Getting Started with the Web

Assoc. Prof. Dr. Kanda Runapongsa Saikaew
(krunapon@kku.ac.th)
Department of Computer Engineering
Khon Kaen University

Agenda

- Installing basic software
- What will your website look like?
- Dealing with files
- HTML basics
- CSS basics
- JavaScript basics
- Publishing your website
- How the Web works

What tools do the professionals use? (1/3)

- **A text editor**, to write code in
  - This could be a text editor (e.g. Sublime, Brackets, Atom or Visual Studio Code), or a hybrid editor (e.g. Dreamweaver or WebStorm)
- **Web browsers**, to test code in
  - Currently, the most-used browsers are Firefox, Chrome, Opera, Safari, Internet Explorer, and Microsoft Edge
  - You should also test how your site performs on mobile devices

What tools do the professionals use? (2/3)

- **A graphics editor**, like GIMP, Paint.NET, or Photoshop, to make images for your web pages.
- **A version control system**, to manage files on servers, collaborate on a project with a team, share code and assets, and avoid editing conflicts
  - Git is the most popular version control system and the GitHub code hosting service, based on Git, is also very popular.

What tools do the professionals use? (3/3)

- **An FTP program**, used on older web hosting accounts to manage files on servers (Git is increasingly replacing FTP for this purpose)
  - There are loads of (S)FTP programs available including Cyberduck, Fetch, and FileZilla.
- **An automation system,** like Grunt or Gulp to automatically perform repetitive tasks, such as minifying code and running tests.
- Templates, libraries, frameworks, etc., to speed up writing common functionality.

What tools do I actually need, right now?

- Installing a text editor
  - For web development, you can probably do better than Notepad or TextEdit. We recommend starting out with Brackets, which is a free editor that offers live previews and code hints
- Installing modern web browsers
- Installing a local web server

How do you set up a local testing server?

- Local files vs. remote files
  - If the web address starts with file:// followed by the path of file on your local hard drive, a local file is being used
  - If the web address starts with http:// or https://, it shows that the file has been received via HTTP

The problem with testing local files

- Some examples won't run if you open them as local files
  - They feature asynchronous requests
    - Some browsers (including Chrome) will not run async request if you run the example from a local file
  - They feature a server-side language
    - Server-side languages (such as PHP or Python) require a special server to interpret the code and deliver the results

Running a simple local HTTP server (1/2)

- One of the easiest ways to run a simple local HTTP server is to use Python's SimpleHTTPServer module

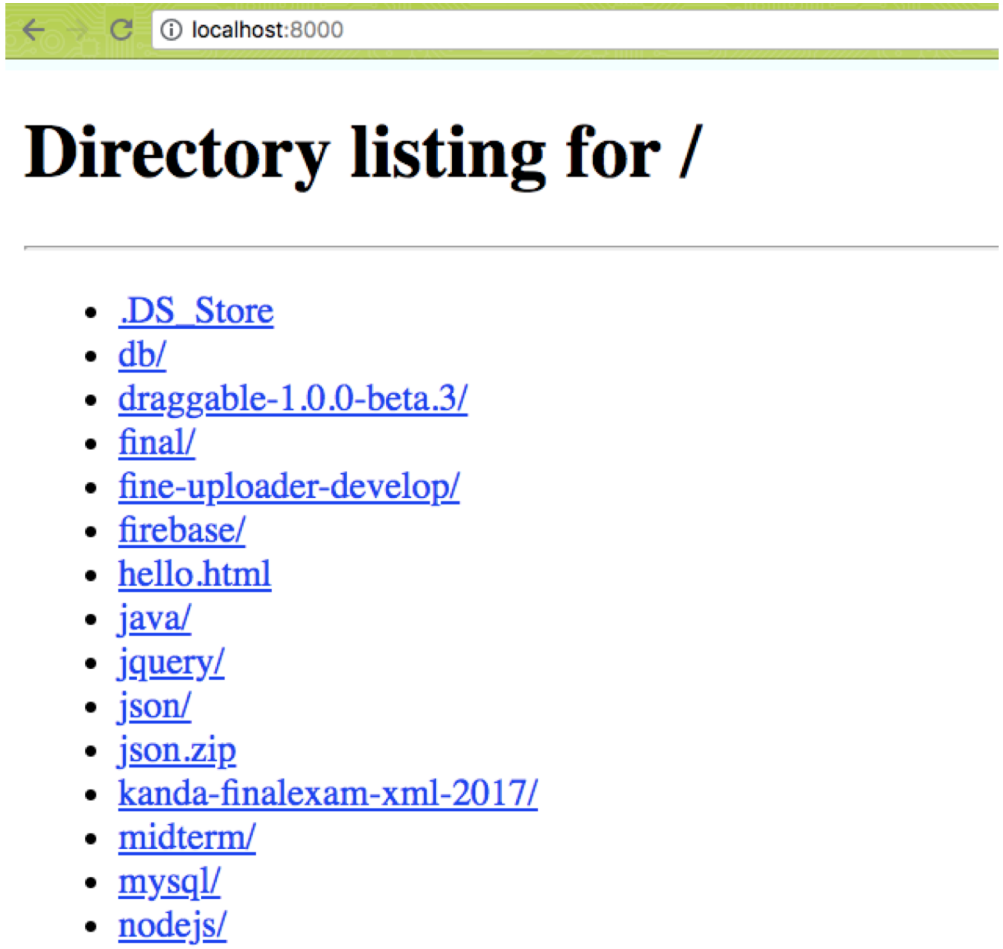
1) Install Python
   - Go to python.org
   - Under the Download section, click the link for Python "3.xxx".
   - At the bottom of the page, choose the *Windows x86 executable installer* and download it.
   - When it has downloaded, run it.
   - On the first installer page, make sure you check the "Add Python 3.xxx to PATH" checkbox.
   - Click *Install*, then click *Close* when the installation has finished.

Running a simple local HTTP server (2/2)

- Enter the command to start up the server in that directory

```
# If Python version returned above is 3.X
python -m http.server
# If Python version returned above is 2.X
python -m SimpleHTTPServer
```

A simple local HTTP server



Agenda

- Installing basic software
- **What will your website look like?**
- Dealing with files
- HTML basics
- CSS basics
- JavaScript basics
- Publishing your website
- How the Web works

What will your website look like?

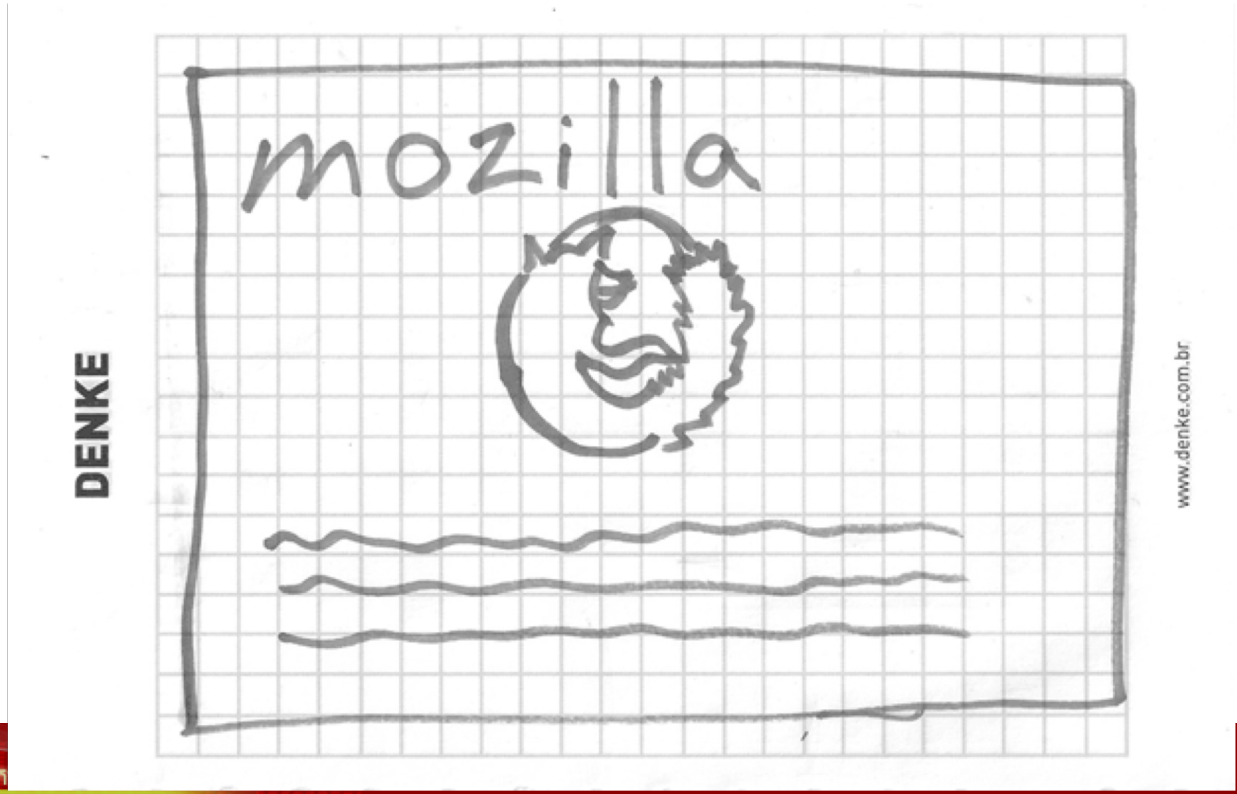
- First things first: planning
- Sketching out your design
- Choosing your assets

First things first: planning

1. What is your website about?
2. What information are you presenting on the subject?
   - Write a title and a few paragraphs
   - Think of an image you'd like to show on your page
3. What does your website look like?
   - What's the background color? What kind of font is appropriate: formal, cartoony, bold and loud, subtle?

