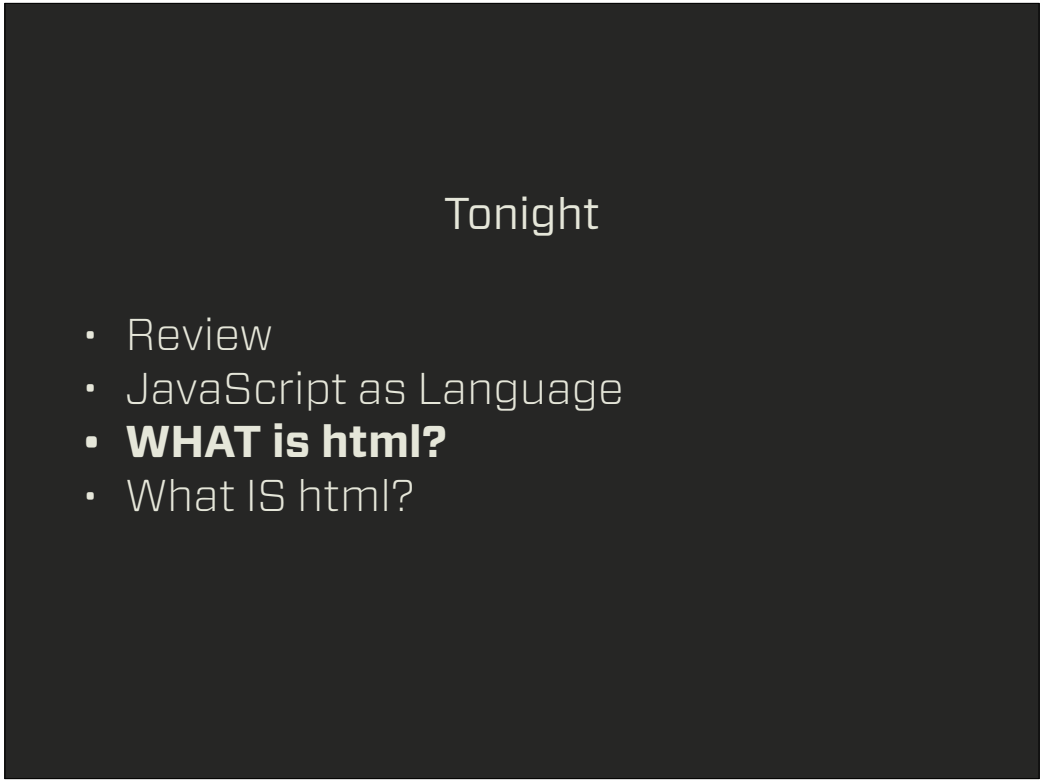
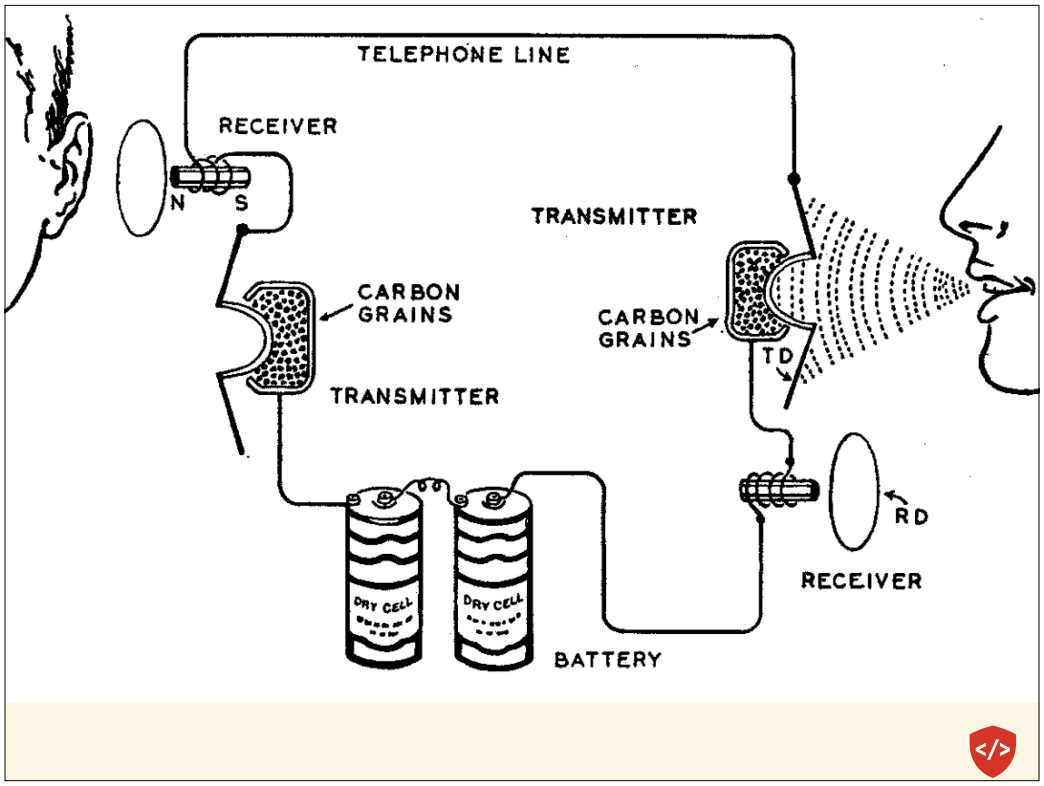
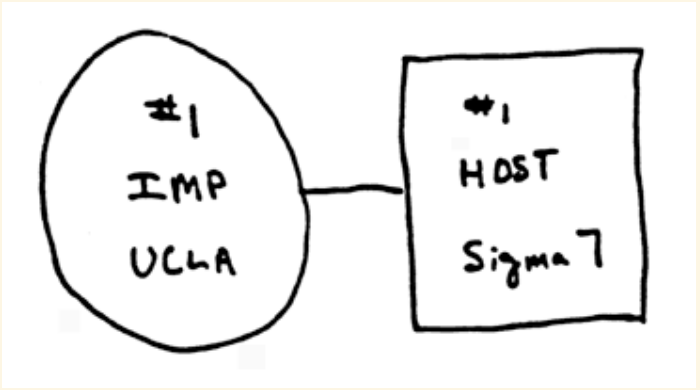


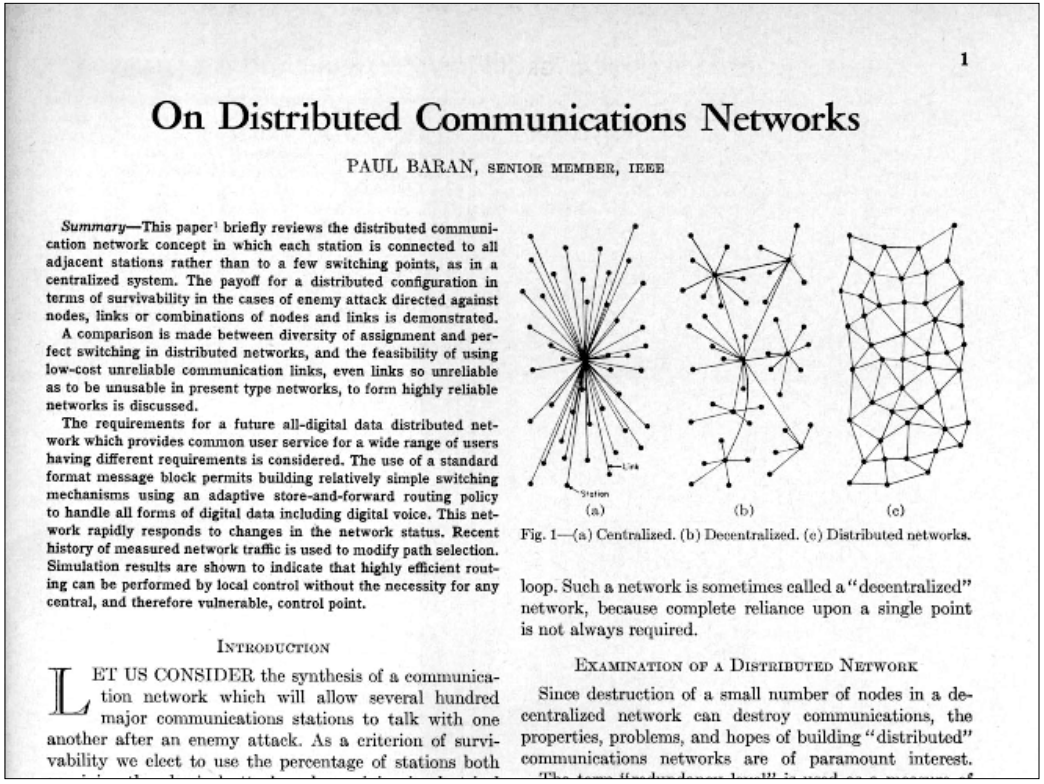
Here’s our plan for tonight.



In the 50s and 60's, most communication happened over telephone "circuits".
These simple analog 1 to 1 connections worked through a "switch" scheme to ensure each caller got the correct receiver.
This relied on circuit-board operators, who would use short patch cables to connect a caller and a receiver.
In 1969, researchers started realizing that this circuit-based system would not work well with digital components, and proposed a new system.



This system allowed 2 remote machines to talk. Digital data was sent between the 2 machines.
But what if you wanted one machine to talk to, or hear from, a lot of machines? potentially in rapid succession? What kind of switching circuit could allow that?



1

On Distributed Communications Networks

PAUL BARAN, SENIOR MEMBER, IEEE

Summary—This paper[1] briefly reviews the distributed communication network concept in which each station is connected to all adjacent stations rather than to a few switching points, as in a centralized system. The payoff for a distributed configuration in terms of survivability in the cases of enemy attack directed against nodes, links or combinations of nodes and links is demonstrated.

A comparison is made between diversity of assignment and perfect switching in distributed networks, and the feasibility of using low-cost unreliable communication links, even links so unreliable as to be unusable in present type networks, to form highly reliable networks is discussed.

The requirements for a future all-digital data distributed network which provides common user service for a wide range of users having different requirements is considered. The use of a standard format message block permits building relatively simple switching mechanisms using an adaptive store-and-forward routing policy to handle all forms of digital data including digital voice. This network rapidly responds to changes in the network status. Recent history of measured network traffic is used to modify path selection. Simulation results are shown to indicate that highly efficient routing can be performed by local control without the necessity for any central, and therefore vulnerable, control point.

Introduction

LET US CONSIDER the synthesis of a communication network which will allow several hundred major communications stations to talk with one another after an enemy attack. As a criterion of survivability we elect to use the percentage of stations both

Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.

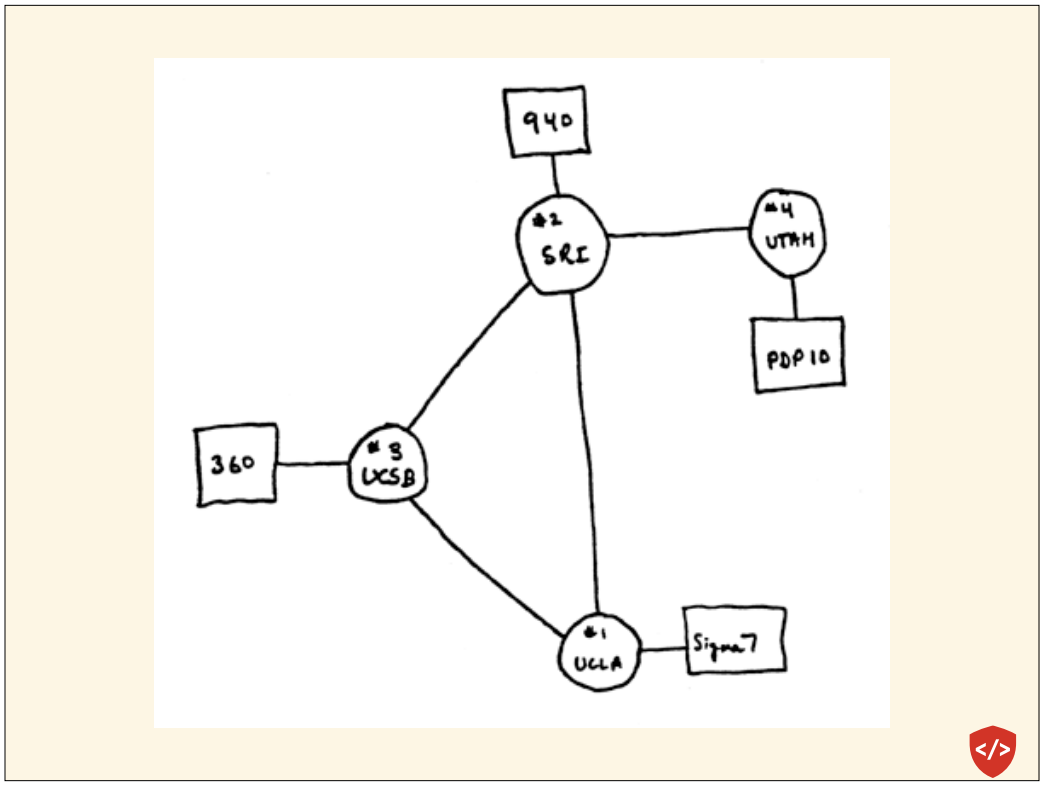
loop. Such a network is sometimes called a "decentralized" network, because complete reliance upon a single point is not always required.

