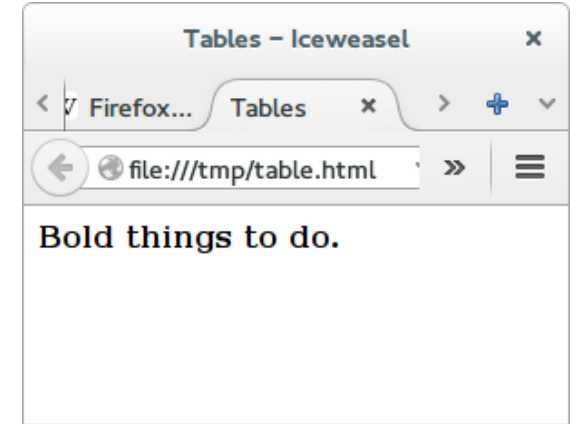


# Web Browser

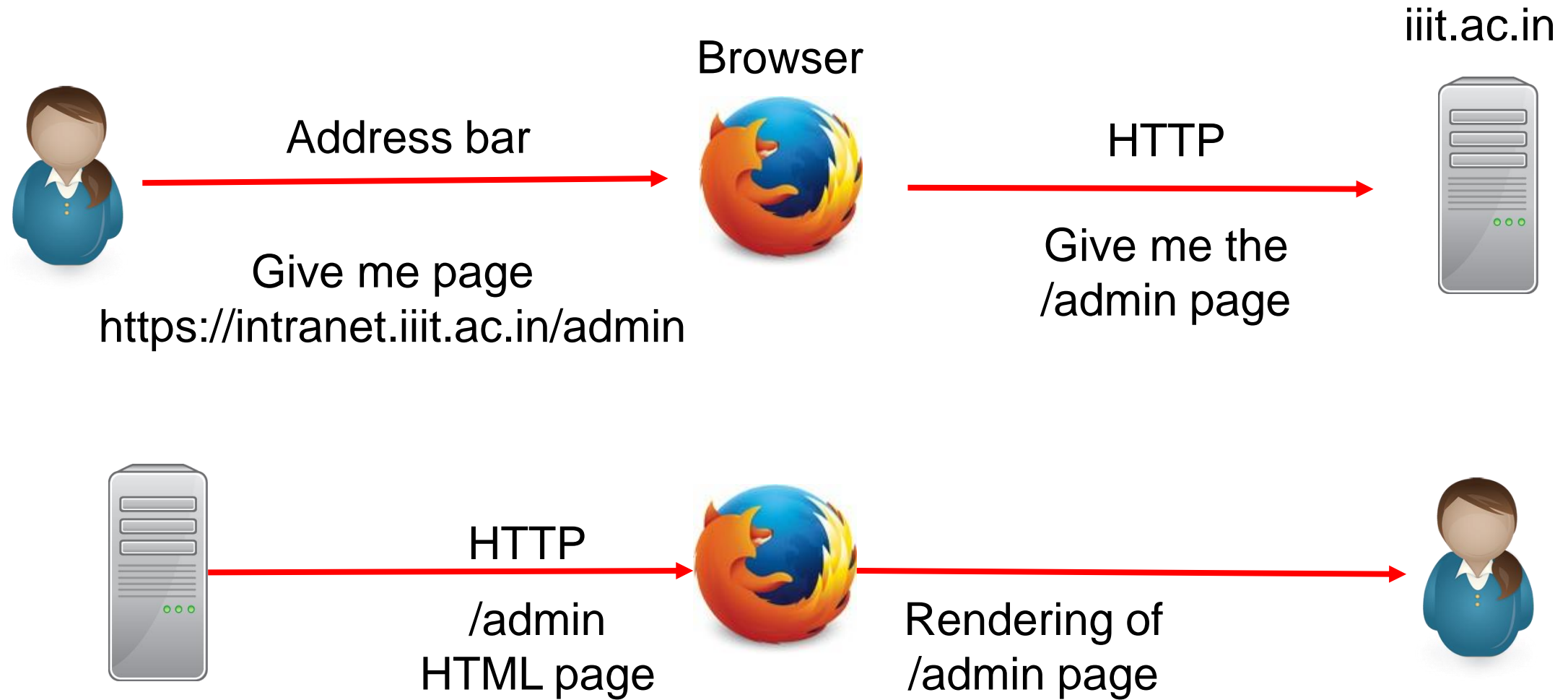


Instruction:  
a paragraph  
with bold text



Displays a paragraph  
with bold text

# Web Page



# HTML – Hypertext Markup Language

- Hyper Text Markup Language -lingua franca for publishing hypertext
- A simple text format to create web pages
- Specify structure, presentation and content of a web page
- Define tags <html> <body> <head>....etc
- Allow to embed other scripting languages to manipulate design layout, text and graphics
- Platform independent
- No special software required. They are by default part of Internet browser application like Chrome, IE, Safari, Mozilla FireFox

# Topic 1: HTML

## What is HTML?

HTML is a language for describing web pages.

