HTTP, HTML, and CSS

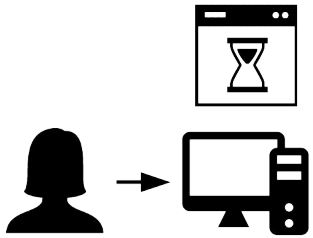
Building Modern Web Applications - VSP2022

Karthik Pattabiraman
Kumseok Jung

Web Applications: What are they?

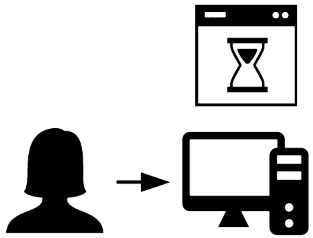
1. **Web Applications**
2. HTTP and HTML
3. CSS

Web Applications: What are they?

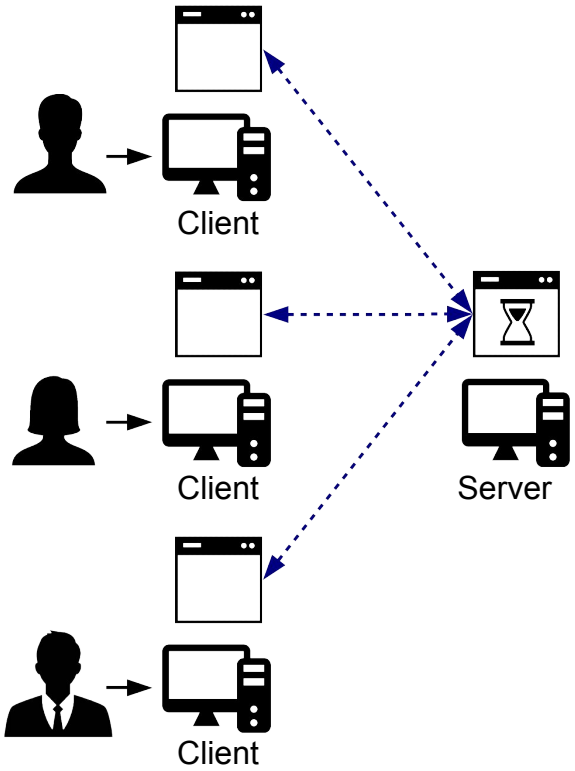


Desktop Application

Web Applications: What are they?



Desktop Application



Web Application

Web Applications: What are they?

- A **client-server software application** in which the client (or UI) runs inside a **web browser**
- What's a **client-server application**?
  - Distributed between 2 machines, client and server
- What's a **web browser**?
  - Software application to view web content

Web Applications: What are they?

Desktop Application	Web Application
Connection to internet **not required**	Connection to internet **required**
Processing on **local device** only	Processing on **local device (client)** and **remote device (server)**
Software **delivered via storage medium**	Software **delivered via network**
Software **installed to the local OS**	Software **interpreted by the browser**
Can run on **local device** only	Can run **from any device**

Web Applications: What are they?

Desktop Application	Web Application
Connection to internet **not required**	Connection to internet **required**
Processing on **local device** only	Processing on **local device (client)** and **remote device (server)**
Software **delivered via storage medium**	Software **delivered via network**
Software **installed to the local OS**	Software **interpreted by the browser**
Can run on **local device** only	Can run **from any device**

HTTP (HyperText Transfer Protocol) and HTML (HyperText Markup Language)

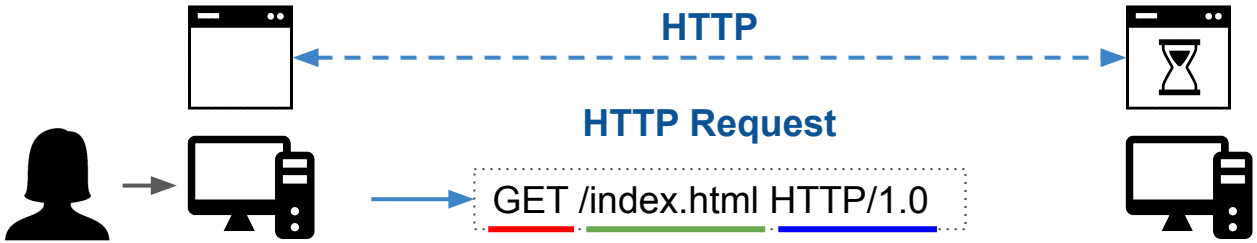
1. Web Applications
2. **HTTP and HTML**
3. CSS

HTTP (HyperText Transfer Protocol)

- Application layer protocol for exchanging HyperText documents (and others)
  - A **standard** defining how web client and web server should exchange information
- HTTP Request
  - Defines the message format a **client** should follow
- HTTP Response
  - Defines the message format a **server** should follow

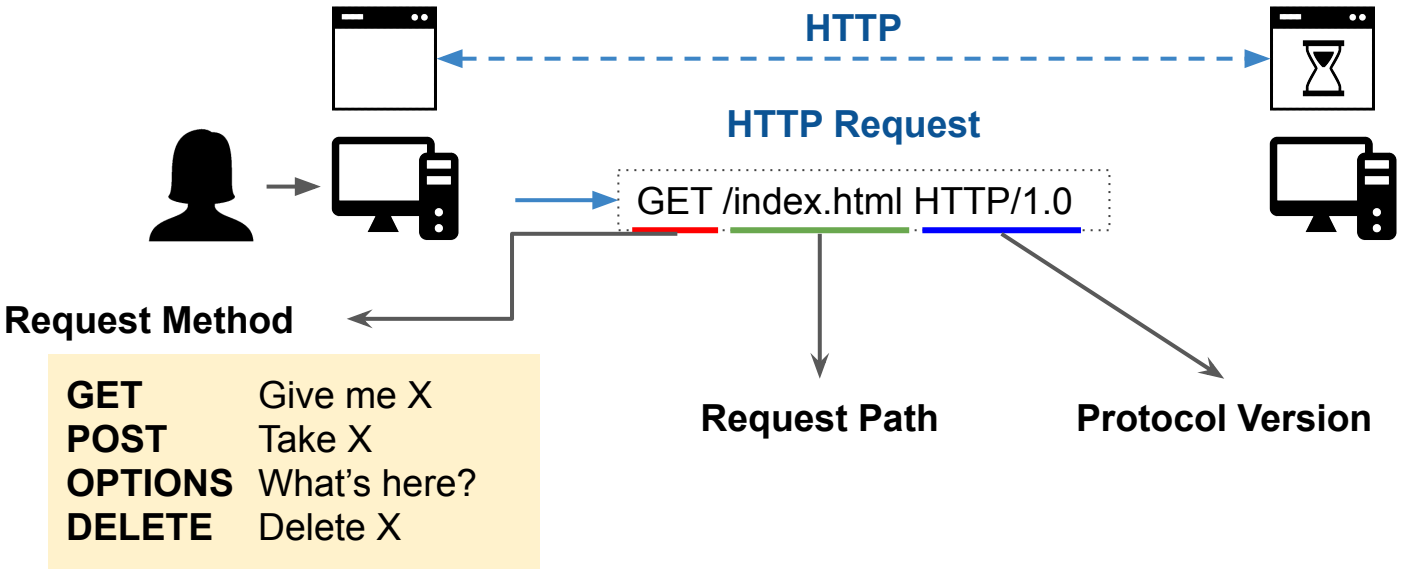
HTTP (HyperText Transfer Protocol)