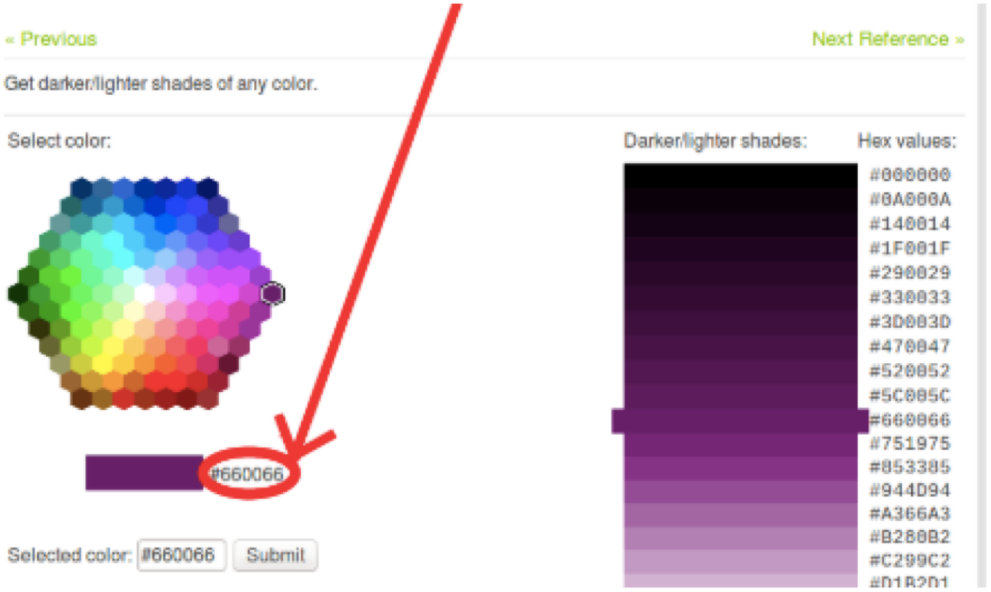
Sketching out your design

- Get a pen and paper and sketch out roughly how you want
- For your first simple webpage, there's not much to sketch out, but you should get in the habit of doing this now



Choosing your assets >> Theme color

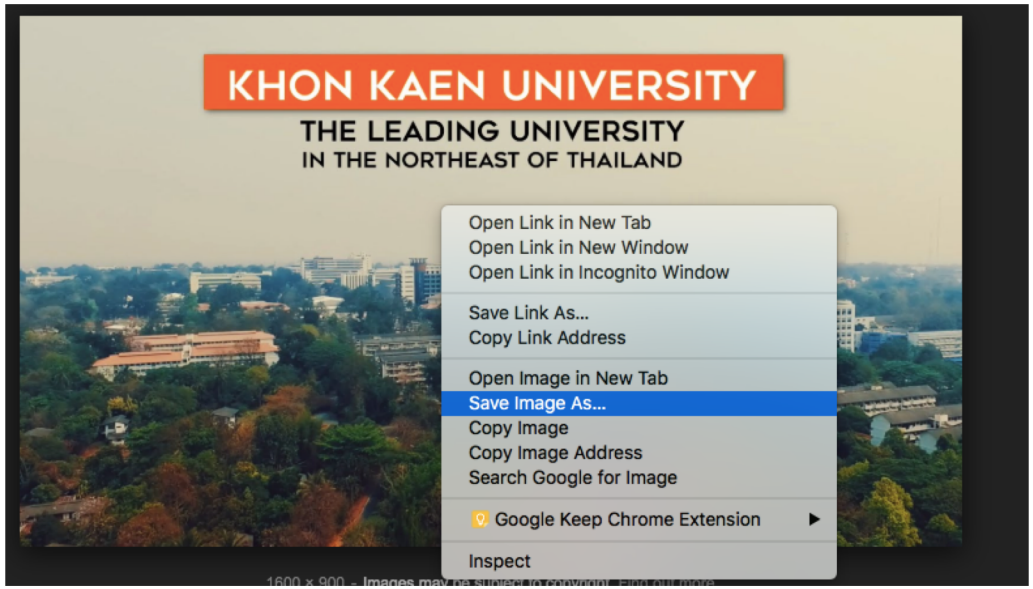
- To choose a color, go to [the Color Picker] and find a color you like. When you click on a color, you'll see a strange six-character code like #660066
- That's called a *hex(adecimal) code*, and represents your color. Copy the code own somewhere safe for now



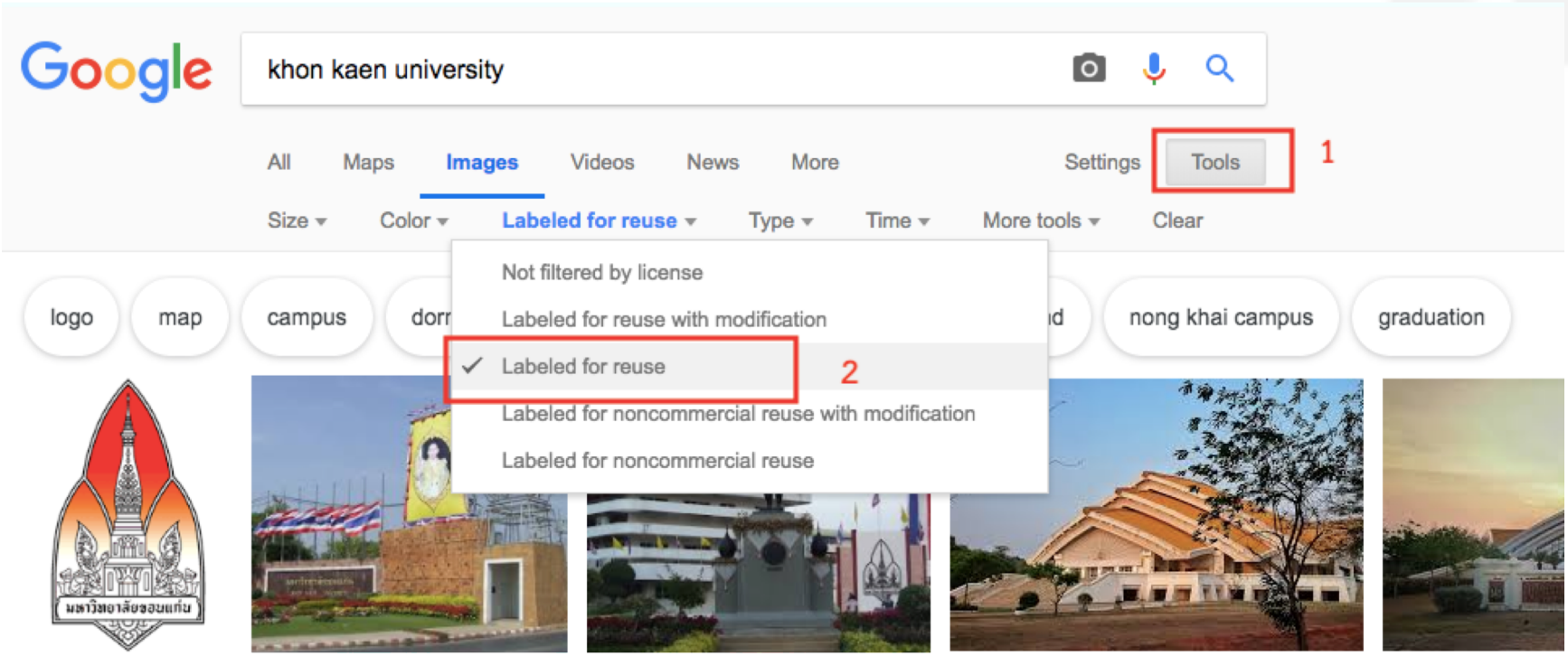
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Colors/Color_picker_tool

Choosing your assets >> Images

- To choose an image, go to https://images.google.com/ and search for something suitable.
- When you find the image you want, click on the image, choose *Save Image As ….*