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is a **markup** language
- A markup language is a set of markup **tags**
- The tags **describe** document content
- HTML documents contain HTML **tags** and plain **text**
- HTML documents are also called **web pages**

## HTML Tags

HTML markup tags are usually called HTML tags

- HTML tags are keywords (tag names) surrounded by **angle brackets** like `<html>`
- HTML tags normally **come in pairs** like `<strong>` and `</strong>`
  - The first tag in a pair is the **start tag**, the second tag is the **end tag**
  - The end tag is written like the start tag, with a **forward slash** before the tag name
  - Start and end tags are also called **opening tags** and **closing tags**
  - There is often text inside the tags:  
`<tagname>content</tagname>` (i.e. `<em> content</em>`)
- Certain HTML tags can also appear alone, like `<img>`

## HTML Elements

"HTML tags" and "HTML elements" are often used to describe the same thing.

But strictly speaking, an HTML element is everything between the start tag and the end tag, including the tags:

`<p>`This is a paragraph.`</p>`

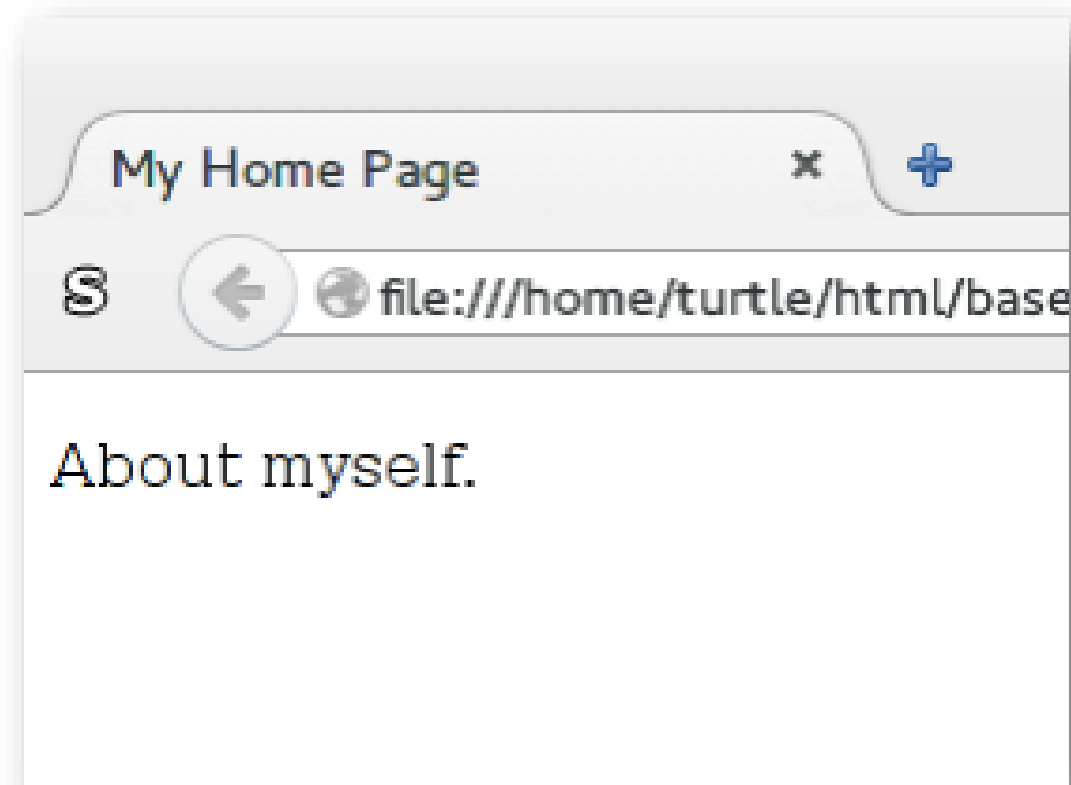
## Web Browsers

The purpose of a web browser (such as Google Chrome, Internet Explorer, Firefox, Safari) is to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page:

You Write:

```
<!DOCTYPE html>
<html>
  <head>
    <title> My Home Page
    </title>
  </head>
  <body>
    <p>About myself.</p>
  </body>
</html>
```

You See:



# Basic Structure

<code>&lt;!DOCTYPE html&gt;</code>	←	HTML declaration
<code>&lt;html&gt;</code>	←	HTML opening tag
<code>&lt;head&gt;</code>	←	Page header opening tag
<code>&lt;title&gt; My Home Page</code>	←	Page title opening tag
<code>&lt;/title&gt;</code>	←	Page title closing tag
<code>&lt;/head&gt;</code>	←	Page header closing tag
<code>&lt;body&gt;</code>	←	Body opening tag
<code>&lt;p&gt;About myself.&lt;/p&gt;</code>	←	Content (inside body)
<code>&lt;/body&gt;</code>	←	Body closing tag
<code>&lt;/html&gt;</code>	←	HTML closing tag

# Heading

You  
Write:-----

`<h1>`This is a  
heading`</h1>`

`<h2>`This is a  
heading`</h2>`

`<h3>`This is a  
heading`</h3>`

`<h4>`This is a  
heading`</h4>`

`<h5>`This is a  
heading`</h5>`

`<h6>`This is a  
heading`</h6>`

You  
See:-----

**This is a heading**

**This is a heading**

**This is a heading**

**This is a heading**

**This is a heading**

**This is a heading**

# Unordered List

You  
Write:

```
<ul>
  <li>Red</li>
  <li>Green</li>
  <li>Blue</li>
</ul>
```

You  
See:

- Red
- Green
- Blue



# Ordered List

You  
Write:

```
<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

You  
See:

1. Coffee
2. Tea
3. Milk

# Link to Another Web Page

You  
Write:

```
<a href="https://fsf.org/">  
Free Software Foundation </a>
```

You  
See:

[Free Software Foundation](https://fsf.org/)

# Image

You  
Write:

```

```

You  
See:



# Comments

You  
Write:

```
<!-- This is a  
comment -->  
<p> This is a  
paragraph </p>
```

You  
See:

This is a paragraph

# Table

You  
Write:

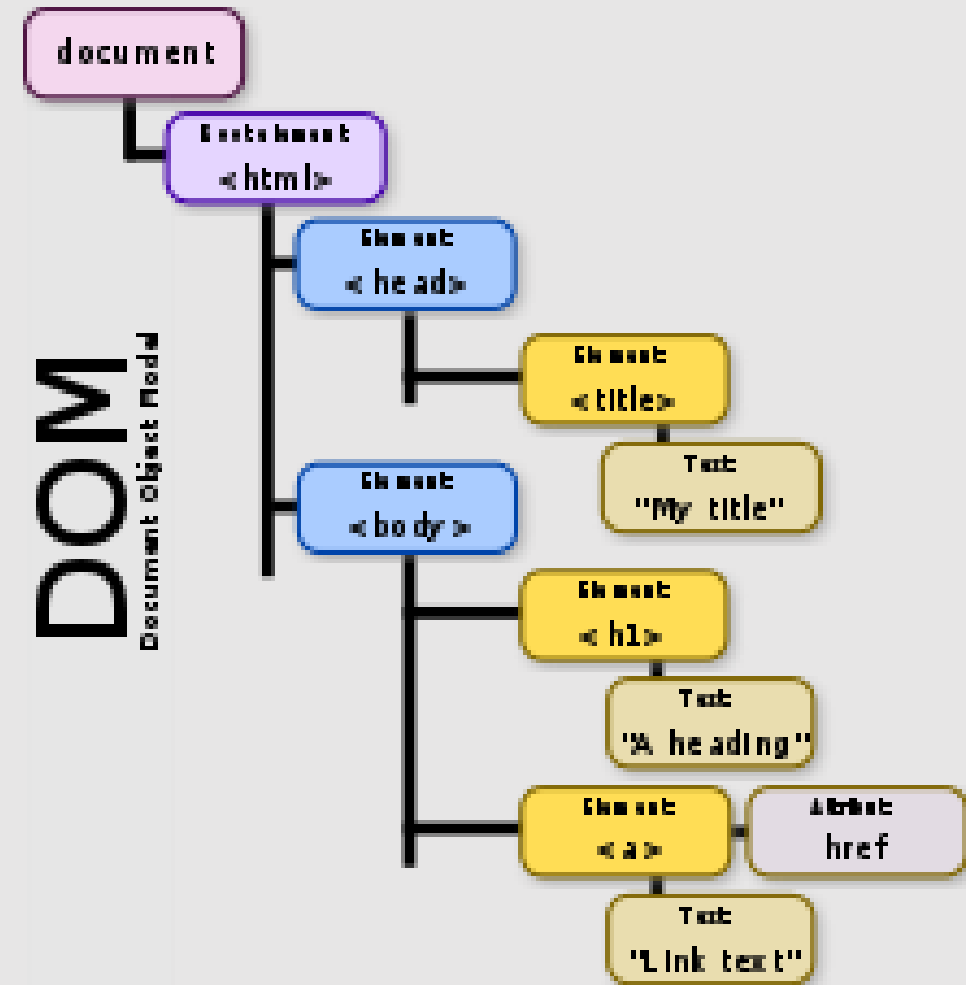
```
<table border="1">
  <tr>
    <th>First name</th>
    <th>Last name</th>
  </tr>
  <tr>
    <td>James</td>
    <td>Kirk</td>
  </tr>
  <tr>
    <td>Spock</td>
    <td></td>
  </tr>
</table>
```

You  
See:

First Name	Last Name
James	Kirk
Spock	

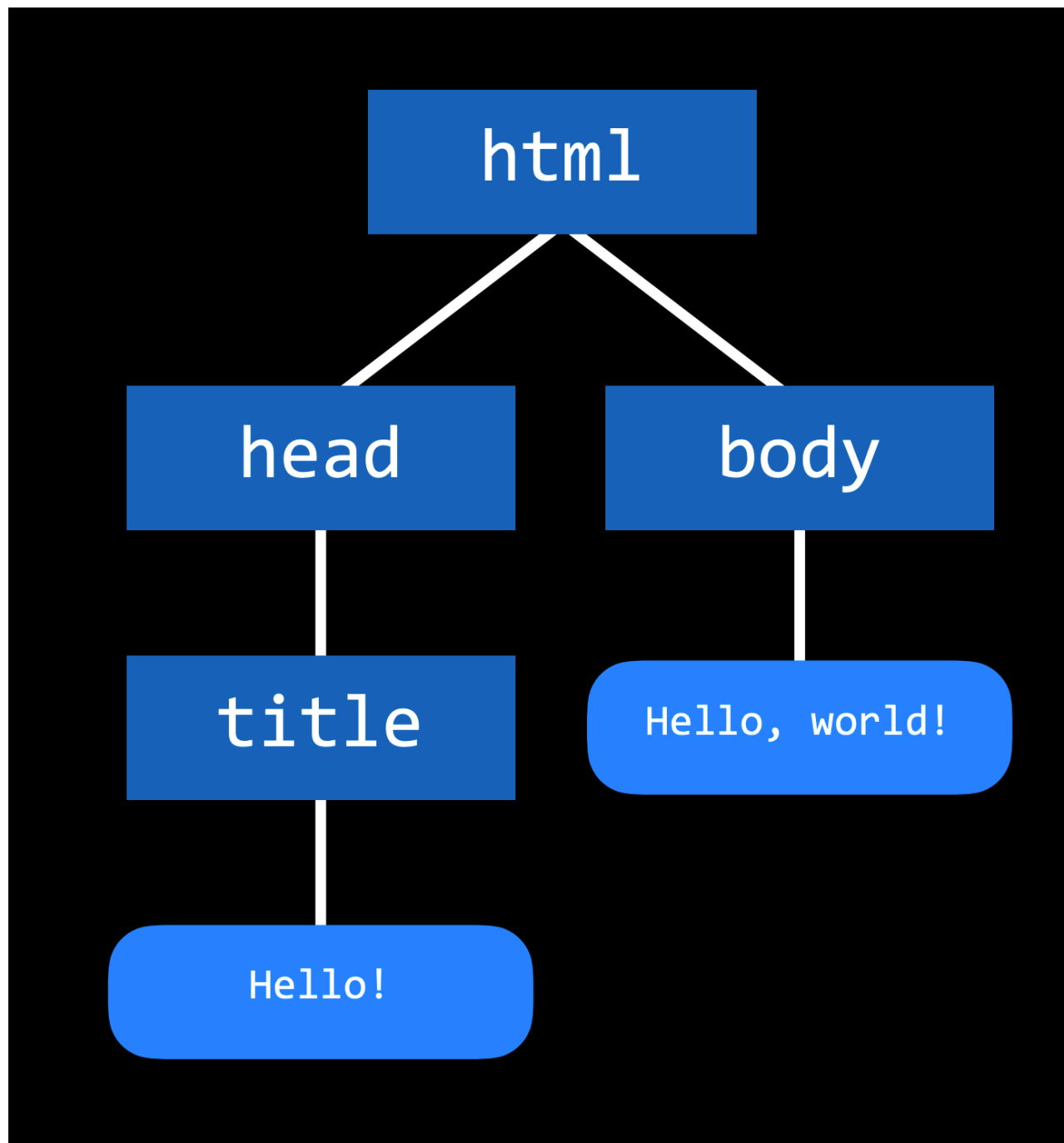
# DOM ( Document Object Modeling)

- The Document Object Model (DOM) is a programming API for HTML and XML documents.
- It defines the logical structure of documents and the way a document is accessed and manipulated.
- HTML, XML, XSLT, JSTL document as a tree structure wherein each node is an object representing a part of the document.



- Finally, we've included the text "Hello, world!" in the body, which is the visible part of our page.

## Document Object Model (DOM)



- The DOM is a convenient way of visualizing the way HTML elements relate to each other using a tree-like structure. Above is an example of the DOM layout for the page we just wrote.

## More HTML Elements

- There are many HTML elements you may want to use to customize your page, including headings, lists, and bolded sections. In this next example, we'll see a few of

## Topic 2: CSS

Here are screenshots of some sample P1.2's

You are logged in with **ak** [Log out](#)

Site created!

# Brian's Web Analytics ( ` 0 ` )

## Sites listing

URL	Registered Visits			
web.mit.edu/akashyap/www	0	Show	Edit	Destroy

[New Site](#)