- HTTP Request
  - Defines the message format a **client** should follow



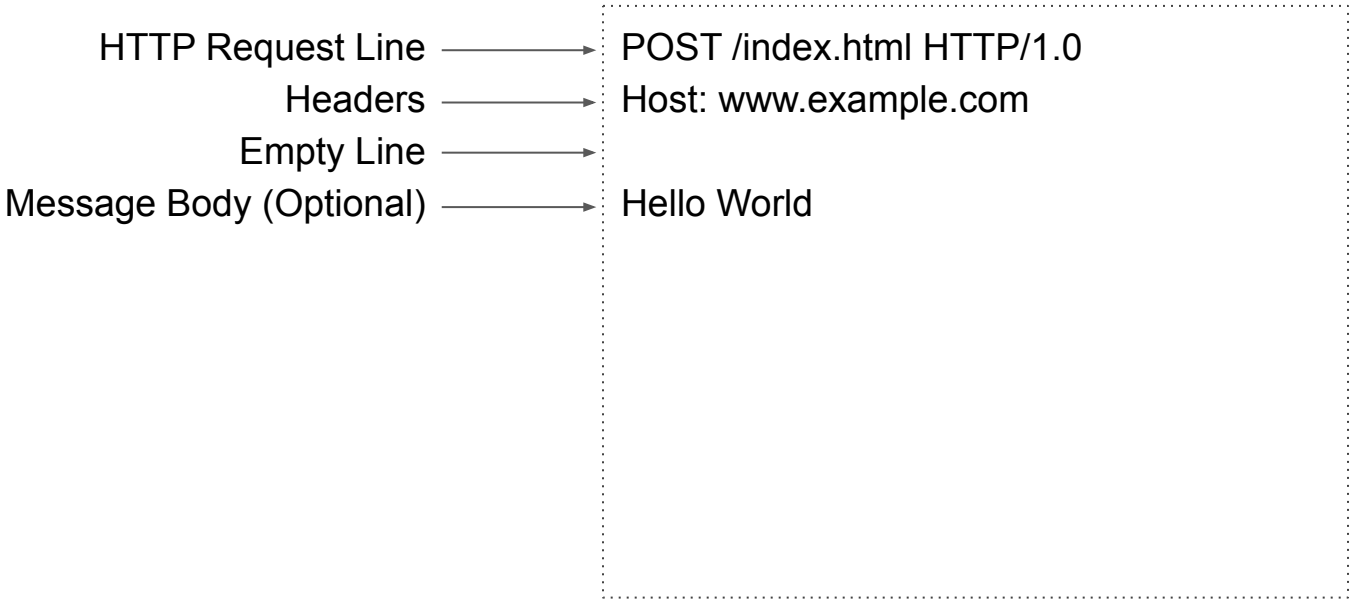
HTTP (HyperText Transfer Protocol)

- HTTP Request
  - Defines the message format a **client** should follow



HTTP (HyperText Transfer Protocol)

- HTTP Request
  - Defines the message format a **client** should follow



HTTP (HyperText Transfer Protocol)

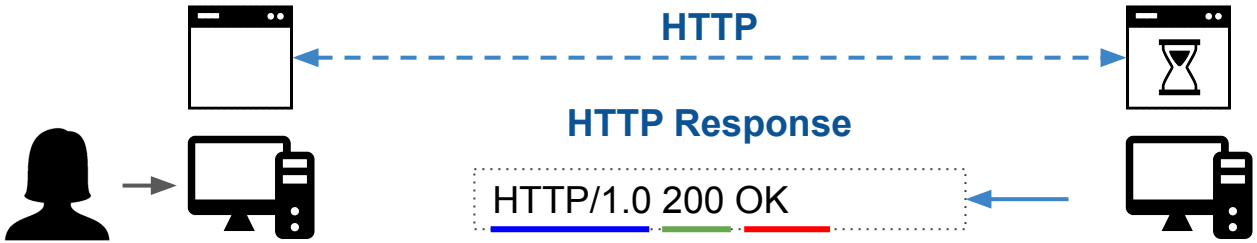
- HTTP Request
  - Defines the message format a **client** should follow

Request Method:

- **GET**
- **HEAD**
- **POST**
- **PUT**
- **DELETE**
- **TRACE**
- **OPTIONS**
- **CONNECT**
- **PATCH**

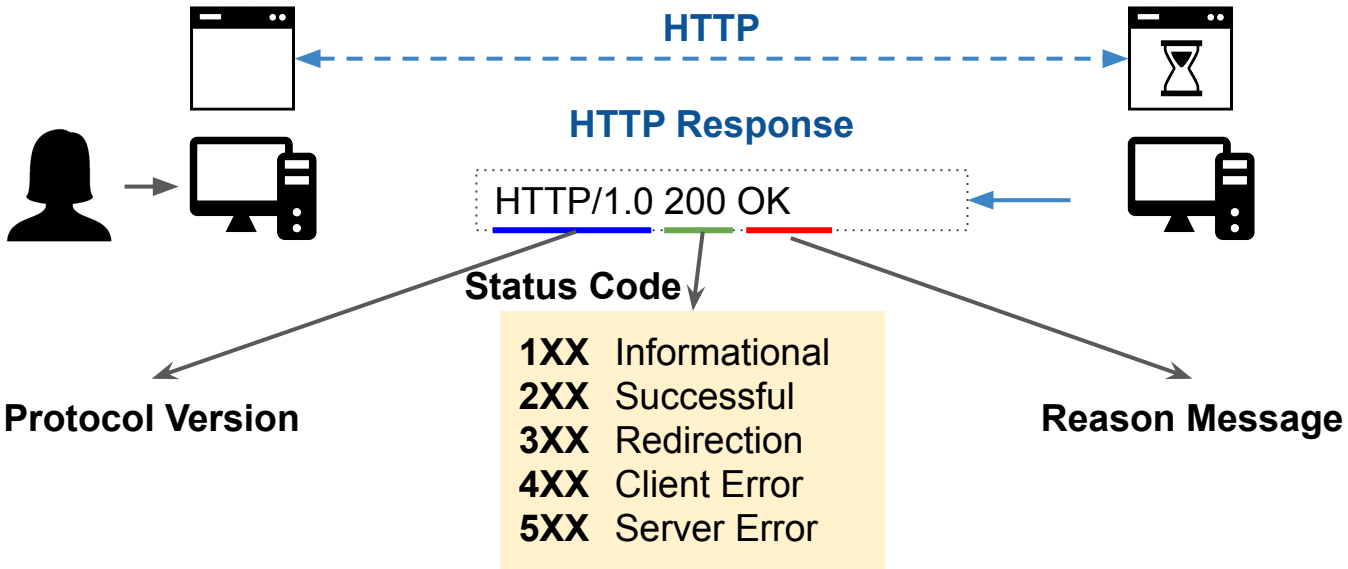
HTTP (HyperText Transfer Protocol)

- HTTP Response
  - Defines the message format a **server** should follow



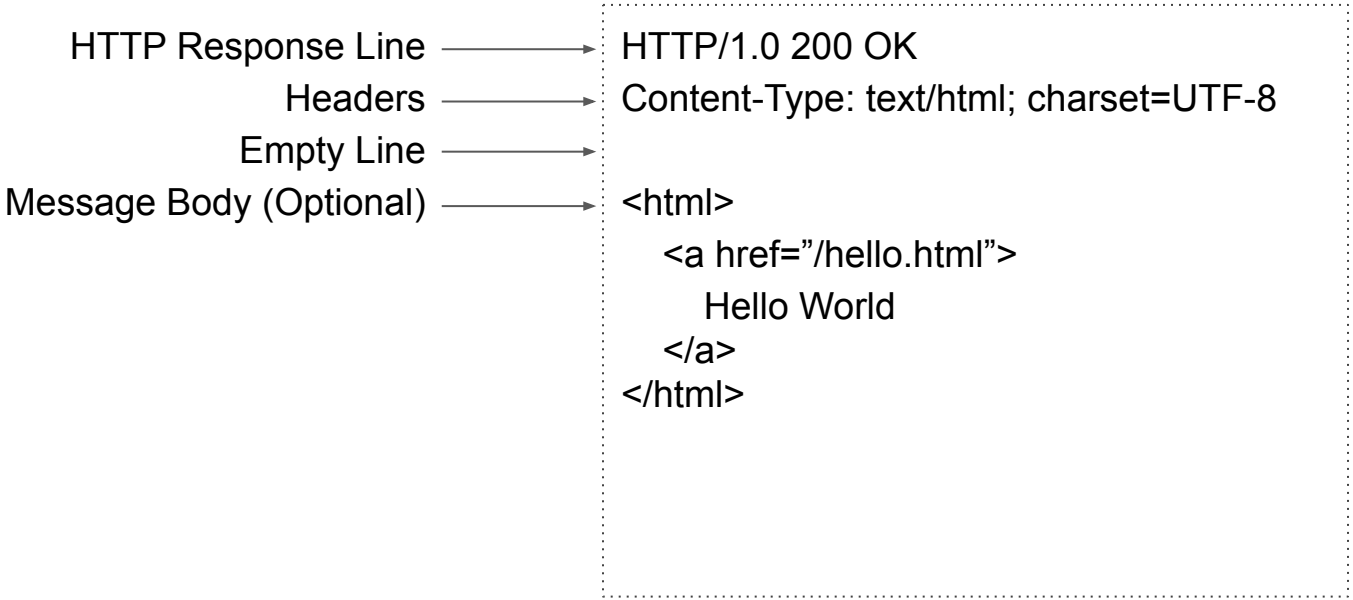
HTTP (HyperText Transfer Protocol)

