Chap # 1

1.1. Using nslookup to find IP addresses
Explanation:
This question asks you to use a tool called nslookup (or a similar one) to find out the **IP addresses** of websites (hosts) your instructor assigned. Every website has one or more IP addresses, which are what your computer uses to connect to the site.
Answer Example (for three websites):
Run the following in your terminal or command prompt:
nslookup www.google.com
nslookup www.facebook.com
nslookup www.yahoo.com
Output might look like:
www.google.com → 142.250.192.196
www.facebook.com → 157.240.22.35
www.yahoo.com → 98.137.11.163

1.2. Send HTTP request using telnet to find Server header
Explanation:
You are asked to manually connect to a website using telnet and send a basic HTTP GET request. From the response, locate the Server: field, which tells you what web server software (like Apache, nginx) the site is using.
Answer Example (for www.google.com):
Step 1: In terminal:
telnet www.google.com 80
Step 2: Type this HTTP request (press Enter after each line, and twice at the end):
GET / HTTP/1.1
Host: www.google.com
Connection: close

Sample Output:
Server: gws
Repeat for 2 more sites like www.facebook.com or www.bbc.com.

1.3. Use HEAD request to get headers only
Explanation:
Instead of asking for the full page (like GET does), HEAD only asks for the headers — the metadata of the response. You'll list **all the header field names** that the server sends back.
Answer Example:
Command (via telnet):
HEAD / HTTP/1.1
Host: www.google.com
Connection: close

Sample Response Headers:
Date:
Content-Type:
Content-Length:
Server:
X-XSS-Protection:
X-Frame-Options:
→ **List these field names only** as your answer.

1.4. Find HTTP methods allowed using OPTIONS
Explanation:

This asks you to send an OPTIONS request to see what HTTP methods (GET, POST, PUT, DELETE, etc.) the server supports for that URL.
Answer Example:
OPTIONS / HTTP/1.1
Host: www.google.com
Connection: close

Sample Output:
Allow: GET, HEAD, OPTIONS
Repeat for 2 more hosts.

1.5. Order MIME types based on accept header
Explanation:
You're given an Accept header with MIME types and quality values (q). Higher q means higher preference. Types without q are treated as 1.0.
You must sort these 4 types by preference:

- image/png
- application/pdf (not listed)
- text/plain
- application/xhtml+xml

Answer:

1. **image/png** → q=1.0
2. **application/xhtml+xml** → q=1.0
3. **text/plain** → q=0.8
4. **application/pdf** → falls under */*; q=0.1

1.6. How websites learn browsing habits from headers
Explanation:
This question asks how websites can know what you're doing **outside** their site, just from headers like Referer, User-Agent, etc., sent with your requests.
Answer:

- The Referer header shows what page you visited **before**.
- Websites can track users across sites using embedded ads/scripts.
- User-Agent helps create a **fingerprint** of your device/browser.
- So, even without cookies, sites can gather data about your behavior.

1.7. UTF-16 in Java vs UTF-8 (pros & cons)
Explanation:
This compares Java's use of UTF-16 (16-bit per char) vs UTF-8 (variable size). You must give **one advantage and one disadvantage** of UTF-16.
Answer:

- ☑ **Advantage (UTF-16):** Fast access and indexing, since most common characters are fixed-length (2 bytes).
- ❌ **Disadvantage:** Wastes space for ASCII text, and not compatible with UTF-8 web standards.

1.8. Can browser load HTML if DNS is down?
Explanation:
This checks if a browser can load a webpage **without DNS** (which translates names like google.com to IP addresses).
Answer:

- **Yes**, if you use the **IP address** directly (e.g., http://142.250.192.196).
- **No**, if you're trying to use a domain name and DNS isn't working.

1.9. Write a minimal GET request
Explanation:

This wants you to manually write an HTTP GET request for a full URL including custom port, query string, and fragment.
Note: the **fragment** (#now) is not sent to the server.
Answer:
GET /hmm/oh/well?isThis=right HTTP/1.1
Host: www.ThisIsATest.net:2012
Connection: close

1.10. Change language preference in browser
Explanation:
You are asked to change your browser's preferred language to see if websites return content in that language, using the Accept-Language header.
Answer (Steps):

1. Go to browser settings → Language → Add German.
2. Move German to top.
3. Visit www.google.com
4. Page should be in German: *"Google auf Deutsch"*
5. Save or screenshot the page to prove change.

