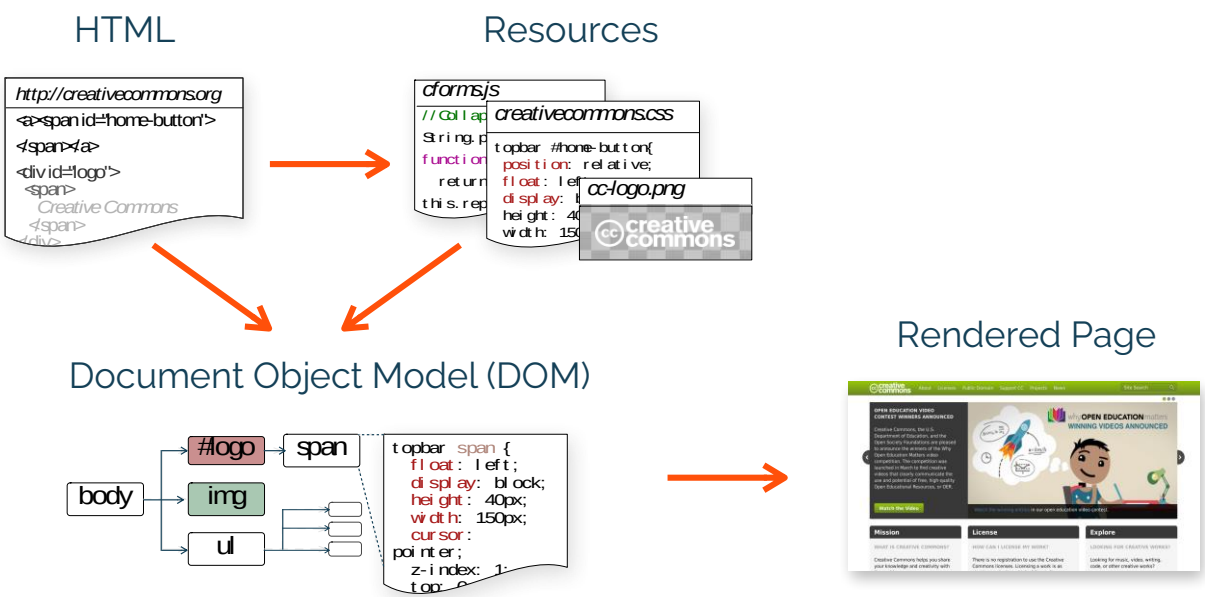
Document Object Model (DOM) and Cascading Style Sheets (CSS)

See https://developer.mozilla.org/en-US/docs/Web/Guide/CSS)

Assignments Released!

Browser's View of HTML: DOM

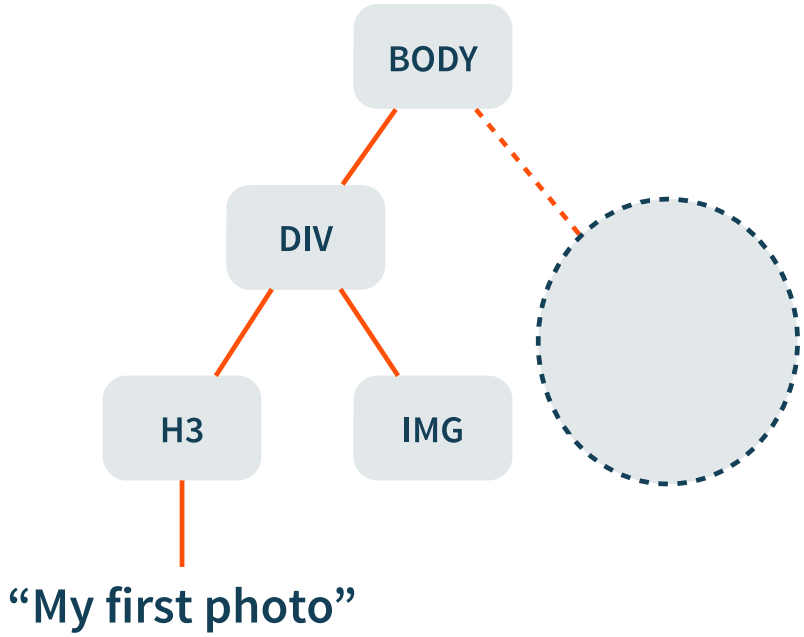
- HTML is parsed by the browser into a tree structure - Document Object Model (**DOM**)