- HTTP Response
  - Defines the message format a **server** should follow



HTTP (HyperText Transfer Protocol)

- HTTP Response
  - Defines the message format a **server** should follow



HTTP (HyperText Transfer Protocol)

- HTTP Response
  - Defines the message format a **server** should follow

Status Code:

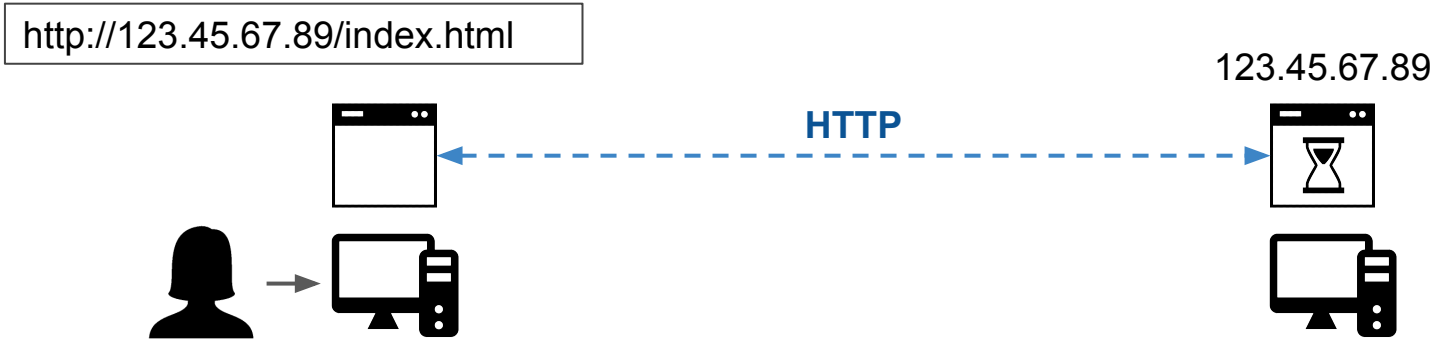
- **1XX**
- **2XX**
- **3XX**
- **4XX**
- **5XX**

HTTP and HTML: The beginnings

Anatomy of a Web Application (1980s)

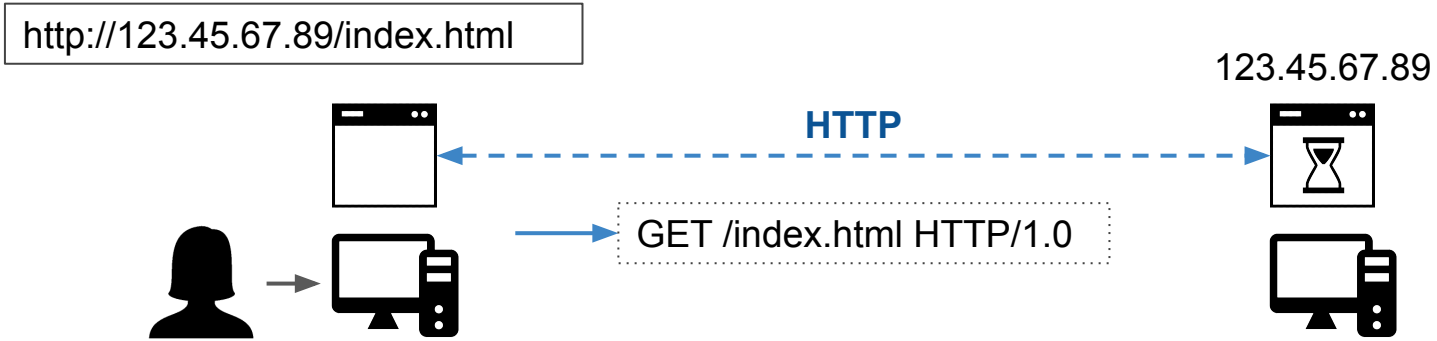
HTTP and HTML: The beginnings

Anatomy of a Web Application (1980s)



HTTP and HTML: The beginnings

Anatomy of a Web Application (1980s)