Examination of a Distributed Network

Since destruction of a small number of nodes in a decentralized network can destroy communications, the properties, problems, and hopes of building "distributed" communications networks are of paramount interest.

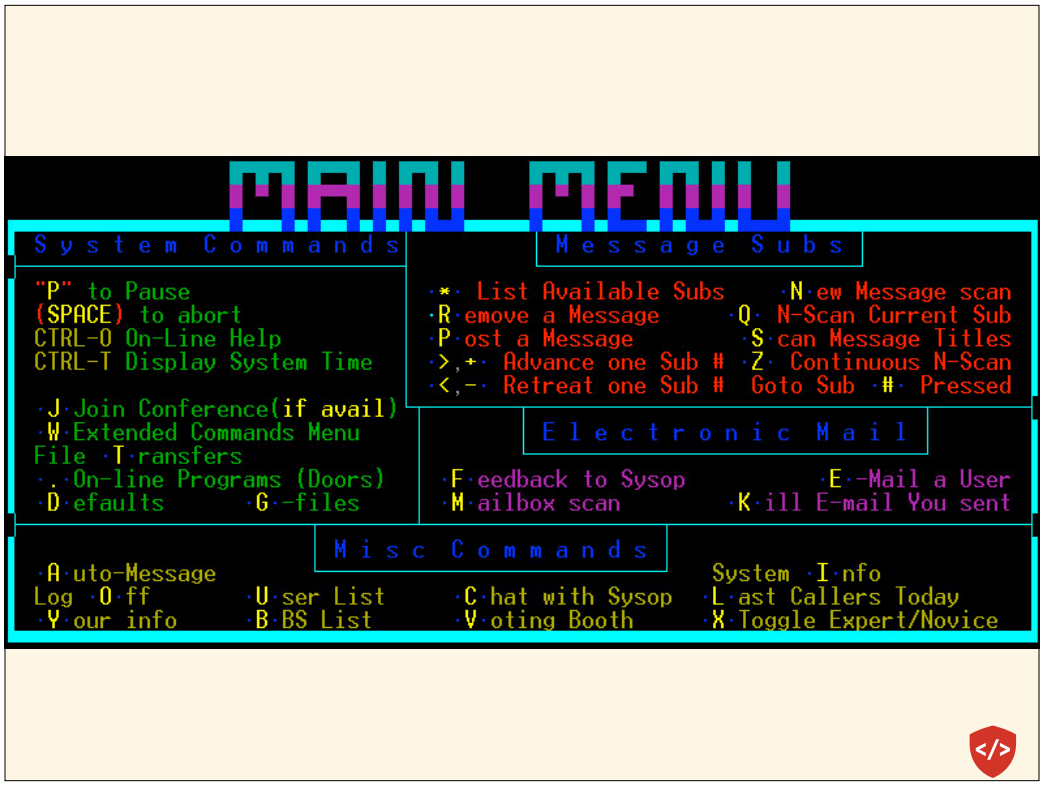
In 1969, as we were landing on the moon, a revolution was happening on the ground.
A new communication protocol, based on switched "packets" of information (rather than circuits) was proposed by researchers at MIT and DARPA, the research wing of the Defense department.

With this new system, it would be possible for 2 computers to talk to each other, without needing to go through a central switch board.

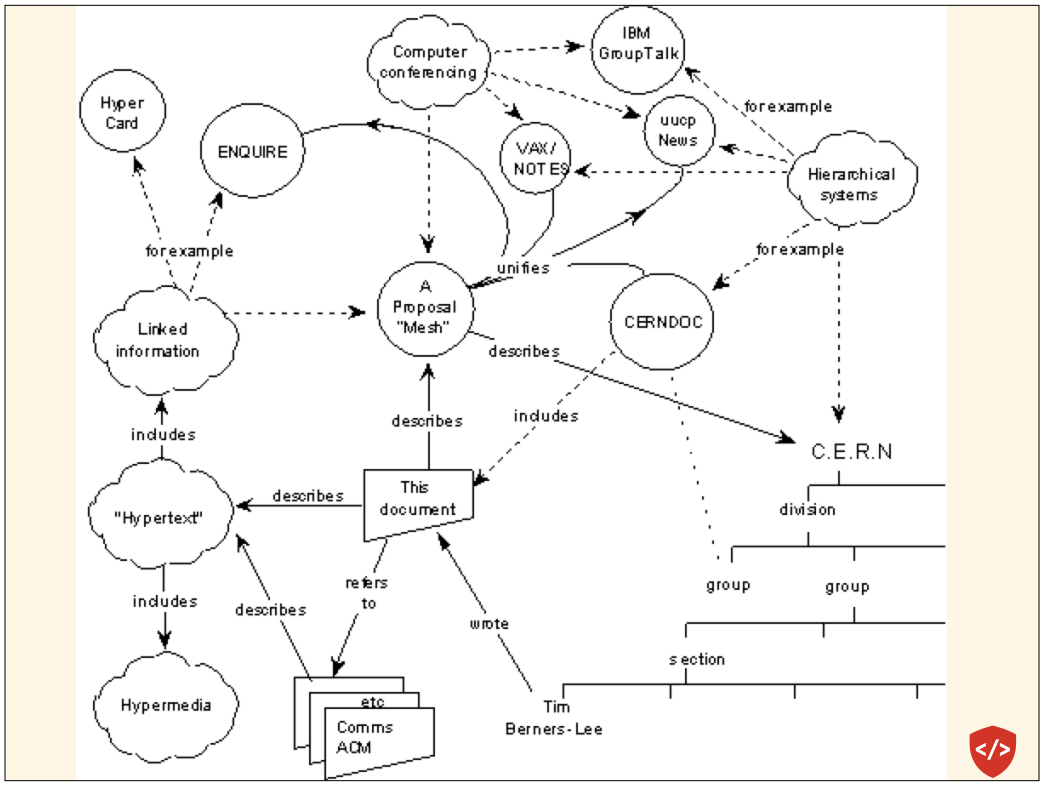


Only a few locations in the US had the most powerful computers. They wanted researchers from other institutions to connect to use the resources available.
Each node in this graph represents a complete, local Network. All the computers within the UCLA network could already talk to each other, for example.

The long-distance connections were known as Internetwork links.
At the same time, the computer scientists building this technology needed a way to communicate about how the process was going.
In 1971, the basics of the modern Email protocol emerged, the first killer-app for the Internet. Honestly, it still is one of the greatest uses.

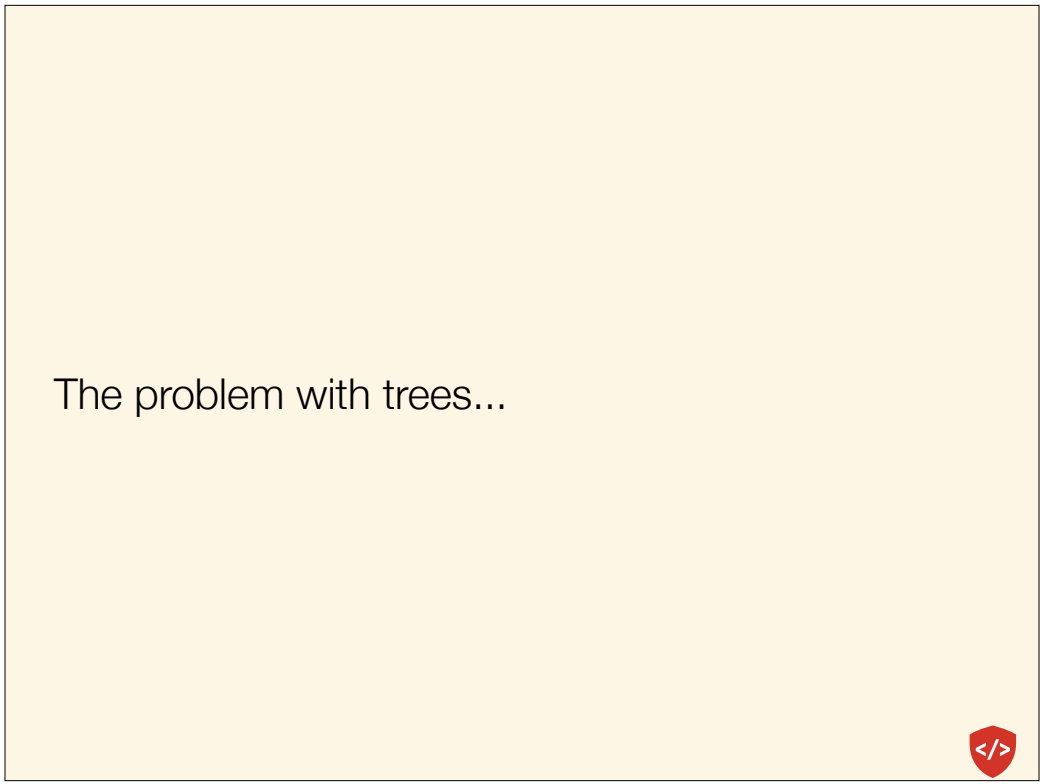


Applications could now be built on top of this Internet.
Not only could computers talk to computers, but users could talk to users, thanks to these applications.
I remember tying up the family phone line for hours chatting with friends on the local BBS when I was in middle school.



In 1989 a researcher at CERN (a particle accelerator in Switzerland) had a problem.
Scientists would spend 2 years at a time at CERN, and the high turnover was making it a headache for everyone to keep track of basic resources and knowledge.

So Tim Berners-Lee proposed a system of connecting together various departments and documentation systems, inspired by an existing technology called HyperCard.

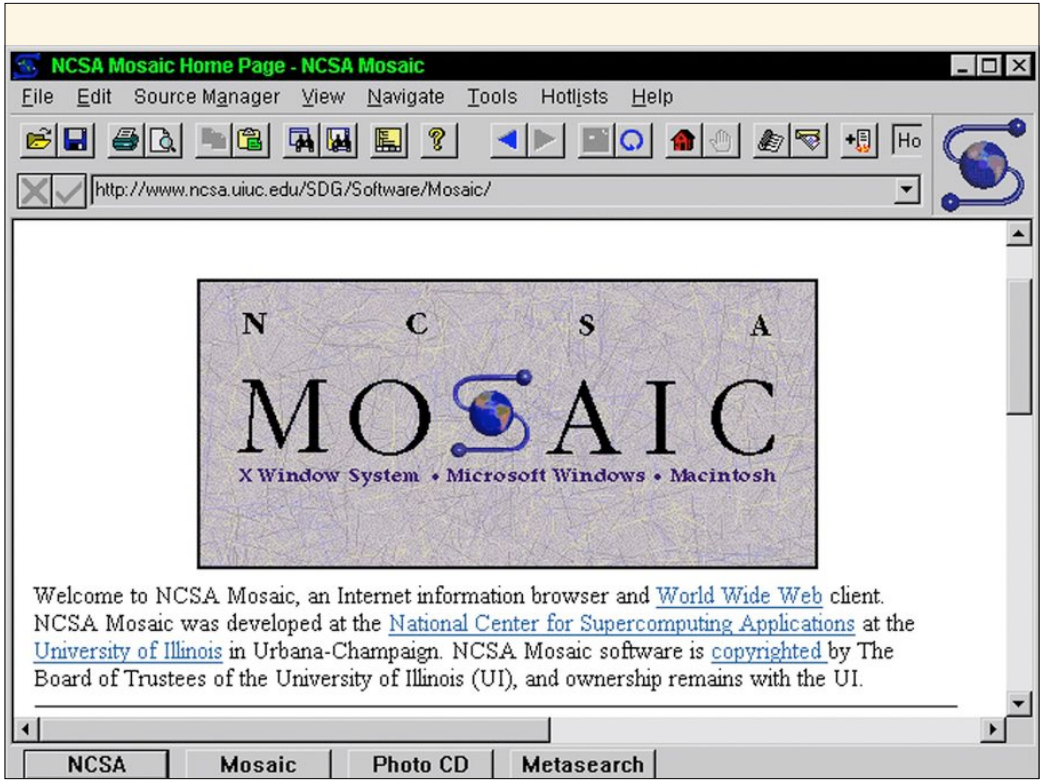


Lots of data was already organized in trees, but that was proving to be insufficient.