DOM: Example

- Often one-to-one correspondence between HTML and the DOM rendered by browser

```
<body>
  <div class="photo">
    <h3>My first photo</h3>
    <img src="picture1.jpg"/>
  </div>
...
</body>
```



DOM: pros and cons?

DOM: pros

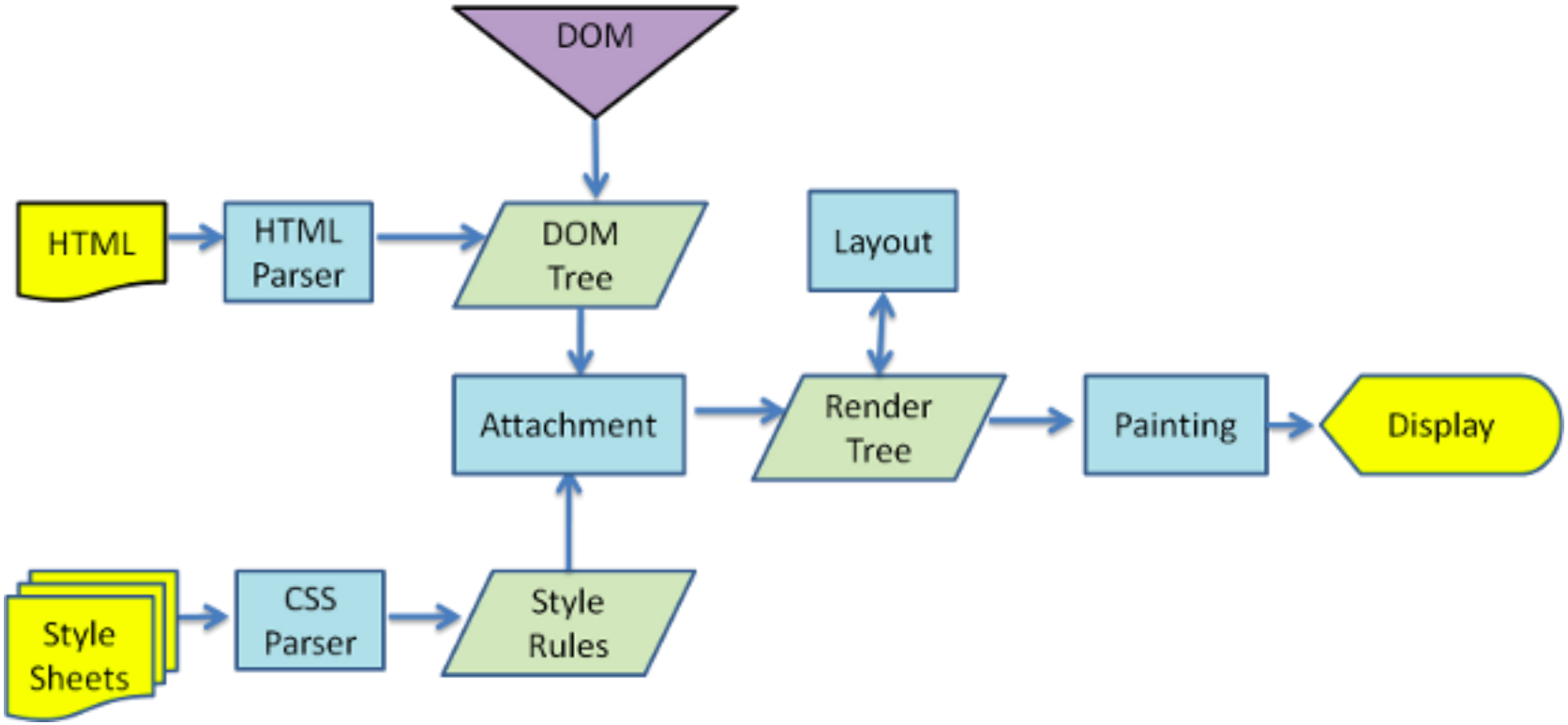
- Common data structure for holding elements of a web page (HTML, CSS, JavaScript etc.)
- Corresponds almost exactly to the browser's rendered view of the document
  - Changes to the DOM are made (almost) immediately to the rendered version of the webpage
  - Heavily used by JavaScript code to make changes to the webpage, and also by CSS to style the page