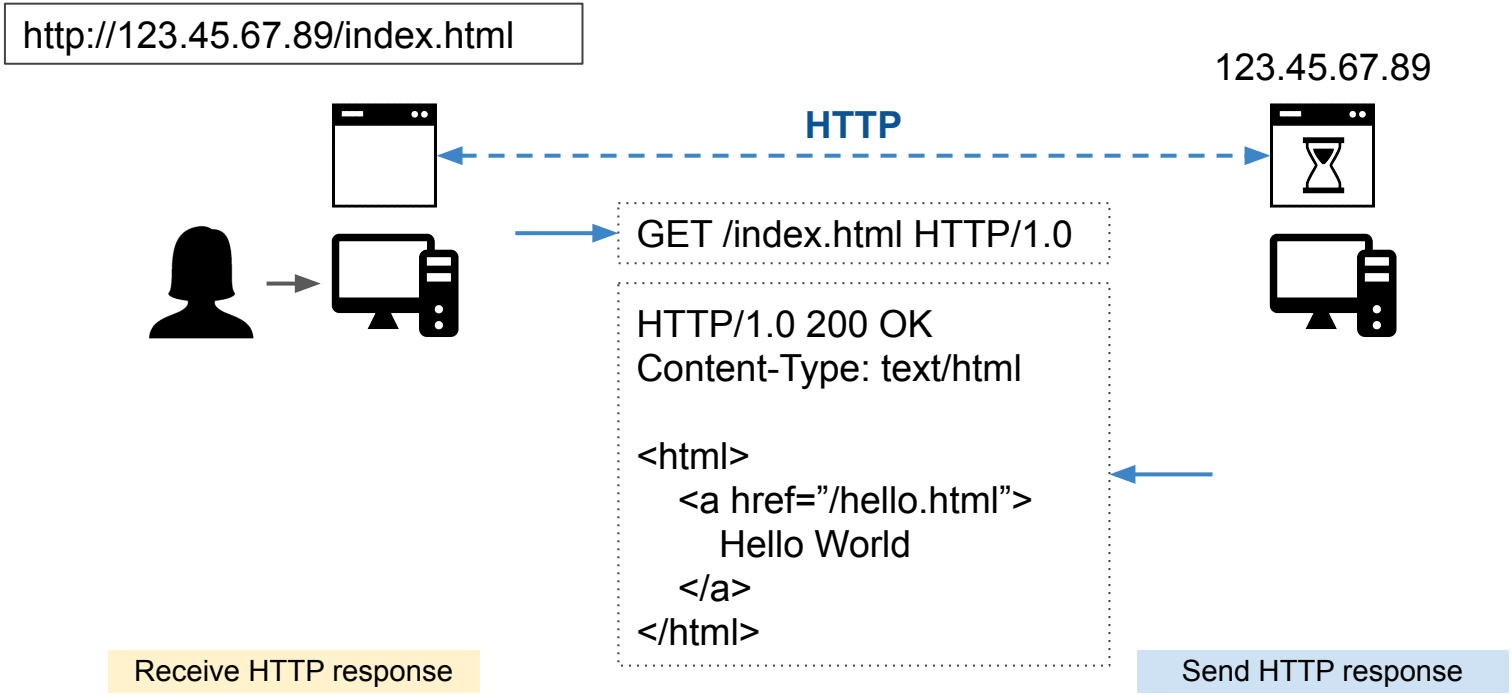


Make HTTP request

Receive HTTP request

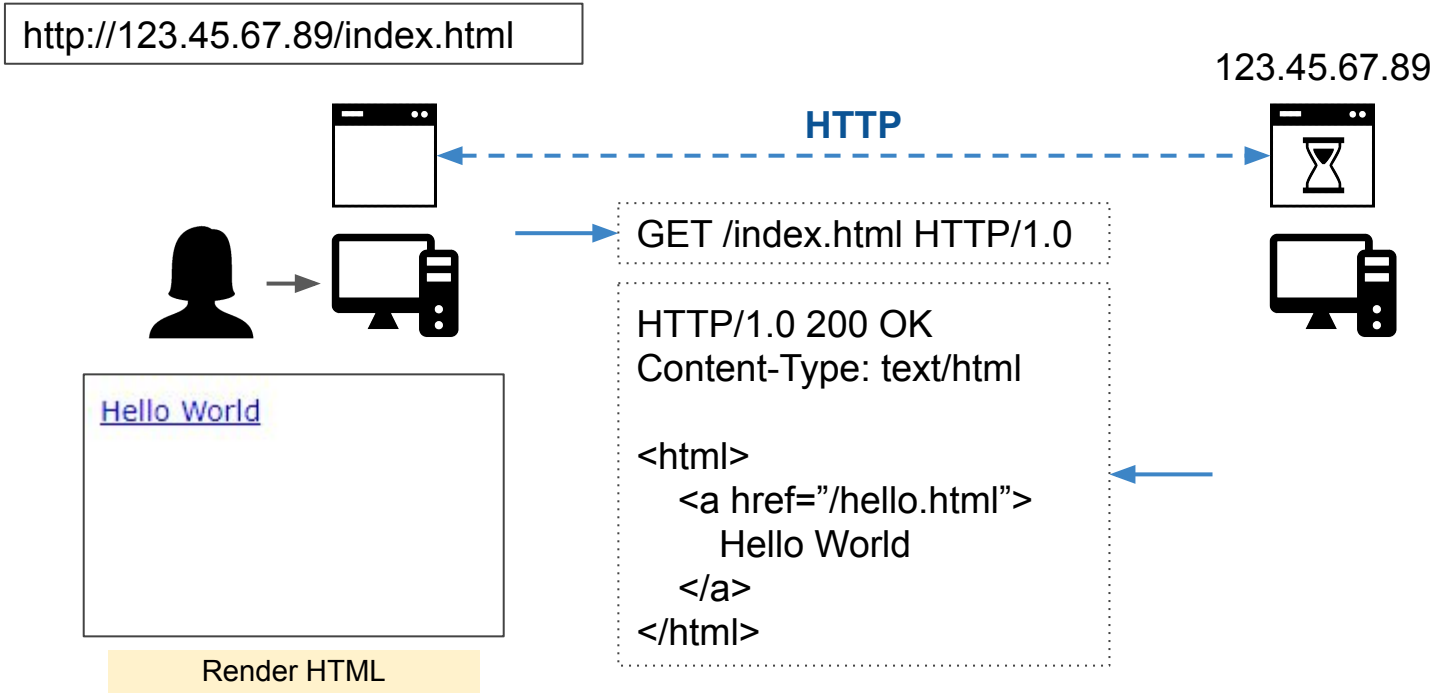
HTTP and HTML: The beginnings

Anatomy of a Web Application (1980s)



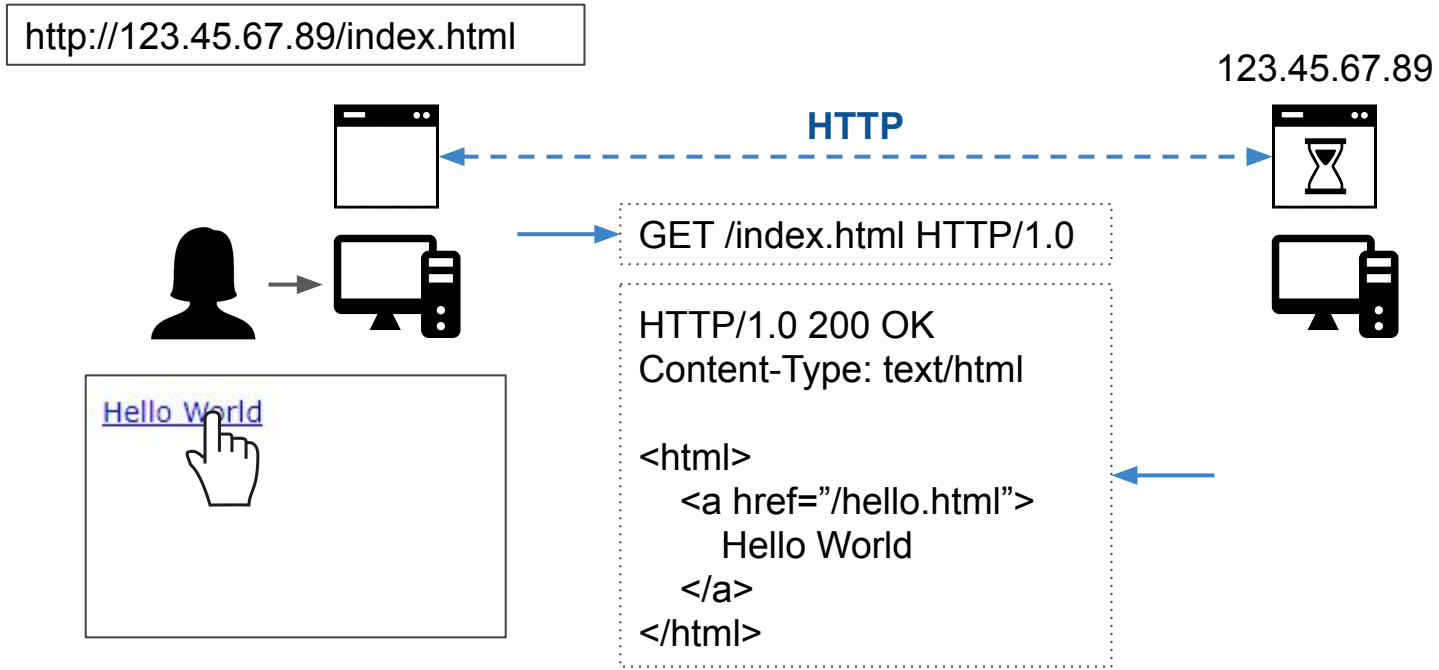
HTTP and HTML: The beginnings

Anatomy of a Web Application (1980s)



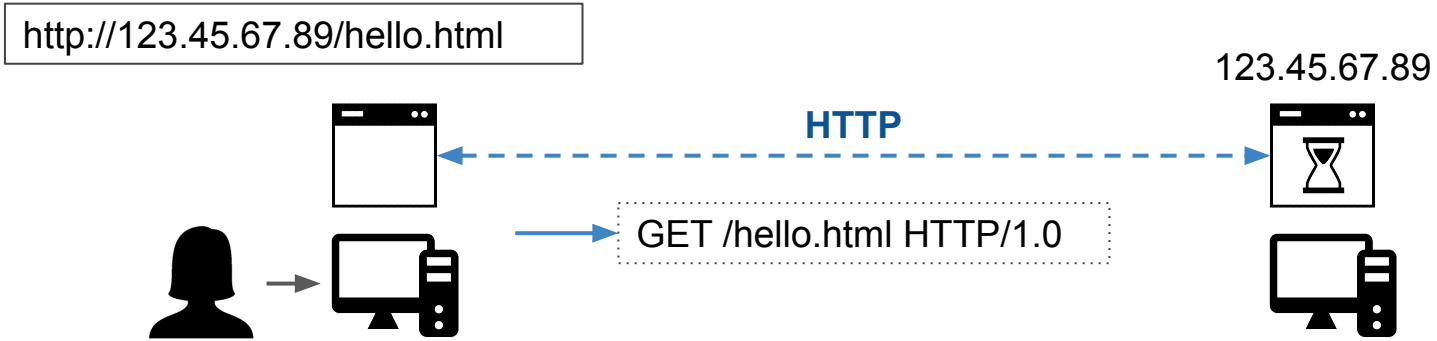
HTTP and HTML: The beginnings

Anatomy of a Web Application (1980s)



HTTP and HTML: The beginnings

Anatomy of a Web Application (1980s)

