- Media types
- Bidirectional text
- Font shadows

CSS3 – Under development






- All of CSS2
- New selectors
- Fancy borders
- User Interaction
- ...

CSS TV

CSS Print Profile

CSS Mobile Profile 2.0

Compatibility of CSS3

					
Animations	✓	✗	✗	✓	✗
Columns	✓	✓	✗	✓	✗
Gradients	✓	✓	✗	✓	✗
Reflections	✓	✗	✗	✓	✗
Transforms	✓	✓	✓	✓	✗
Transforms 3D	✗	✗	✗	✓	✗
Transitions	✓	✓	✓	✓	✗
FontFace	✓	✓	✓	✓	✓

How we see an HTML document?

The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in [England](#) and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained [English estate gardens](#).

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

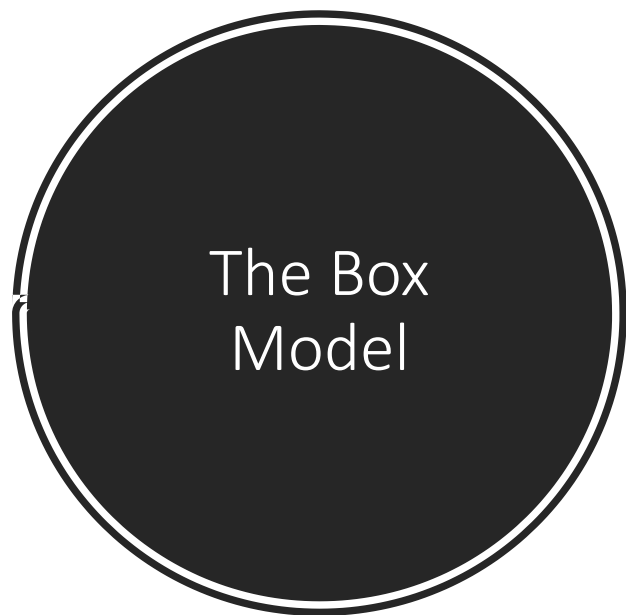
How CSS sees an HTML document?

The Cottage Garden

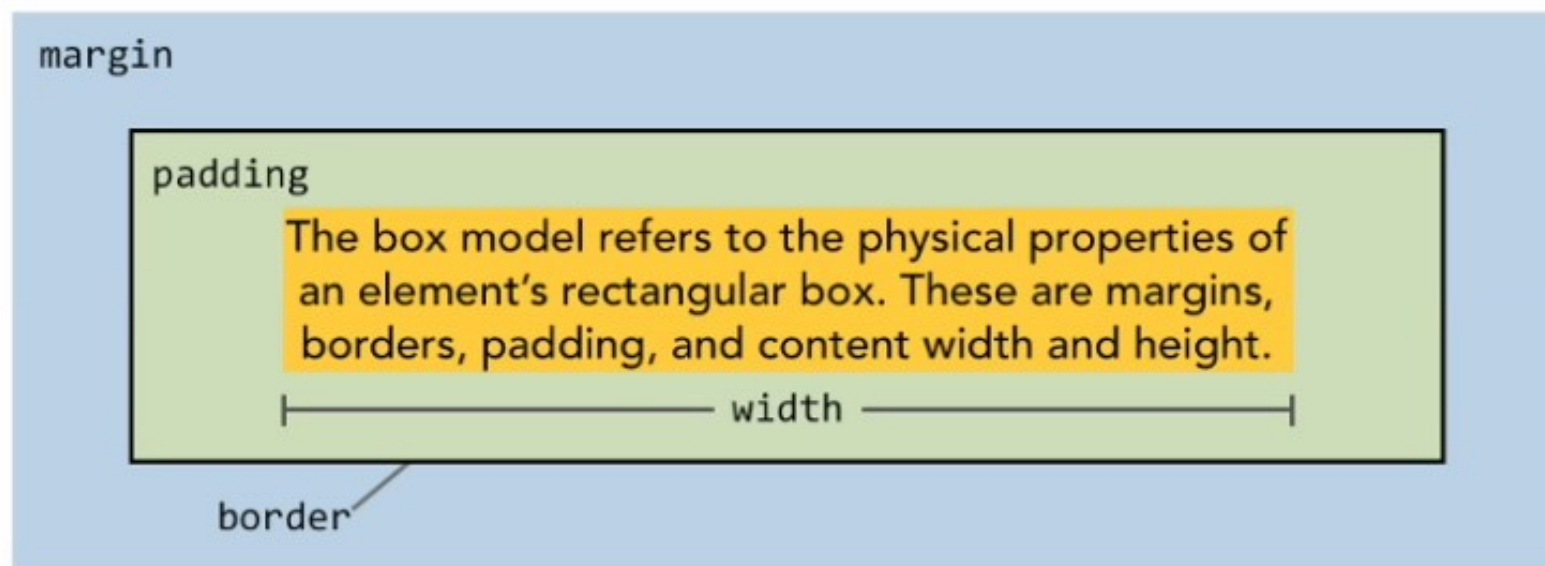
The [cottage garden](#) is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in [England](#) and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained [English estate gardens](#).