DOM: Disadvantages

- No isolation between different parts of the DOM tree for a script as long as it's from the **same origin**
  - All scripts from the same origin (i.e., domain) can access the entire DOM tree from that origin
  - Highly dynamic - difficult to reason about DOM state
- DOM is also somewhat browser-specific
- Can be a significant bottleneck in rendering webpages in parallel as it is a single global structure

What do web browsers do to render a page?

What do web browsers do ?



Example from Webkit: Used by Chrome and Safari

Class Activity

- Draw the DOM tree structure corresponding to the HTML

Draw the DOM tree structure corresponding to the HTML:

```
<html>
    <head>
    <meta charset="UTF-8">
        <title>World News Headlines</title>
    </head>
    <body>
        Some news stories
        <h1 id="title">Local News</h1>
        <div class="contact">
            <!-- This is a comment -->
            <h2>Sports</h2>
            <p id="snews">Headlines: </p>
        </div>
    </body>
```

Draw the DOM tree structure corresponding to the HTML

```
<html>
  <head>
  <meta charset="UTF-8">
    <title>World News Headlines</title>
  </head>
  <body>
    Some news stories
    <h1 id="title">Local News</h1>
    <div class="contact">
      <!-- This is a comment -->
      <h2>Sports</h2>
      <p id="snews">Headlines: </p>
    </div>
  </body>
```

```
└HTML
  ├HEAD
  │ ├#text:
  │ ├META charset="UTF-8"
  │ ├#text:
  │ ├TITLE
  │ │ └#text: World News Headlines
  │ └#text:
  ├#text:
  └BODY
    ├#text: Some news stories
    ├H1 id="title"
    │ └#text: Local News
    ├#text:
    ├DIV class="contact"
    │ ├#text:
    │ ├#comment: This is a comment
    │ ├#text:
    │ ├H2
    │ │ └#text: Sports
    │ ├#text:
    │ ├P id="snews"
    │ │ └#text: Headlines:
    │ └#text:
    └#text:
```

CSS

Key concept: Separate style from content

Content (what to display) is in HTML files

Formatting information (how to display it) is in separate style sheets (.css files).

Result: define style information once, use in many places

- Consider can you make all the text in the app slightly bigger?
- Or purple is our new company color.

DRY principle: Don't Repeat Yourself

CSS: Philosophy and Motivation

- Language for specifying how (HTML) documents are presented to users (Separate from content)

- Declarative – set of rules and their actions
  - Makes it easy to modify and maintain the website

- Allows different rules to be specified for different display formats (e.g., printing versus display)

Including CSS in HTML: Example

```
<html>
  <head>
   <title>Sample document</title>
  <link rel="stylesheet" href="style1.css">
  </head>
  <body>
    <p>
      <strong>C</strong>ascading
      <strong>S</strong>tyle
      <strong>S</strong>heets
    </p>
  </body>
</html>
```

style1.css

Style sheet contain one or more **CSS Rules**

Selector

```
body {
  font-family: Tahoma, Arial, sans-serif;
  color: black;
  background: white;
  margin: 8px;
}
```

Declaration Block

Property

Value

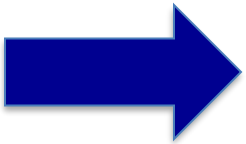
CSS by Tagname

```
1  strong {color: red;}
```

Tag name to match

Attribute: Value

```
<p>
    <strong>C</strong>ascading
    <strong>S</strong>tyle
    <strong>S</strong>heets
  </p>
```



Cascading Style Sheets

How does CSS work ?