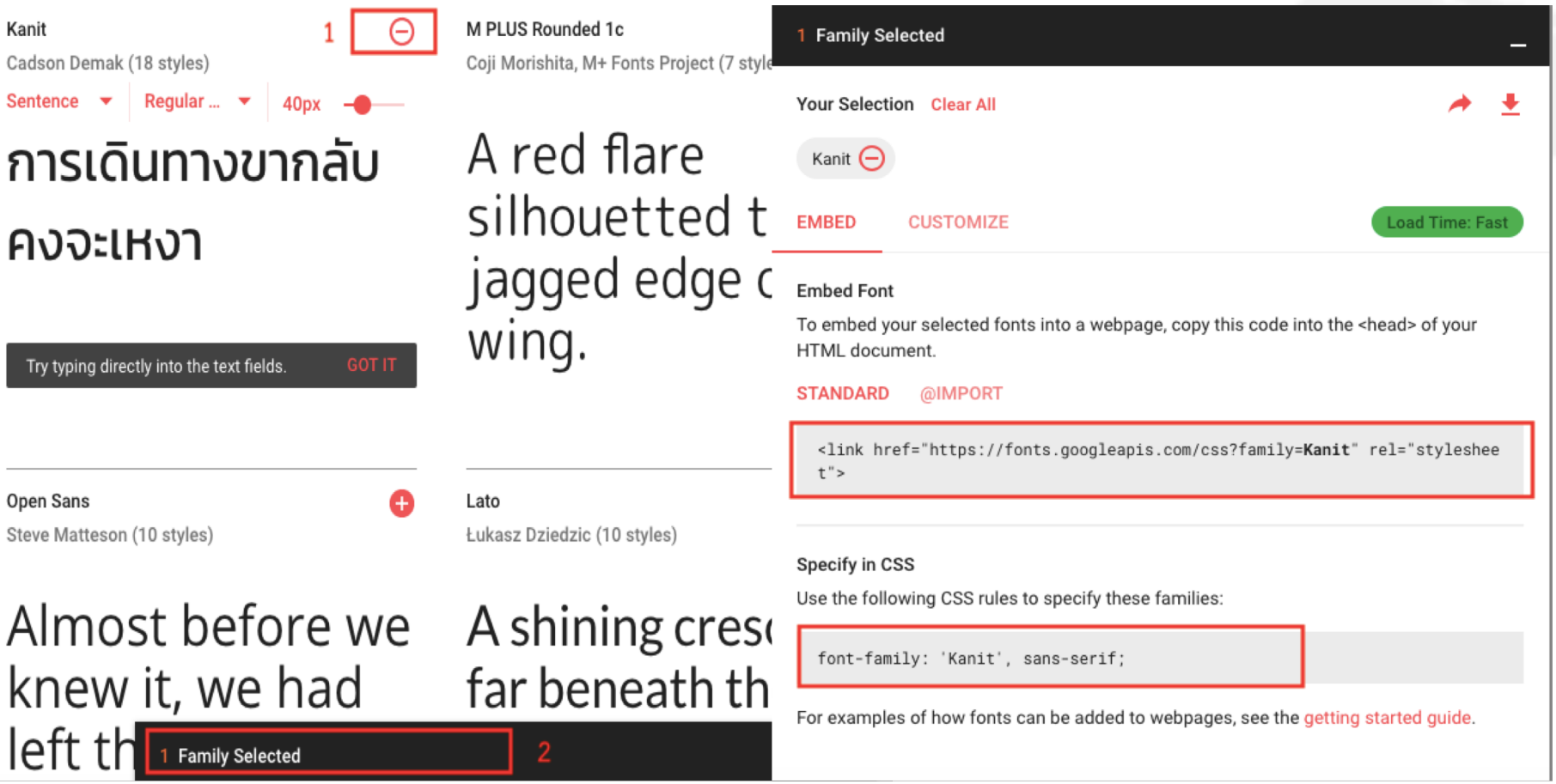
Checking image usage rights



Choosing your assets >> Fonts

- Go to https://fonts.google.com/ and scroll down the list until you find one you like. You can also use the controls on the right to further filter the results.
- Click the "plus" (Add to) icon next to the font you want.
- Click the "* Family Selected" button in the panel at the bottom of the page ("*" depends on how many fonts you selected).
- In the popup box, you can see and copy the lines of code Google gives you into your text editor to save for later.

Using Google fonts



Agenda

- Installing basic software
- What will your website look like?
- **Dealing with files**
- HTML basics
- CSS basics
- JavaScript basics
- Publishing your website
- How the Web works

Where should your website live on your computer?

- Keep all the related files in a single folder that mirrors the published website's file structure on the server
- Choose a place to store your website projects
  - Here, create a new folder called web-projects (or similar). This is where all your website projects will live.
- Inside this first folder, create another folder to store your first website in
  - Call it test-site (or something more imaginative).

An aside on casing and spacing

- Many computers, particularly web servers, are case-sensitive
- Browsers, web servers, and programming languages do not handle spaces consistently
  - It's better to separate words with dashes, rather than underscores: my-file.html vs. my_file.html
  - The Google search engine treats a hyphen as a word separator, but does not treat an underscore that way.

What structure should your website have? (1/2)

- **index.html**
  - This file will generally contain your homepage content
  - Using your text editor, create a new file called index.html and save it just inside your test-site folder.
- **images folder**
  - This folder will contain all the images
  - Create a folder called images, inside your test-site folder.

What structure should your website have? (2/2)

- **styles folder**: This folder will contain the CSS code used to style your content
  - Create a folder called styles, inside your test-site folder.
- **scripts folder**: This folder will contain all the JavaScript code used to add interactive functionality to your site
  - Create a folder called scripts, inside your test-site folder.

File paths

- To make files talk to one another, you have to provide a file path between them
  - Basically a route so one file knows where another one is
- To demonstrate this, we will insert a little bit of HTML into our index.html file

Sample index.html

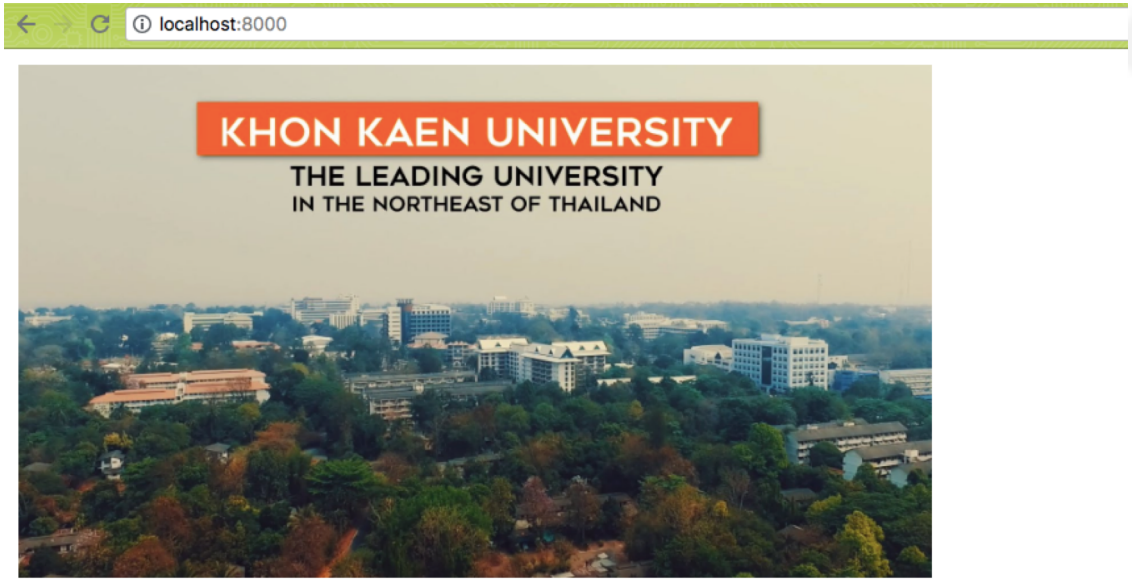
1) Copy the image you chose earlier into your images folder

2) Open up your index.html file, and insert the following code into the file exactly as shown