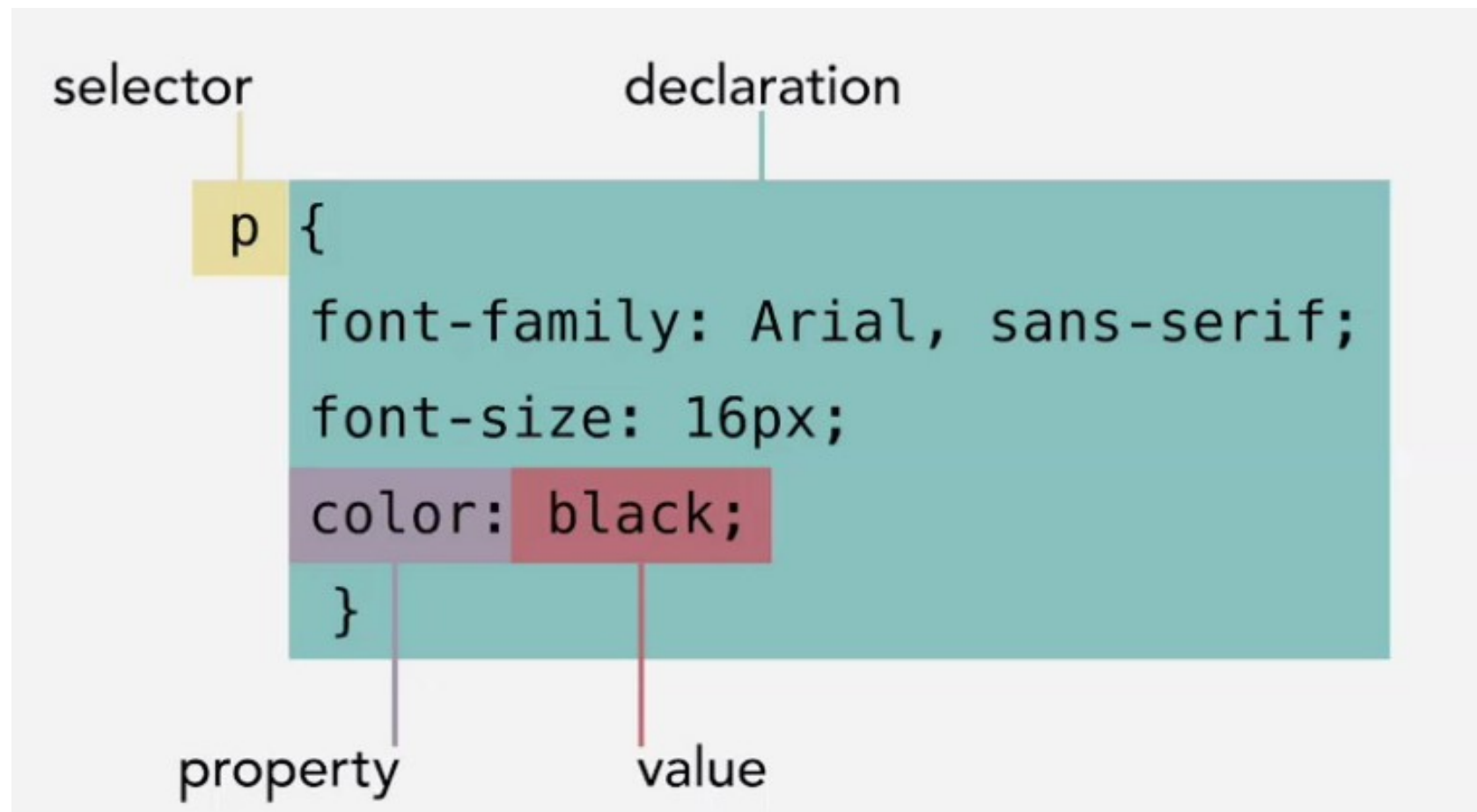
The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.