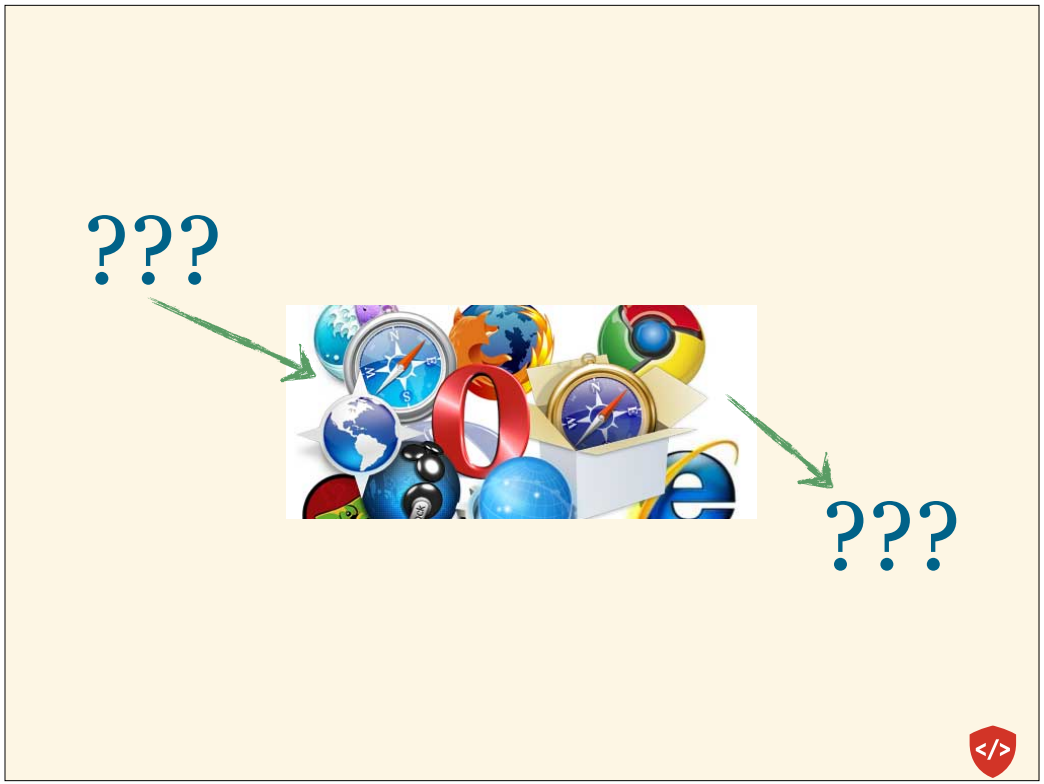
A better representation of the data was more of a MESH, where nodes could be connected arbitrarily.

He called it a hyper-text transport protocol (HTTP).

Read TBL's thoughts here, in his original proposal: www.w3.org/History/1989/proposal.html



By 1993 there was a publicly available way to browse this web of interconnected documents: a “browser” created by a company called NCSA.
“The Mosaic browser was named for its support of multiple internet protocols.” – WikiP (protocols like FTP, gopher, newsgroups, and of course HTTP)
The first was strikingly similar to our modern browser.



Browsers old and new all work in the same basic way. We give something to the browser, and we get something back.

URLWTF??

Some URLs are nice:

http://twitter.com/brookr

https://codefellows.basecamp.com/todo_lists.html

Some URLs are not:

https://plus.google.com/u/0/105482132808648480108/posts

http://www.amazon.com/dp/B00EOE0WKQ?pf_rd_i=507846

You can make a reasonable guess at what some URLs will show you.

Others: not so much. Guess what these urls go to?

URLWTF??

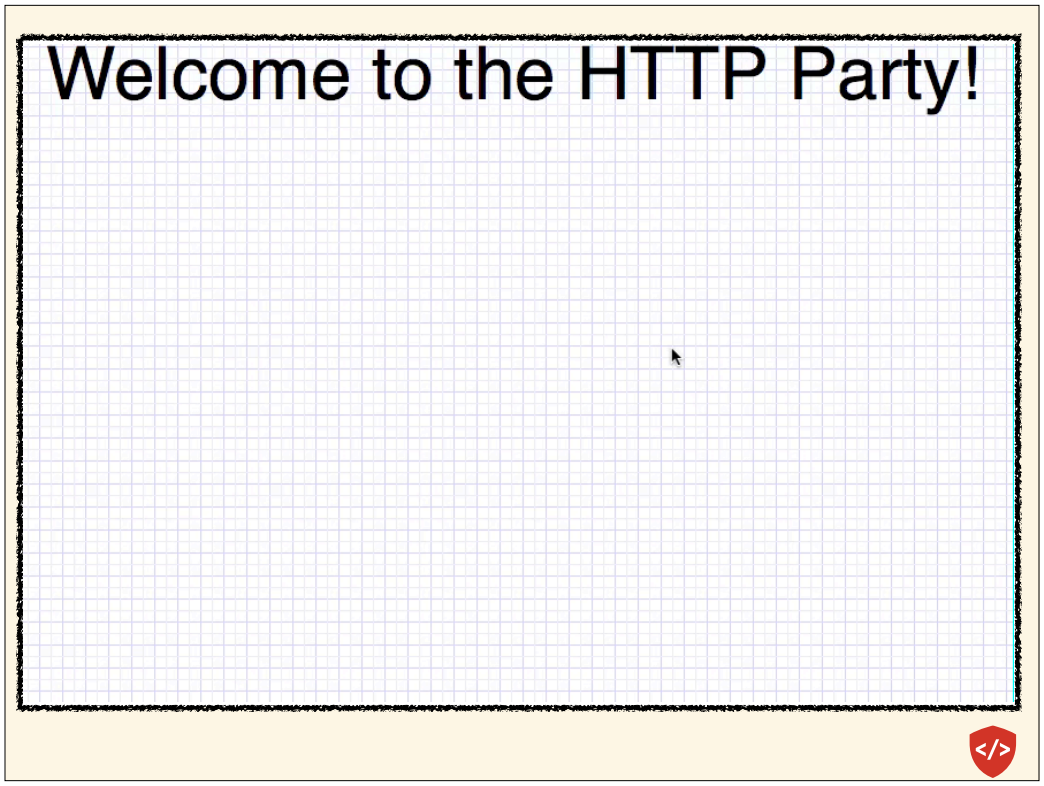
- But they all have the same parts:

https://codefellows.basecamp.com/todo_lists.html?a=b

- Protocol: https://
- [sub]domain: codefellows.basecamp.com
- path: /todo_lists
- format: .html
- parameters: ?due_frame=later&a=b&key=value

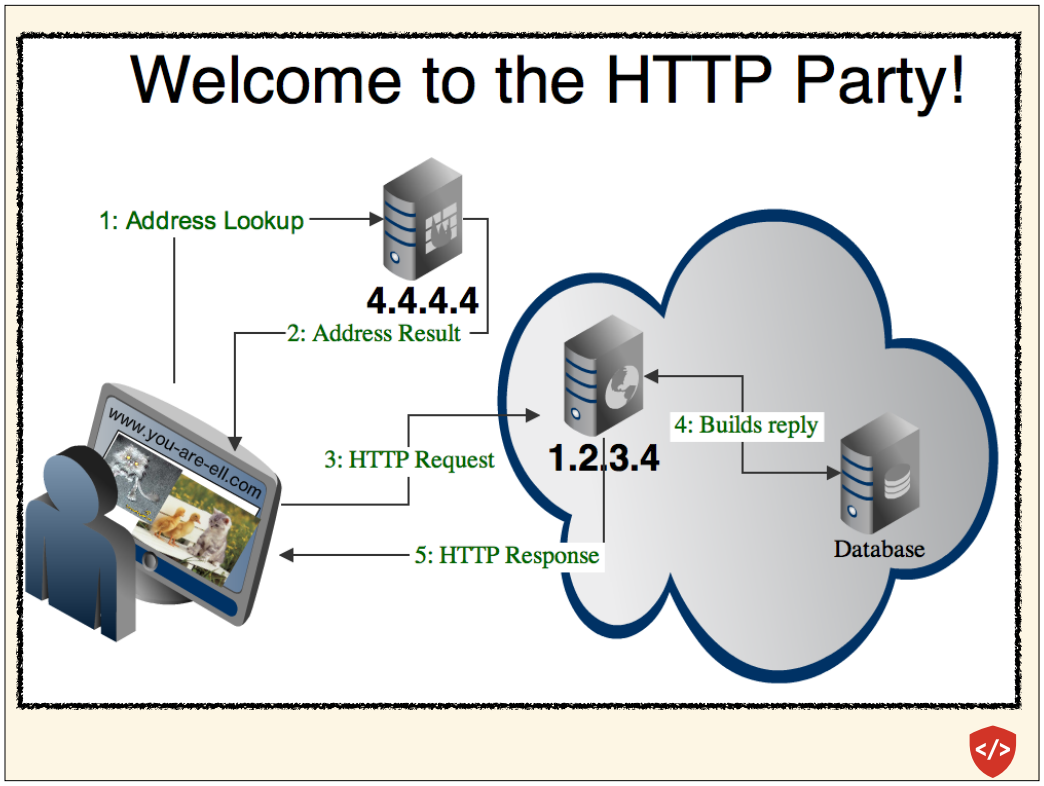
These pieces give the browser all the info needed to get that web page.

How: Using the HTTPS protocol.
From whom: This specific sub-domain of this domain.
From where at that domain: At this path.
What are we looking for: This format, modified by these paramaters.



This is how your URL gets converted into web page contents.

SHOW VIDEO



This is how your URL gets converted into web page contents.

The DNS Dance

- Go ask at a known DNS Server
- Global index of domain names
- Give a name
- Get an IP address
- Now ready to ask for that page!

To make that happen, the browser converts the domain into an address.

REAL WORLD EXAMPLE: Like looking up someone's phone number in the phone book. You know where to look, you know their name. The book tells you the number to reach them directly.

However people do that these days, I don't even know!

If you've ever registered your own domain name before, setting up DNS is part of the process. This is why: people need to be able to find it.

The HTTP Party

- The browser creates a HTTP Request Object
- HTTP Request has 3 parts:
  - **URL** (twitter.com)
  - **Method** (GET)
  - **Headers** (sender info: user agent, cookies, etc)

Image: http://rdr.zazzle.com/img/imt-prd/pd-172254667462737476/isz-m/tl-R.S.V.P+Wedding+Postage+Stamp.jpg

The HTTP Party

- The server:
  - receives the request
  - builds a response
  - sends it back to the client

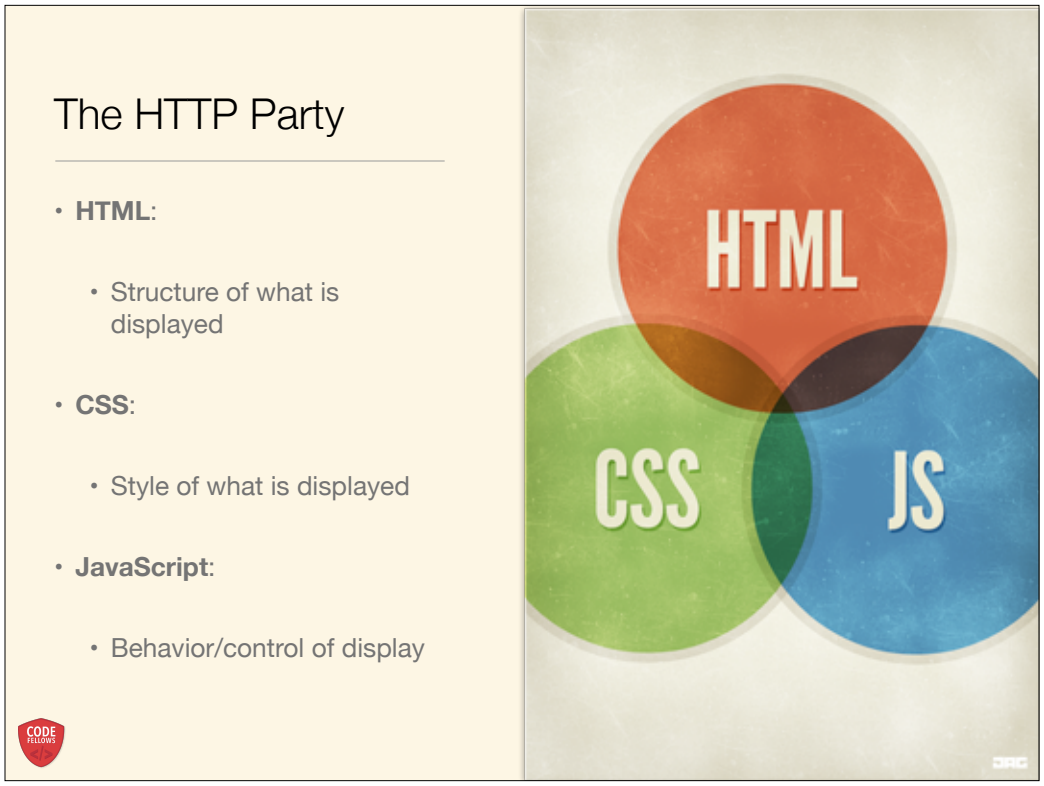
The server's response could vary greatly, from a simple error message, to a plain static file, to a dynamically generated page that matches the unique user's unique and wonderful request.