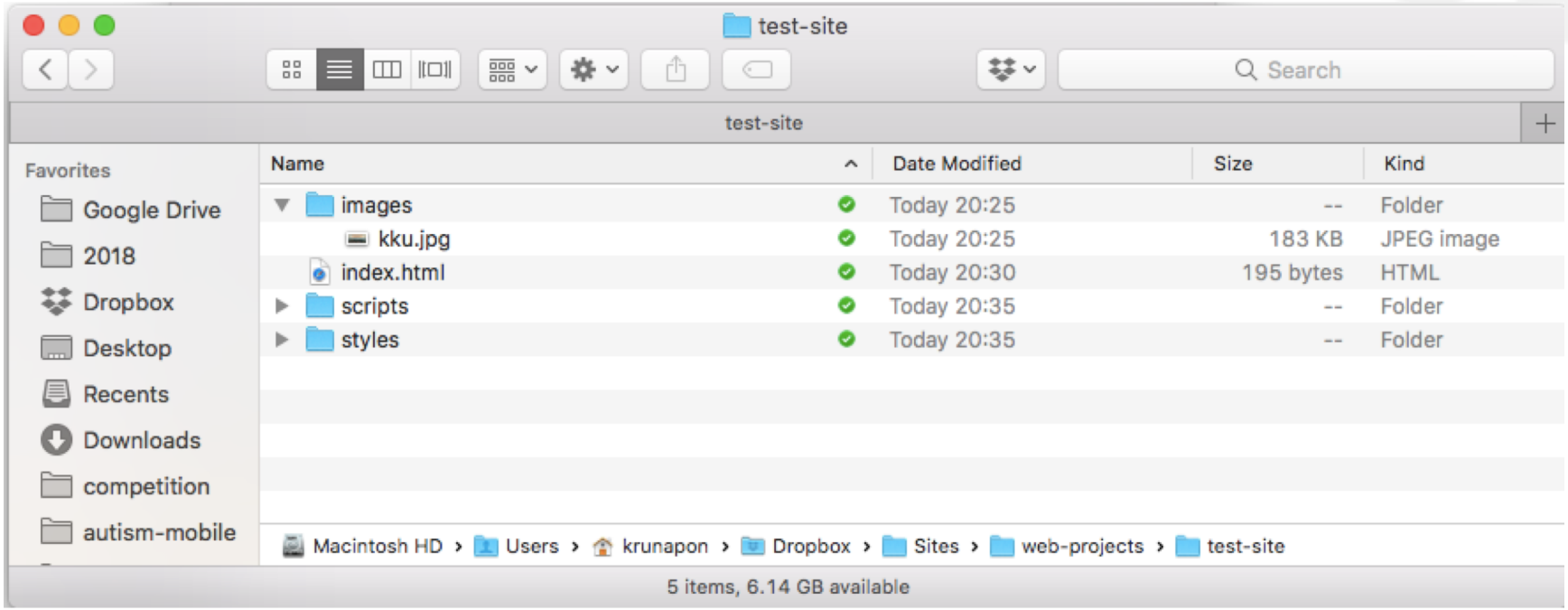
test-site — -bash — 80×24

```
macair:test-site krunapon$ more index.html
<!DOCTYPE html>
<html>
  <head>
        <meta charset="utf-8">
        <title>My test page</title>
  </head>
  <body>
        <img src="images/kku.jpg" width="50%" height="50%" alt="My test image">
  </body>
</html>
```

Example web page result



The structure of the folder



Agenda

- Installing basic software
- What will your website look like?
- Dealing with files
- **HTML basics**
- CSS basics
- JavaScript basics
- Publishing your website
- How the Web works

HTML basics

- HTML (Hypertext Markup Language) is the code that is used to structure a web page and its content
- Content should be structured within
  - A set of paragraphs
  - A list of bulleted points
  - Images and data tables

What is HTML?

- HTML is not a programming language
- It is a markup language that defines the structure of your content
- HTML consist of a series of elements, which you use to enclose, or wrap, different parts of the content to make it appear a certain way

<!DOCTYPE html>

- The <!DOCTYPE> declaration must be the very first thing in your HTML document, before the <html> tag.
- The <!DOCTYPE> declaration is not an HTML tag
- It is an instruction to the web browser about what version of HTML the page is written in.

HTML Tags: <html> and<head>

- <html></html>
  - This element wraps all the content on the entire page, and is sometimes known as the root element
- <head></head>
  - This element acts as a container for all the stuff you want to include on the HTML page that *isn't* the content you are showing to your page's viewers
  - This includes things like keywords and a page description that you want to appear in search results, CSS to style our content, character set declarations, and more

HTML Tags: <meta> and <title>

- <meta charset="utf-8">
  - This element sets the character set your document should use to UTF-8
- <title></title>
  - This sets the title of your page
  - It is also used to describe the page when you bookmark/favourite it

HTML Tags: <body>

- This contains *all* the content that you want to show to web users when they visit your page
  - whether that's text, images, videos, games, playable audio tracks, or whatever else

Images

```
<img src="images/kku.jpg" width="50%" height="50%" alt="My test image">
```

- It embeds an image into our page in the position it appears
- It does this via the src (source) attribute, which contains the path to our image file
- Attributes width and height of an image
- Attribute alt specifies descriptive text for users who cannot see the image

Marking up text

- Heading elements allow you to specify that certain parts of your content are headings — or subheadings — of your content
- HTML contains six heading levels, <h1>–<h6> although you'll commonly only use 3–4 at most

```
1 <h1>My main title</h1>
2 <h2>My top level heading</h2>
3 <h3>My subheading</h3>
4 <h4>My sub-subheading</h4>
```

Paragraphs

- <p> elements are for containing paragraphs of text; you'll use these frequently when marking up regular text content

```
1 | <p>This is a single paragraph</p>
```

Lists

- Marking up lists always consist of at least two elements
- The most common list types are ordered and unordered lists:
  - **Unordered lists** are for lists where the order of the items doesn't matter, like a shopping list. These are wrapped in a <ul> element.
  - **Ordered lists** are for lists where the order of the items does matter, like a recipe. These are wrapped in an <ol> element
- Each item inside the lists is put inside an <li> (list item) element.

Sample Paragraphs and Lists

```
<p>At Mozilla, we're a global community of</p>

<ul>
  <li>technologists</li>
  <li>thinkers</li>
  <li>builders</li>
</ul>

<p>working together ... </p>
```

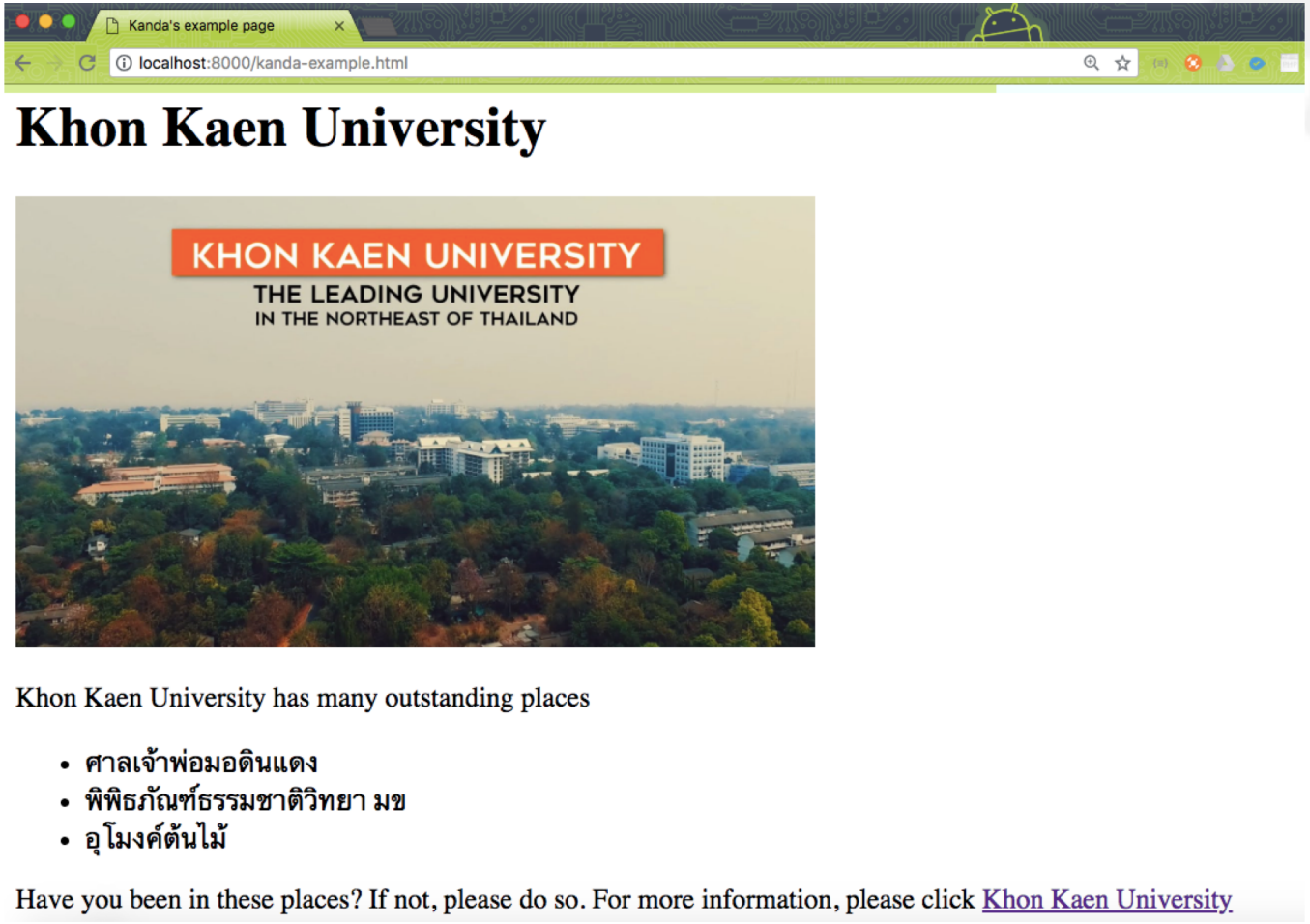
Links

- Links are very important — they are what makes the web a web!
- To add a link, we need to use a simple element — <a> — the "a" being short for "anchor”

Steps to insert a link

1. Choose some text. We chose the text “KKU”
2. Wrap the text in an <a> element, like so:<a>KKU
3. Give the <a> element an href attribute, like so:Mozilla Manifesto
4. Fill in the value of this attribute with the web address that you want the link to link to:KKU

HTML Page Example



Agenda

- Installing basic software
- What will your website look like?
- Dealing with files
- HTML basics
- **CSS basics**
- JavaScript basics
- Publishing your website
- How the Web works

What is CSS?