- Apply styles to the DOM tree of the web page
- CSS rule applies to DOM nodes matching tag, and their descendants (unless overridden)

```
P
├─STRONG
│ └─"C"
├─"ascading"
├─STRONG
│ └─"S"
├─"tyle"
├─STRONG
│ └─"S"
└─"heets"
```

```
strong {color: red;}
```

Tags that match the style rules in DOM tree

CSS Inheritance

- All descendants of a DOM node inherit the CSS styles ascribed to it unless there is a “more-specific” CSS rule that applies to them

- Always apply style rules in top down order from the root of the DOM tree and overriding the rules as and when appropriate

What style would the output be?

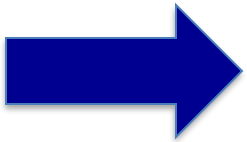
```
1  p {color:blue; text-decoration:underline}
2  strong {color:red}
```

```
<p>
  <strong>C</strong>ascading
  <strong>S</strong>tyle
  <strong>S</strong>heets
</p>
```

CSS Inheritance Example

```
1 p {color:blue; text-decoration:underline}
2 strong {color:red}
```

```
<p>
  <strong>C</strong>ascading
  <strong>S</strong>tyle
  <strong>S</strong>heets
</p>
```



Cascading Style Sheets

CSS by Class and IDs

- CSS rules can also apply to elements of a certain class or an element with a specific ID

```
.key {
  color: green;
}
```

```
#principal {
  font-weight: bolder;
}
```

```
<p class="key" id="principal">
```

CSS selector types

CSS Selector	CSS	HTML

Tag name	`h1 {`
	`  color: red;`
	`}`
		`<h1>Today's Specials</h1>`

Class attribute	`.large {`
	`  font-size: 16pt;`
	`}`
		`<p class="large">...`

Tag and Class	`p.large {...}`	`<p class="large">...`

Element id	`#p20 {`
	`  font-weight: bold;`
	`}`
		`<p id="p20">...`

CSS Rules and Priority

- What to do when rules conflict with each other ?
  - Always apply the “most specific selector”

- “Most-specific” (‘>’ represents specificity):
  - Selectors with IDs > Classes > Tags
  - Direct rules get higher precedence over inherited rules (as before)

CSS Class and IDs: Example

```html
<!doctype html>
<html>
  <head>
  <meta charset="UTF-8">
  <title>Sample document</title>
  <link rel="stylesheet" href="style1.css">
  </head>
  <body>
    <p id="first">
      <strong class="carrot">C</strong>ascading
      <strong class="spinach">S</strong>tyle
      <strong class="spinach">S</strong>heets
    </p>
    <p id="second">
      <strong>C</strong>ascading
      <strong>S</strong>tyle
      <strong>S</strong>heets
    </p>
  </body>
</html>
```

```css
strong { color: red; }
.carrot { color: orange; }
.spinach { color: green; }
#first { font-style: italic; }
```

?

CSS Class and IDs: Example

```html
<!doctype html>
<html>
  <head>
  <meta charset="UTF-8">
  <title>Sample document</title>
  <link rel="stylesheet" href="style1.css">
  </head>
  <body>
    <p id="first">
      <strong class="carrot">C</strong>ascading
      <strong class="spinach">S</strong>tyle
      <strong class="spinach">S</strong>heets
    </p>
    <p id="second">
      <strong>C</strong>ascading
      <strong>S</strong>tyle
      <strong>S</strong>heets
    </p>
  </body>
</html>
```

```css
strong { color: red; }
.carrot { color: orange; }
.spinach { color: green; }
#first { font-style: italic; }
```