THE BOX MODEL



CSS Syntax





Bill

.....

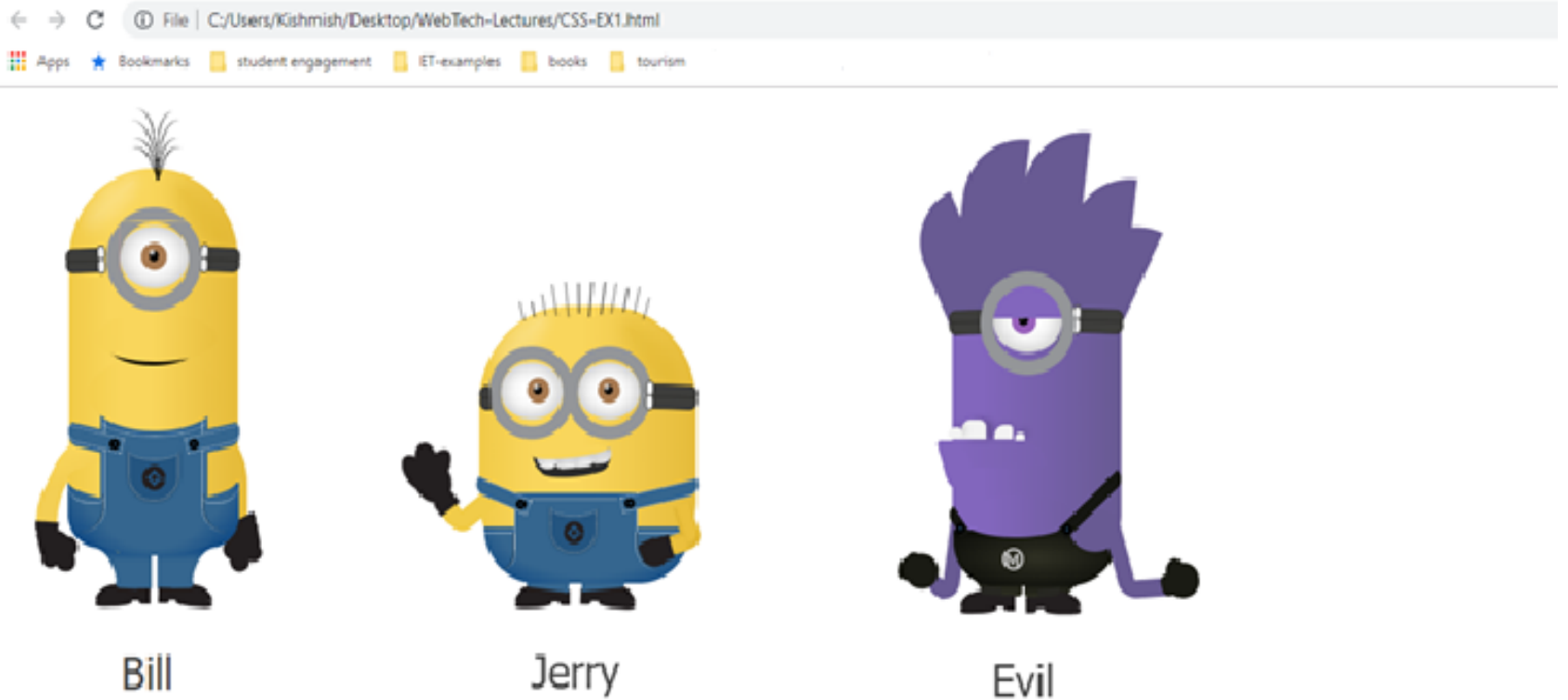
Jerry

.....