1.11. How server knows port even without it in Host field
Explanation:
This asks how the server can still know what port the client connected to even if it's not written in the Host header.
Answer:

- The **TCP connection** includes the port.
- Web servers **listen on specific ports** (e.g., 80 for HTTP).
- So the server knows the port **from the socket**, not from the header.

Chap # 2

Exercise 2.1

Explanation: Count the number of tags and elements in Figure 2.1, and calculate characters of content excluding whitespace.
Answer: Without the actual figure, we can't count accurately. But generally:

Tags: Each opening and closing tag counts.
Elements: Each pair of opening and closing tags is one element.
Characters of content: Count actual text content only (not markup), excluding leading/trailing spaces.

Exercise 2.2

Explanation: Draw a DOM-style tree of the XHTML document from Figure 2.12.
Answer: Based on assumption:

html
head
title
body
(other nested elements like h1, p, etc.)

Exercise 2.3
Explanation: Compare developing HTML for internal company use vs. public web.
Answer: Internal use is easier because:
- ❖ Limited browser/platform variation.
- ❖ Controlled network and security.
- ❖ Easier testing and maintenance.

Exercise 2.4
Explanation: Use XHTML with extra spacing between paragraphs.
Answer:
This is paragraph one.
This is paragraph two.
This is paragraph three.

Exercise 2.5
Explanation: Evaluate spacing options for readability and browser resizing.
Answer: Using   (non-breaking space then space) ensures:

Sentences won't break awkwardly on resize.
   might keep both spaces on one line, which can look awkward.
Space +   can break in between.

Exercise 2.6
Explanation: Properly encode special characters in input value.
Answer:
<input type="text" value="An example is written as <x, b>." />

Exercise 2.7

Explanation: Use all six heading levels with paragraph.
Answer:
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
This is a paragraph of text.
In Mozilla 1.4, headings use progressively smaller bold fonts, paragraph uses normal text font.

Exercise 2.8
Explanation: Identify markup error in code snippet.
Answer: Use < instead of < in pre block.
<pre>
// Java code example
if (a < b && c != d) {
System.out.println("Time for donuts!");
}
</pre>

Exercise 2.9
Explanation: Image appearance on different resolutions.
Answer:

Image appears **smaller** on higher resolution display.
More pixels per inch compress the same pixel size image.

Exercise 2.10
Explanation: Allow visitors to stretch image using only XHTML.
Answer:

Browser resizing window stretches image horizontally.

Exercise 2.11
Explanation: Convert relative URL to absolute based on base URL.
Answer:
Base: http://www.example.com/hw1/detail/page7.html
Relative: ../images/icon5.gif
Absolute: http://www.example.com/hw1/images/icon5.gif

Exercise 2.12

Explanation: Write relative paths for files in nested folders.
Answer:

From App1 to legal/copyright.html: legal/copyright.html
From legal/copyright.html to App1/logo.jpg: ../logo.jpg

Exercise 2.13
Explanation: Identify misuse of inline formatting () around block elements.
Answer:

dl, dt, dd are block elements and shouldn't be inside .
Fix:
Apply only to inline content, or use CSS for styling.

Exercise 2.14
Explanation: Convert HTML 4.01 table to valid XHTML 1.0.
Answer:

```
<table border="1" summary="This table gives some statistics about fruit flies: average height and weight, and percentage with red eyes (for both males and females).">
 <caption><em>A test table with merged cells</em></caption>
 <tr>
  <th rowspan="2"></th>
  <th colspan="2">Average</th>
  <th rowspan="2">Red<br />eyes</th>
 </tr>
 <tr>
  <th>height</th>
  <th>weight</th>
 </tr>
 <tr>
  <th>Males</th>
  <td>1.9</td>
  <td>0.003</td>
  <td>40%</td>
 </tr>
 <tr>
  <th>Females</th>
  <td>1.7</td>
  <td>0.002</td>
  <td>43%</td>
```

```
  </tr>
</table>
```

Exercise 2.15
Explanation: Compare classid (IE) vs. type (Mozilla).
Answer:

classid: Precise control over component loading (advantage in IE).
type: More flexible and portable, lets browsers use preferred plugins.

