HTML5

First Steps

Background

- Web starts in 1989
  - (Sir) Tim Berners-Lee
  - Enquirer 1980 - His personal project
  - “Classical” Hypertext can be traced back to
    - Jorge Francisco Isidoro Luis Borges Acevedo, The Garden of Forking Paths, 1941
    - Vannevar Bush, Memex 1945
  - Ideas similar to Hypertext can be traced back to
    - Agostino Ramelli, Book-Wheel, 1588
- Hypertext has been very active area starting in the 1960’s
  - Ted Nelson = Xanadu

…

- Many different HTML versions
- HTML5
- Main trends ->
  - Structure -> Trees
  - Clear(er) semantics,
  - Towards declarative constructs,
  - Incorporating media
  - Supporting evolving platforms
- Web Standards try to be compatible to older versions
- Web is based on best effort

Material

- https://dev.w3.org/html5/spec-LC/
- https://html.spec.whatwg.org/multipage/

Doctype

```
<! Doctype>

          <! DOCTYPE html>




<!--

          Comment1

          …….

        CommentN

–->
```

Tags

- Tags
    - Markup
    - <Tag …. > Something </Tag>
    - Some tags have no end e.g.
        - <br> -> line break

<a href="http://www.cs.usask.ca"> link </a>

- Tags can be nested

Attributes

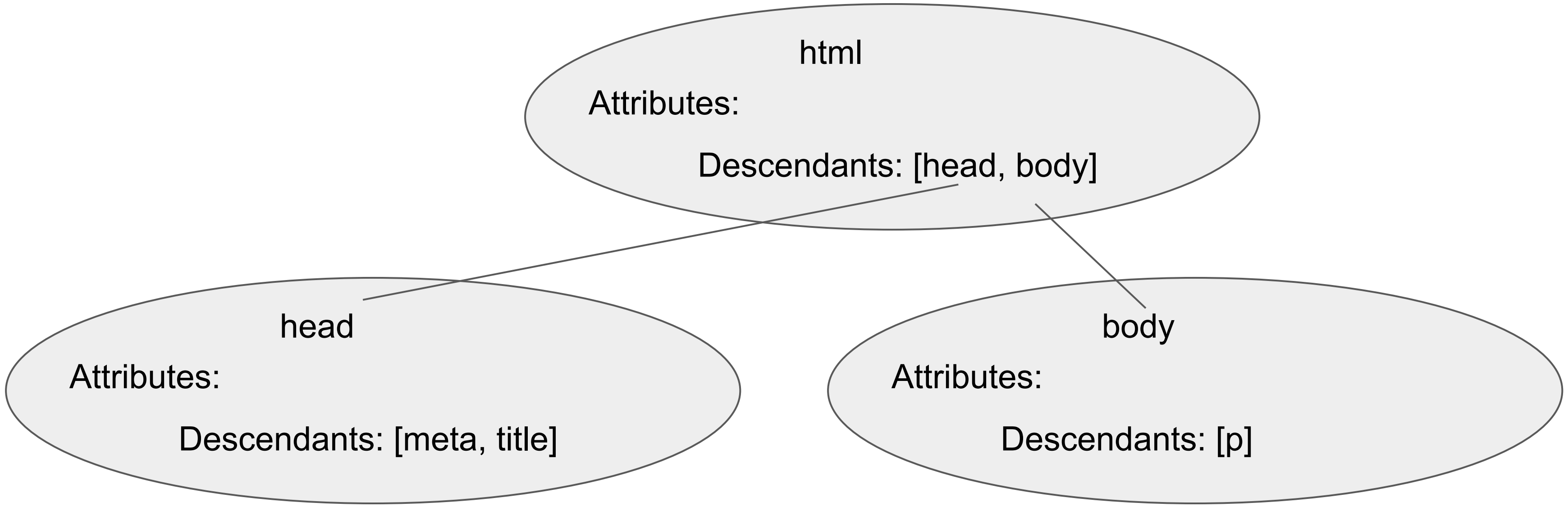
- Tags have attributes
  - <a **href**="http://www.cs.usask.ca"> link</a>
- Common Attributes
  - https://html.spec.whatwg.org/multipage/dom.html#global-attributes
- Tag specific
  - href
  - target
  - download
  - ping
  - rel
  - hreflang
  - type
  - referrerpolicy