M

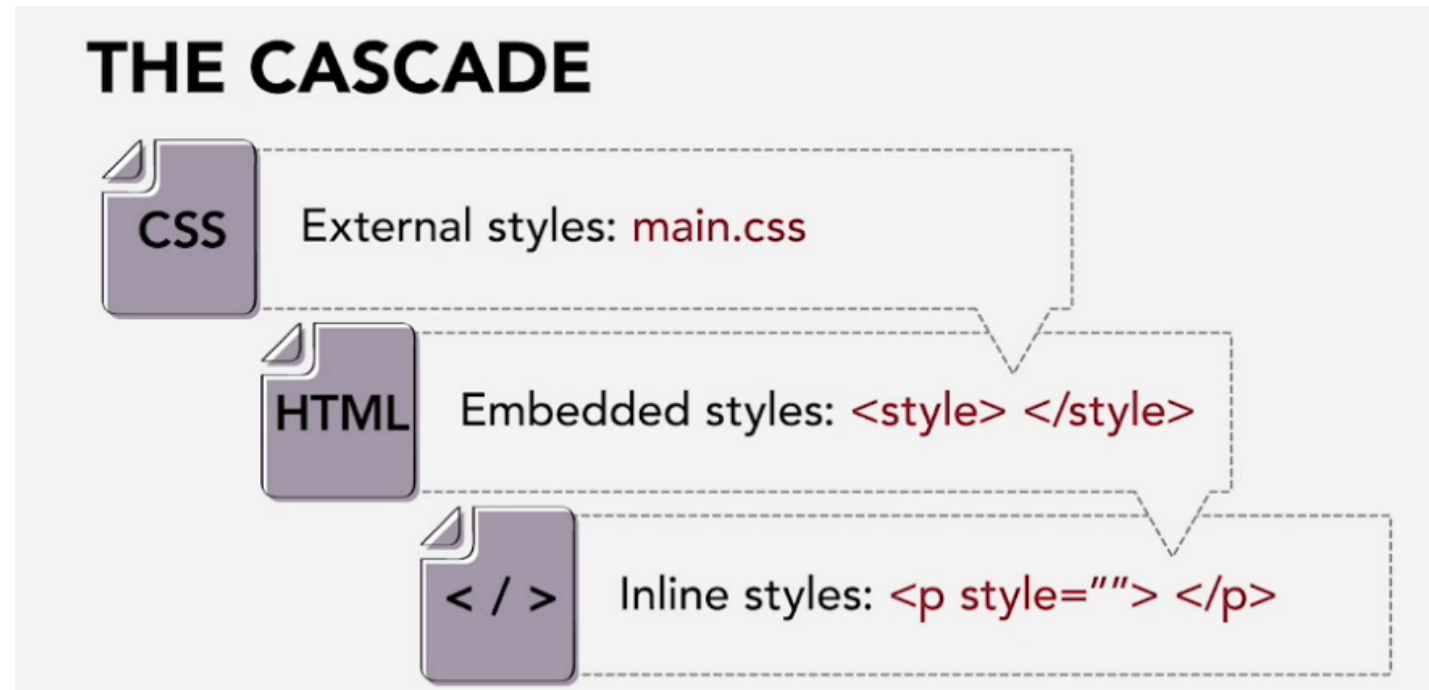
Evil

Rendering with CSS



Types of CSS

- There are three ways of specifying styles:
 - External style sheet (multiple pages can link to this).
 - Internal style sheet (applies only to specify web page).
 - Inline style (applies only to the specific element).



Where to place your CSS?

