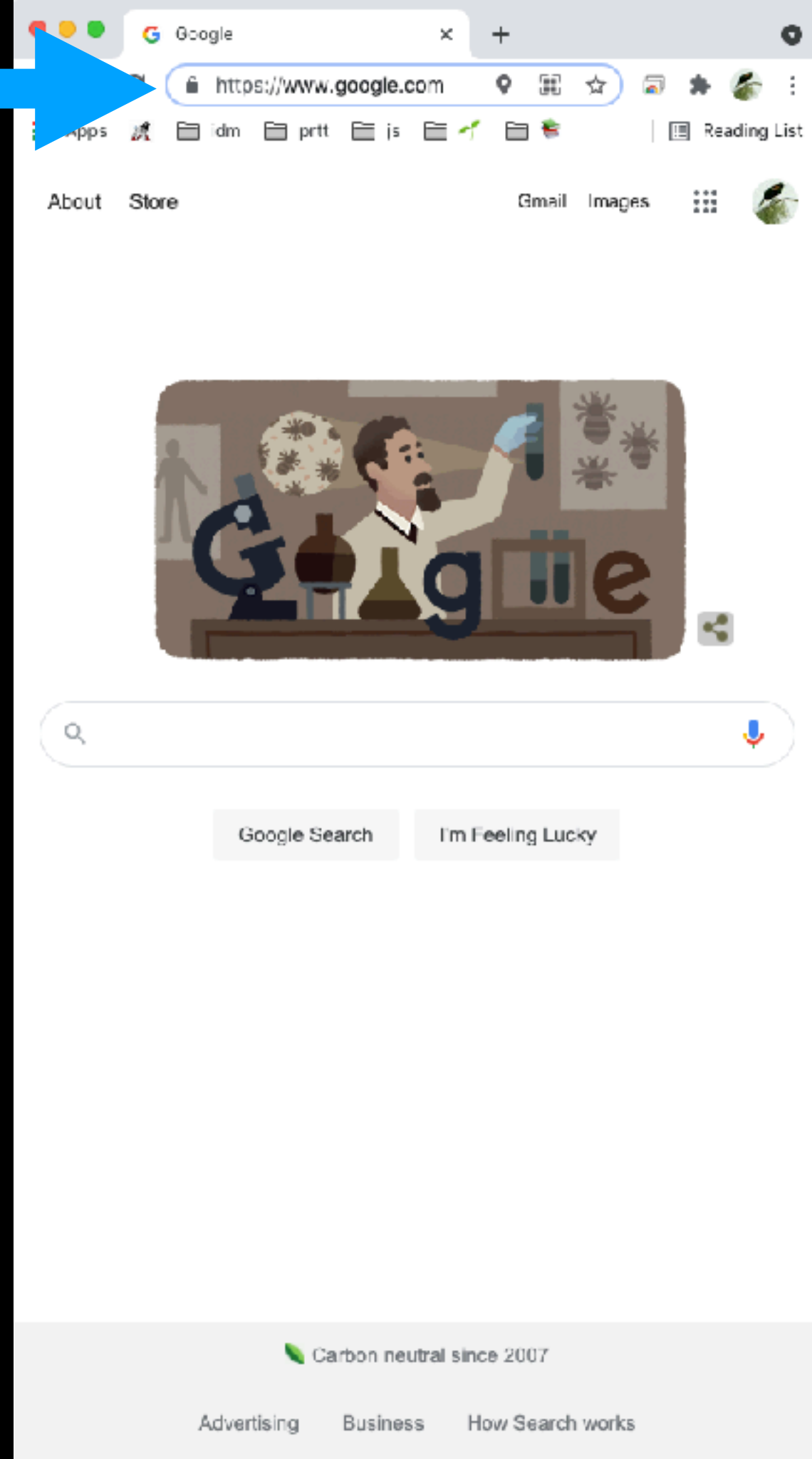


www

The web is an information system where documents and other resources are identified by **Uniform Resource Locators** which may be interlinked by hyperlinks, and are accessible over the **internet**.