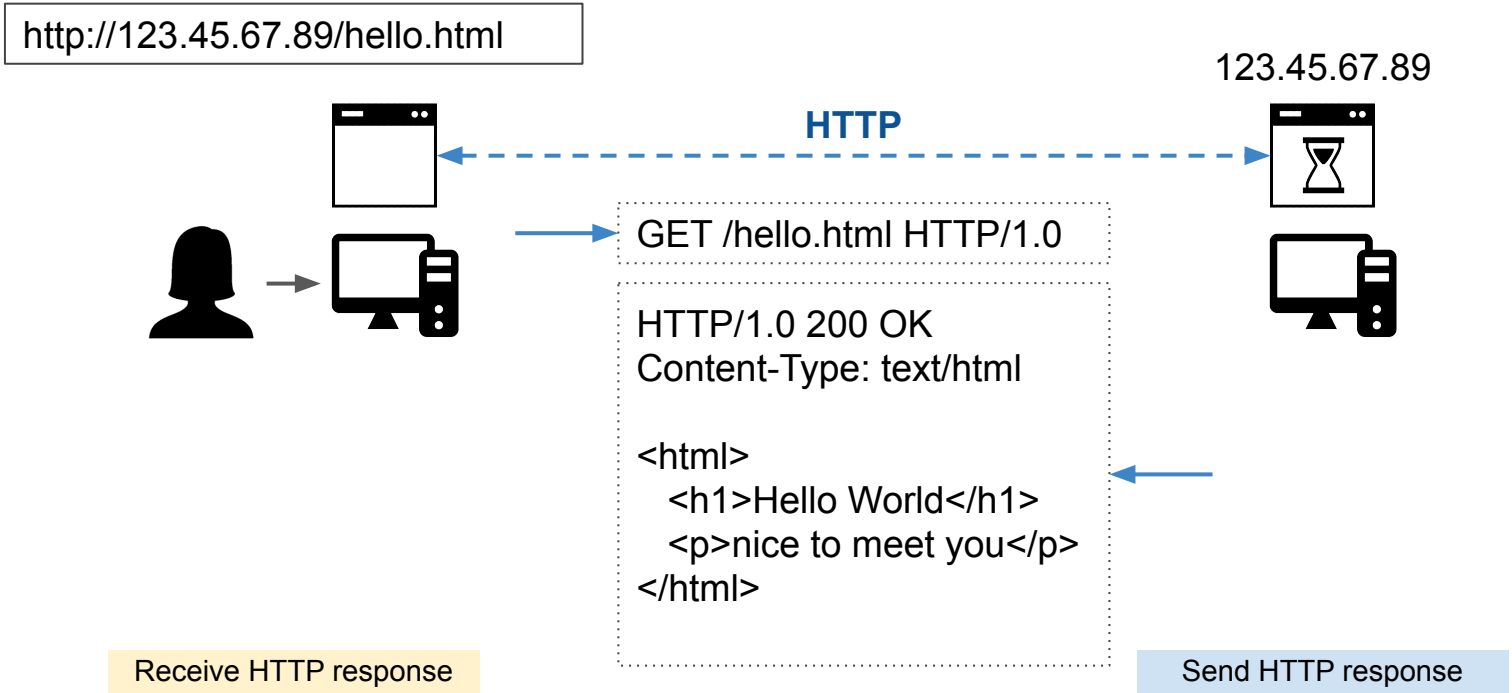


Make HTTP request

Receive HTTP request

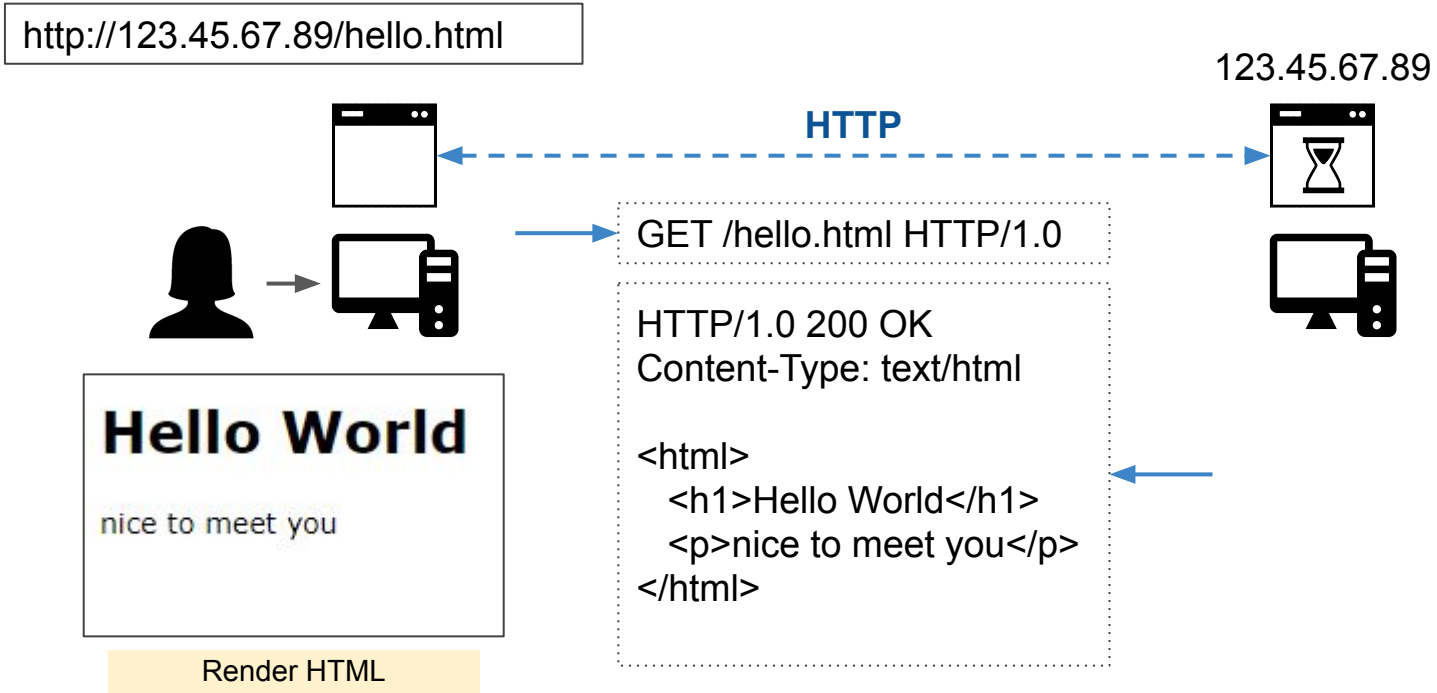
HTTP and HTML: The beginnings

Anatomy of a Web Application (1980s)



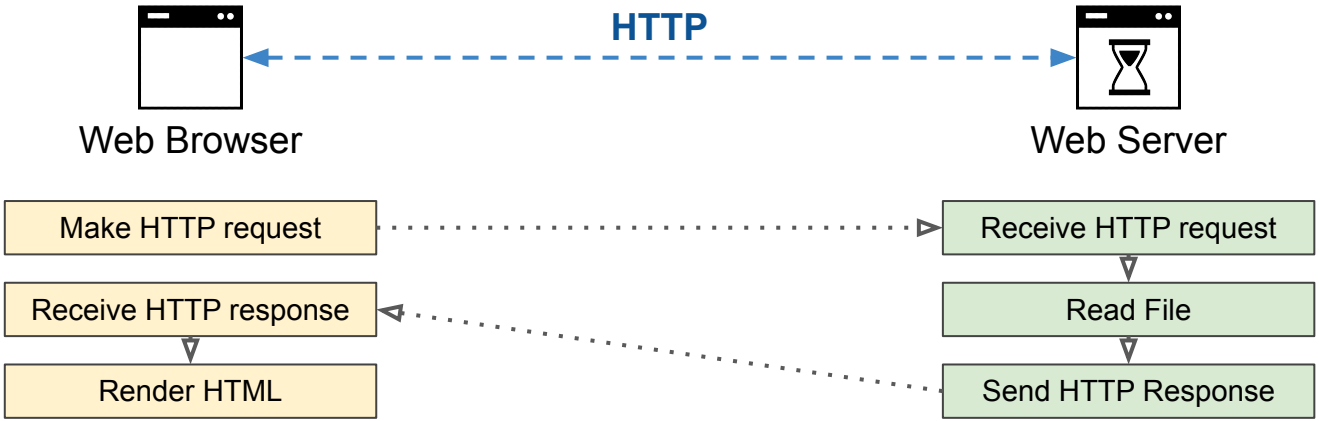
HTTP and HTML: The beginnings

Anatomy of a Web Application (1980s)



HTTP and HTML: The beginnings

Anatomy of a Web Application (1980s)



HTTP and HTML: The beginnings

- **3 essential components** of a web application
  - **Server**: To “serve” the web-page and to send content to the client
  - **Client**: To receive content from the server and display them on the web browser window
  - HTTP connection for client-server interactions
- Everything else is optional

HTTP and HTML: Early Web Applications

- Do not have (much) interactivity on the client
- Getting new content involves navigating to a new webpage or reloading the webpage
  - Typically through a hyperlink
  - Limited information passing from the client (typically in the form of a URL with ‘?’)
- E.g., http://www.google.com?search=“UBC”

HTML (HyperText Markup Language)