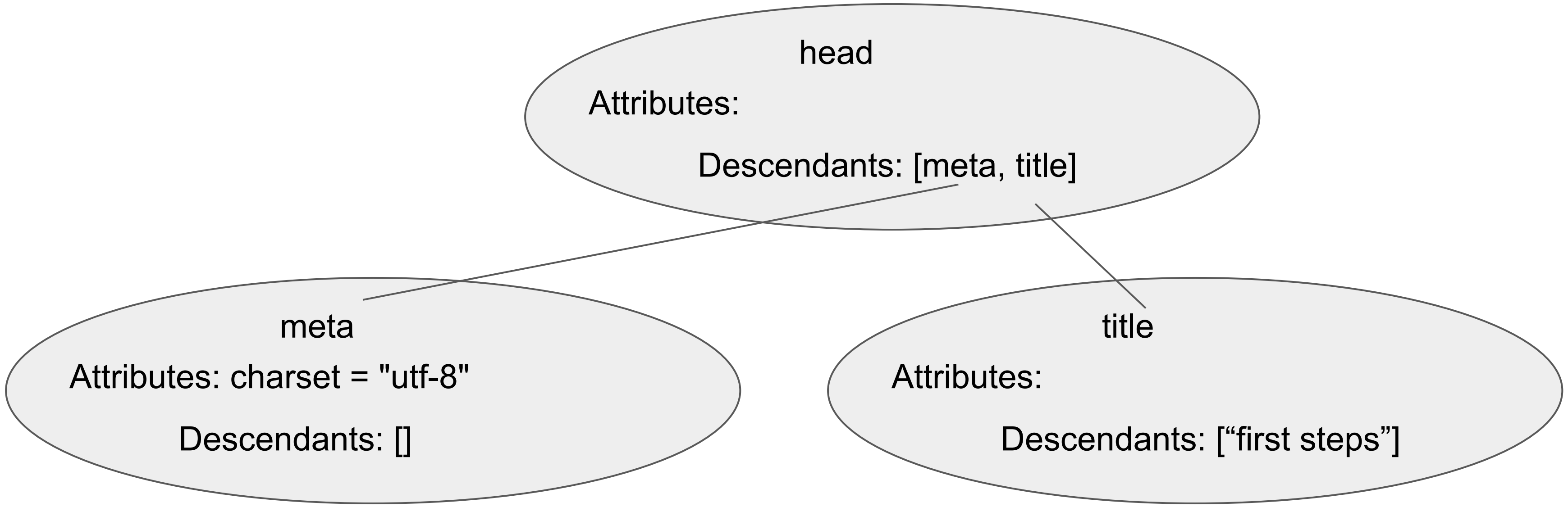
Example

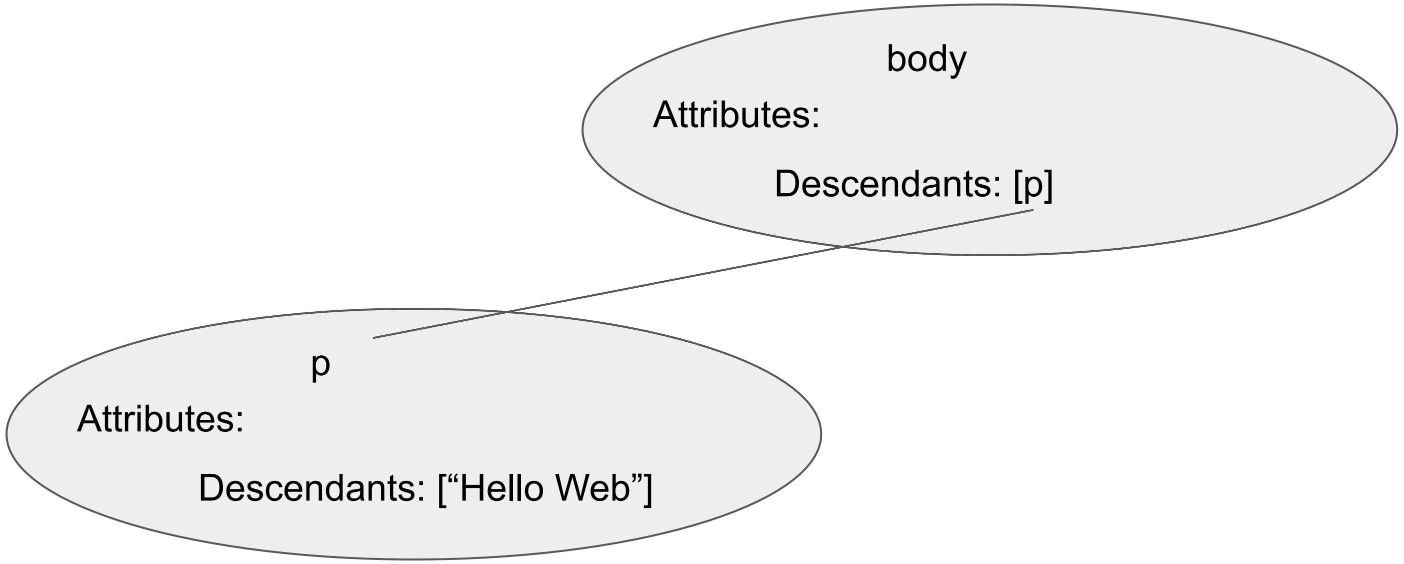
```
<!DOCTYPE html>

<! -- Comment -->

<html>
     <head>
          <meta charset = "utf-8">
          <title> First Steps </title>
     </head>
     <body>
          <p> Hello Web </p>
     </body>
</html>
```


html
Attributes:
Descendants: [head, body]
head
Attributes:
Descendants: [meta, title]
body
Attributes:
Descendants: [p]


head
Attributes:
Descendants: [meta, title]
meta
Attributes: charset = "utf-8"
Descendants: []
title
Attributes:
Descendants: [“first steps”]


body
Attributes:
Descendants: [p]
p
Attributes:
Descendants: [“Hello Web”]

DOM Manipulation

```
 <!DOCTYPE html>

<html>
   <head>

       <meta charset="UTF-8">
    <title>Demo 1</title>

   </head>
   <body>

       <p id="p1">Hello world!</p>

       <button onclick="ChangeText()"> Change text </button>
       <button id="b2"> Change color </button>

       <script type = "text/javascript">


              function ChangeText()
              {
                     document.getElementById("p1").innerHTML="New text!";
              }

              function ChangeColor()
              {
                     document.getElementById("p1").style.color = "blue";
              }

              document.getElementById("b2").addEventListener("click", ChangeColor);
                </script>


           </body>
</html>
```

```
<!DOCTYPE html>


<html>


<head>
   <meta charset="UTF-8" />
   <title>Fetch Text File Example</title>
</head>


<body>
   <h1>Load Text Content Using Fetch</h1>
   <button id="loadButton">Load
       Text File</button>
   <div id="content" style="white-space: pre;"></div>
```

```
<script>
        const loadButton = document.getElementById('loadButton');
        const contentDiv = document.getElementById('content');
        loadButton.addEventListener('click', () => {
            fetch('files.txt').then(response => {
                if (!response.ok) {
             throw new Error(`HTTP error! Status: ${response.status}`);
             }
                return response.text();
            }).then(text => {
                contentDiv.textContent = text;
            }).catch(error => {
                console.error('Fetch error:', error);
                contentDiv.textContent = 'Error loading file.';
            });
        });
    </script>
```

```
</body>

</html>
```