The HTTP Party

- An HTTP Response also has 3 parts:
  - **Status code** (200, 301, 404, 500, etc)
  - **Headers** (info about the server & file sent)
  - **Body** (the content of the page)
    - HTML, CSS, JavaScript

The response is stateless: It's simply a reply to the request. It doesn't consider the requestor's history, or who they are, or where they come from. Some of that info can be used to build the response, but only if it was included in the request.

Anyone know what any of these codes mean?



Some combination of different amounts of HTML/CSS/JS will be rendered by the browser to produce the final result shown to the user.

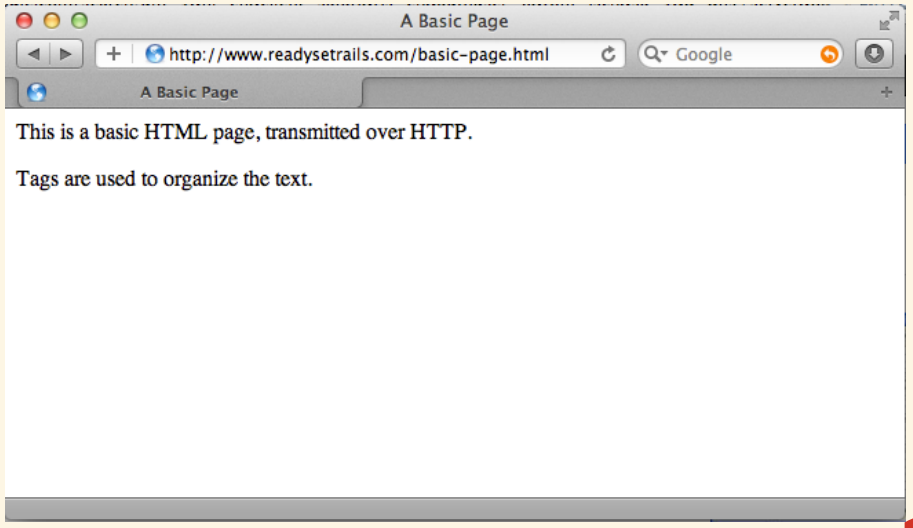
Image: http://cache1.bigcartel.com/product_images/51574645/html-css-js-mockup.png

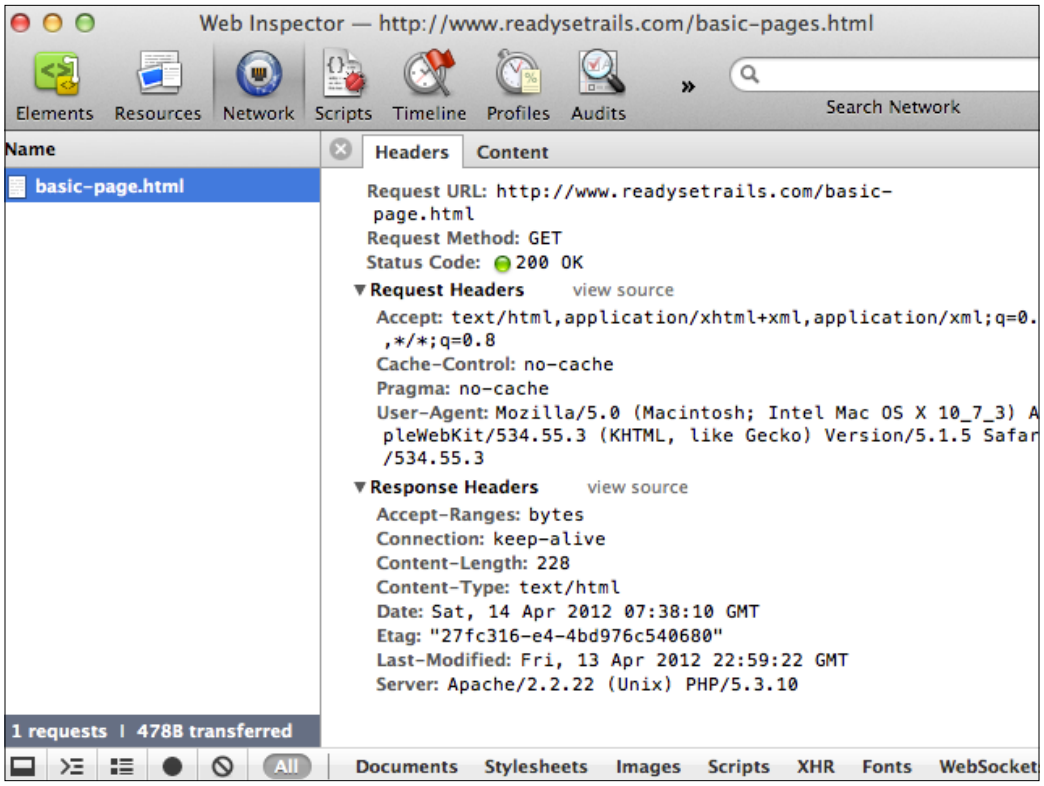
It’s a Markup language!

The HTML that comes back from the server is a set of instructions that tells the browser how to display the content.

Full of links that the browser fetches individually... “Loading 34 of 44...”

The HTTP Party Example

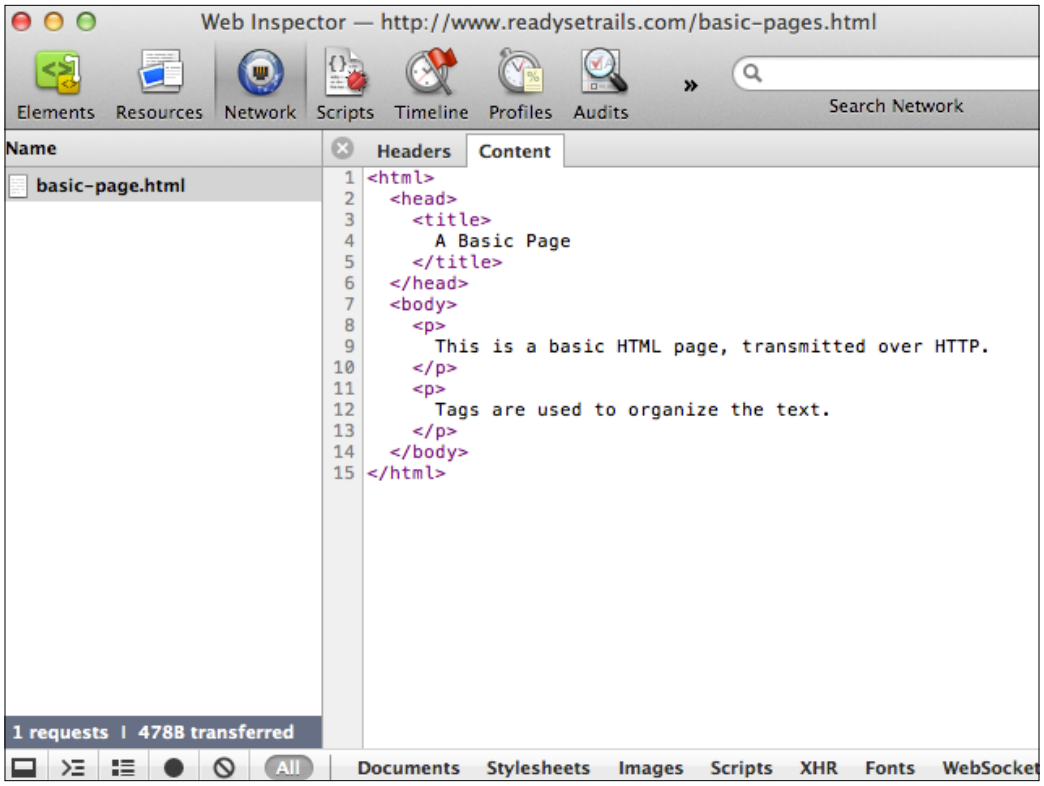




Let's take a look behind the scenes, and see what actually happened there.

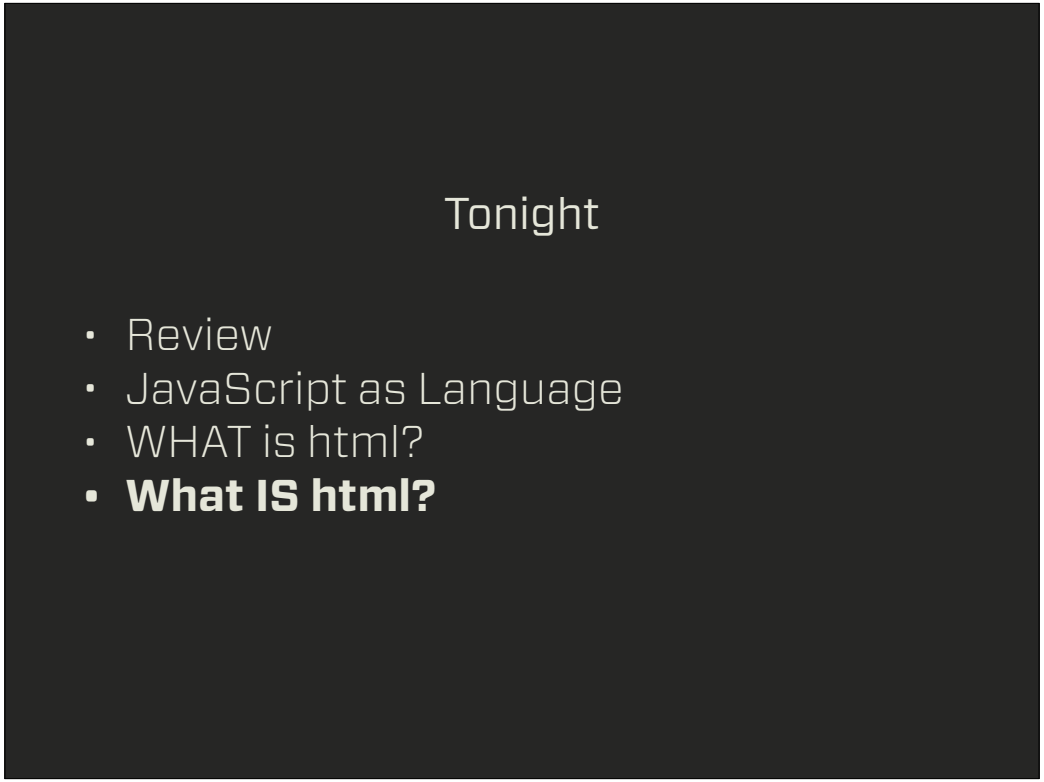
We see the request parts: URL, Method, Headers