- Inside an HTML element
- Inside the head section of an HTML page
- In an external CSS file
 - in the head of my document

Remember, browsers always render the page in a linear manner

Inline Styles

- Each browser has default display style.
- Inline styles can be used to customize how elements are displayed. Examples:

```
<body style="background-color:yellow">
```

```
<h1 style="text-align:center">
```

```
<p style="font-family:courier new; color:red; font-size:20px">
```

Disadvantages of Inline Styles

- If all h2 headings need to use a certain font and alignment, then every time we specify a h2, we need to duplicate presentation attributes:

`<h2 style="text-align:center; font-size:10px"> blah blah </h2>`

- Each time we want to change the style for h2, need to search and replace everywhere.
- For large web pages/sites this leads to complexity and maintenance issues. Could be tens or hundreds of occurrences across tens of web pages

Embedded Style

- They specify uniform styles for each element in given page.
- Are specified in the head of your HTML document:

```
<head>  
<style>  
  body {  
    font-family: Helvetica, Arial, sans-serif;  
    color: #665544;  
    padding: 22px;}  
</style>  
</head>
```

- Older browsers do not support embedded style.

Embedded Style

- Advantages:
 - Uniform display of each element type within a given web page.
 - Easy to change – one location at head section.
- Disadvantages:
 - To make several web pages look same, need to duplicate the embedded style in each one.

External Style Sheet

- Style rules stored in a separate file (e.g., mystyle.css).
- Each page wishing to use this style links to this external file:

```
<head>
```

```
  <link rel="stylesheet" type="text/css" href="mystyle.css" />
```

```
</head>
```

- External style sheet contains list of style rules
- Advantages:
 - Many pages can share same style easily
 - Easy maintenance – just change external style sheet

How does CSS work?

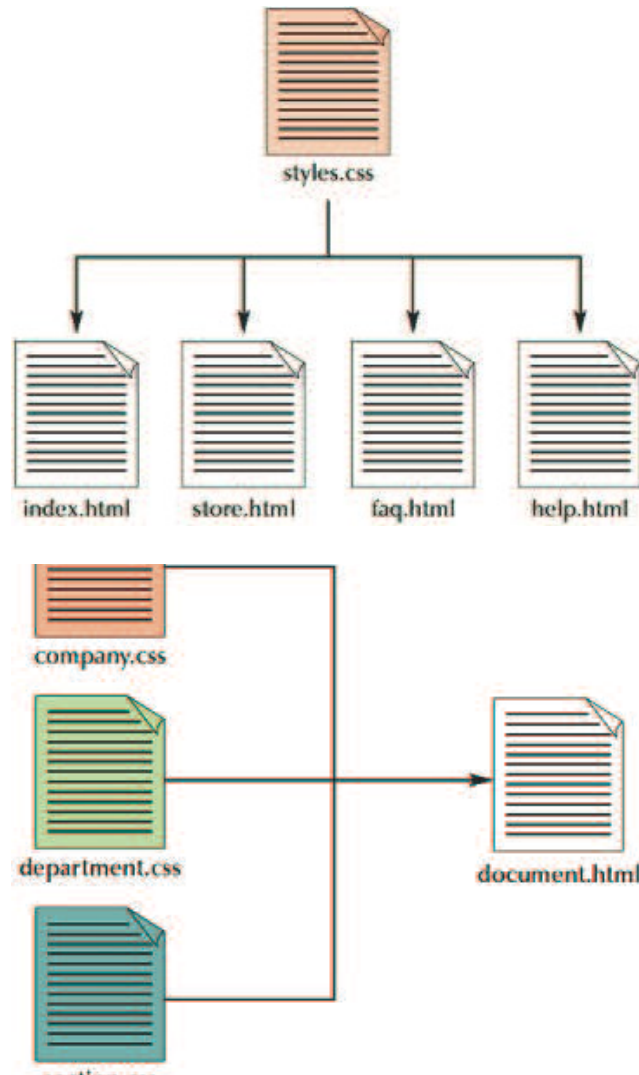
- **Web browsers assume a default set of values for each property**, which may vary from element to element. For example, browsers will render `<p>` text differently from `` text or `` text.
- **CSS defines an extensive set of presentation properties**, including color, font size, font family, margins, borders, and many more.
- **A CSS style sheet changes that default rendering** by naming an element or set of elements, and the new property values:

- `p {color: blue; font-size: small}`

Learning CSS

- **Focus on Selectors:** These allow you to efficiently target elements
- **Properties:** Learn which properties can be used for specific elements and which values they will accept
- **Implementing CSS:** Where you can add it to your HTML, how to externalize CSS into a single external file
- **Specificity, Inheritance and Cascade:** These three principles guide how rules are rendered and what happens if rules conflict with each other.
- **Browser Support:** Know what is supported by each browser and the minor rendering differences between them.