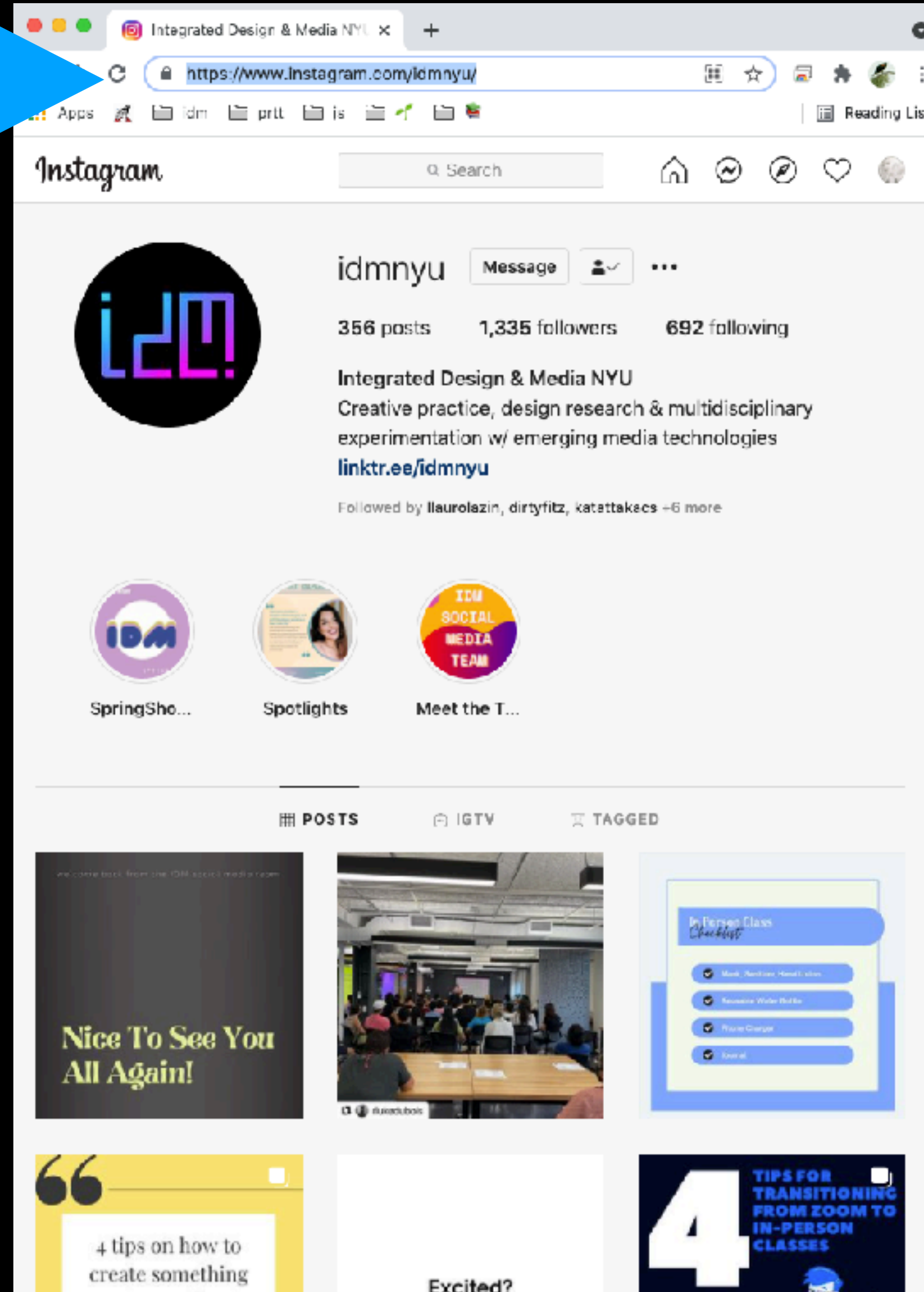
we colloquially call a **url** an address

the **url** for this webpage is: <https://www.google.com>



The resources fo the Web are transferred via the **Hypertext Transfer Protocol** may be accessed by users by a software application called a **browser**, and are published by a software application called a **server**.