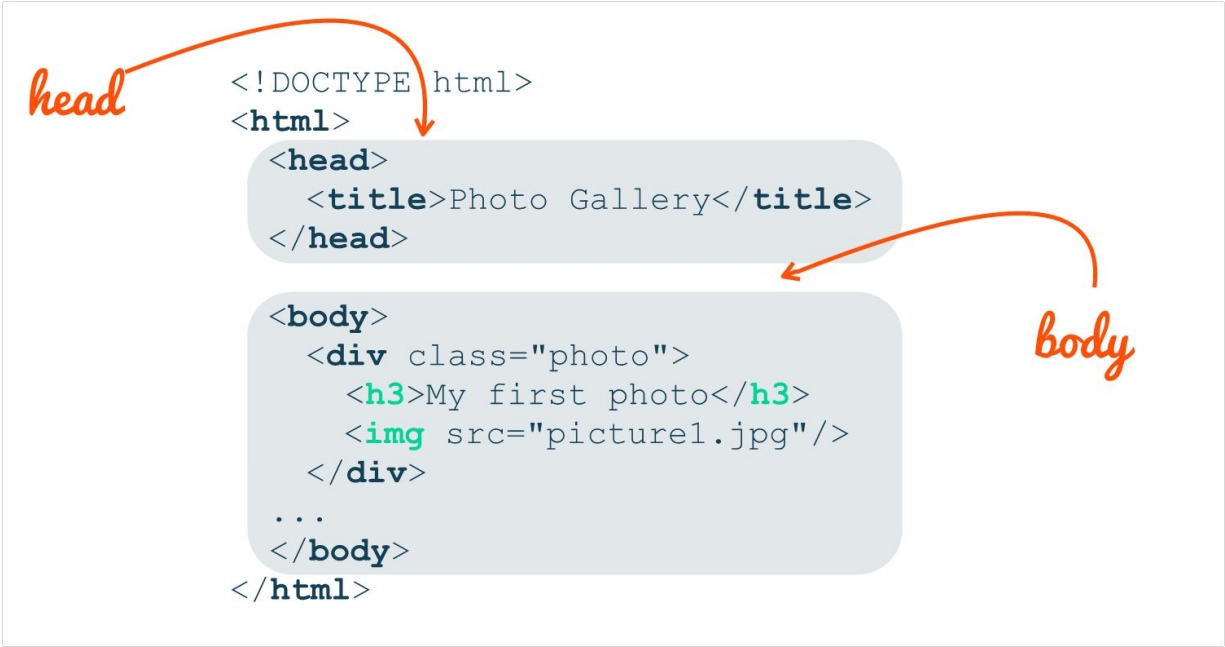
- Hypertext markup language to describe the structure and contents of the initial page
  - Also has pointers to the JavaScript code (e.g., `<script>`)
- Is retrieved by the browser and parsed into a tree called the Document Object Model (DOM)
  - Common way for elements to interact with the page
  - Can be read and modified by the JavaScript code
  - Modifications to the DOM are rendered by browser

HTML (HyperText Markup Language)

- Hierarchical way to organize documents and display them (typically in a web browser)
- Combines semantics (document structure) with presentation (document layout)
- Allows tags to be interspersed with document content e.g., `<head>` - these are not displayed, but are directives to the layout engine

HTML (HyperText Markup Language)

Example:



HTML: <head>

- Is typically NOT displayed by web browser
- Contains metadata to describe the page
  - Title of the webpage: `<title>TITLE</title>`
  - Style of the webpage: `<style>style rules</style>`
  - Link to CSS stylesheets: `<link rel="stylesheet" type="text/css" href="">`
  - For search engines: `<meta name="" content="">`
  - Embed JavaScript: `<script>` Javascript code `</script>`
    OR `<script src="Javascript file"></script>`

HTML: <body>

- Contains the actual contents of the page with HTML tag descriptors for the structure
- Common tags used in HTML

	group elements spanning multiple lines line break before and after
	group elements within a single line
<p>	new paragraph
 	line break

HTML: <body>

`<h1>,…,<h6>`	headings
`<img src=''>`	images
`<a href=''>`	hyperlink

`<table><tr><td>`	tables
`<ul><li>`	unordered list
`<ol><li>`	ordered list
`<form><input>`	forms that take in user input

HTML:

- `<div>`'s are a way to separate different sections of a page and have no meaning by themselves
  - Used to group together semantically related elements in the same portion of the document
  - Allows semantic attributes such as CSS styling or JavaScript code to be applied to div elements
  - div elements can be nested within each other
- Use of `<div>`'s allows easier rendering of pages, and adds semantic meaning to webpages (good)

HTML:

- Div element is used to group menu items
  - id can be used within JavaScript (JS) code to access it – must be unique to the element
  - class is used for indicating type of element – need not be unique, and is used for multiple elements in JS
  - background is a style to apply to all elements in div

```
<div id="menu1" class="Menu" style="background: #d3e7dd;">
    <a href="index.html">Home</a> |
    <a href="about.html">About Us</a> |
    <a href="faq.html">FAQ</a> |
    <a href="contact.html">Contact Us</a>
</div>
```

HTML:

- span is an inline version of div, for separating small chunks of the document without a line break. Cannot contain other div elements in it
- Mostly for applying styling rules to small segments of the webpage without line-breaks

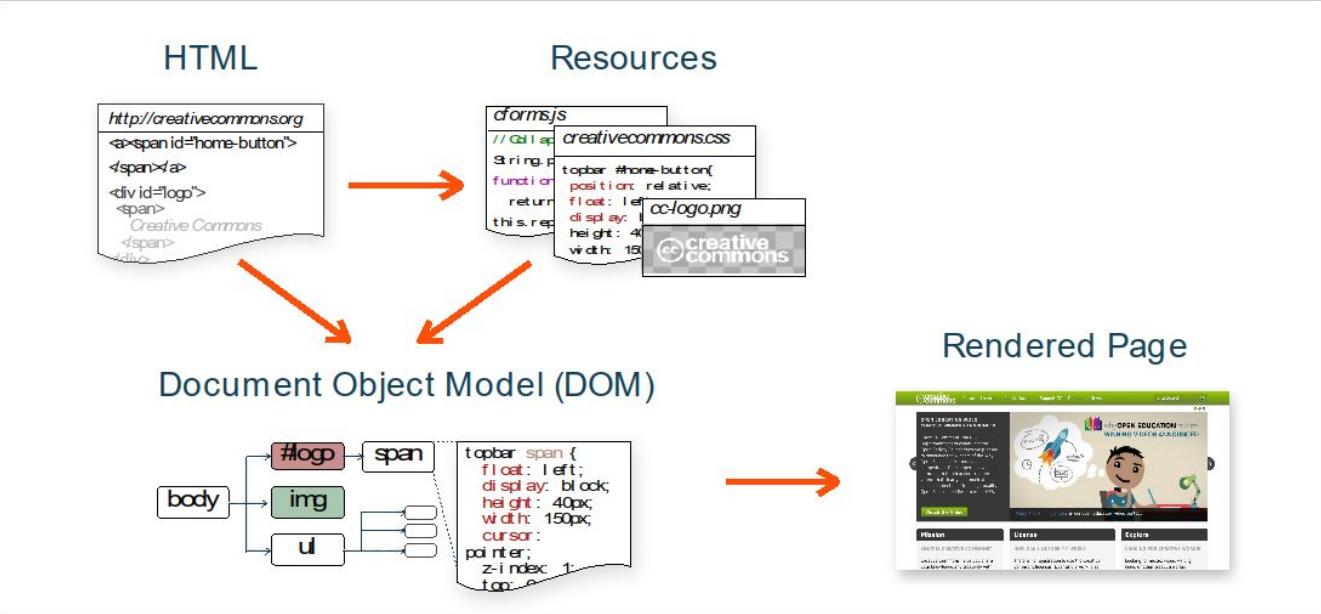
```
1 <div id="menu1" class="Menu" style="background: #d3e7dd;">
2     <a href="index.html">Home</a> |
3     <a href="about.html">About Us</a> |
4     <a href="faq.html">FAQ</a> |
5     <span class="italics"><a href="contact.html">Contact Us</
          a></span>
6 </div>
```

HTML: Why use and ?