## All activity for your sites

Site ID	Visited from	Platform	User
---------	--------------	----------	------

Logged in as akak. [Log out](#)

You have successfully logged into your account.

You have successfully logged into your account.

Look below for your currently tracked sites.

[Back](#) [Add Site](#)

**ID   Pagename   URL   Visits**

The functionality is present in all these pages, but the sites themselves are visually unappealing. (In fact, perhaps good visual design is essential to your site's functionality - e.g. people abandoned MySpace as a social networking tool since they couldn't tolerate looking at the cluttered, eye-soring profiles vs. Facebook's incredibly clean [c. 2007] layout.) Maybe we can find a way to style the pages so that our sites could look... like this!



# Separating Presentation Details

- Separating content and its presentation details is important
- Content is consumed using various media
  - Computer monitor
  - Braille terminal
  - Text to speech system
  - Automated systems
- Focus on content during development
- Easily improve/change the presentation later
- Saves a lot of effort

# CSS Properties

- Color
- Background and Borders
- Basic Box
- Flexible Box
- Text
- Text Decoration
- Fonts
- Writing Modes
- Table
- Lists and Counters
- Animation
- Transform
- Transition
- Basic User Interface
- Multi-column
- Paged Media
- Generated Content
- Filter Effects
- Image/Replaced Content
- Masking
- Speech

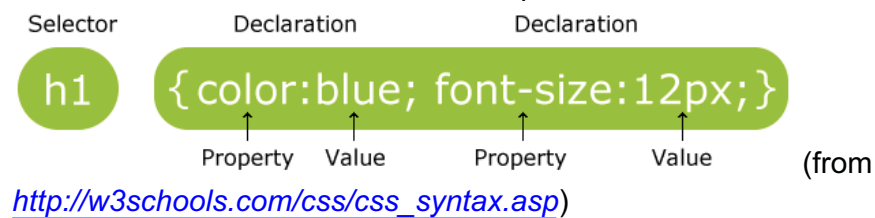
# CSS Syntax

Selector                      Property                      Value