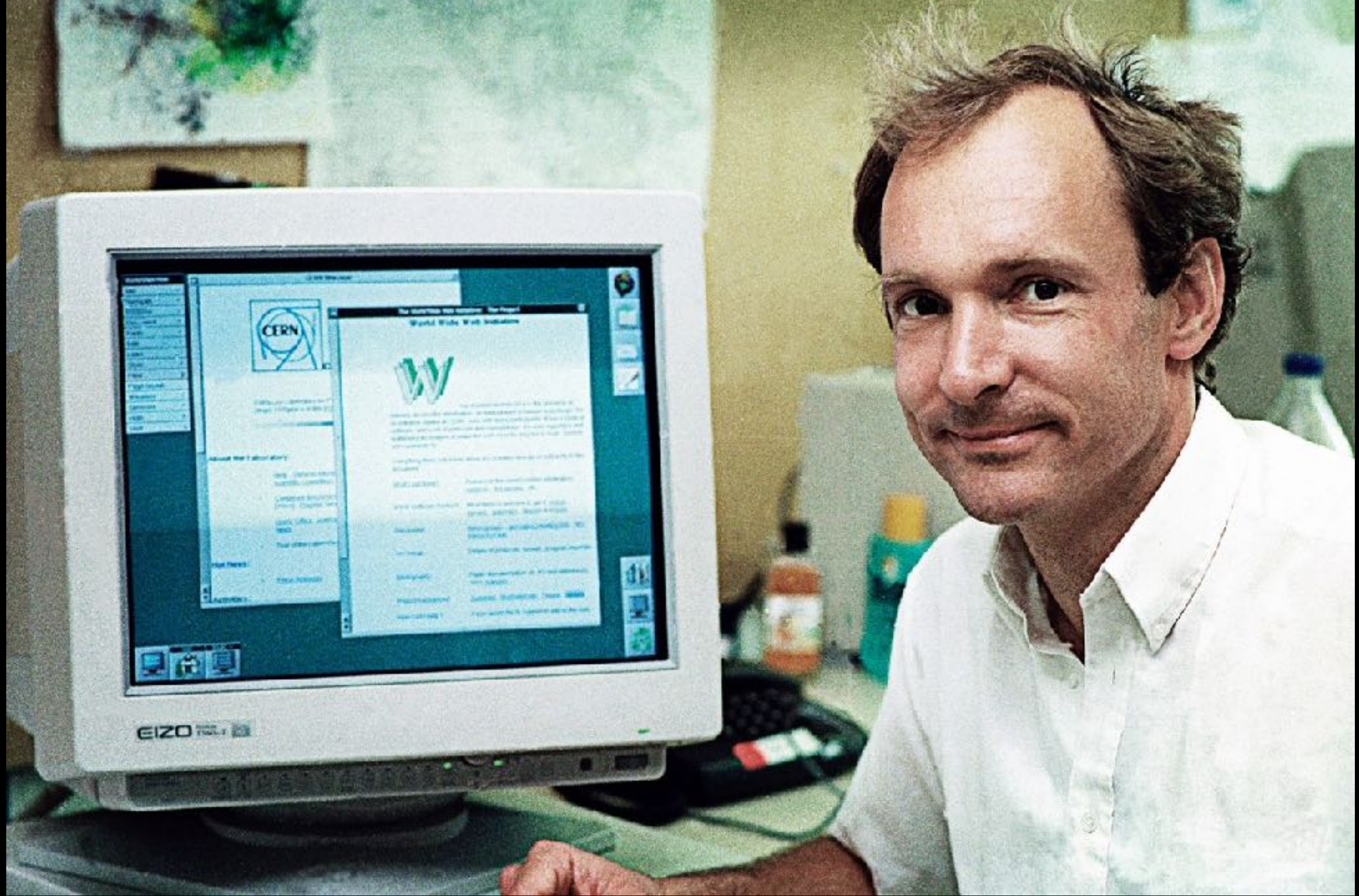
web apps are called that for a reason



the url for this webpage is:

<https://www.instagram.com/idmnyu/>





**1989** - Tim Berners Lee invents the WWW

**1990** - Tim Berners writes the first web browser at CERN in Switzerland

**1991** - The browser released outside CERN + to the general public

**1993 -4** - Web begins to enter everyday use



# UNIX

operating system

Bell Labs 1969 (proprietary)

1973 - released outside Bell

## Unix Commands

**pwd** - "present working directory" - where am i in the system!?.

**ls** - list everything in this directory

**cd** - "change diretory [insert name of folder]" - only if the folder is a child of the one you are in

type in just **cd** - terminal will direct you to the root of the drive  
powershell will do nothing just leave you in directory