- Divs and spans are very useful to break a page into semantically related elements
  - Search engines like Google rely on these to find related information
  - Provide hooks to your webpage from CSS, and especially JavaScript code (more on this later)
  - Makes it easier to render across platforms
- However, overuse of these makes webpages hard to read, and also slower (Especially on mobile)

HTML: Browser's View of HTML - DOM

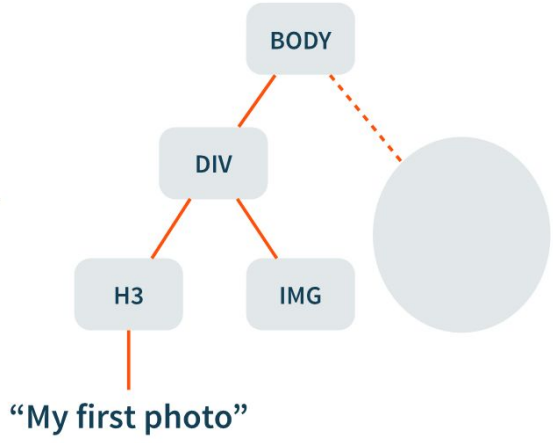
HTML is parsed by the browser into a tree structure - Document Object Model (DOM)



HTML: 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>
```



HTML: Why is DOM important?

- Common data-structure for holding elements of a web-page (HTML, CSS, JavaScript etc.)
  - No need to worry about parsing HTML, CSS 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

HTML: Disadvantages of DOM

- No isolation between different parts of the DOM tree for a script as long as its from the same origin
  - All scripts from same origin (i.e., domain) can access the entire DOM tree from that origin
  - Scripts can clobber the DOM and leave it that way
  - Highly dynamic - difficult to reason about DOM state
- DOM is also very browser-specific (not standard)
- Can be a significant bottleneck in rendering webpages in parallel as it is a single global structure

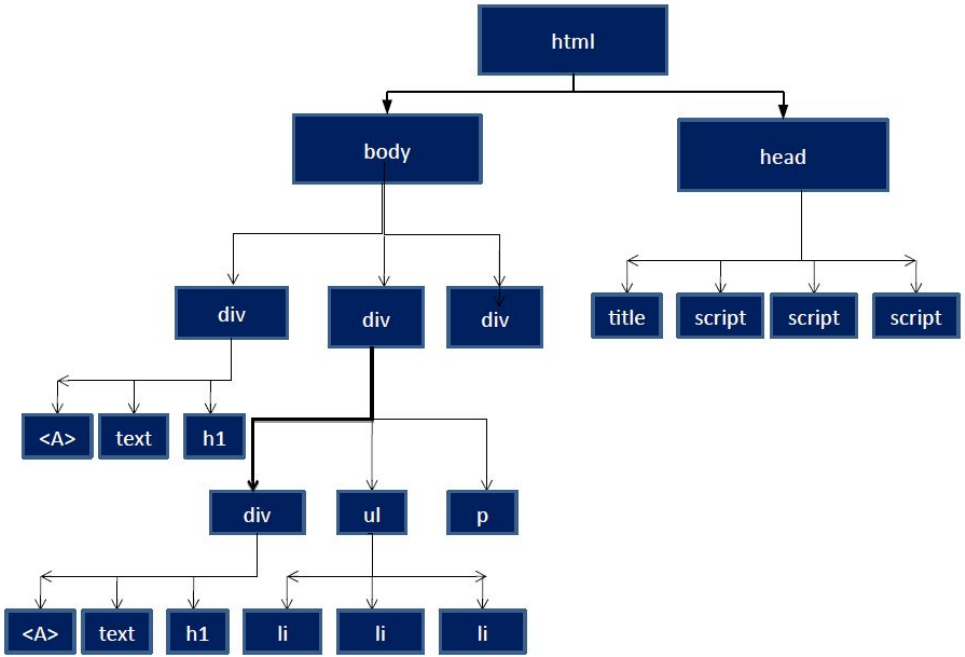
Class Activity: DOM

Draw the DOM tree corresponding to the following HTML code:

```
1  <html>
2  <head>
3     <title> ... </title>
4     <script> ... </script>
5     <script> ... </script>
6     <script> ... </script>
7  </head>
8  <body>
9     <div> <A> ... </A> <text> ... </text> <h1> ... </h1> </
          div>
10    <div>
11       <div> <A> ... </A> <text> ... </text> <h1> ... </h1> <
             /div>
12       <ul> <li> ... </li> <li> ... </li><li> ... </li> </ul>
13       <p> ... </p>
14    </div>
15    <div> ... </div>
16 </body>
17 </html>
```

Class Activity: DOM

Draw the DOM tree corresponding to the following HTML code:

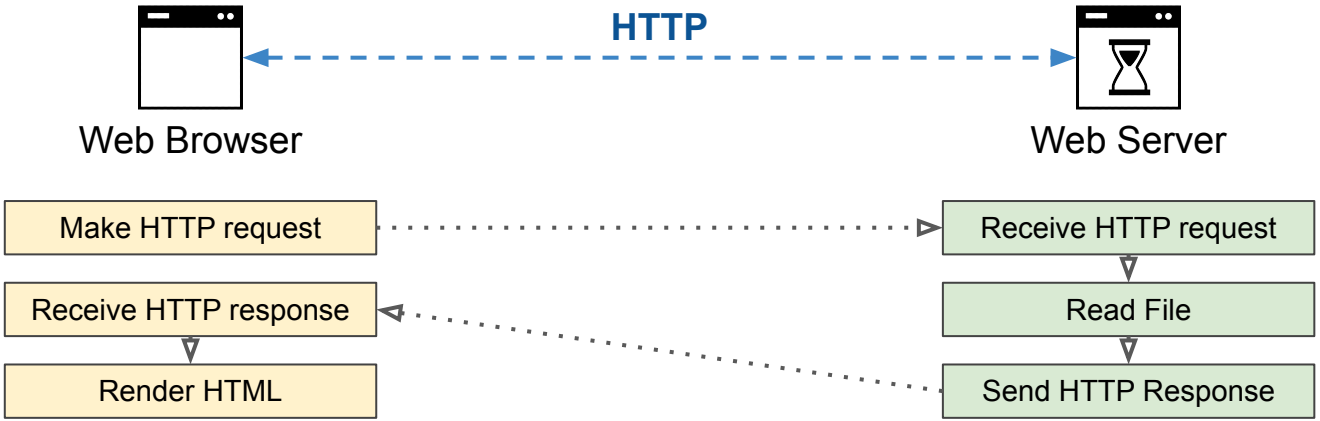


CSS (Cascading Style Sheets)

1. Web Applications
2. HTTP and HTML
3. **CSS**

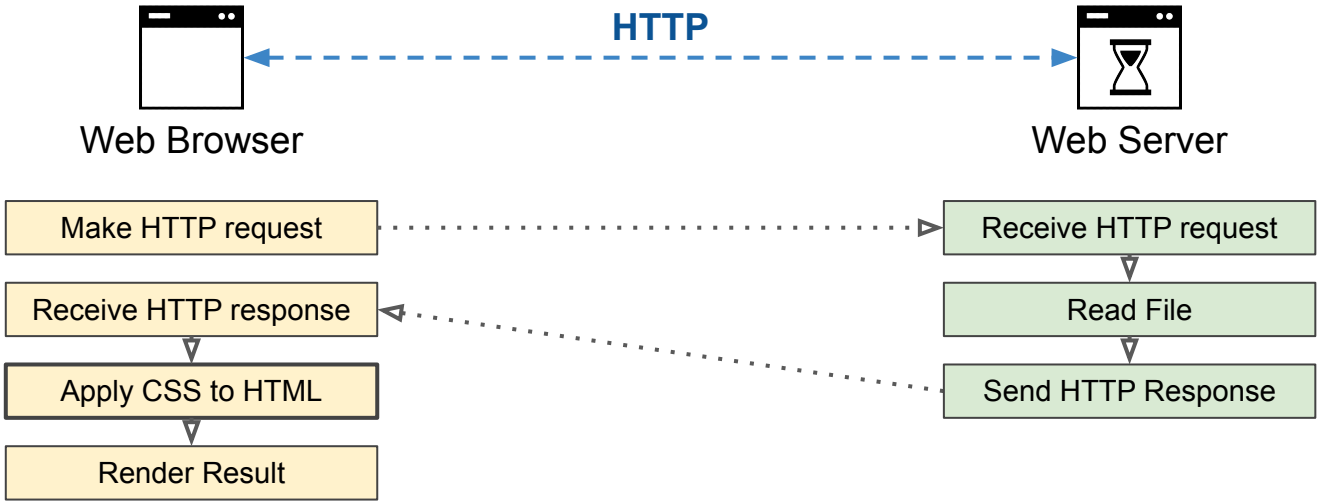
CSS: More Job for Browsers

Anatomy of a Web Application (1980s)



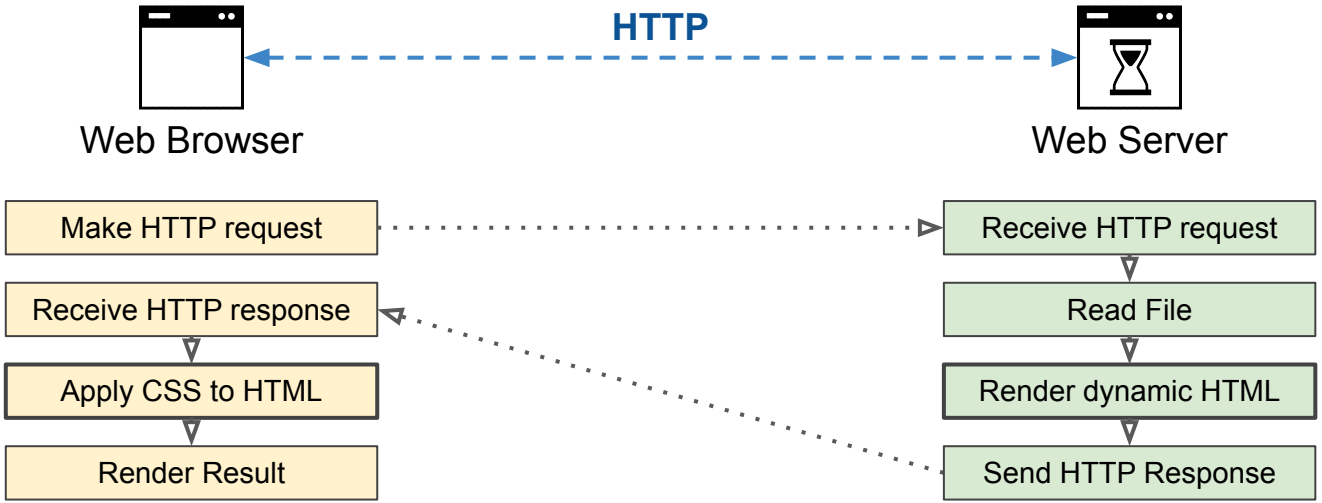
CSS: More Job for Browsers

Anatomy of a Web Application (1990s)



CSS: More Job for Browsers

Anatomy of a Web Application (1990s)



CSS (Cascading Style Sheets)

- CSS (Cascading style sheets) separate the content of the page from its presentation
- Written in the form of declarative rules with a element on LHS and action to apply on RHS
- Ensure uniformity by applying the rule to all elements of the webpage in the DOM

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)

CSS: Example