- Like HTML, CSS is not really a programming language. It is not a *markup language* either — it is a *style sheet language*
- This means that it lets you apply styles selectively to elements in HTML documents
- For example, to select **all** the paragraph elements on an HTML page and turn the text within them red, you'd write this CSS:

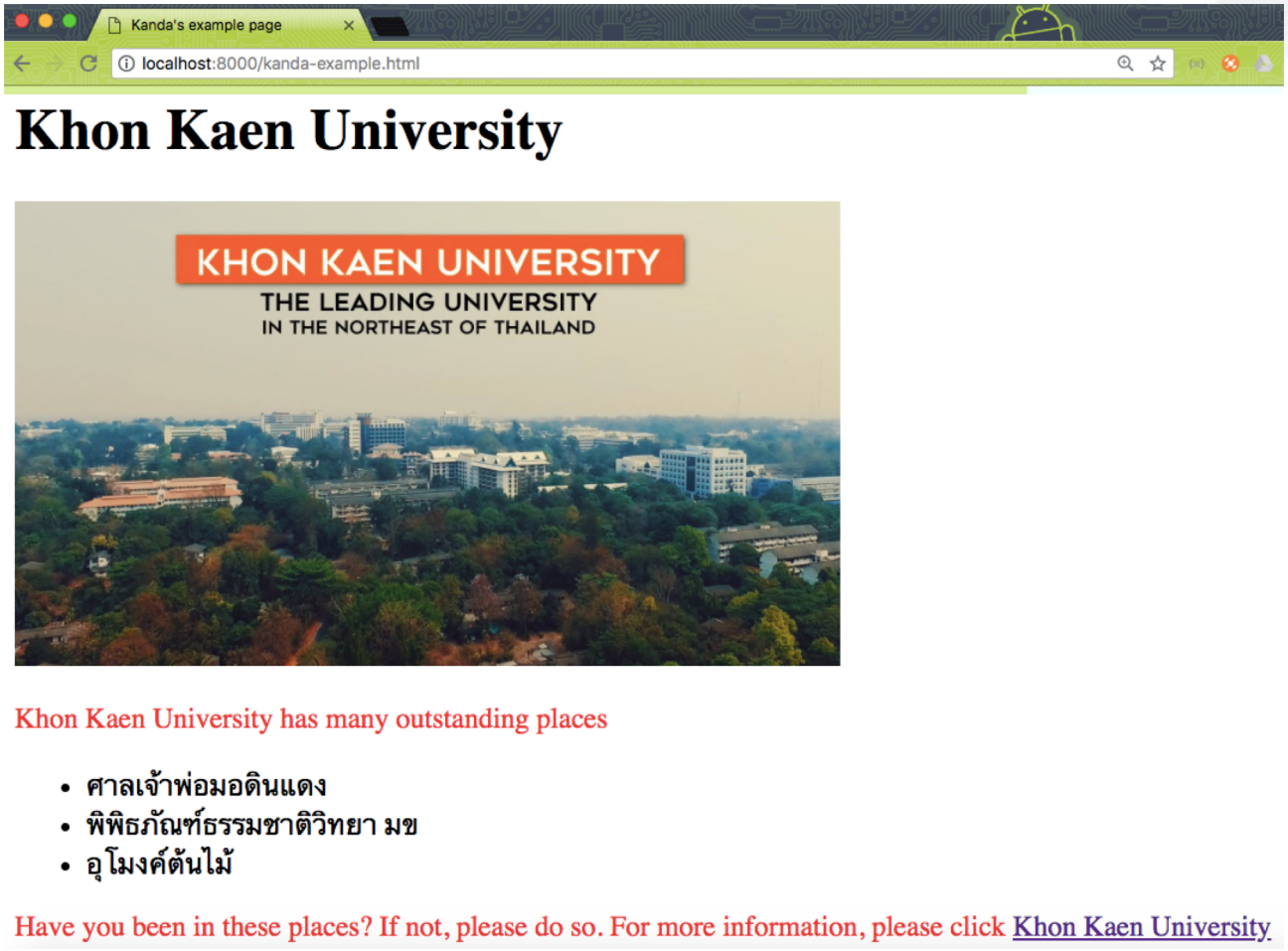
```
1 | p {
2 |   color: red;
3 | }
```

File style.css in styles directory

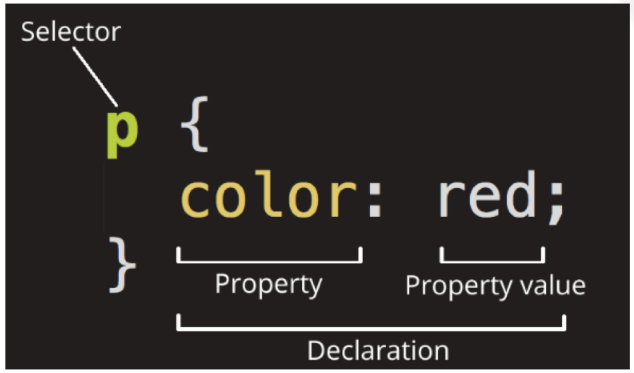
- Let's try it out: paste those three lines of CSS into a new file in your text editor, and then save the file as style.css in your styles directory.
- But we still need to apply the CSS to your HTML document. Otherwise, the CSS styling won't affect how your browser displays the HTML document
- Open your <your-name>-example.html file and paste the following line somewhere in the head (that is, between the <head> and </head> tags):

```
<head>
      <meta charset="utf-8">
      <title>Kanda's example page</title>
      <link href="styles/style.css" rel="stylesheet" type="text/css">
</head>
```

HTML page with CSS style



Anatomy of a CSS ruleset



Selector

The HTML element name at the start of the rule set. It selects the element(s) to be styled (in this case, p elements)

Declaration

A single rule like color: red; specifying which of the element's **properties** you want to style.

Properties

Ways in which you can style a given HTML element. (In this case, color is a property of the <p> elements.) In CSS, you choose which properties you want to affect in your rule

Property value

To the right of the property after the colon, we have the property value, which chooses one out of many possible appearances for a given property (there are many colorv alues besides red).

Multiple rule sets

- Each rule set (apart from the selector) must be wrapped in curly braces ({}).
- Within each declaration, you must use a colon (:) to separate the property from its values.
- Within each rule set, you must use a semicolon (;) to separate each declaration from the next one

```
1 | p {
2 |   color: red;
3 |   width: 500px;
4 |   border: 1px solid black;
5 | }
```

Selecting multiple elements

- You can also select multiple types of elements and apply a single rule set to all of them
- Include multiple selectors separated by commas

```
1 | p, li, h1 {
2 |   color: red;
3 | }
```

Different types of selectors

Selector name	What does it select	Example
Element selector (sometimes called a tag or type selector)	All HTML element(s) of the specified type.	`p` Selects `<p>`
ID selector	The element on the page with the specified ID. On a given HTML page, you're only allowed one element per ID (and of course one ID per element).	`#my-id` Selects `<p id="my-id">` or `<a id="my-id">`
Class selector	The element(s) on the page with the specified class (multiple class instances can appear on a page).	`.my-class` Selects `<p class="my-class">` and `<a class="my-class">`
Attribute selector	The element(s) on the page with the specified attribute.	`img[src]` Selects `<img src="myimage.png">` but not `<img>`
Pseudo-class selector	The specified element(s), but only when in the specified state, e.g. being hovered over.	`a:hover` Selects `<a>`, but only when the mouse pointer is hovering over the link.