We see the response parts: Status, Headers, Content

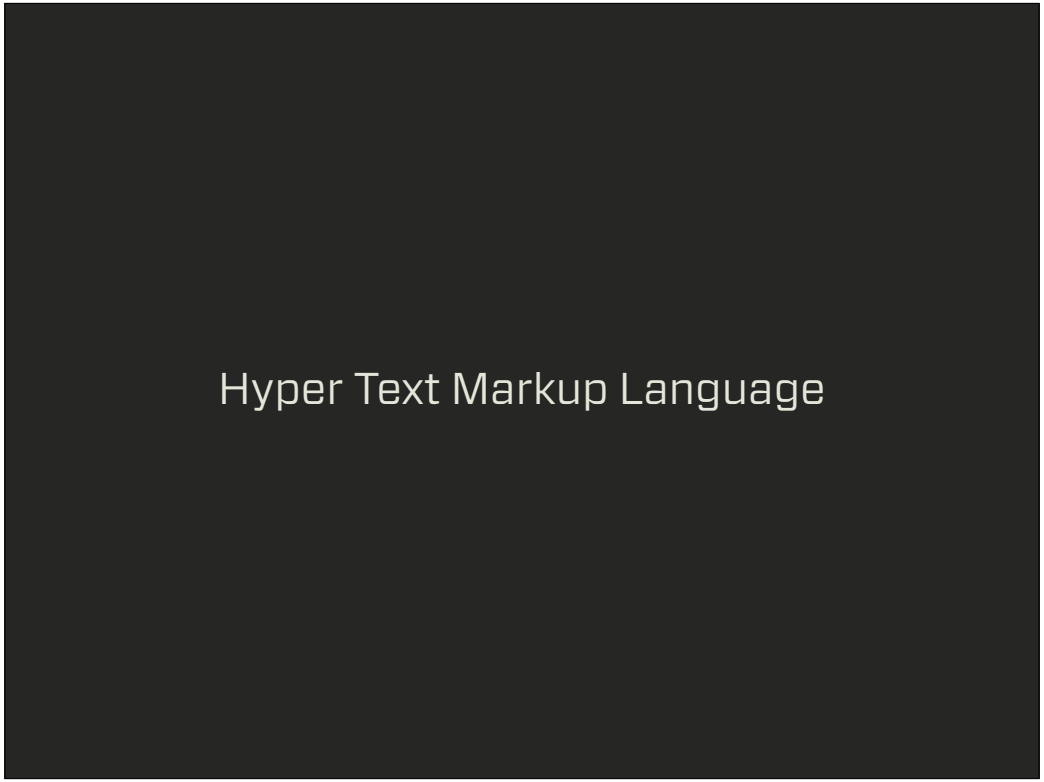


Content is HTML, but might have also included CSS or JavaScript

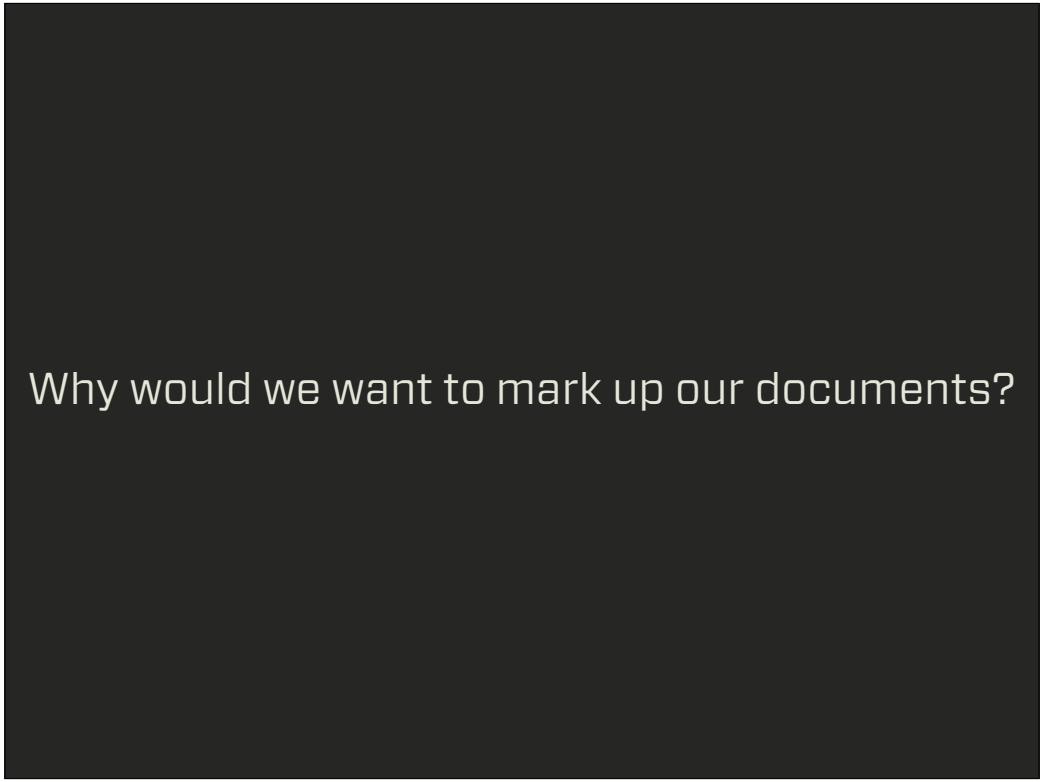
The web turned 25 earlier this year. Check it out, get involved: http://www.webat25.org/



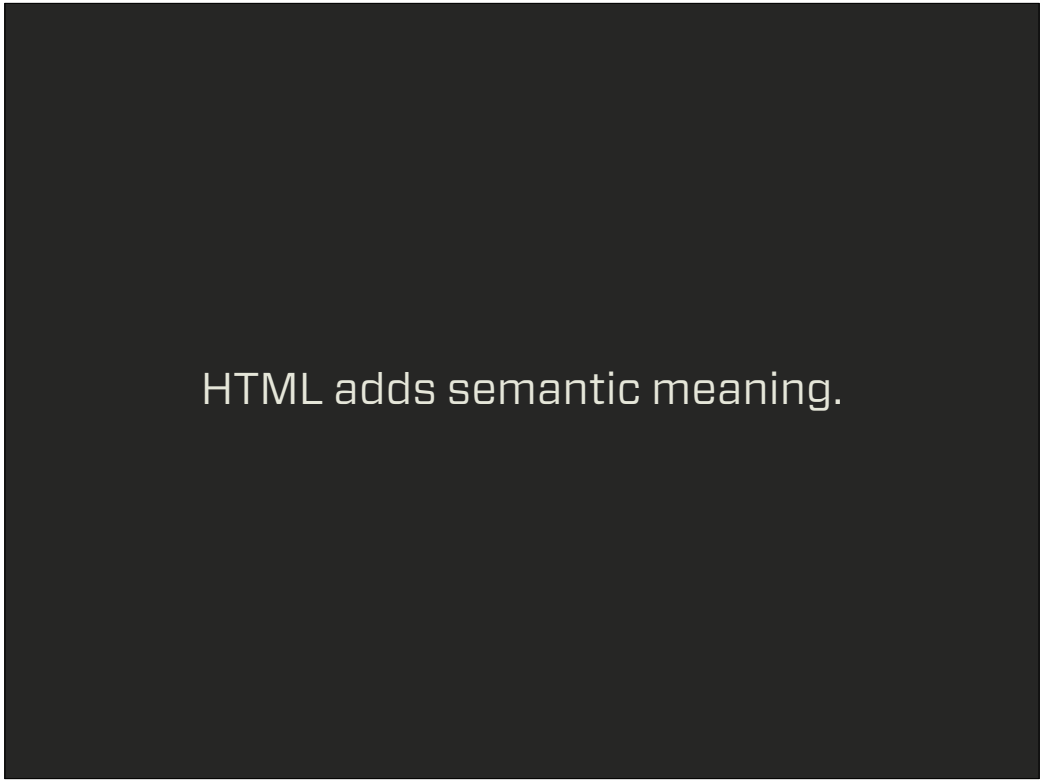
Here's our plan for tonight.



But what is a Markup Language? Let’s see by example.



But what is a Markup Language? Let's see by example.

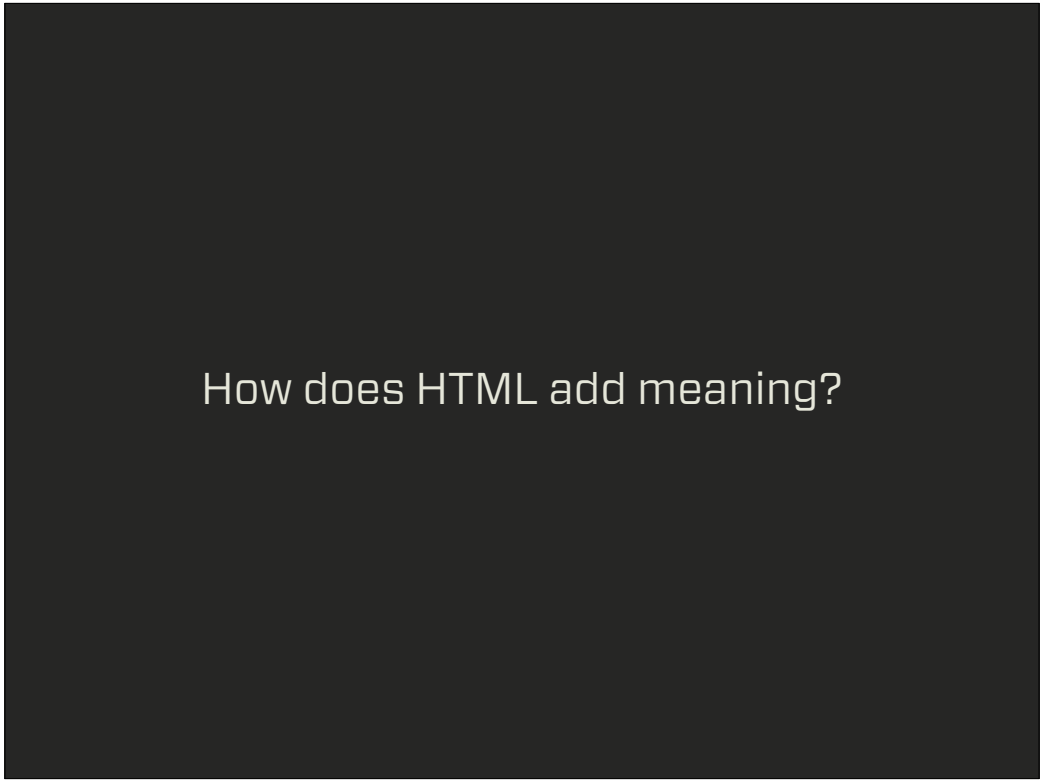


The real goal of HTML is to give greater meaning to the content our documents contain.

Vocabulary

- *Semantics*:
  - The study of meaning

Vocabulary review



Let’s get into the details.

Tags

- The language we use to “mark up” our docs
- Usually open and closing pairs
- Only certain tags are recognized
- Examples:
  - Paragraph: some text
  - Title: <title>My page!</title>
  - Header: <h1>Welcome...</h1>

Elements

- Designators that define objects in the doc
- Create structure around content
- Contain opening tags, closing tags, content
- Examples:
  - Paragraph: some text
  - Title: <title>My page!</title>
  - Header: <h1>Welcome...</h1>

Attributes

- Properties that provide additional info
- Gives the tag more details
- Examples:
  - Paragraph: some text
  - Header: <h1 class="red">Welcome...</h1>
  - Image: <img src="cats.png">

Data-attributes

- New with HTML5
- We can create custom attributes
- Starts with “data-”

```
<ul id="vegetable-seeds">
  <li data-spacing="10cm" data-sowing-time="March to June">Carrots</li>
  <li data-spacing="3cm" data-sowing-time="March to September">Radishes</li>
</ul>
```

Required Structure

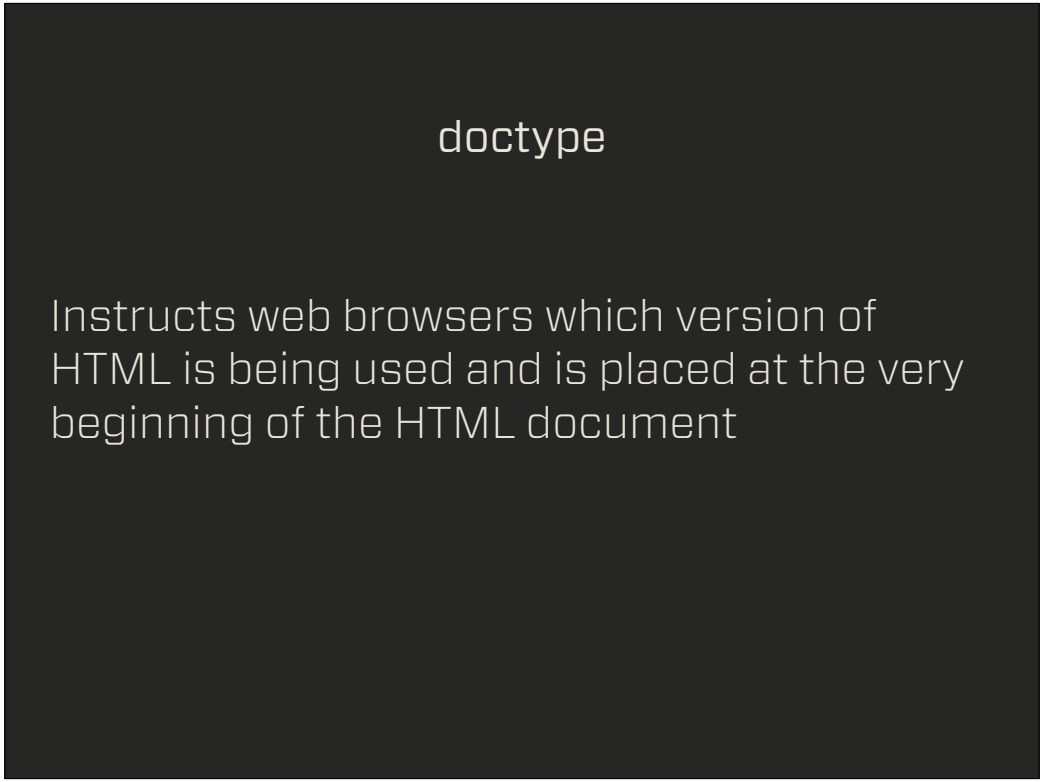
- All HTML docs must have a standard structure
  - doctype
  - html
  - head
  - body

The doctype declaration is used to instruct web browsers which version of HTML is being used and is placed at the very beginning of the HTML document.