Alternative

- Promise
    - Specification:
        - https://tc39.es/ecma262/multipage/control-abstraction-objects.html#sec-promise-objects
    - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
    -
- XMLHttpRequest
    - Specification:
        - https://xhr.spec.whatwg.org/#interface-xmlhttprequest
    - https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

```html
<!DOCTYPE html>

<html>

<head>
   <meta charset="UTF-8" />
   <title>Fetch Text File Example</title>
</head>

<body>
   <h1>Load Text Content Using XMLHttpRequest</h1>
   <button id="loadButton">Load
       Text File</button>
   <div id="content" style="white-space: pre;"></div>
```

```
 <script>
        const loadButton =
document.getElementById('loadButton');
        const contentDiv =
document.getElementById('content');

        loadButton.addEventListener('click', () =>
          {
            const xhr = new XMLHttpRequest();
            xhr.open('GET', 'files.txt', true);
xhr.onload = function () {
            if (xhr.status >= 200 && xhr.status < 300) {
                contentDiv.textContent = xhr.responseText;
            }
            else {
                console.error(`HTTP error! Status: ${xhr.status}`);
                contentDiv.textContent = 'Error loading file.';
            }
        };
        xhr.onerror = function () {
            console.error('Network error:', xhr.status);
            contentDiv.textContent = 'Error loading file.';
        };
        xhr.send();
    });
</script>
```

```
</body>

</html>
```

Style (example from html definition)

```
<p>My sweat suit is <span style="color: green; background:
transparent">green</span> and my eyes are <span
style="color: blue;
background: transparent">blue</span>.</p>
```

Style

- Inline
- Style File

Inline style

- Use the style attribute (in tag) to declare a style for an individual element
- Every CSS property is followed by a colon and the value of the attribute
- Multiple property declarations are separated by a semicolon

```
<h1
  style = "font-size: 18pt; color: #FF0000">
hello
</h1>
```

Style tag

```
<style type = "text/css">
    h1      { font-family: helvetica, tahoma; color: #FFFF00}
    h2      { font-size: 18pt; color: #00FF00 }
    .ralph1 { color: #FF0000}
     #ralph2 {color: #00FF00}
</style>
...
<body>
<h1 style = "font-size: 18pt; color: #0000FF"> hello </h1>
<h1> hello1  </h1>
<h1 class="ralph1"> hello2 </h1>
<h1 id="ralph2"> hello3 </h1>
```

Style in Separate File

```
<!DOCTYPE html>
<html>
        <head>
                <title>Demo page</title>

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

        </head>

        <body>

        ……


        </body>
</html>
```

CSS File

```
h1 { font-family: helvetica, tahoma; color: #FFFF00}
h2 { font-size: 18pt; color: #00FF00 }
.ralph1 { color: #FF0000}
 #ralph2 {color: #00FF00}
```

CSS Statements

- Style defines rules
  - Each rule consists of selector and body
  - Rule body begins with { and ends with }
  - Different properties are separated by ;
  - Different values are separated by ,

- h1   { font-family: helvetica, tahoma; color: #FFFF00;}

- Selector {attribute: value;}
- Selector {attribute1: value; …..; attributeN: value; }

Selectors

- Selector is name for rule/style
- Selector Types
  - Tag
    - h1 { font-family: helvetica, tahoma; color: #FFFF00;}
  - Class
    - .ralph1 { color: #FF0000}
  - ID
    - #ralph2 {color: #00FF00}

CSS File

```
<style type = "text/css">
     h1  { font-family: helvetica, tahoma; color: #FFFF00}
     h2  { font-size: 18pt; color: #00FF00 }
     .ralph1 { color: #FF0000}
    #ralph2 {color: #00FF00}
</style>
```

Grouping & Nesting

- Standalone
  - h1 { font-family: helvetica, tahoma; color: #FFFF00;}
- Grouping
  - h1, h2 { font-family: helvetica, tahoma; color: #FFFF00;}
- Nested
  - h1 h2 { font-family: helvetica, tahoma; color: #FFFF00;}

- .class
  - class
- #id
  - id
- tag
  - standalone
- tag1 tag2
  - Select if tag2 is in tag1

- tag1>tag2
  - Select when tag1 is a parent of tag2
- tag1+tag2
  - Select tag2 that is right after tag1
- [attribute]
  - Select all tags with attribute
- [attribute=value]
  - Select all tags with attribute equal value

- [attribute~=value]
    - Select all tags with attribute containing value
- :link
    - Select unvisited links
- :visited
    - Select visited links
- :active
    - Select active link
- :hover
    - Select links with “mouse over”