```
1  <html>
2      <head>
3          <title>Sample document</title>
4          <link rel="stylesheet" href="style1.css">
5      </head>
6      <body>
7          <p>
8              <strong>C</strong>ascading
9              <strong>S</strong>tyle
10             <strong>S</strong>heets
11         </p>
12     </body>
13 </html>
```

CSS: Example

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

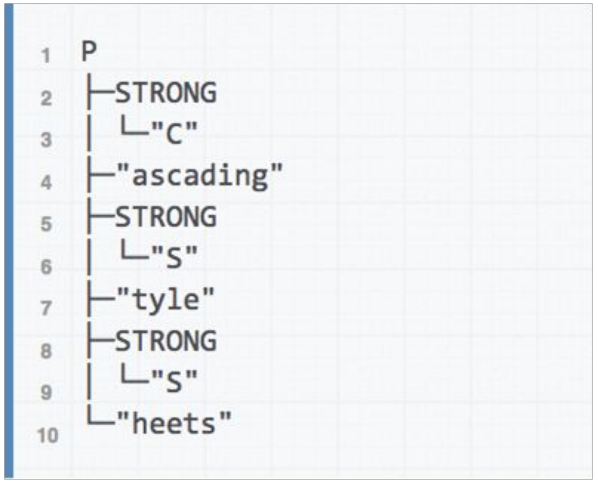
- **`strong`** → tag to match
- **`color: red;`** → attribute: value

```
1 <p>
2     <strong>C</strong>ascading
3     <strong>S</strong>tyle
4     <strong>S</strong>heets
5 </p>
```

Result: Cascading Style Sheets

CSS: How does it work?

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



Here, all STRONG tags will be matched; all descendants of STRONG tags will be styled.

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
    - Can be implemented with an in-order traversal

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

```
1 <p>
2     <strong>C</strong>ascading
3     <strong>S</strong>tyle
4     <strong>S</strong>heets
5 </p>
```

Result:

CSS: Class and ID

- 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: 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: 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)

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

```
1  Selector : pseudo-class {
2      property: value
3  }
```

CSS: Example

```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>
```

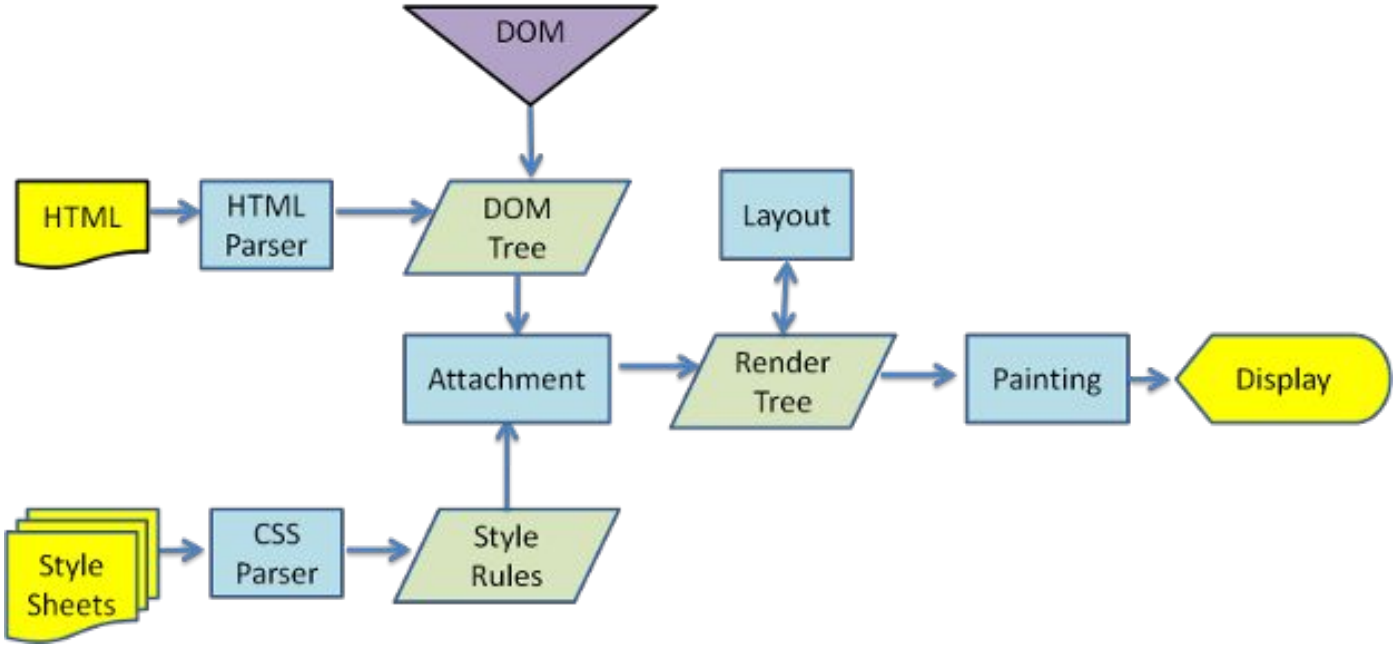
```css
div.menu-bar ul ul {
  display: none;
}

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 second 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 second element

What do Web Browsers do?



Source: https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

Class Activity: CSS Rules

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

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

Source: 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, Piscataway, NJ, USA, 408-418.

Class Activity: HTML - 1

Use a rich text editor (i.e., Word, LibreOffice Writer, Google Docs, etc.) to draft a preview of the output of this example.

```
<html>
<head>
  <link rel="stylesheet" href="activity.css">
</head>
<body>
<p id="news" style="font:normal">World
        <span class="sports">Sports News</span>
</p>
<div class="sports">Football</div>
</body>
</html>
```

/lectures/lecture-1/activity1.html

Class Activity: HTML - 2

Use a rich text editor (i.e., Word, LibreOffice Writer, Google Docs, etc.) to draft a preview of the output of this example.

```
<html>
<head>
  <link rel="stylesheet" href="activity.css">
</head>
<body>
<p id="news" style="font:normal">World
        <span class="latest">Sports News</span>
</p>
</body>
</html>
```

/lectures/lecture-1/activity2.html