The head of the document is used to outline any meta data, the document title, and links to any external files.

Any context included within the head tags is not visible within the actual web page itself.

All of the content visible within the web page will fall within the body tags.



The head of the document is used to outline any meta data, the document title, and links to any external files.

Any context included within the head tags is not visible within the actual web page itself.

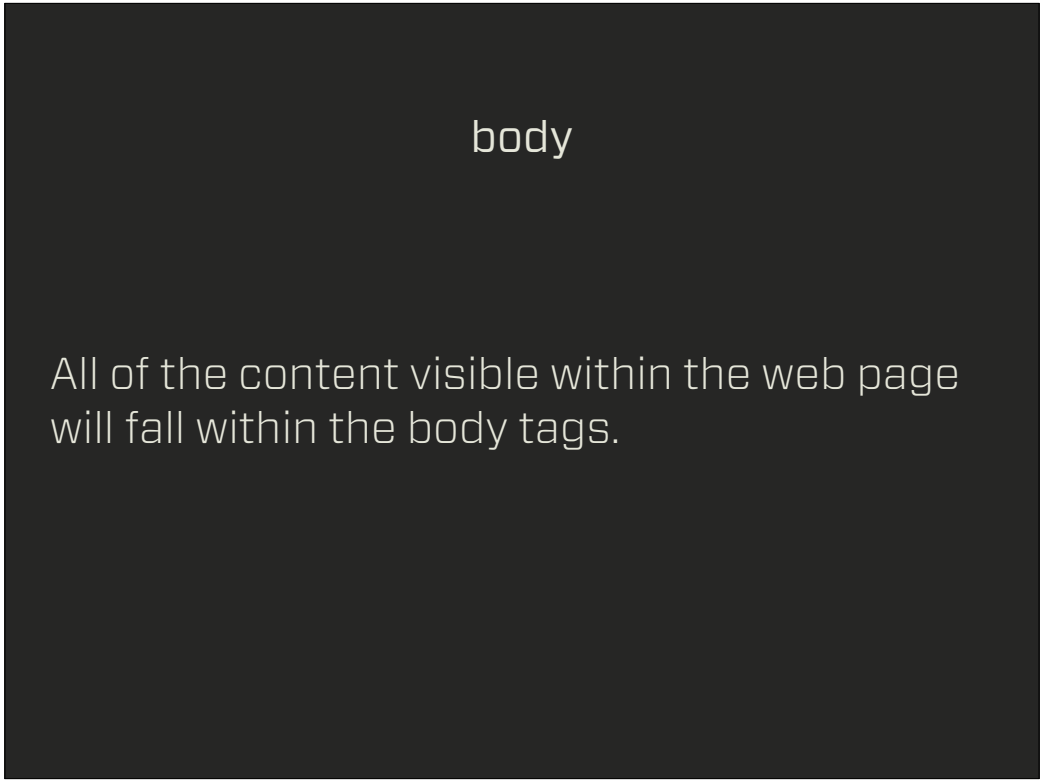
All of the content visible within the web page will fall within the body tags.

head

Outlines any meta data, the document title, and links to any external files.

Any context included within the head tags is not visible within the actual web page itself.

All of the content visible within the web page will fall within the body tags.

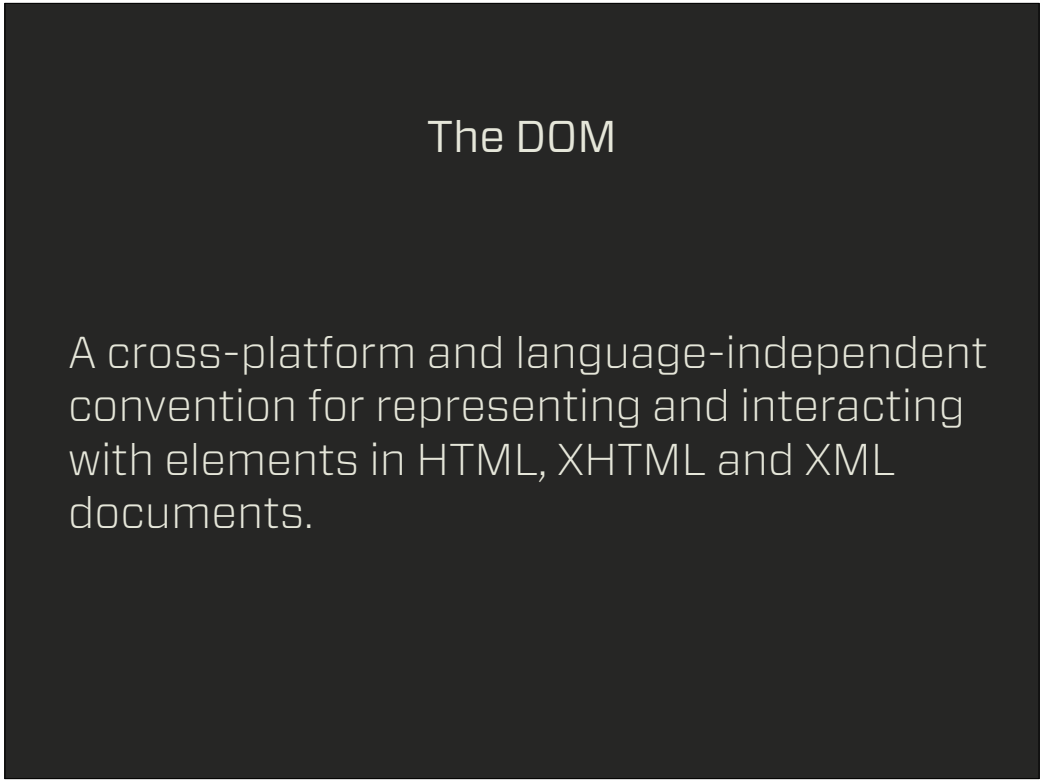

body
All of the content visible within the web page will fall within the body tags.

Essential Structure

```
<!DOCTYPE html>
    <head>
        <meta charset="utf-8">
        <title></title>
        <link rel="stylesheet" href="css/main.css">
    </head>
    <body>
        <h1>Hello World</h1>
        <p>This is a website.</p>
        <script src="js/main.js"></script>
    </body>
</html>
```

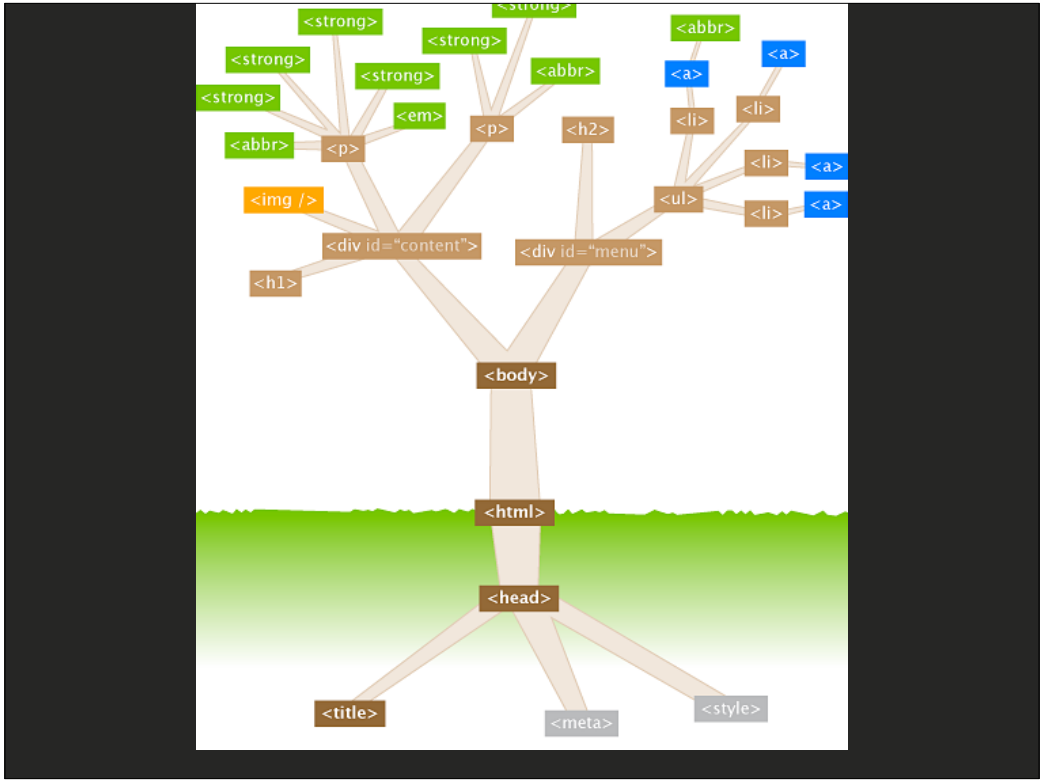
The DOM

Document-Object Model

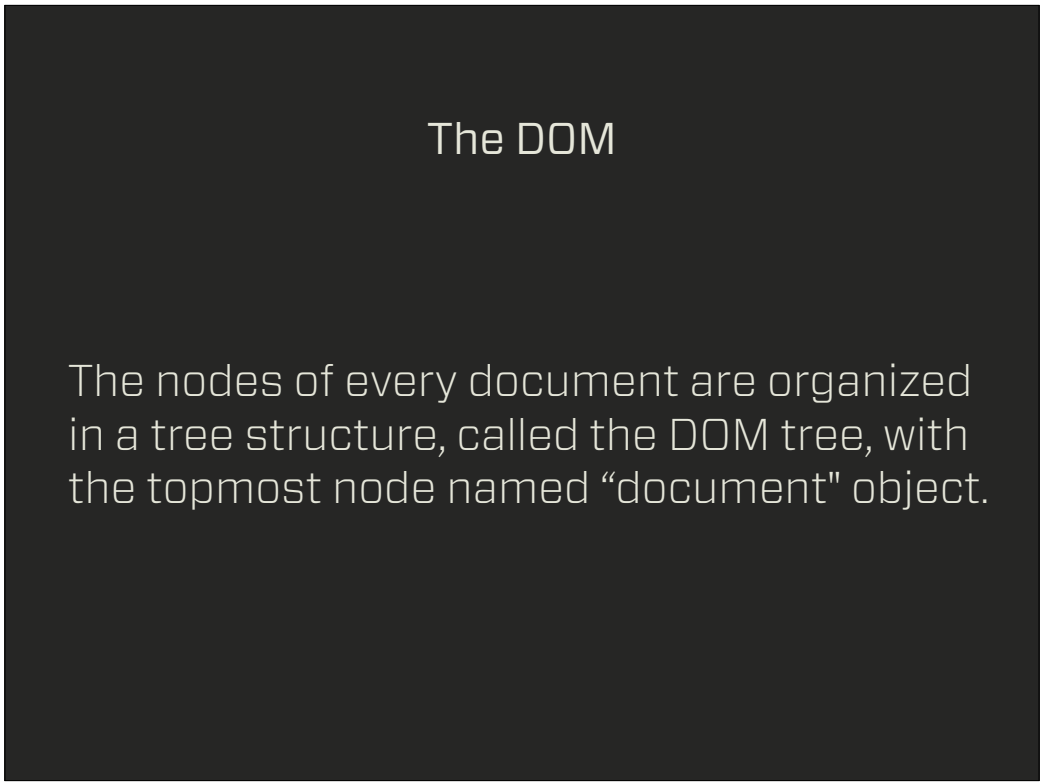


NOTE: The DOM tree is NOT ‘view source’

With the advent of JS MVVM frameworks and other DOM manipulators, much can happen after the fact that ‘view source’ will not show.



Remember this tree?



When an HTML page is rendered in browsers, the browser downloads the HTML into local memory and automatically parses it to display the page on screen.

The DOM is also the way JavaScript understands the state of the browser in HTML pages.

