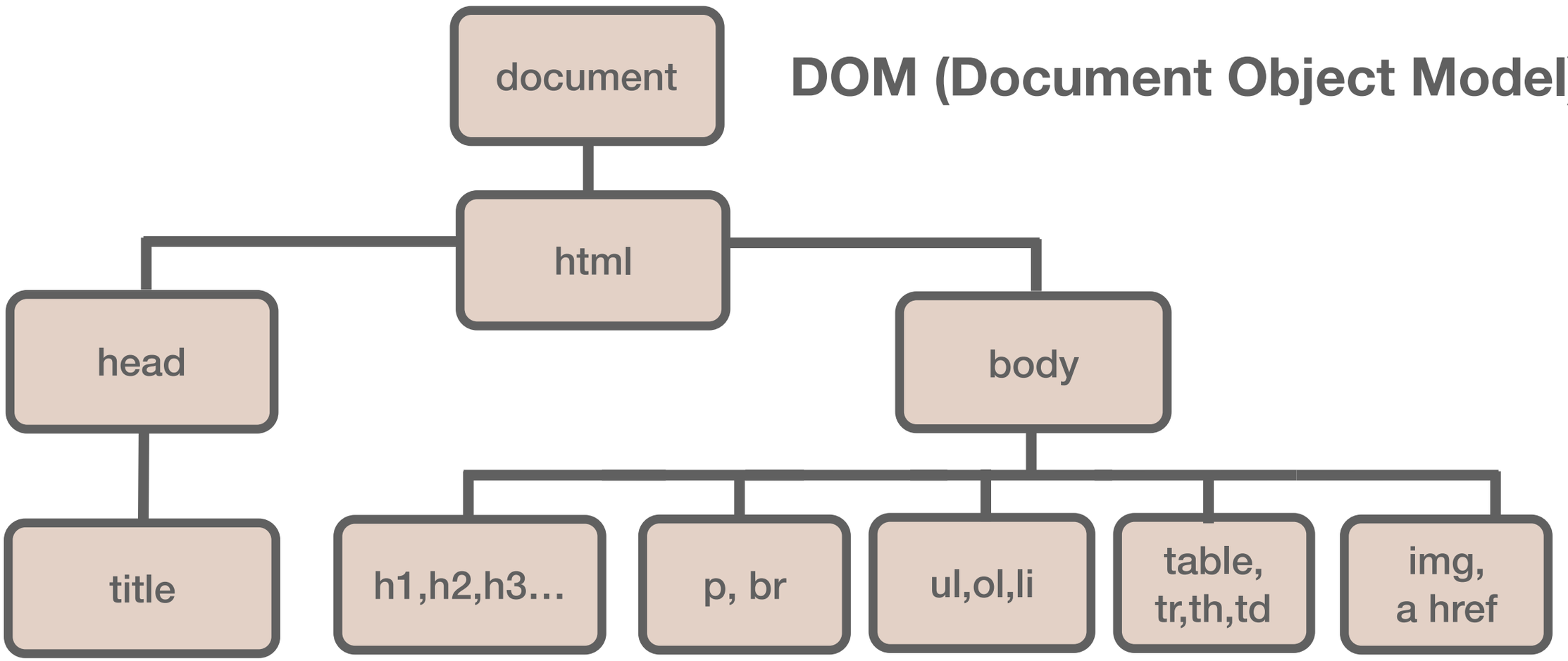
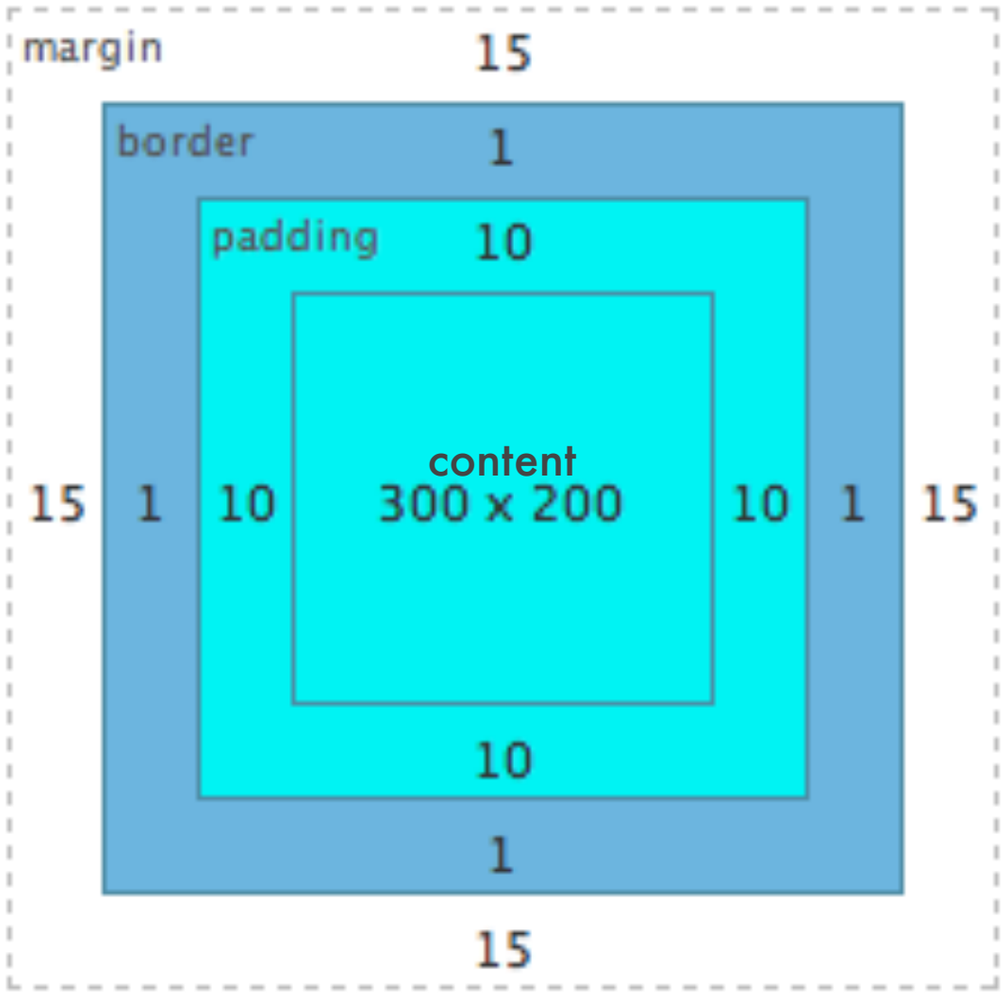
AFTERNOON SESSION