```
body {  
    color: red;  
    font-weight: bold;  
}  
  
h1 {  
    font-size: 20px;  
}
```

Rule

CSS files are a series of zero or more presentation rules. Here's a sample rule:



This rule demands that content between `<h1>` tags be blue and of font-size 12 pixels.

**How does the browser know which HTML elements a rule targets?** Notice that a rule consists of 1 or more selectors. Selectors specify which elements the presentation rule is targeting. In this example, the presentation rule is targeting all `<h1>` HTML elements.

### Most common selectors

<code>htmlElementType</code>	targets all HTML elements of type <code>htmlElementType</code> (e.g. <code>h1</code> , <code>span</code> , <code>div</code> , etc.)
<code>.className</code>	targets elements of class <code>className</code>
<code>#idName</code>	targets one element with id <code>idName</code> (assuming HTML is well-formed - ids should be unique per document!)
<code>*</code>	targets all elements
<code>selector1, selector2</code>	targets all elements specified by <b>either</b> <code>selector1</code> or <code>selector2</code>
<code>selector1 selector2</code>	targets all elements specified by <code>selector2</code> that are children (both direct and indirect) of elements specified by <code>selector1</code>
<code>selector1[attr=value]</code>	targets all elements specified by <code>selector1</code> with a specified attribute, e.g. <code>input[type=password]</code>

Selectors can also be combined! For example,  
`img.classA.classB#idName[alt="Profile image"]`

There are a lot more ways to select elements; for a complete list check out the W3C specs:  
<http://www.w3.org/TR/selectors/>.

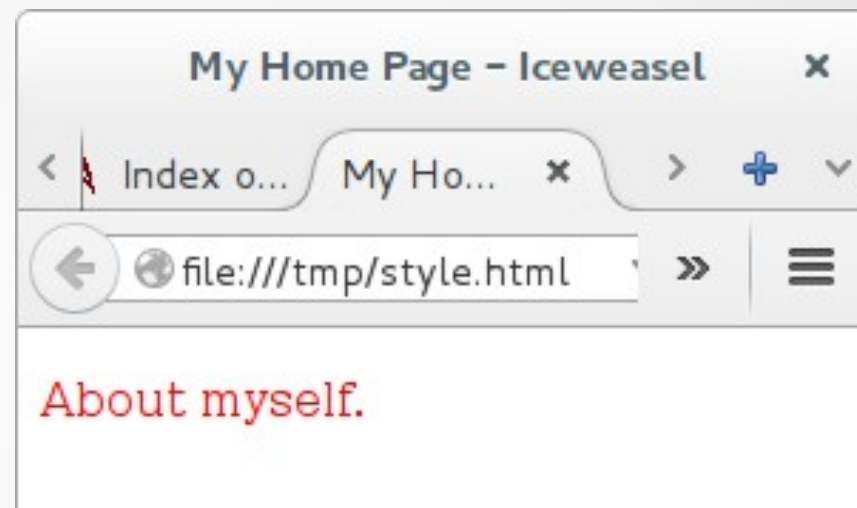
**How does the browser know how to display particular selected elements?** Notice that a rule also consists of declarations, which are simply property:value pairs. The browser renders targeted elements based on these property:value pairs. There are lots of properties - no need to memorize all of them. Check a reference like the Mozilla Developer Network (MDN):

# In-line CSS (Bad)

You Write:

```
<!DOCTYPE html>
<html>
  <head>
    <title> My Home Page
  </title>
</head>
<body style="color: red;">
  <p>About myself.</p>
</body>
</html>
```

You See:

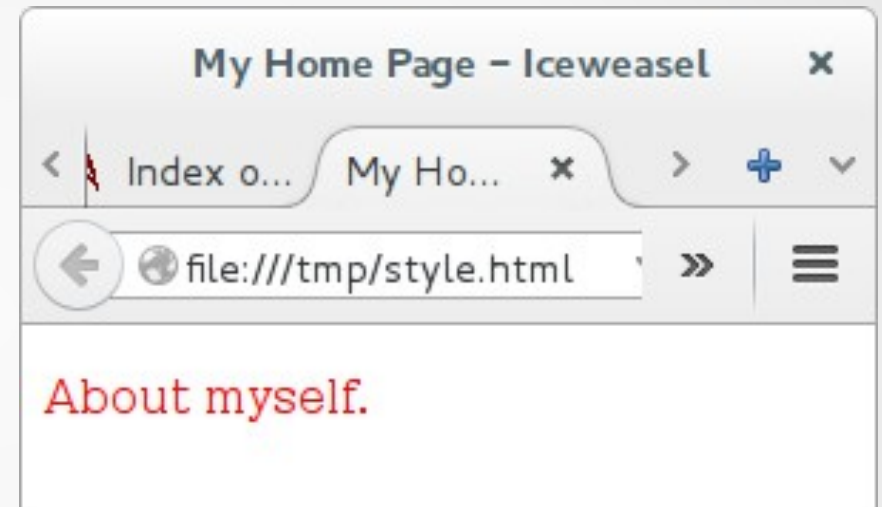


# Internal CSS (Better)

You Write:

```
<head>
  <title> My Home </title>
  <style>
    body {
      color: red;
    }
  </style>
</head>
```

You See:



# External CSS (Good)

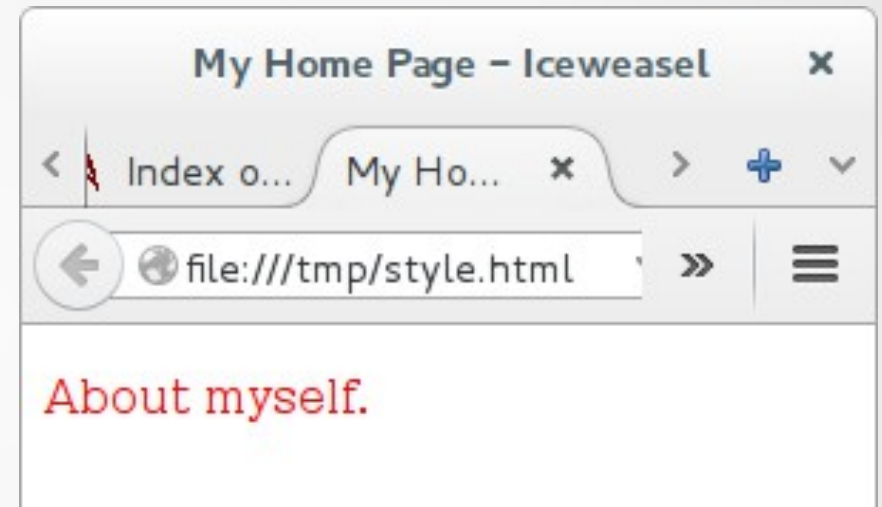
In HTML You Write:

```
<head>
  <title> My Home </title>
  <link
    rel="stylesheet"
    href="style.css">
</head>
```

In *style.css* You Write:

```
body {
  color: red;
}
```

You See:



# CSS Comments

```
/* This is a comment */
```

```
body {  
    color: red;  
    font-weight: bold;  
}  
  
h1 {  
    font-size: 20px;  
}
```



```
td, th {  
    border: 1px solid black;  
    padding: 2px;  
}
```

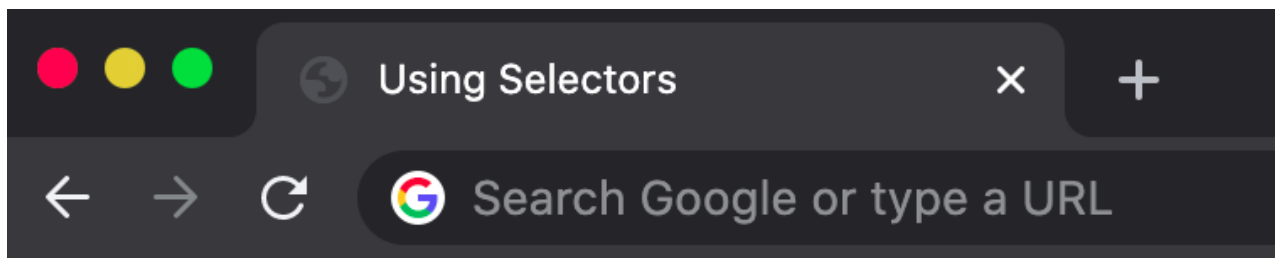
- This is a good introduction into what are known as **CSS selectors** ([https://www.w3schools.com/cssref/css\\_selectors.asp](https://www.w3schools.com/cssref/css_selectors.asp)). There are many ways to determine which HTML elements you are styling, some of which we'll mention here:
  - **element type**: this is what we've been doing so far: styling all elements of the same type.
  - **id**: Another option is to give our HTML elements an id like so: `<h1 id="first-header">Hello!</h1>` and then applying styling using `#first-header{...}` using the hashtag to show that we're searching by id. Importantly, no two elements can have the same id, and no element can have more than one id.
  - **class**: This is similar to id, but a class can be shared by more than one element, and a single element can have more than one class. We add classes to an HTML element like this: `<h1 class="page-text muted">Hello!</h1>` (note that we just added two classes to the element: `page-text` and `muted`). We then style based on class using a period instead of a hashtag: `.muted {...}`
- Now, we also have to deal with the problem of potentially conflicting CSS. What happens when a header should be red based on its class but blue based on its id? CSS has a specificity order that goes:
  1. In-line styling
  2. id
  3. class
  4. element type
- In addition to the comma for multiple selectors, there are several other ways to specify which elements you would like to style. This table from lecture provides a few, and we'll go through a few examples below:

<code>a, b</code>	Multiple Element Selector
<code>a b</code>	Descendant Selector
<code>a &gt; b</code>	Child Selector
<code>a + b</code>	Adjacent Sibling Selector
<code>[a=b]</code>	Attribute Selector
<code>a:b</code>	Pseudoclass Selector
<code>a::b</code>	Pseudoelement Selector

**Descendant Selector:** Here, we use the descendant selector to only apply styling to list items found within an unordered list:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Using Selectors</title>
    <style>
      ul li {
        color: blue;
      }
    </style>
  </head>
  <body>
    <ol>
      <li>foo</li>
      <li> bar
        <ul>
          <li>hello</li>
          <li>goodbye</li>
          <li>hello</li>
        </ul>
      </li>
      <li>baz</li>
    </ol>

  </body>
</html>
```