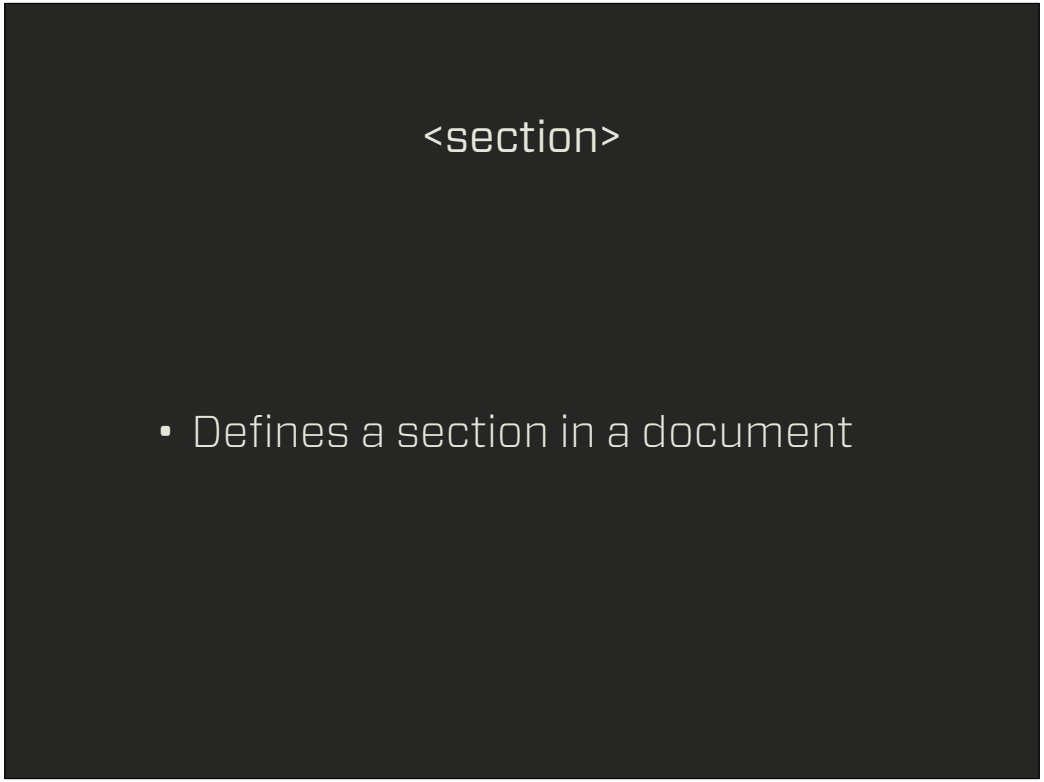
The DOM

Web browsers rely on layout engines to parse HTML into a DOM.

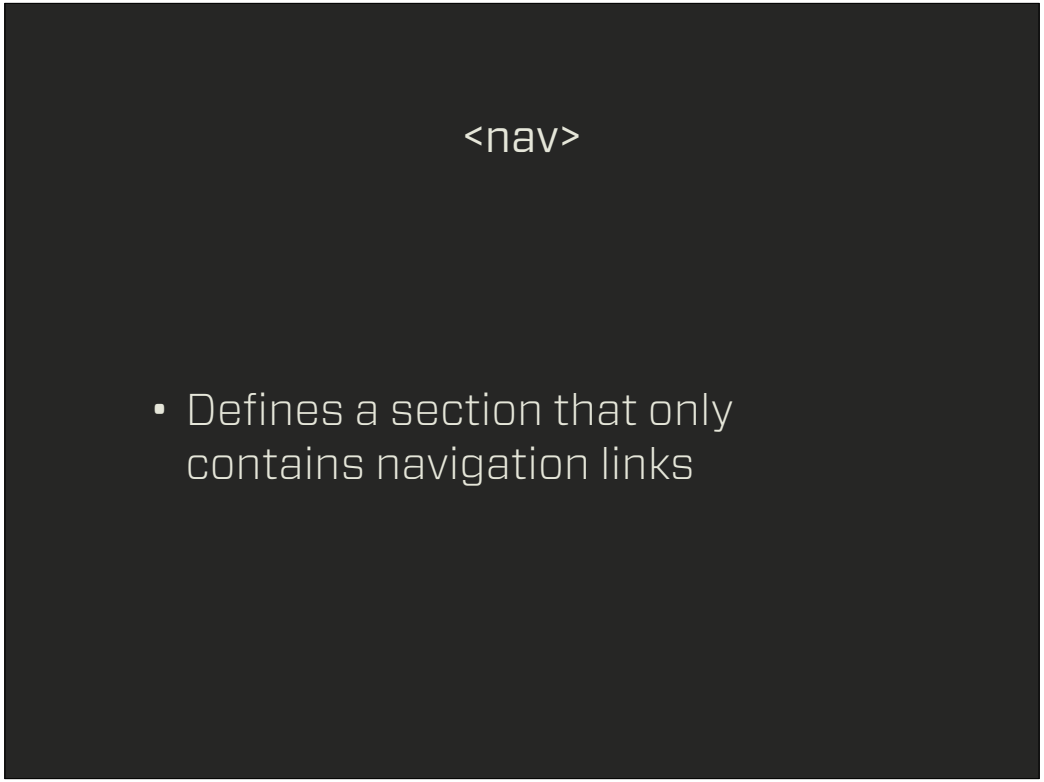
- Trident/MSHTML: MS Internet Explorer
- Gecko: Mozilla/Firefox
- Webkit: Safari/iOS
- Blink: Chrome and Opera

This is where cross-browser differences arise.

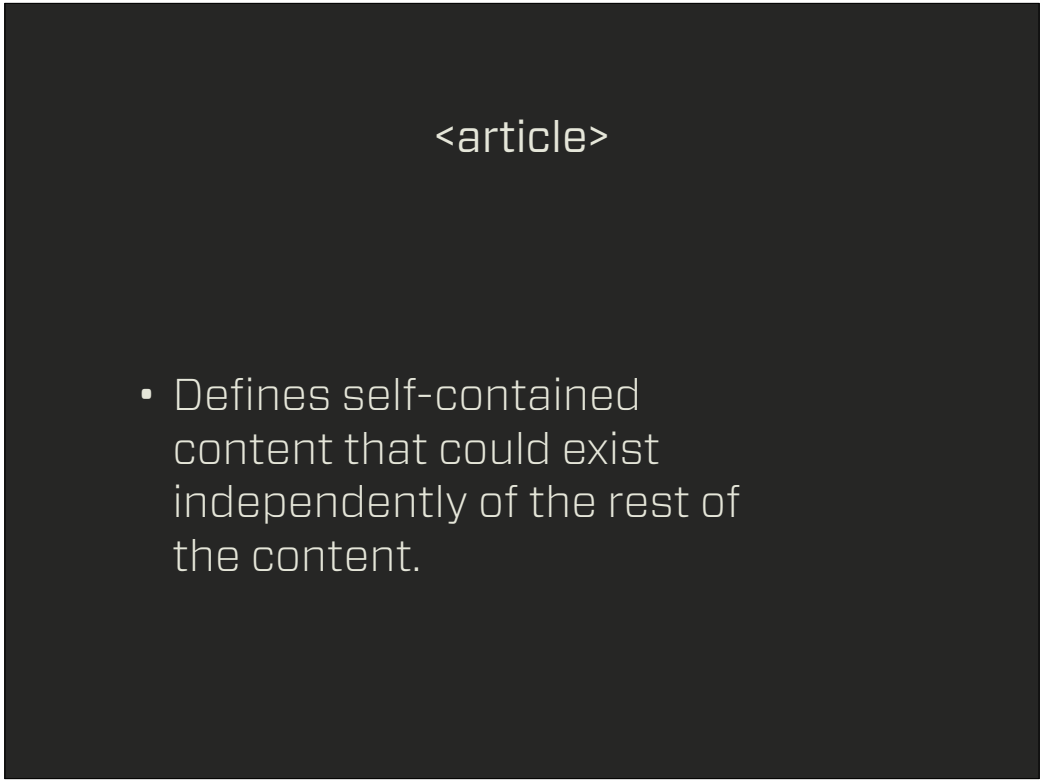
Lets look at some key elements now available in HTML5



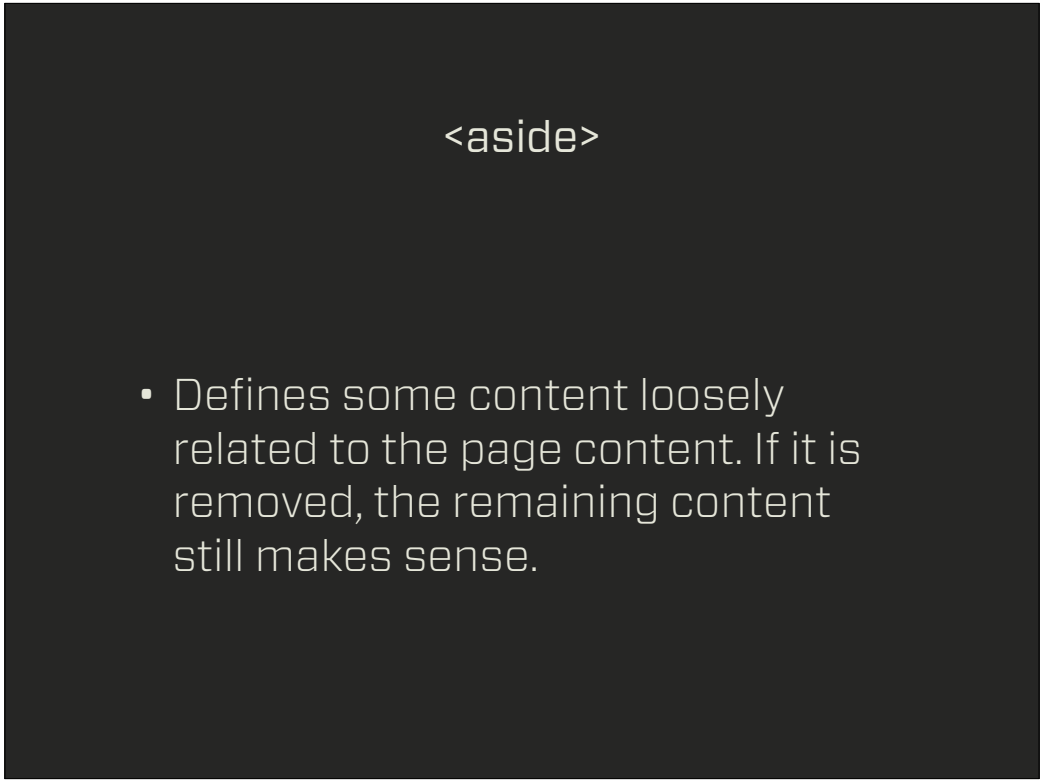
Lets look at some key elements now available in HTML5



Lets look at some key elements now available in HTML5



Lets look at some key elements now available in HTML5

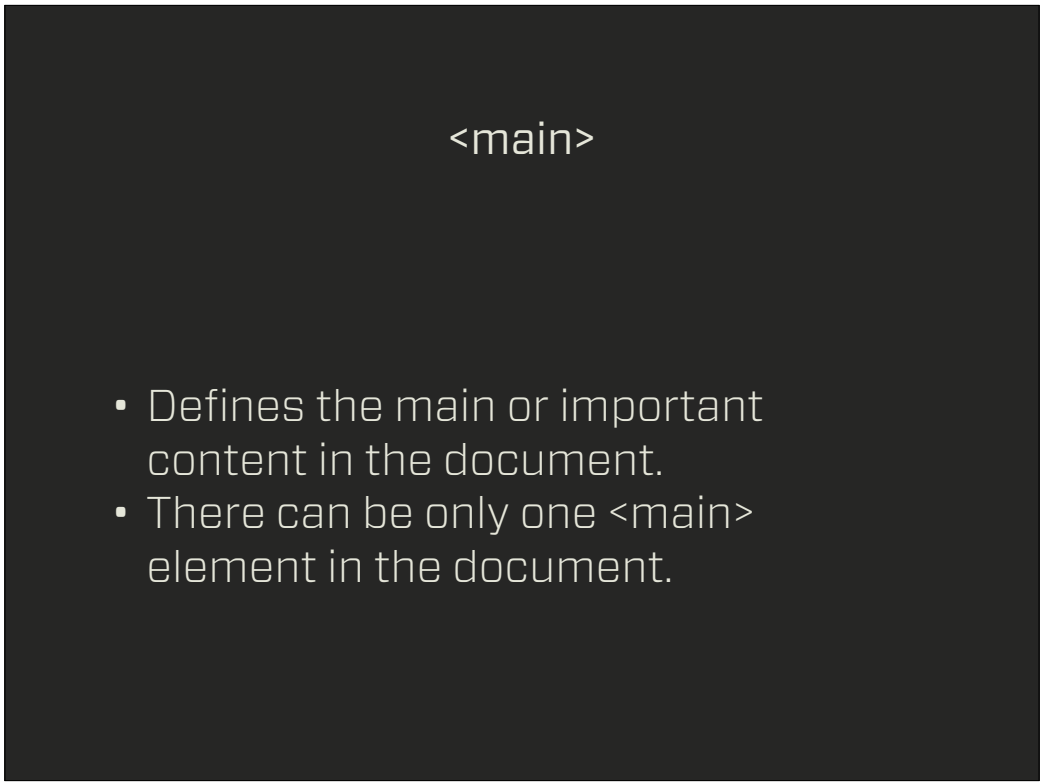


Lets look at some key elements now available in HTML5

<header>

- Not to be confused with <head>
- Defines the header of a page or section. It often contains a logo, the title of the Web site, and a navigational table of content.