Cascading Style Sheets

Cascading Style Sheets

CSS Selectors based on Relationships

- Selectors can also be based on relationships between elements in the DOM tree
  - A E : Any element E that is a **descendant** of A
  - A > E: Any element E that is a **child** of A
  - E : first-child: Any element E that is the first child of its parents
  - B + E : Any element E that is the next sibling of B element (i.e., B and E have the same parent)

Describe what each CSS rule does!

```html
<div class="menu-bar">
  <ul>
    <li>
      <a href="example.html">Menu</a>
      <ul>
        <li>
          <a href="example.html">Link</a>
        </li>
        <li>
          <a class="menu-nav" href="example.html">Submenu</a>
          <ul>
            <li>
              <a class="menu-nav" href="example.html">Submenu</a>
              <ul>
                <li><a href="example.html">Link</a></li>
                <li><a href="example.html">Link</a></li>
                <li><a href="example.html">Link</a></li>
                <li><a href="example.html">Link</a></li>
              </ul>
            </li>
            <li><a href="example.html">Link</a></li>
          </ul>
        </li>
      </ul>
    </li>
  </ul>
</div>
```

```css
div.menu-bar ul ul {
  display: none;
}

div.menu-bar li:hover > ul {
  display: block;
}
```

Describe what each CSS rule does!

```html
<div class="menu-bar">
  <ul>
    <li>
      <a href="example.html">Menu</a>
      <ul>
        <li>
          <a href="example.html">Link</a>
        </li>
        <li>
          <a class="menu-nav" href="example.html">Submenu</a>
          <ul>
            <li>
              <a class="menu-nav" href="example.html">Submenu</a>
              <ul>
                <li><a href="example.html">Link</a></li>
                <li><a href="example.html">Link</a></li>
                <li><a href="example.html">Link</a></li>
                <li><a href="example.html">Link</a></li>
              </ul>
            </li>
            <li><a href="example.html">Link</a></li>
          </ul>
        </li>
      </ul>
    </li>
  </ul>
</div>
```

```css
div.menu-bar ul ul {
  display: none;
}

div.menu-bar li:hover > ul {
  display: block;
}
```

The first rule says that for all ‘div’ elements of class ‘menu-bar’, in which an ul element is a **descendant** of another ul, do not display the element

The second rule says that for all ‘div’ elements of class ‘menu-bar’, in which an ul element is a **child** of an li element, display it and the entire block, if the mouse hovers over the element

CSS Pseudo-Class Selectors

- CSS Selectors can also involve actions external to the DOM called pseudo-classes
  - Visited: Whether a page was visited in the history
  - Hover: Whether the user hovered over a link
  - Checked: Whether a check box was checked

```
selector : pseudo-class {
      property: value
}
```

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

CSS Specificity

Based on the CSS specification[1], we can calculate the overall specificity weight (SW) of a selector **S** composed of different selector types as follows:

SW(S) = tuple(a, b, c, d)

where a, b, c, and d are calculated as follows:

a = 1 if the declaration is inline, 0 otherwise

b = number of ID types in S

c = number of class types in S

d = number of element/tag types in S

1) http://www.w3.org/TR/CSS2/cascade.html

Calculate the Specificity

```
1) .latest { color: green;}
```

```
2) #news span { color: red; }
```

SW(S) = tuple(a, b, c, d)

a = 1 if the declaration is inline, 0 otherwise
b = number of ID types in S
c = number of class types in S
d = number of element/tag types in S

Calculate the Specificity

```
1).latest { color: green;}    SW: {0,0,1,0}

2)#news span { color: red; } SW: {0,1,0,1}
```

SW(S) = tuple(a, b, c, d)

a = 1 if the declaration is inline, 0 otherwise
b = number of ID types in S
c = number of class types in S
d = number of element/tag types in S