1. foo
2. bar
  - hello
  - goodbye
  - hello
3. baz

**Attributes as Selectors:** We can also narrow down our selection based on the attributes we assign to HTML elements using brackets. For example, in the following list of links, we choose to only make the link to Amazon red:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Using Selectors</title>
    <style>
      a[href="https://www.amazon.com/"] {
        color: red;
      }
    </style>
  </head>
  <body>
    <ol>
      <li><a href="https://www.google.com/">Google</a></li>
      <li><a href="https://www.amazon.com/">Amazon</a> </li>
      <li><a href="https://www.facebook.com/">Facebook</a></li>
    </ol>

    </body>
  </html>
```

# CSS Selectors – Tag Name

- Select all elements in a page matching a HTML tag name

```
<body>  
  <h1>Section 1</h1>  
  
  <p>About myself.</p>  
  <h2>Sub section 1.1</h2>  
  <p>My interests are...</p>  
</body>
```

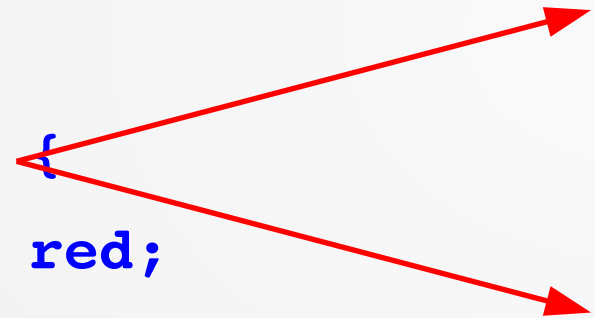
**p {**  
 **color: red;**  
**}**

The diagram illustrates the 'Tag Name' selector. On the left, a CSS rule is shown: `p { color: red; }`. Two red arrows originate from the `p` selector. One arrow points to the first paragraph in the HTML code: `<p>About myself.</p>`. The other arrow points to the second paragraph: `<p>My interests are...</p>`. This demonstrates that the `p` selector targets all elements with the tag name `p`.

# CSS Selectors – Class Name

- Select all elements in a page have a “class” attribute
- An element can have multiple classes

```
.heading {  
    color: red;  
}
```




The diagram consists of two red arrows originating from the opening curly brace of the `.heading {` CSS rule. One arrow points to the `<h1 class="heading">` tag in the HTML code block, and the other points to the `<h2 class="heading">` tag. This illustrates that the `.heading` selector applies to all elements with the `heading` class, regardless of their tag type.

```
<body>  
  
    <h1 class="heading">  
        Section 1</h1>  
    <p>About myself.</p>  
  
    <h2 class="heading">  
        Sub section 1.1</h2>  
    <p>My interests are...</p>  
  
</body>
```

# CSS Selectors – ID

- Select all elements in a page a ID attribute
- ID must be unique in the entire document

```
#first-p {  
    color: red;  
}  
  
<body>  
    <h1>Section 1</h1>  
    <p id="first-p">  
        About myself.</p>  
    <h2>Sub section 1.1</h2>  
    <p>My interests are...</p>  
</body>
```



# Grouping Selectors

```
h1 { color: red; }
```

```
h2 { color: red; }
```

is same as:

```
h1, h2 { color: red; }
```

# Grouping Selectors

```
p { color: red; }
```

```
p { background: black; }
```

is same as:

```
p { color: red; background: black; }
```



# Combining Selectors

```
p.even { color: red; }
```

means all elements with “p” tag **and** “even” class, and similarly

```
p#first { color: red; }
```

means all elements with “p” tag **and** “first” as ID attribute

# Other Selectors

- [P]
  - Select elements containing attribute P
- [P=Q]
  - Select elements containing attribute P and value Q
- :hover :link :visited
  - Select elements with a particular state
- :before :after
  - Select pseudo elements before/after an element
- And more...

# Pseudo Selectors

defines a special state of an element or a phantom state that can be targeted with CSS.

Can be psuedo-class or a pseudo-element

**Pseudoclass** : style the selected elements ***only*** when they are in certain state

Ex: only when element is hovered over by the mouse pointer, or a checkbox when it is disabled or checked, or an input is required,

**Pseudoelement**: A CSS pseudo-element is used to style specified parts of an element.

- Style the first letter, or line, of an element, Insert content before, or after, the content of an element

Some pseudoclasses:

- `:default`
- `:fullscreen`
- `:focus`
- `:hover`
- `:invalid`
- `:lang()`
- `:link`
- `:not()`
- `:optional`
- `:out-of-range`
- `:read-only`
- `:required`
- `:target`
- `:valid`
- `:visited`
- `.....`

Some pseudoelements

- `::after`
- `::before`
- `::first-letter`
- `::first-line`
- `::selection`
- `....`