DOM (Document Object Model)



The Box Model



Third hands-on exercise:

https://www.w3.org/Style/Examples/011/firstcss.en.html (Step 1 to 5)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
  <title>My first styled page</title>
</head>
<body>
<!-- Site navigation menu -->
<ul class="navbar">
  <li><a href="index.html">Home page</a>
  <li><a href="musings.html">Musings</a>
  <li><a href="town.html">My town</a>
  <li><a href="links.html">Links</a>
</ul>

<!-- Main content -->
<h1>My first styled page</h1>

<p>Welcome to my styled page!
<p>It lacks images, but at least it has style. And it has links, even if they don’t go
anywhere&hellip;
<p>There should be more here, but I don't know what yet.

<!-- Sign and date the page, it's only polite! —>
<address>Made 5 April 2004<br>
  by myself.</address>
</body>
</html>
```

Code View - Step 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
  <title>My first styled page</title>
  <style type="text/css">
  body {
    color: purple;
    background-color: #d8da3d }
  </style>
</head>
<body>
<!-- Site navigation menu -->
<ul class="navbar">
  <li><a href="index.html">Home page</a>
  <li><a href="musings.html">Musings</a>
  <li><a href="town.html">My town</a>
  <li><a href="links.html">Links</a>
</ul>

<!-- Main content -->
     :
     :
</body>
</html>
```

Code View - Step 2 (Add colours)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
  <title>My first styled page</title>
  <style type="text/css">
  body {
    color: purple;
    background-color: #d8da3d }
 h1 {
    font-family: Helvetica, Geneva, Arial,
         SunSans-Regular, sans-serif }
  </style>
</head>
<body>
<!-- Site navigation menu -->
<ul class="navbar">
  <li><a href="index.html">Home page</a>
  <li><a href="musings.html">Musings</a>
  <li><a href="town.html">My town</a>
  <li><a href="links.html">Links</a>
</ul>
<!-- Main content -->
      :
      :
</body>
</html>
```

Code View - Step 3 (Add fonts)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
  <title>My first styled page</title>
  <style type="text/css">
  body {
    color: purple;
    background-color: #d8da3d }
 ul.navbar {
    position: absolute;
    top: 2em;
    left: 1em;
    width: 9em }
 h1 {
    font-family: Helvetica, Geneva, Arial,
         SunSans-Regular, sans-serif }
  </style>
</head>
<body>
<!-- Site navigation menu -->
<ul class="navbar">
  <li><a href="index.html">Home page</a>
  <li><a href="musings.html">Musings</a>
  <li><a href="town.html">My town</a>
  <li><a href="links.html">Links</a>
</ul>
<!-- Main content -->
```