**mkdir** - "make directory" - make a folder

**rm** - "remove file" - deletes a file forever (**not the trash**). doesn't work on directories

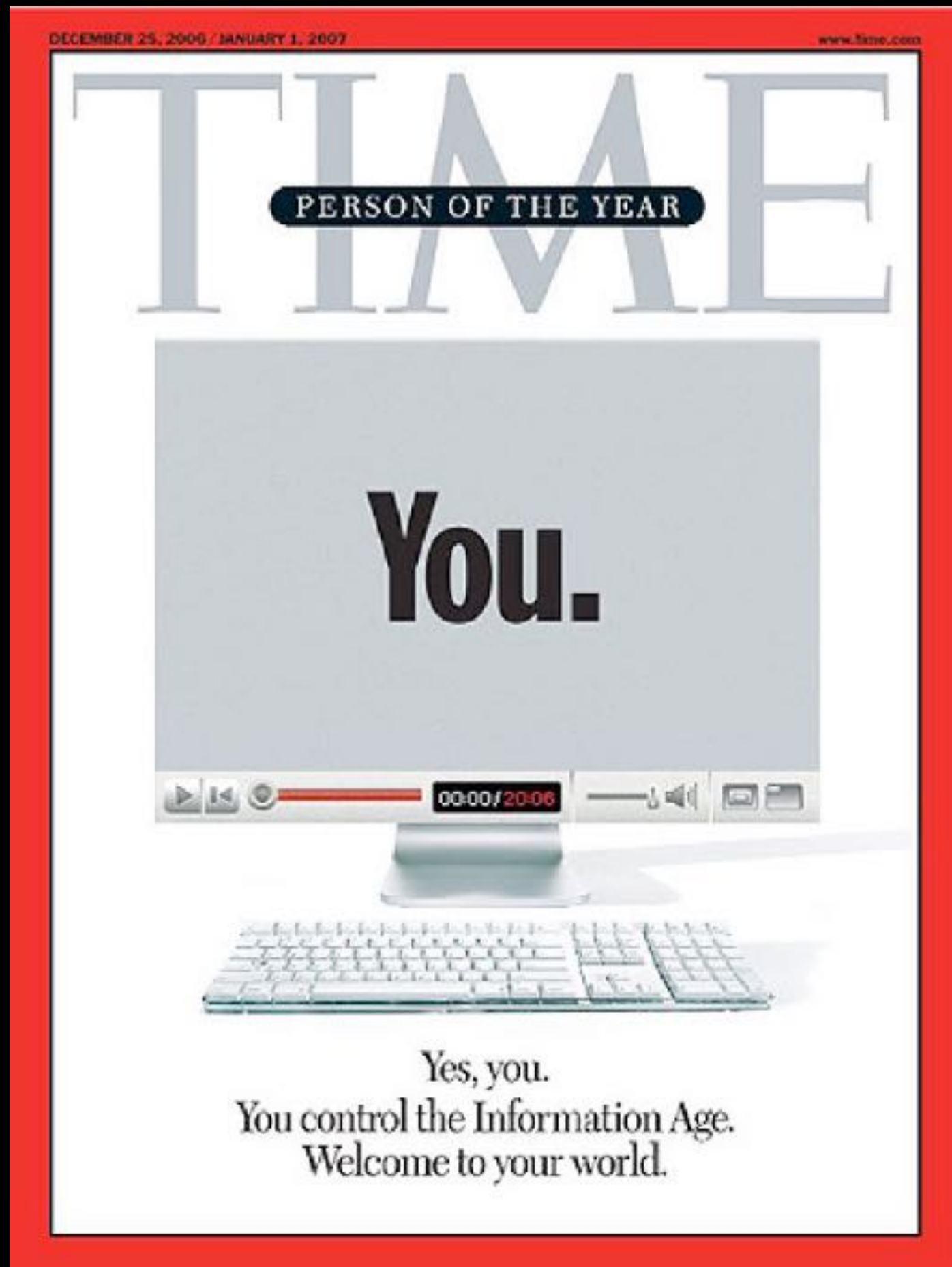
**mv** - "move" - move file

**..** - goes back to the parent directory



**Youtube, init(): February 2005**

**You: Person of the Year  
Time Magazine, December 2006**



# Web. 1.0

HTML + CSS + JS

1989 - 2004

## HTML - Hyper Text Mark Up

is a grammar for structuring web pages. It defines paragraphs, headings, data tables + media elements. HTML describes the content of the page - not how it looks.

## CSS - Cascading Style Sheet

rules for styling a web page. Setting colors, positioning, typeface, layout and even timing. It can be used to consider the design of your page across different platforms and screen sizes.