Fonts and text

- let's start adding some more rules and information to our style.css file to make our example look nice

1. Add this line to style.css

<link href='https://fonts.googleapis.com/css?family=Open+Sans' rel='stylesheet' type='text/css'>

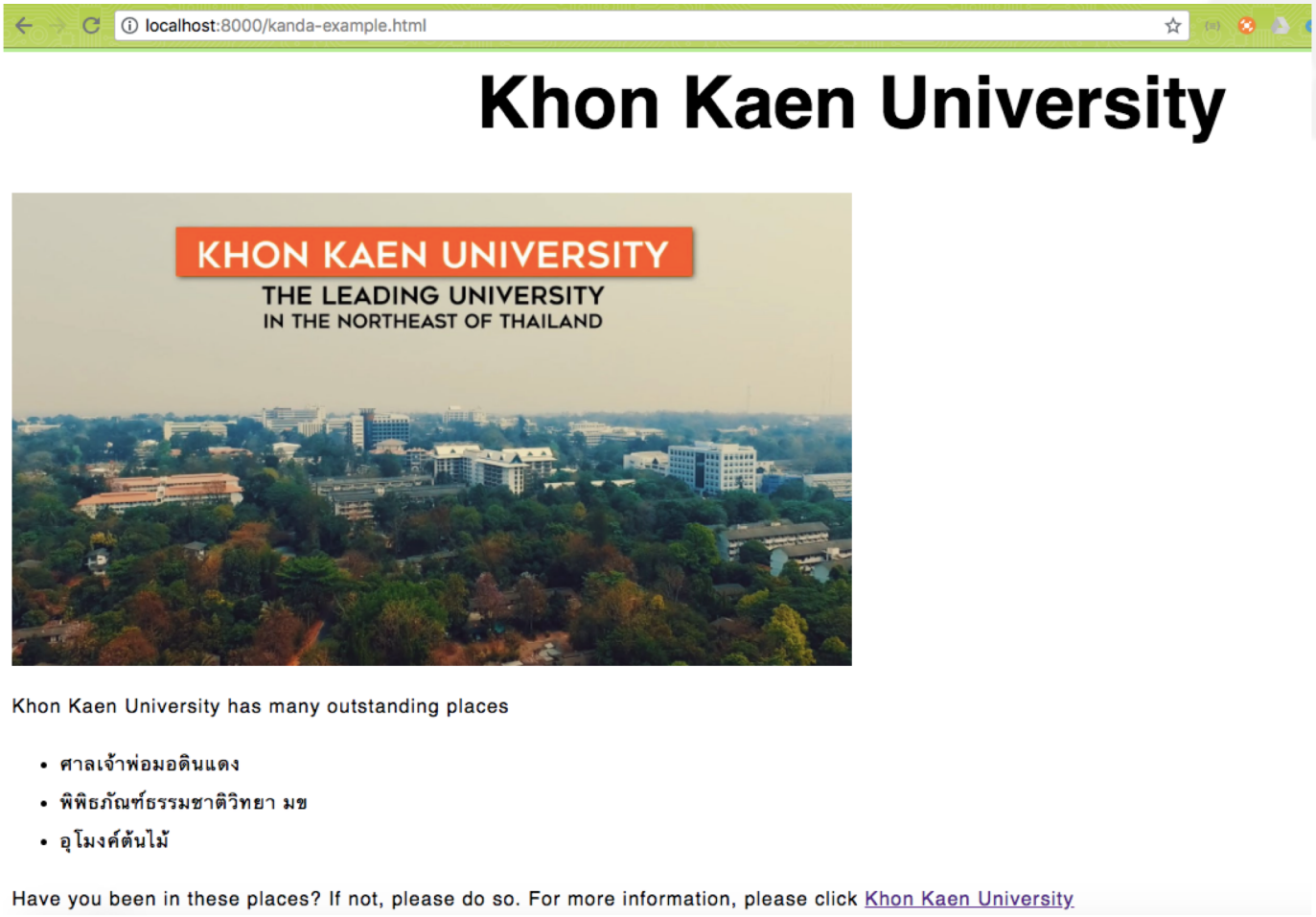
2. Delete the existing rule you have in your style.css file

3. Have this content in style.css file

```
1  html {
2    font-size: 10px; /* px means 'pixels': the base font size
3    font-family: 'Open Sans', sans-serif; /* this should be t
4  }
```

```
1  h1 {
2    font-size: 60px;
3    text-align: center;
4  }
5
6  p, li {
7    font-size: 16px;
8    line-height: 2;
9    letter-spacing: 1px;
10 }
```

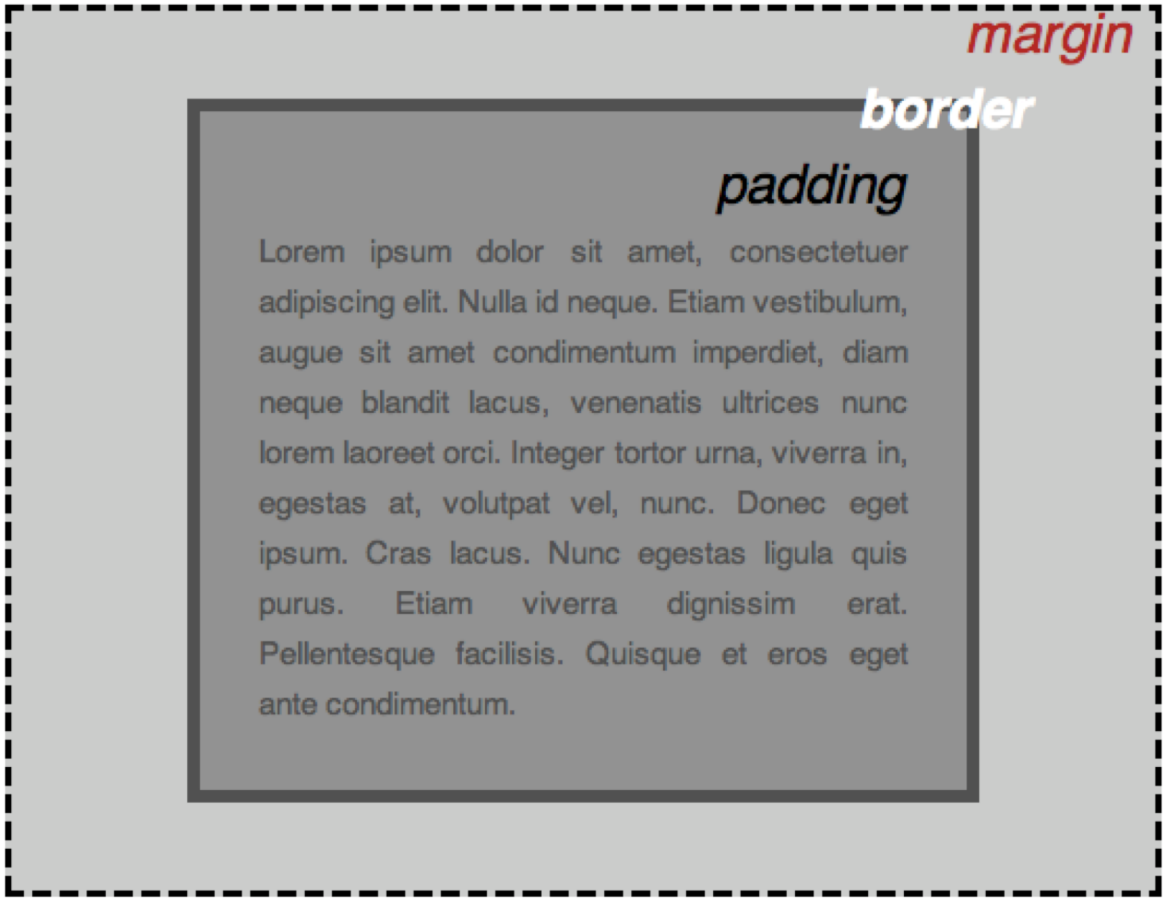
Example web page result with CSS



CSS layout and box model

- Most of the HTML elements on your page can be thought of as boxes sitting on top of each other
- CSS layout is based principally on the *box model*
- Each of the blocks taking up space on your page has properties like this:
  - padding, the space just around the content (e.g., around paragraph text)
  - border, the solid line that sits just outside the padding
  - margin, the space around the outside of the element

Padding, border, and margin



Changing the page color and set the properties of the <body> element

```
1 | html {
2 |   background-color: #00539F;
3 | }
```

```
1 | body {
2 |   width: 600px;
3 |   margin: 0 auto;
4 |   background-color: #FF9500;
5 |   padding: 0 20px 20px 20px;
6 |   border: 5px solid black;
7 | }
```

Properties of <body> element

```
1  body {
2    width: 600px;
3    margin: 0 auto;
4    background-color: #FF9500;
5    padding: 0 20px 20px 20px;
6    border: 5px solid black;
7  }
```

- **width: 600px;** — this forces the body to always be 600 pixels wide.
- **margin: 0 auto;** — When you set two values on a property like margin or padding, the first value affects the element's top **and** bottom side (make it 0 in this case), and the second value the left **and** right side (here, auto is a special value that divides the available horizontal space evenly between left and right)
- **background-color: #FF9500;** — as before, this sets the element's background color
- **padding: 0 20px 20px 20px;** — we have four values set on the padding, to make a bit of space around our content. This time we are setting no padding on the top of the body, and 20 pixels on the left, bottom and right. The values set top, right, bottom, left, in that order
- **border: 5px solid black;** — this simply sets a 5-pixel–wide, solid black border on all sides of the body.