Lets look at some key elements now available in HTML5

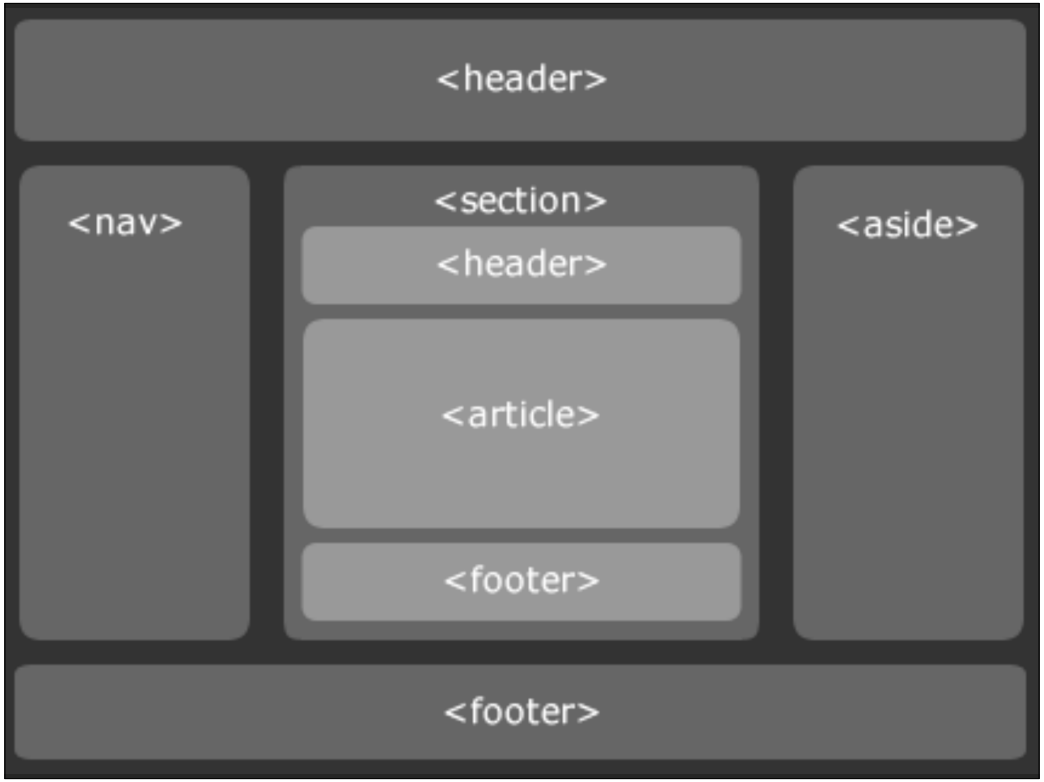


Lets look at some key elements now available in HTML5

<footer>

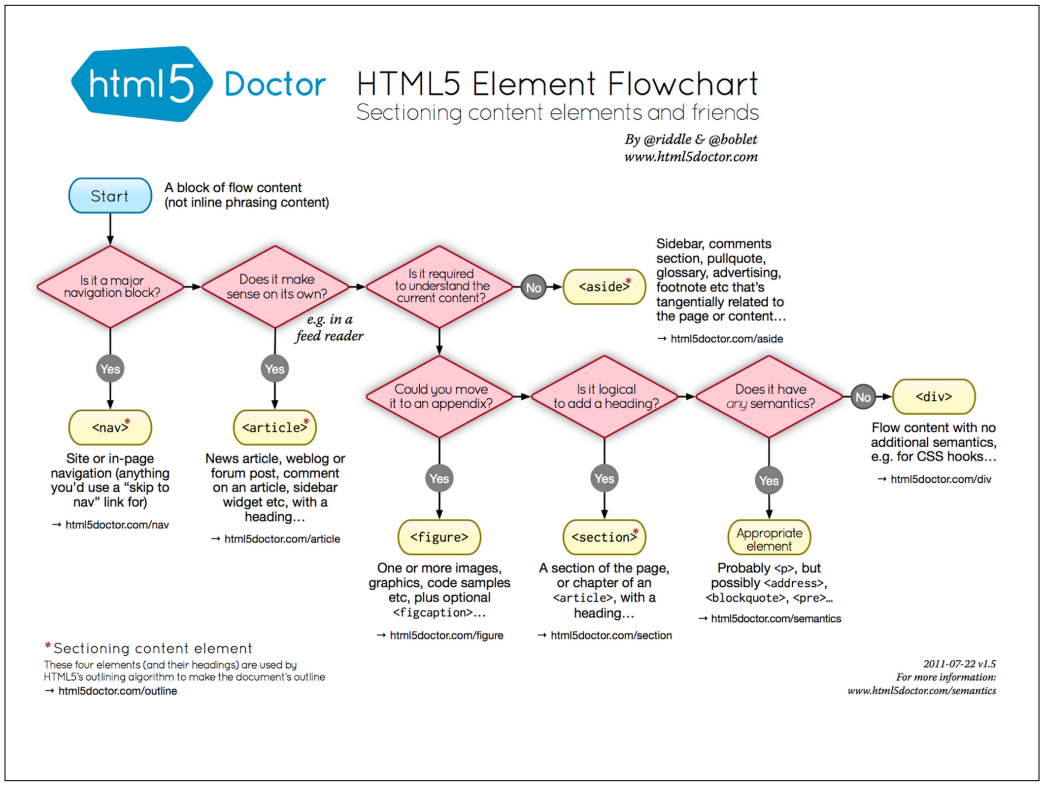
- Defines the footer for a page or section. It often contains a copyright notice, some links to legal information, or addresses to give feedback.

Lets look at some key elements now available in HTML5



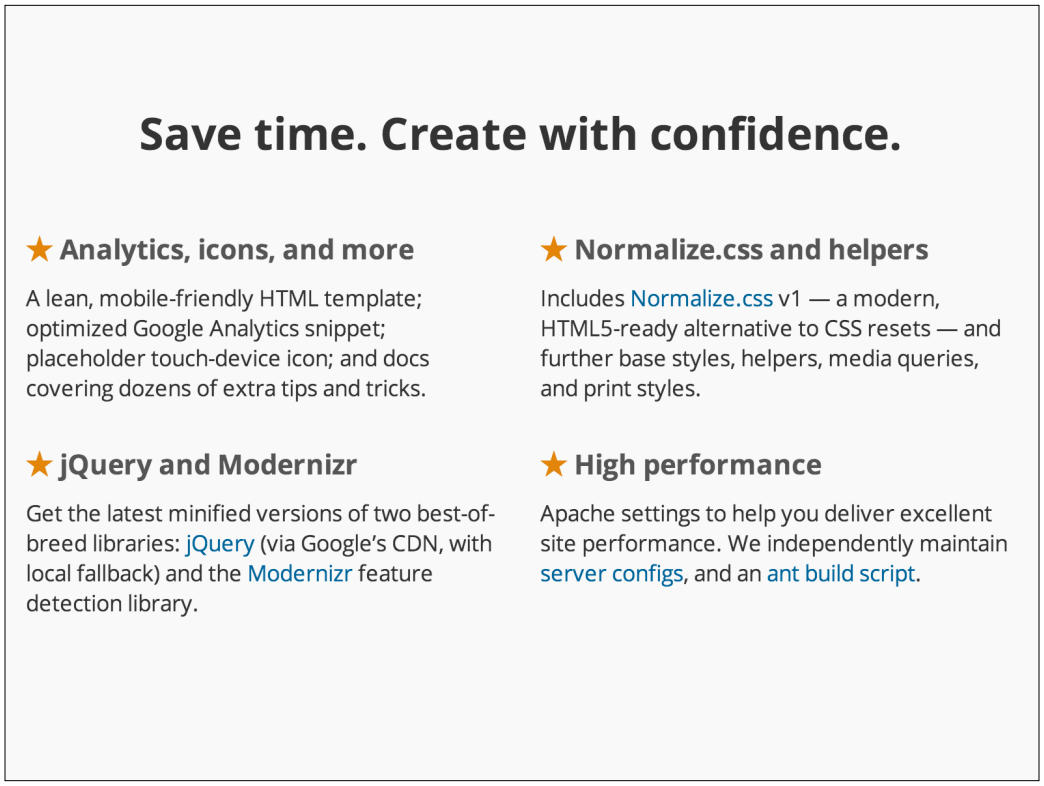
HTML Structure when you put it all together…

More overview of HTML5, courtesy of Dale Sande, Code Fellows Front-end instructor: https://speakerdeck.com/anotheruiguy/html5-platform-overview



How do you decide what to use?

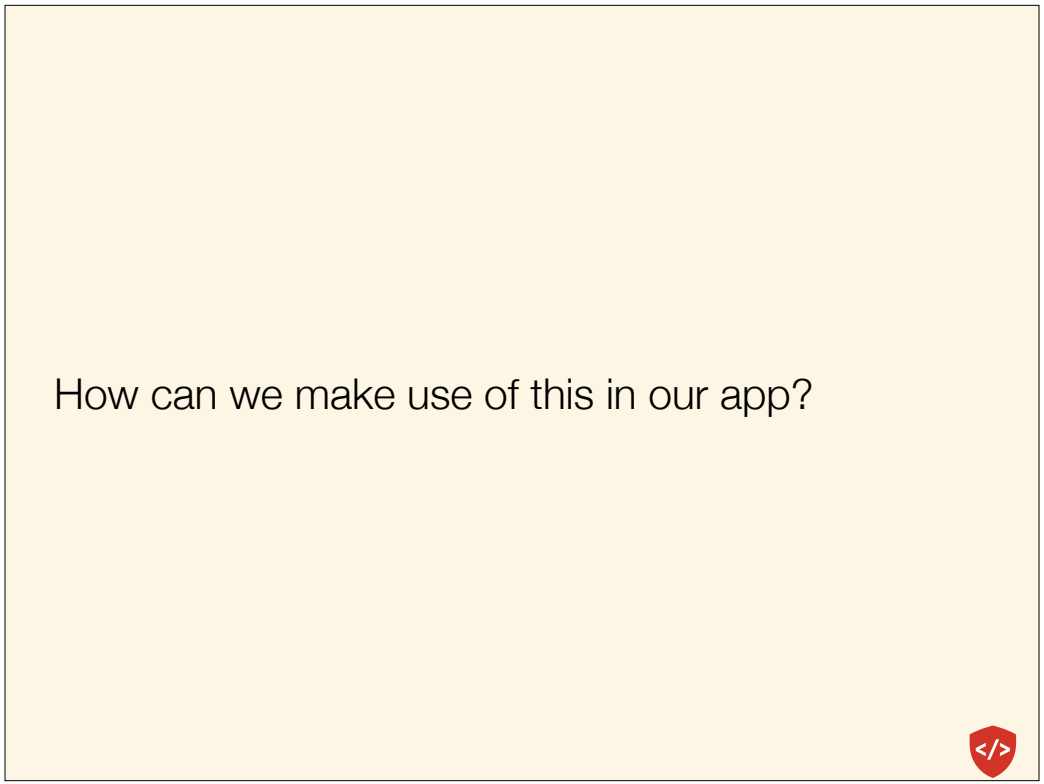
More overview of HTML5, courtesy of Dale Sande, Code Fellows Front-end instructor: https://speakerdeck.com/anotheruiguy/html5-platform-overview



Another great resource: HTML5 Boilerplate!

Read more and watch the video:
http://html5boilerplate.com/

Source code at: https://github.com/h5bp/html5-boilerplate (look at the index.html file for the main details of what it is doing).



HTML-ify your project!

More overview of HTML5, courtesy of Dale Sande, Code Fellows Front-end instructor: https://speakerdeck.com/anotheruiguy/html5-platform-overview