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>  Intro to CSS</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>
```

```
    This Webpage though....
```

```
    </h1>
```

```
    <p>
```

```
    here is a page of text. here is a page of text. here...
```

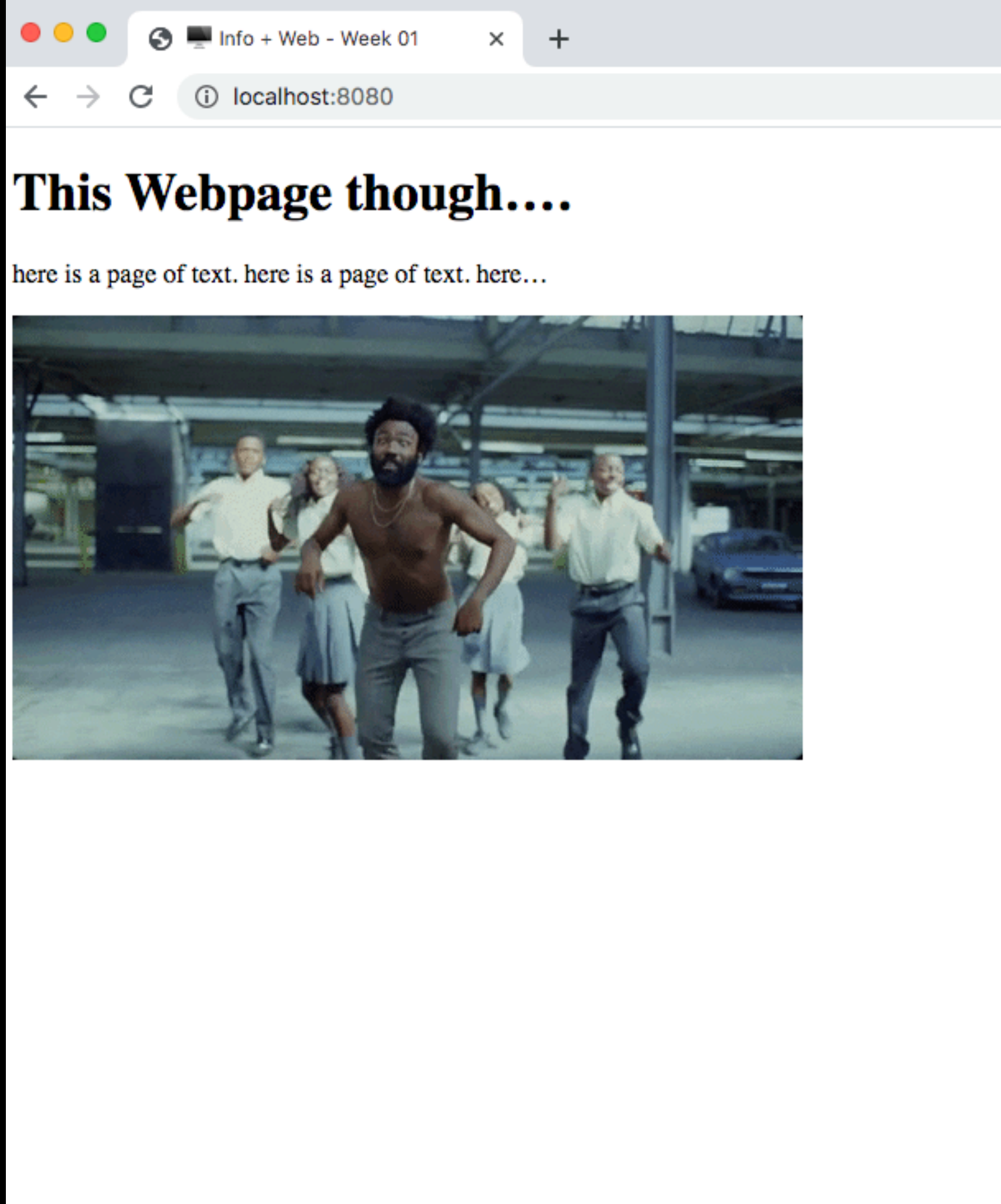
```
    </p>
```

```
    <img src = "https://media.giphy.com/media/  
    IYf4uAJEWVo1FffnVD/giphy.gif">
```