Positioning and styling our main page title

```
1  h1 {
2    margin: 0;
3    padding: 20px 0;
4    color: #00539F;
5    text-shadow: 3px 3px 1px black;
6  }
```

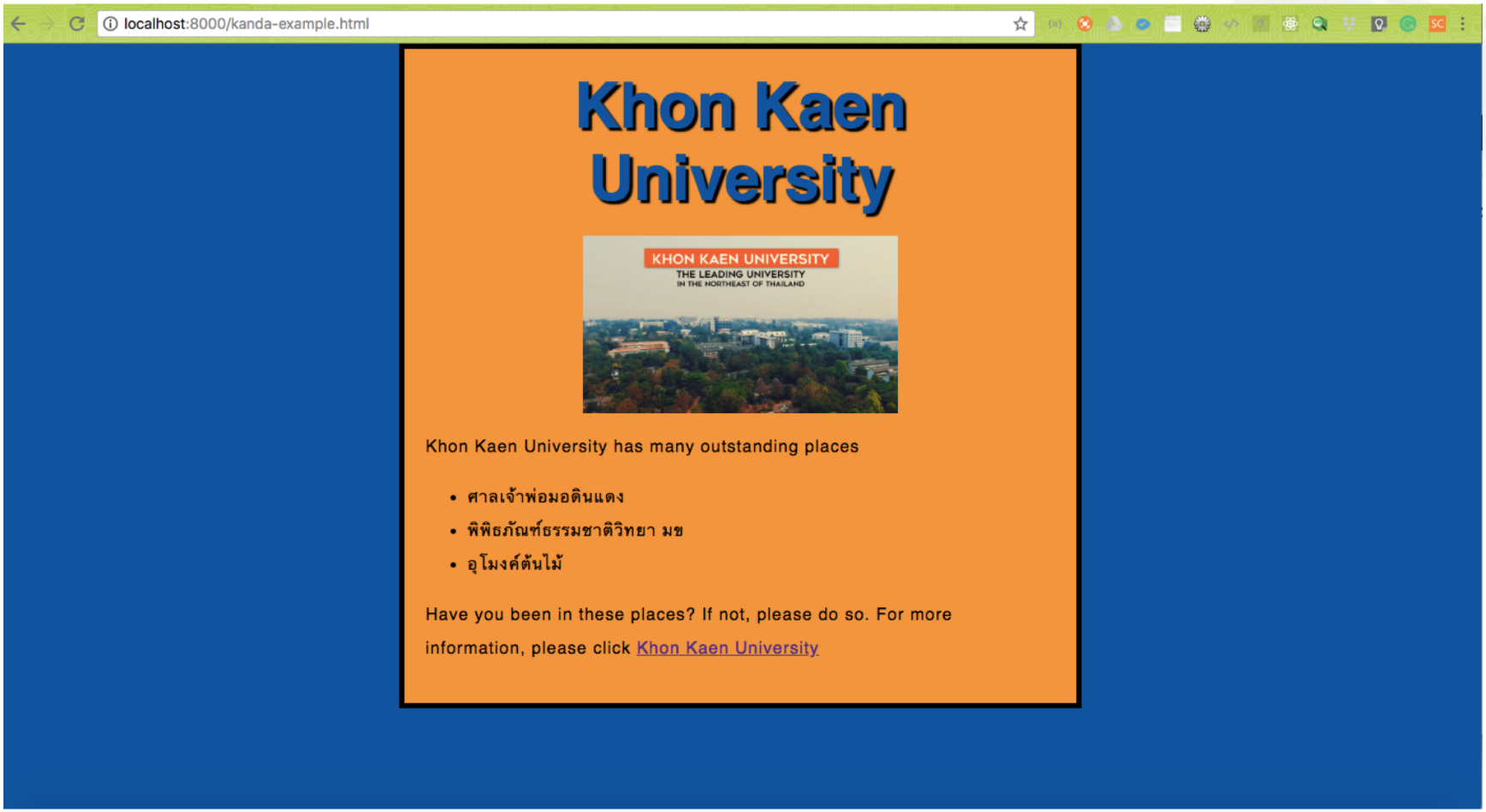
- margin:0;
  - You may have noticed there's a horrible gap at the top of the body
  - That happens because browsers apply some **default styling** to the <h1> element (among others)
  - To get rid of the gap we overrode the default styling by setting margin: 0;
- padding: 20 px 0;
  - set the heading's top and bottom padding to 20 pixels
- text-shadow: 3px 3px 1px black;
  - The first pixel value sets the **horizontal offset** of the shadow from the text — how far it moves across: a negative value should move it to the left
  - The second pixel value sets the **vertical offset** of the shadow from the text — how far it moves down, in this example; a negative value should move it up
  - The third pixel value sets the **blur radius** of the shadow — a bigger value will mean a more blurry shadow
  - The fourth value sets the base color of the shadow

Centering the image

```
1 | img {
2 |   display: block;
3 |   margin: 0 auto;
4 | }
```

- Center the image to make it look better
- Use the margin: 0 auto
  - Top and bottom to be values 0
  - auto is a special value that divides the available horizontal space evenly between left and right

Example web page result with CSS and Box models



Agenda

- Installing basic software
- What will your website look like?
- Dealing with files
- HTML basics
- CSS basics
- **JavaScript basics**
- Publishing your website
- How the Web works

What is JavaScript?

- JavaScript ("JS" for short) is a full-fledged dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on websites
- JavaScript (JS) is a programming language mostly used to dynamically script webpages on the client side, but it is also often utilized on the server-side, using packages such as Node.js.
- It was invented by Brendan Eich, co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation.

A “hello world” example

- Within the “scripts” folder, create a new file called main.js
- In an html file, enter the following element on a new line just before the closing </body> tag

```
        </ul>
        <p>Have you been in these places?  If not, please do so.  For more info
rmation, please click <a href="http://www.kku.ac.th">Khon Kaen University</a></
p>
        <script src="scripts/main.js"></script>
  </body>
</html>
```

Variables

- Variables are containers that you can store values in
- You start by declaring a variable with the var keyword

```
1 | var myVariable;
```

```
1 | var myVariable = 'Bob';
```

Variable	Explanation	Example
String	A sequence of text known as a string. To signify that the value is a string, you must enclose it in quote marks.	`var myVariable = 'Bob';`
Number	A number. Numbers don't have quotes around them.	`var myVariable = 10;`
Boolean	A True/False value. The words `true` and `false` are special keywords in JS, and don't need quotes.	`var myVariable = true;`
Array	A structure that allows you to store multiple values in one single reference.	`var myVariable = [1,'Bob','Steve',10];` Refer to each member of the array like this: `myVariable[0]`, `myVariable[1]`, etc.
Object	Basically, anything. Everything in JavaScript is an object, and can be stored in a variable. Keep this in mind as you learn.	`var myVariable = document.querySelector('h1');` All of the above examples too.

Comments

```
1 | /*
2 | Everything in between is a comment.
3 | */
```

```
1 | // This is a comment
```

Operators

Operator	Explanation	Symbol(s)	Example
Addition	Used to add two numbers together or glue two strings together.	`+`	`6 + 9;` `"Hello " + "world!";`

Subtraction, Multiplication, Division	These do what you'd expect them to do in basic math.	`-`, `*`, `/`	`9 - 3;`
			`8 * 2; // multiply in JS is an asterisk`
			`9 / 3;`

Assignment	You've seen this already: it assigns a value to a variable.	`=`	`var myVariable = 'Bob';`
Equality	Does a test to see if two values are equal to one another and returns a `true`/`false` (Boolean) result.	`===`	`var myVariable = 3;` `myVariable === 4;`

Not, Does-not-equal	Returns the logically opposite value of what it precedes; it turns a `true` into a `false`, etc. When it is used alongside the Equality operator, the negation operator tests whether two values are *not* equal.	`!`, `!==`	The basic expression is `true`, but the comparison returns `false` because we've negated it:
			
			`var myVariable = 3;`
			`!(myVariable === 3);`
			
			Here we are testing "is `myVariable` NOT equal to 3". This returns `false` because `myVariable` IS equal to 3.
			