Higher specificity wins when competing for the style of the same DOM element

```
1).latest { color: green;}    SW: {0,0,1,0}

2)#news span { color: red; } SW: {0,1,0,1}
```

SW(S) = concatenate(a, b, c, d)

a = 1 if the declaration is inline, 0 otherwise
b = number of ID types in S
c = number of class types in S
d = number of element/tag types in S

!important rule

The **!important property** in CSS means that all subsequent rules on an element are to be ignored, and the rule denoted by !important is to be applied. This rule overrides all previous styling rules.

```
h1 {
  background-color: red !important;
}
```

Class Activity: CSS Rules

- What's the effect of the following CSS spec. on the HTML code in the next two slides? Why?

```
#news { background-color: silver; font: italic; color: black; }

.sports { color: blue; text-decoration: underline; }

H3, H4 { font-family: sans-serif; }

.latest { color: green;}

#news span { color: red; }

P.select { font-size: medium; }
```

Class Activity: HTML -1

```
#news { background-color: silver; font: italic; color: black; }

.sports { color: blue; text-decoration: underline; }

H3, H4 { font-family: sans-serif; }

.latest { color: green;}

#news span { color: red; }

P.select { font-size: medium; }
```

```
<HTML>
 <HEAD>
  <LINK href="example.css" rel="stylesheet" type="↩
       text/css"/>
 </HEAD>
 <BODY>
   <P id='news' style='font:normal'>World
     <SPAN class='sports'>Sports news</SPAN>
   </P>
   <DIV class='sport'>Football</DIV>
 </BODY>
</HTML>
```

Class Activity: HTML -1

```
#news { background-color: silver; font: italic; color: black; }

.sports { color: blue; text-decoration: underline; }

H3, H4 { font-family: sans-serif; }

.latest { color: green;}

#news span { color: red; }

P.select { font-size: medium; }
```

```
<HTML>
 <HEAD>
  <LINK href="example.css" rel="stylesheet" type="↩
       text/css"/>
 </HEAD>
 <BODY>
   <P id='news' style='font:normal'>World
     <SPAN class='sports'>Sports news</SPAN>
   </P>
   <DIV class='sport'>Football</DIV>
 </BODY>
</HTML>
```

World Sports news
Football

Class Activity: HTML - 2

```
#news { background-color: silver; font: italic; color: black; }

.sports { color: blue; text-decoration: underline; }

H3, H4 { font-family: sans-serif; }

.latest { color: green;}

#news span { color: red; }

P.select { font-size: medium; }
```

```
<HTML>
 <HEAD>
 <LINK href="example.css" rel="stylesheet" type="↩
      text/css"/>
 </HEAD>
 <BODY>
   <P id='news' style='font:normal'>World
     <SPAN class='latest'>latest news</SPAN>
   </P>
 </BODY>
</HTML>
```

Class Activity: HTML - 2

```
#news { background-color: silver; font: italic; color: black; }

.sports { color: blue; text-decoration: underline; }

H3, H4 { font-family: sans-serif; }

.latest { color: green;}

#news span { color: red; }

P.select { font-size: medium; }
```

```
<HTML>
 <HEAD>
 <LINK href="example.css" rel="stylesheet" type="↩
      text/css"/>
 </HEAD>
 <BODY>
   <P id='news' style='font:normal'>World
     <SPAN class='latest'>latest news</SPAN>
   </P>
 </BODY>
</HTML>
```

World latest news

Detecting Unused CSS Rules

Paper: Ali Mesbah and Shabnam Mirshokraie. 2012. **Automated analysis of CSS rules to support style maintenance**. In Proceedings of the 34th International Conference on Software Engineering (ICSE '12). IEEE Press, 408-418.
https://people.ece.ubc.ca/amesbah/resources/papers/icse12.pdf

Building a Web app

- You are hired as a software engineer
- The goal is to construct a web application that **recommends movies to users based on their profiles and hobbies**
- In pairs, devise a plan in which you outline how you would go about achieving the goal.