```
  </body>
```

```
</html>
```

html looks like this:





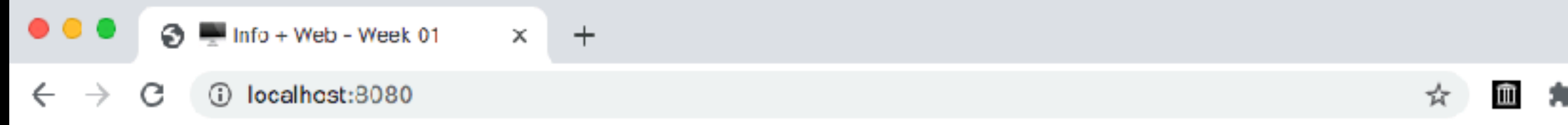
## CSS - Cascading Style Sheet

```
h1 {  
    color: #FF4500  
}
```

```
body {  
    background-color: #000080;  
    font-family: sans-serif;  
    color: rgb(255,255,255);  
}
```

```
body p {  
    font-family: 'Courier New';  
}
```

html + css:



*This Webpage though7.*

here is a page of text. here is a page of text. here...



The key to understanding how **HTML** + **CSS** works is to imagine that there is an invisible box around every **HTML** element.

Block level elements are outlined w/ red + inline elements in green.

**<body>** creates 1st box, then **<h1>**, **<h2>**, **<p>**, **<i>** + **<a>** each create their own boxes within it.

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

## < HTML >

### 3 categories of HTML elements

1 - **block**: large blocks of content has height + width  
**<p>, <h1>, <blockquote>, <ol>, <ul>, <table>**

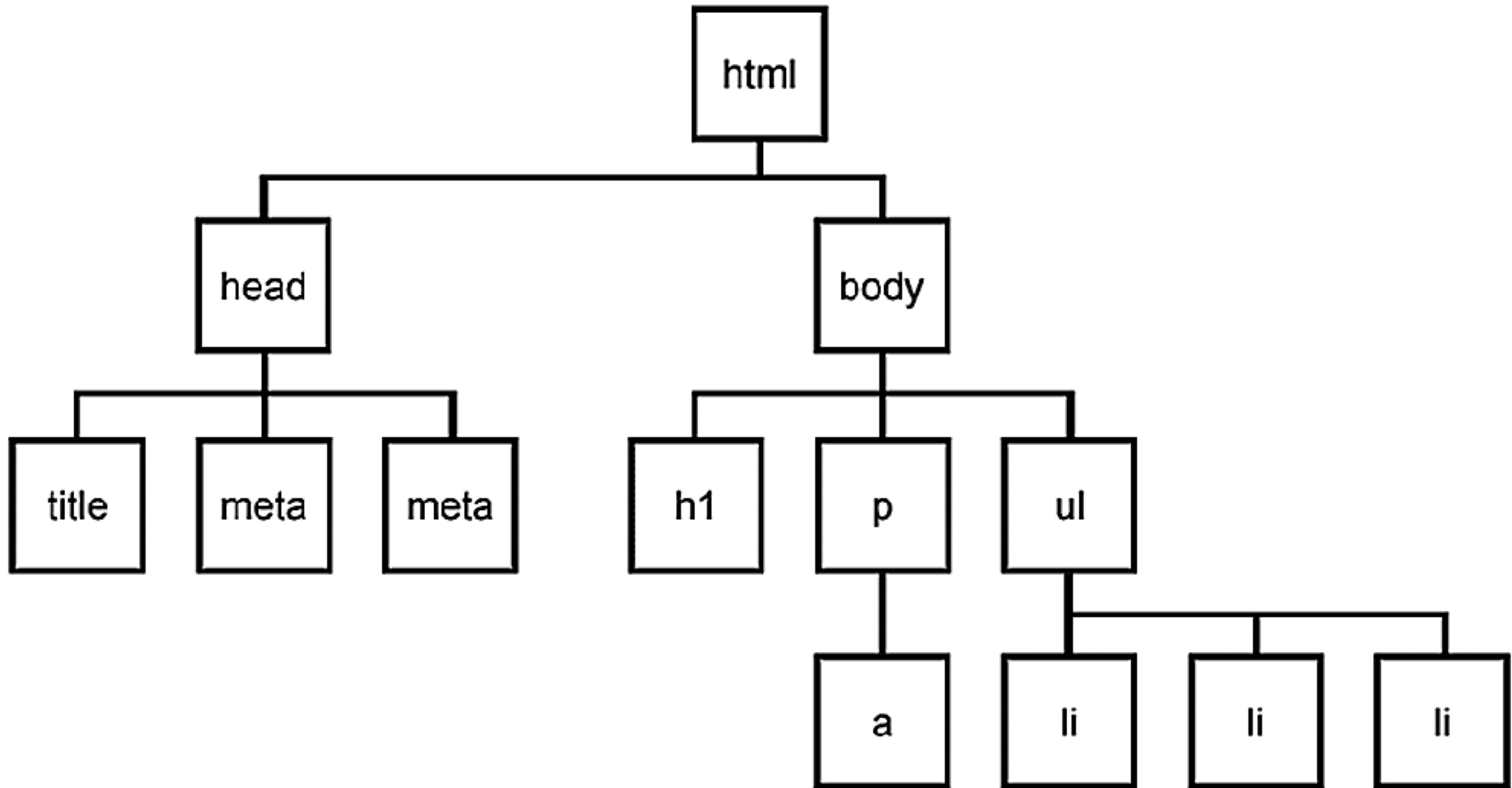
2 - **inline**: small about of content, no height or width  
**<a>, <em>, <strong>, <br>, <span>, <time>**

a. **inline block**: inline content w/ height + width

3 - **metadata**: information about the page, usually not visible  
**<title>, <meta>, <script>**



# Parent / Child Element Structure



The `<head>` element contains the metadata for a web page. Metadata is information about the page that isn't displayed directly on the web page. Unlike the information inside of the `<body>` tag, the metadata in the head is information about the page itself.

# Text tags

- **h1, h2, h3, h4, h5, h6** are text tags for headings
- **p** is a tag for paragraphs
- **b** is for bold, **i** is for italics
- **<strong>** is for **bold** **<em>** is for *italics*
- **ul, ol, li** are used for making lists
  - **ul**: unordered lists
  - **ol**: ordered lists
  - **li**: an individual list tag
- **<br/>** will break to a new line

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

`<div>`s can contain any text or other HTML elements, such as links, images, or videos. Remember to always add two spaces of indentation when you nest elements inside of `<div>`s for better readability.



# Semantic HTML

HTML should be coded to represent the data that will be populated and not based on its default presentation styling. Presentation (how it should look), is the sole responsibility of CSS.

Some of the benefits from writing semantic markup are as follows:

- Search engines will consider its contents as important keywords to influence the page's search rankings (see SEO)
- Screen readers can use it as a signpost to help visually impaired users navigate a page
- Finding blocks of meaningful code is significantly easier than searching through endless divs with or without semantic or namespaced classes
- Suggests to the developer the type of data that will be populated
- Semantic naming mirrors proper custom element/component naming

<https://developer.mozilla.org/en-US/docs/Glossary/Semantics>

`<p>`

`<h1>` - `<h6>`

## Semantic elements

`<main>`

dominant content of the `<body>` element

`<article>`

A document, page or site. This is usually a root container element after body

`<section>`

Generic section of a document

`<header>`

Intro section of a document

`<footer>`

Footer at end of a document or section

`<nav>`

Navigational section

Use these **before** div when appropriate.

# Semantic elements

## `<aside>`

represents a portion of a document whose content is only indirectly related to the document's main content. Asides are frequently presented as sidebars or call-out boxes.

## `<details>`

creates a disclosure widget in which information is visible only when the widget is toggled into an "open" state.

## `<figcaption>`

represents a caption or legend describing the rest of the contents of its parent `<figure>` element.

## `<mark>`

represents text which is marked or highlighted for reference or notation purposes, due to the marked passage's relevance or importance in the enclosing context.

## `<summary>`

element specifies a summary, caption, or legend for a `<details>` element's disclosure box. Clicking the `<summary>` element toggles the state of the parent `<details>` element open and closed.

## `<time>`

represents a specific period in time.

The `<em>` tag will generally render as *italic* emphasis.

The `<strong>` will generally render as bold emphasis.



`<br>`

The line break element is unique because it is only composed of a starting tag. You can use it anywhere within your HTML code and a line break will be shown in the browser.

tag attribute value

```
<video src= "filepath/file.mov" alt= "this is the video" height="300"></video>
```

```
<html attribute= "value" attribute= "value" attribute= "value"> </html>
```

# Absolute Links

OPENING  
LINK TAG

URL WE ARE  
DIRECTED TO



TEXT WE  
CLICK ON

CLOSING  
TAG