Exercise 2.16
(a) Explanation:
head.misc allows any number (including 0) of elements like script, style, meta, etc.
(b) Explanation:
Head can contain:

Either: title followed optionally by base, all possibly interleaved with head.misc items.
Or: base followed optionally by title, again with head.misc interleaved.

(c) Succinct explanation:
The head must contain a title and may optionally contain a base element. Other items like scripts or styles can appear in any order before, between, or after them.

Chap 3

Exercise 3.1: Simple CSS Style Rules
Given declarations:
background declaration: background-color: silver;
text declaration: font-size: larger;

(a) Write CSS rules:

Apply **background declaration** to all elements.
Apply **text declaration** to all elements.

Answer:

```
div {
  background-color: silver;
```

```
}

strong {
  font-size: larger;
}
```

Explanation:
You select elements by their tag names (div, strong) and apply the respective declarations.

(b) Write a single style rule applying both declarations to both <p> and elements.

Answer:

```
p, em {
  background-color: silver;
  font-size: larger;
}
```

Explanation:
Multiple selectors separated by commas share the same declaration block.

(c) Write a single rule applying background declaration to elements with id="Nevada" and to elements with class shiny.

Answer:

```
#Nevada, .shiny {
  background-color: silver;
}
```

Explanation:
ID selectors use #id, class selectors use .class.

(d) Write a rule applying text declaration to elements with class bigger.

Answer:

```
span.bigger {
  font-size: larger;
}
```

(e) Write a rule applying text declaration to elements that are descendants of other elements.

Answer:

```
span span {
  font-size: larger;
}
```

Explanation:
This selects any inside another .

(f) Write a rule applying background declaration when the cursor hovers over a hyperlink (<a>).

Answer:

```
a:hover {
  background-color: silver;
}
```

Exercise 3.2: External Style Sheets and @import

(a) Create 3 external style sheets each with a subset of rules from 3.1. Write a complete XHTML 1.0 Strict document using:

One **non-persistent preferred** style sheet
One **alternate** style sheet
One style sheet used **only for print**

(b) Use @import **so first style sheet imports the second, which imports the third. The XHTML treats the first as persistent.**

Answer:

style1.css (persistent preferred)

```
@import url("style2.css");

div { background-color: silver; }
strong { font-size: larger; }
```

style2.css (alternate)

```
@import url("style3.css");

p, em { background-color: silver; font-size: larger; }
span.bigger { font-size: larger; }
```

style3.css (print only)

```
a:hover { background-color: silver; }
span span { font-size: larger; }
```

XHTML Document:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>CSS @import Example</title>
  <link rel="stylesheet" type="text/css" href="style1.css" />
  <link rel="alternate stylesheet" type="text/css" href="style2.css"
title="Alternate" />
  <link rel="stylesheet" type="text/css" href="style3.css" media="print"
/>
```

```
</head>
<body>
  <!-- Content here -->
</body>
</html>
```

Explanation:

style1.css is persistent and imports style2.css, which imports style3.css.
style2.css is alternate, can be toggled.
style3.css applies only for printing.

Exercise 3.3: Embedded Style Sheet
Set font-family to "Gill Sans Bold SmallCaps & OSF" for all elements.
Answer:

```
<style type="text/css">
  * {
    font-family: "Gill Sans Bold SmallCaps & OSF";
  }
</style>
```

Or full embedded example:

```
<html>
<head>
  <style type="text/css">
    * {
      font-family: "Gill Sans Bold SmallCaps & OSF";
    }
  </style>
</head>
<body>
  <p>Sample text</p>
</body>
</html>
```

Exercise 3.4: CSS Cascade and Specificity
Given Author, User, User Agent styles, find final applied styles:

Style origin	Selector	Declarations
Author	div	color: blue
Author	p	color: green; font-size: smaller !important
Author	.hmm	color: fuchsia
User	p	color: white; background-color: black; font-size:

Style origin	Selector	Declarations
		larger !important
User	body	color: yellow
User Agent	body	color: black

(a) For <p> **elements:**

color?

Author says color: green
User says color: white
User important declarations override author, but color is not marked !important in user styles.
However, font-size is important in both author and user styles.

Specificity: Both author and user styles have specificity of p selector.

User color: white **overrides author** color: green **because user styles override author unless author has !important on color (which it doesn't).**

background-color?

Only user defines it: background-color: black

font-size?

Author: font-size: smaller !important
User: font-size: larger !importan

Both !important rules, user styles override author (user has precedence over author styles). So font-size = larger

.

If <p> **belongs to** .hmm **class?**

.hmm sets color: fuchsia
.hmm specificity is higher than p selector
So color: fuchsia overrides color: white from user styles.

Summary (a):

Property	Normal <p>	<p class="hmm">
color	white	fuchsia
background-color	black	black
font-size	larger	larger

(b) For **elements:**

Author: color: blue

User: no color on div
User agent: no color on div
For :
.hmm = color: fuchsia (author style)
Since .hmm has higher specificity, color will be fuchsia.
Does the containing element affect it? No, color is set directly on div or .hmm.

(c) For **elements inside** <body>**, no** .hmm **class:**
ol has no styles specified.
body color: user style says color: yellow.
So <ol> inherits color: yellow.
If <body class="hmm">:
.hmm sets color: fuchsia but only on elements with class .hmm.
Body has .hmm, so color: fuchsia on body, so ol inherits fuchsia.

(d) If user agent rule changes to:
* { color: black }
This universal selector applies color black everywhere.
User style body { color: yellow } overrides user agent because user styles have precedence.
So for <ol>, inherits from body (yellow or fuchsia if body has .hmm).

Exercise 3.5: Class quote with margins
(a) Write class quote with top/bottom margin 0, left/right margin 4em using shortcut declaration.
Answer:

```
.quote {
  margin: 0 4em;
}
```

Explanation:
margin: [top/bottom] [left/right] shorthand.

(b) Why use em **instead of** px**?**
em units scale with the font size, so margin adapts to text size.
px is fixed, which may look inconsistent if font size changes or zoom.

Exercise 3.6: Pixel length changes with resolution
If monitor resolution changes from 1024×768 to 1280×1024, what happens to 1px length?
Answer:

Pixel size decreases because more pixels fit in the same physical screen size (assuming physical size unchanged).
So 1px represents a smaller physical length on the higher resolution.

Exercise 3.7: Picture framing
(a) Frame around <img> **with brown border, inside edges touching image edges, 10px spacing between images.**

```
img {
  border: 2px solid brown;
  margin: 5px; /* margin is half of 10px to give 10px between adjacent images */
  display: inline-block; /* to respect margin */
}
```

(b) Add a 3px tan mat between image and border

```
img {
  border: 2px solid brown;
  padding: 3px;
  background-color: tan;
  margin: 5px;
  display: inline-block;
}
```

Exercise 3.8: Client area wider than canvas
Why can the client area be wider than the canvas?
Answer:
Because the browser window may have extra horizontal padding, margin, or other elements affecting the total width. Also, the canvas may have fixed width smaller than the viewport.

Exercise 3.9: em and ex related to height, no width unit
Why no width unit related to character width?
Answer:
Character width varies widely across fonts and characters, so CSS doesn't define a standard width unit. Height is more consistent and related to font metrics, making em and ex meaningful.

Exercise 3.10: Stairstep effect
Create HTML and CSS so right sides line up centered, each step same height.

Answer (short version):

```
<style>
  .step {
    height: 3em;
    line-height: 3em;
    margin-left: auto;
    margin-right: auto;
    width: fit-content;
    text-align: right;
  }
  .step1 { width: 10em; }
  .step2 { width: 8em; }
  .step3 { width: 6em; }
</style>

<div class="step step1">Step 1</div>
<div class="step step2">Step 2</div>
<div class="step step3">Step 3</div>
```

Exercise 3.11: Box coordinates

Given:

```
<body>
  <div id="div1">
    <div id="div2"></div>
  </div>
  <div id="div3"></div>
</body>
```

With CSS:

```
body { margin:5px; border:0; padding:2px }
div { margin:3px; border:1px solid blue; padding:4px }
```

Find upper-left corner of content area of each div relative to upper-left corner of the body's containing block.

Calculation steps:

Body's margin: 5px → content starts 5px inside viewport.
Body padding: 2px inside body content box.

For div1:

Margin: 3px outside border
Border: 1px
Padding: 4px

Position of div1 content:

5px (body margin) + 2px (body padding) + 3px (div1 margin) + 1px (div1 border) + 4px (div1 padding) = 15px

So, div1 content starts at (15px, 15px).
For div2 inside div1:
div1 content origin is (15,15), inside that:
Add div2 margin, border, padding:
3 + 1 + 4 = 8px
So div2 content origin is:
(15 + 8, 15 + 8) = (23px, 23px)
For div3 (sibling to div1):
After div1, but vertical offset is not asked.
Assuming no layout flow considered:
Same margin, border, padding:
5 + 2 + 3 + 1 + 4 = 15px
So position is also (15px, 15px), but horizontally shifted based on flow (usually below div1).

Exercise 3.12: Line height and baseline

Normal line height: 1.2em
Baseline height above bottom of character cell: 0.2em
New line-height: 2em

Find half-leading and baseline height above bottom of line box

---

Half-leading:
Leading = line-height - font size = 2em - 1em = 1em
Half-leading = 0.5em
Baseline height above bottom of line box:
Baseline inside character cell = 0.2em
Extra space below baseline due to leading = 0.5em
So baseline above bottom of line box = 0.2em + 0.5em = 0.7em

Exercise 3.13: Image height vs. line box
(a) If image height (1.5em) > line height (2em), line box height must increase.
Use font cell height + baseline info to calculate if line box fits image height.
(b) An inline element with font size 2× default font size may cause line box height to increase, but not necessarily because of vertical-align behavior.

Chap # 4

4.1:
Explanation: Check if a variable has the undefined type/value.
Answer:
```
if (typeof testVar === "undefined") {
   console.log("undefined");
} else {
   console.log("defined");
}
```

4.2:
Explanation: Understand variable scope and global assignment in a function.
Answer:
Output: 6
Reason: window.i = 6; assigns 6 to the global i, even though there's a local var i; in the function.

4.3:
Explanation: Type coercion and comparison in JavaScript.
Answer:
Output: true
Explanation:

- "007" is coerced to number 7.
- Number("007") is also 7.
- So, 7 == "007" → true.

4.4:
Explanation: Object references and parameter passing in functions.
Answer:
Output: 10
Explanation: Both o1 and o2 refer to the same object, so modifying o1.j inside the function affects o2.j.

4.5:
Explanation: Create an object with a property.
Answer:
```
var myObj = {
   color: "red"
};
```

4.6:
Explanation: Understand obfuscated code and window object usage.
Answer:
This line builds the string "alert" dynamically and then calls window.alert().
```
// "al" + "father".slice(4, 6) + "t" → "al" + "er" + "t" = "alert"
var weird = "alert";
window.alert("Weird, but it works.");
```

4.7:
Explanation: Clarify ternary operator precedence.
Answer:
```
a = b ? (z /= (y ? x : (w ? v : u))) : (d += e);
```

4.8:
Explanation: Simulating overflow using 32-bit operations.
Answer:

```
function multiplyAndOverflow(a, b) {
  return (a * b) | 0; // Forces result to 32-bit signed int
}
```

4.9:
Explanation: Why the exception happens (variable scope).
Answer:
Exception occurs at: myVar += value;
Reason: myVar is not defined in the function scope or as this.myVar.
Fix:

```
function addTo(value) {
  this.myVar += value;
}
```

4.10:
Explanation: this context when calling functions.
Answer:
Output: 1,2
Explanation:

- o1.rusty(1); sets o1.x = 1
- o2.rusty(2); sets o2.x = 2

4.11:
Explanation: Display 2D array with visual grid using alert().
Answer:

```
function drawGrid(arr) {
  let result = "";
  for (let i = 0; i < arr.length; i++) {
    result += arr[i].join(" | ") + "\n";
  }
  alert(result);
}
```

4.12:
Explanation: Return the median without modifying the original array.
Answer:

```
function median(arr) {
  let sorted = arr.slice().sort((a, b) => a - b);
  return sorted[Math.floor(sorted.length / 2)];
}
```

4.13:
Explanation: Validate phone numbers using regex.
Answer:

```
function isValid(phone) {
  const pattern = /^(\(\d{3}\)\s?|\d{3}[-\/ ]?)\d{3}[- ]?\d{4}$/;
  return pattern.test(phone);
}
```