			`var myVariable = 3;`
			`myVariable !== 3;`

Conditionals

- Conditionals are code structures which allow you to test if an expression returns true or not, running alternative code revealed by its result
- A very common form of conditionals is the if ... else statement

```
1 var iceCream = 'chocolate';
2 if (iceCream === 'chocolate') {
3   alert('Yay, I love chocolate ice cream!');
4 } else {
5   alert('Awwww, but chocolate is my favorite...');
6 }
```

Functions

- Functions are a way of packaging functionality that you wish to reuse
- When you need the procedure you can call a function, with the function name. instead of rewriting the entire code each time

1.
```
1 | var myVariable = document.querySelector('h1');
```

2.
```
1 | alert('hello!');
```

```
1 | function multiply(num1,num2) {
2 |   var result = num1 * num2;
3 |   return result;
4 | }
```

```
1 | multiply(4,7);
2 | multiply(20,20);
3 | multiply(0.5,3);
```

Agenda

- Installing basic software
- What will your website look like?
- Dealing with files
- HTML basics
- CSS basics
- JavaScript basics
- **Publishing your website**
- How the Web works

Getting hosting and a domain name

- If you want total control over your published website, then you'll probably need to spend money to buy
  - Hosting — rented file space on a hosting company's web server. You put your website files on this space, and the web server serves the content to web users who request it
  - A domain name — the unique address where people can find your website, like http://www.mozilla.org, or http://www.bbc.co.uk. You rent your domain name for so many years from a **domain registrar**

Tips for finding hosting and domains

- There are a few free services available like Neocities, Blogger, and WordPress. Again, you get what you pay for, but they are ideal for your initial experiments
  - Free services mostly don't require FTP software for uploads either
  - You can just drag and drop right inside their web interface.
- Sometimes companies provide both hosting and domains in one package.
-

Using an online tool like Github

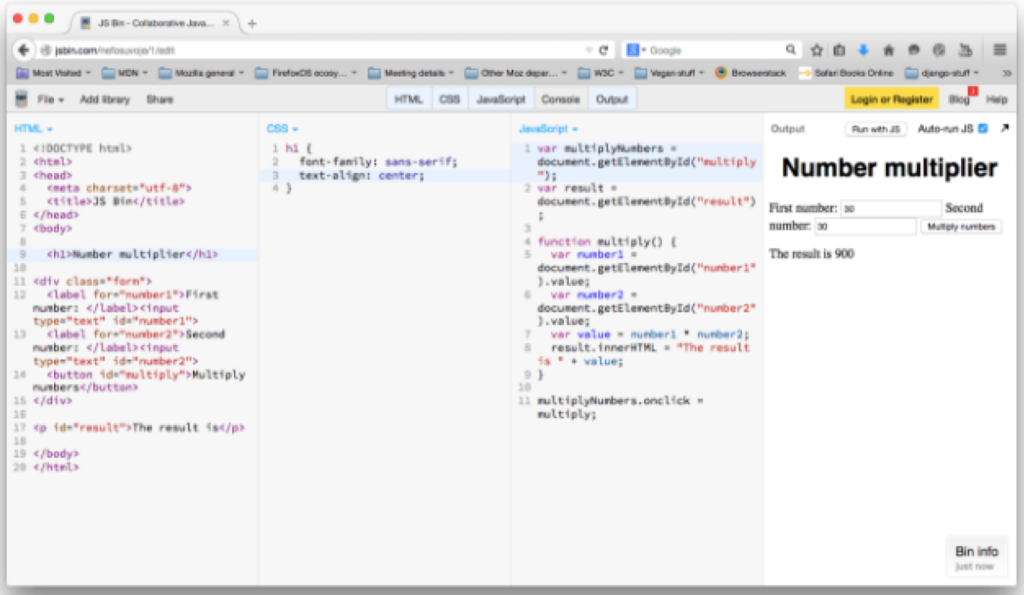
- GitHub is a "social coding" site
- It allows you to upload code repositories for storage in the Git **version control system**
- You can then collaborate on code projects, and the system is open-source by default, meaning that anyone in the world can find your GitHub code, use it, learn from it, and improve on it
- GitHub has a very useful feature called GitHub Pages, which allows you to expose website code live on the web.

Using an online tool like Google App Engine

- Google App Engine is a powerful platform that lets you build and run applications on Google's infrastructure
  - whether you need to build a multi-tiered web application from scratch or host a static website
- See How do you host your website on Google App Engine? for more information.
- Unlike most hosting, such tools are usually free to use, but you only get a limited feature-set.

Using a web-based IDE such as Thimble

- There are a number of web apps that emulate a website development environment, allowing you to enter HTML, CSS and JavaScript and then display the result of that code when rendered as a website — all in one browser tab
  - JSFiddle, Thimble, JS Bin, CodePen



Publishing via Github

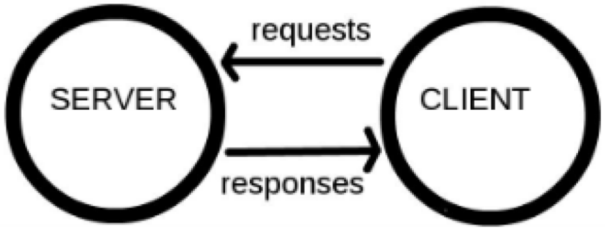
- First of all, sign up for GitHub and verify your email address.
- Next, you need to create a repository for your files to go in.
- On this page, in the *Repository name* box, enter *username*.github.io, where *username* is your username. So for example, our friend bobsmith would enter *bobsmith.github.io*
- Also check *Initialize this repository with a README* and then click *Create repository*.

Agenda

- Installing basic software
- What will your website look like?
- Dealing with files
- HTML basics
- CSS basics
- JavaScript basics
- Publishing your website
- **How the Web works**

Clients and servers

- Computers connected to the web are called **clients** and **servers**. A simplified diagram of how they interact might look like this



- Clients are the typical web user's internet-connected devices (for example, your computer connected to your Wi-Fi, or your phone connected to your mobile network) and web-accessing software available on those devices (usually a web browser like Firefox or Chrome)
- Servers are computers that store webpages, sites, or apps. When a client device wants to access a webpage, a copy of the webpage is downloaded from the server onto the client machine to be displayed in the user's web browser.

The other parts of the toolbox

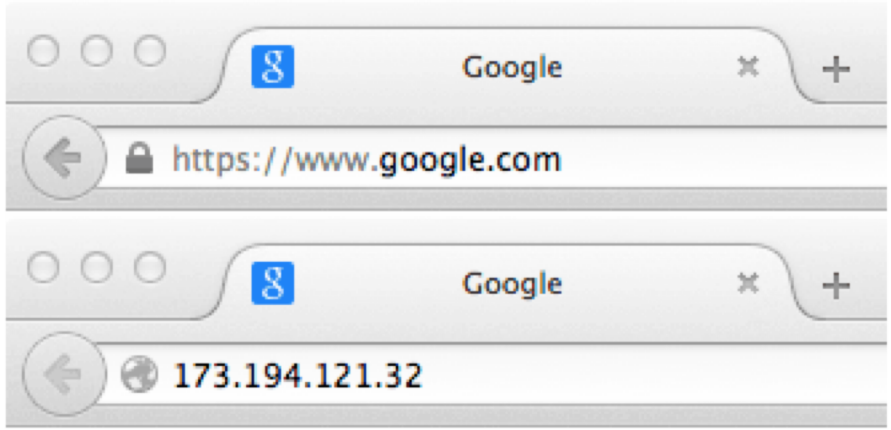
- **Your internet connection**: Allows you to send and receive data on the web
- **TCP/IP**: Transmission Control Protocol and Internet Protocol are communication protocols that define how data should travel across the web
- **DNS**: Domain Name Servers are like an address book for websites
- **HTTP**: Hypertext Transfer Protocol is an application protocol that defines a language for clients and servers to speak to each other
- **Component files**: A website is made up of many different files, which are like the different parts of the goods you buy from the shop. These files come in two main types
  - **Code files**: Websites are built primarily from HTML, CSS, and JavaScript, though you'll meet other technologies a bit later.
  - **Assets**: This is a collective name for all the other stuff that makes up a website, such as images, music, video, Word documents, and PDFs.

What happens, exactly?

1. The browser goes to the DNS server, and finds the real address of the server that the website lives on (you find the address of the shop)
2. The browser sends an HTTP request message to the server, asking it to send a copy of the website to the client (you go to the shop and order your goods). This message, and all other data sent between the client and the server, is sent across your internet connection using TCP/IP
3. Provided the server approves the client's request, the server sends the client a "200 OK" message, which means "Of course you can look at that website! Here it is", and then starts sending the website's files to the browser as a series of small chunks called data packets (the shop gives you your goods, and you bring them back to your house)
4. The browser assembles the small chunks into a complete website and displays it to you (the goods arrive at your door — new shiny stuff, awesome!).

DNS explained

- Real web addresses aren't the nice, memorable strings you type into your address bar to find your favorite websites. They are special numbers that look like this: 63.245.215.20
- This is called an IP address, and it represents a unique location on the Web. However, it's not very easy to remember, is it? That's why Domain Name Servers were invented



Packets explained

- Earlier we used the term "packets" to describe the format in which the data is sent from server to client
- Basically, when data is sent across the web, it is sent as thousands of small chunks, so that many different web users can download the same website at the same time
- If web sites were sent as single big chunks, only one user could download one at a time, which obviously would make the web very inefficient and not much fun to use.

References

- https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web