```
<a href="https://www.youtube.com/watch?v=qcnnI6HD6DU"> absolute link</a>
```

**< a href** — stands for *hyperlink reference*

# RELATIVE Links

Linking to pages on the same site. It's faster to simply direct to the file path. Rather than put the entire url address.

if the file is in the same folder: `<a href="index.html">Homepage</a>`

if the file is in the parent folder: `<a href="../index.html">Homepage</a>`

if the file is in the child folder: `<a href="images/photos.html">Photos</a>`

id attribute: `<a href="#thisID">Jump to a different element on page</a>`

```
<li><a href="#theFoot">id attribute link</a></li>
```

# Why **index.html**?

The main homepage of a site written in HTML (and the homepage of each section in a child folder) is called index.html.

Web servers are usually set up to return the index.html file if no file name is specified. Therefore, it's always a good idea to name your "home" page index.html

The `<img>` tag has a required attribute called `src`. The `src` attribute must be set to the image's source, or the location of the image. In some cases, the value of `src` must be the *uniform resource locator* (URL) of the image. A URL is the web address or local address where a file is stored.



# Images: relative vs. absolute url

```
<img src= "images/potato07.png" alt= "spud" >
```

```
<img src= "https://pngriver.com/wp-content/uploads/2018/04/Download-Potato-PNG-Pic.png" alt= "spud" >
```

The **<img>** tag is for images, which can be on your local directory or on another webpage.  
Read all about **<img>** tag [here](#). The same goes for **<video>** + **<audio>** tags

The **alt** attribute, which means alternative text, brings meaning to the images on our sites. The **alt** attribute can be added to the image tag just like the **src** attribute. The value of **alt** should be a description of the image.

```

```

1. If an image fails to load on a web page, a user can mouse over the area originally intended for the image and read a brief description of the image. This is made possible by the description you provide in the **alt** attribute.
2. Visually impaired users often browse the web with the aid of screen reading software. When you include the **alt** attribute, the screen reading software can read the image's description out loud to the visually impaired user.
3. The **alt** attribute also plays a role in Search Engine Optimization (SEO), because search engines cannot "see" the images on websites as they crawl the internet. Having descriptive **alt** attributes can improve the ranking of your site.

Like the `<img>` tag, the `<video>` tag requires a `src` attribute with a link to the video source.

Unlike the `<img>` tag however, the `<video>` element requires an opening and a closing tag.

# <video /> structure

main  
tag

poster

width/  
height

control  
attributes

```
<body>

  <!-- Adding video tag -->
  <video poster="media/listen.jpg" width="400px" preload loop autoplay controls>
    <source src="media/listen.mp4"/>
    <source src="media/listen.webm"/>
  </video>
</body>
```

different  
sources

After the `src` attribute, the `width` and `height` attributes are used to set the size of the video displayed in the browser.

The `controls` attribute instructs the browser to include basic video controls: pause, play and skip. Unlike the `<img>` tag however, the `<video>` element requires an opening and a closing tag.

The text, "Video not supported", between the opening and closing video tags will only be displayed if the browser is unable to load the video.



# <audio /> structure

main  
tag

control  
attributes

```
<audio controls autoplay loop>
  <source src="audio/virginia.mp3" />
  <source src="audio/virginia.ogg" />
  <p>This browser does not support this audio format</p>
</audio>
```

different  
sources

text is the  
file cannot  
be found

## Some Media Attributes

**Preload** - what preloads when the page loads

**Controls** - if the play/stop buttons are visible

**Autoplay** - if the video should start playing  
automatically

**Loop** - if the video should loop on completion

# Attributes

If we want to expand an element's tag, we can do so using an attribute. Attributes are content added to the opening tag of an element and can be used in several different ways, from providing information to changing styling. Attributes are made up of the following two parts:

- 1) The **name** of the attribute
- 2) The **value** of the attribute

One commonly used attribute is the `id`.

We can use the `id` attribute to specify different content (such as `<div>`s) and is really helpful when you use an element more than once.

```
<div id="intro">  
  <h1>Technology</h1>  
</div>
```

**<span>** contains short pieces of text or other HTML. They are used to separate small pieces of content that are on the same line as other content.

```
<div>  
  <h1>Technology</h1>  
</div>
```

```
<div>  
  <p> Wherever there's a  
    <span>computer</span>, there's a skilled  
    person developing, maintaining, hacking,  
    advancing or simply using it.</p>  
</div>
```