Understanding Cascading Order



- You can link a single style sheet to multiple documents in your Web site by using the link element
- You can also link a single document to several style sheets

Resolving Rule Conflict

- Use !important to override.
 - **p {color: blue !important}**
- All user rules and author rules have more weight than the default supplied by the web browser.
- If two rules of the same origin conflict, the more specific rule applies.
- If the rules are equally specific, the last-specified rule is chosen.

Resolving Rule Conflict

```
<html>
  <head>
    <title>How CSS Rules Cascade</title>
    <style type="text/css">
      * {font-family: Arial, Verdana, sans-serif;}
      h1 { font-family: "Courier New", Courier, monospace;}
      i { color: green;}
      i { color: red;}
      b { color: pink;}
      p b { color: blue !important;}
      p b { color: violet;}
      p#intro { font-size: 100%;}
      p { font-size: 75%;}
    </style>
  </head>
  <body>
    <h1>Potatoes</h1>
    <p id="intro">There are <i>dozens</i> of different <b>potato</b> varieties.</p>
    <p>They are usually described as early, second early and maincrop potatoes.</p>
  </body>
</html>
```

Potatoes

There are *dozens* of different **potato** varieties.

They are usually described as early, second early and maincrop potatoes.

Later styles over-rule earlier styles

- If you define a style property, and later define an alternative style property for the same thing, the later definition over-rides the earlier one.

```
<style>
  body {
    background-color:yellow;
    font-weight:bold;}
  div {
    background-color:#afa;
    font-weight:normal;}
</style>
```

```
<p>Some text here, inherits properties of the body.</p>
```

```
<div>
```

```
<p>However, the div's rules over-ride the body's rules, as the div's rules apply
later (i.e. nearer to this text in the document).</p>
```

```
</div>
```

Style specificity (prioritization) for one document

- Three ways to apply a style to an HTML document:
 - **Inline Styles**
 - **Embedded Styles**
 - **External Styles**
- Style prioritization
 - **Inline > embedded > external**

The more local a style is defined, the higher priority has

Inheritance

- Property values are normally *inherited*.
- Suppose there is a **<h1>** element with an emphasizing element (****) inside:
 - **<h1>**The headline ****is**** important!**</h1>**
- If no color has been assigned to **** element, the emphasized “is” will inherit the color of the parent element, so if **<h1>** has the color blue, the content enclosed in the **** element will be in blue as well.

Inheritance

```
<html>
  <head>
    <title>Inheritance</title>
    <style type="text/css">
      body {
        font-family: Arial, Verdana, sans-serif;
        color: #665544;
        padding: 10px;}
      .page {
        border: 1px solid #665544;
        background-color: #efefef;
        padding: inherit;}
    </style>
  </head>
  <body>
    <div class="page">
      <h1>Potatoes</h1>
      <p>There are dozens of different potato varieties.</p>
      <p>They are usually described as early, second early and maincrop potatoes.</p>
    </div>
  </body>
</html>
```

Potatoes

There are dozens of different potato varieties.

They are usually described as early, second early and maincrop potatoes.

Styles inherited from a parent

- Some styles, like font family, text-alignment etc., are automatically inherited by child elements from their parent element (i.e. by an element contained inside another one).
- Others, like padding are not automatically inherited.

```
<div style="font-family:serif; border:1px solid red; padding:10px;">
```

This text will be in a serif font.

```
<p>
```

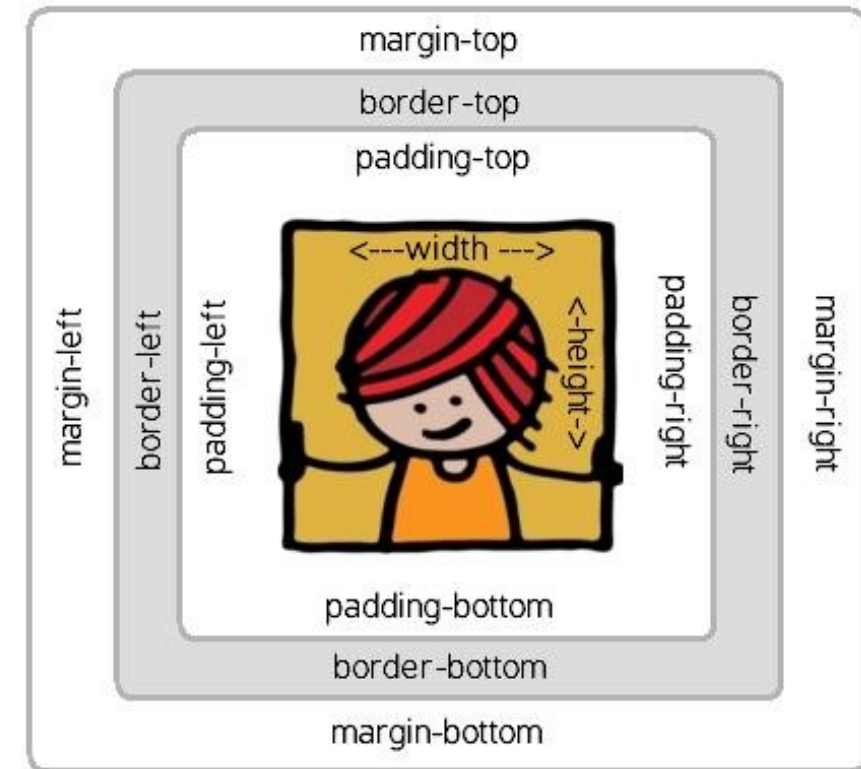
This text is also in a serif font, because font is inherited by default.

But the border and padding properties are not inherited from the parent div.

```
</p>
```

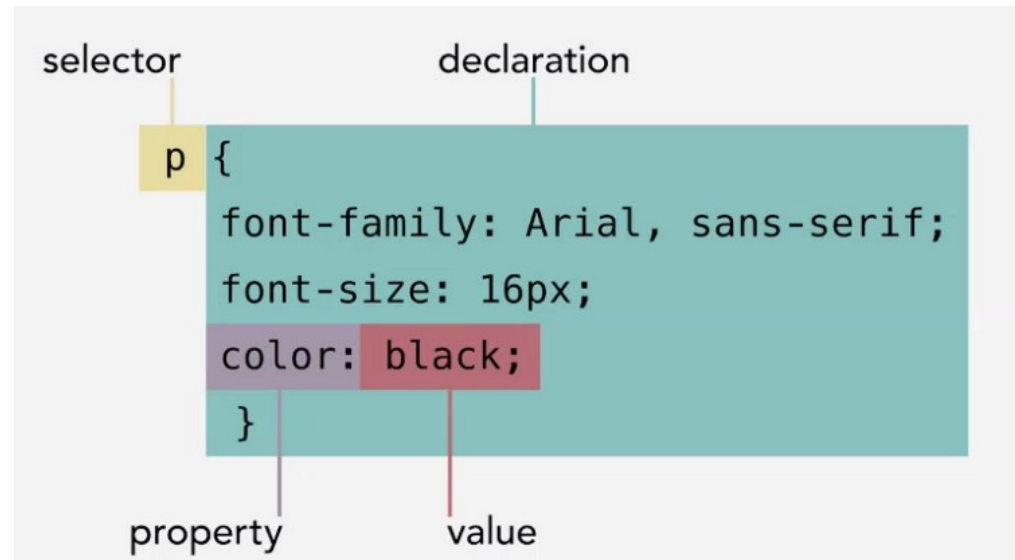
Summarizing

- CSS treats HTML element as it appears inside its own box and uses rules to indicate how that element should look
- You can use CSS to control the dimensions of a box
- You can also control the borders, margins and padding of each box with CSS



Summarizing

- Rules are made up of
 - **Selectors**, that specify the element the rules apply
 - **Declarations**, that indicate what these elements should look like
 - **Properties** of the element that you want to change
 - **Values** you want to give on those defined properties



Thank you!