Code View - Step 4 (Add navbar)

```
<html>
<head>
 <title>My first styled page</title>
 <style type="text/css">
            :
 ul.navbar {
   position: absolute;
   top: 2em;
   left: 1em;
   width: 9em }
 h1 {
   font-family: Helvetica, Geneva, Arial,
        SunSans-Regular, sans-serif }
ul.navbar li {
   background: white;
   margin: 0.5em 0;
   padding: 0.3em;
   border-right: 1em solid black }
  ul.navbar a {
   text-decoration: none }
  a:link {
   color: blue }
  a:visited {
   color: purple }
  </style>
            :
```

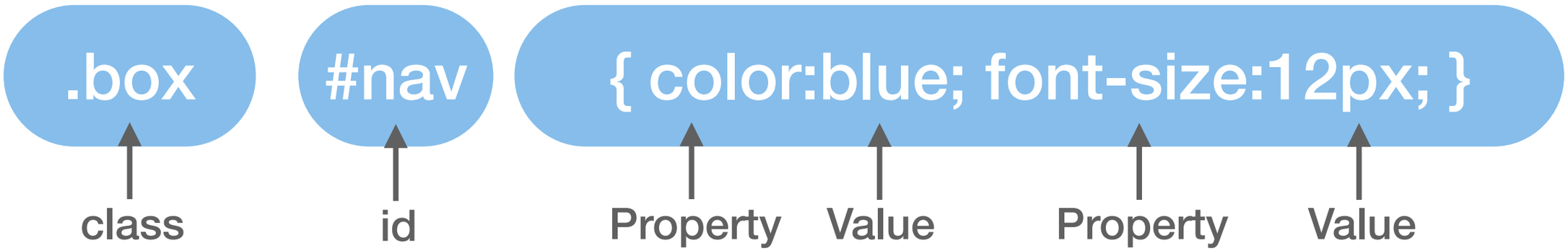
Code View - Step 5 (Styling the navbar)

ADJECTIVE

CSS

Custom (e.g. class or id)
Selector

Declaration Declaration



Put the Internal Style Definition into an External File

4 Ways of Positioning Display Box

- Static - default position of a box following the normal document flow (not affected by top, left, right, bottom pos.)

- Fixed - it always stay on the same location as defined by the positions (top and left or bottom or right) even the page is scrolled. Unlike absolute, its parent is the viewport.

- Relative - relative when used with top and left position pair or bottom and right position pair will allow the object box to be moved to a new location relative to its current position (not container).

- Absolute - take the positioning out of the document flow and place it at a location (top and left position) as defined in relationship to its containing (or parent) element (context). The container/parent should be set to relative.

The Grammar of CSS

- Styles define how to display HTML elements
- Each style description is made up of a Selector and Declaration
- Selector defines which HTML element should be used for display and the declaration defines how
- Each declaration contains properties and values
- There are base and custom selectors (ID and CLASS are custom selectors)
- Style definition can be placed inline, in the head section or in an external file (e.g. style.css)

VERB

JS

JavaScript = act on a HTML tag, CSS property or respond to an event triggered by user action

Understanding Computational Thinking as Foundation to JavaScript Programming

What is “Computational Thinking”?

Introduction to computational thinking

Before computers can be used to solve a problem, the problem itself and the ways in which it could be resolved must be understood. Computational thinking techniques help with these tasks.

Revise | Test

< 1 2 >

What is computational thinking?

Computers can be used to help us solve problems. However, before a problem can be tackled, the problem itself and the ways in which it could be solved need to be understood.

Computational thinking allows us to do this.

More Guides

Introduction to computational thinking

Decomposition >

Pattern recognition >

Abstraction >

Algorithms >

Evaluating solutions >

Decomposition

Pattern

Abstraction

Algorithm

Automation & Testing

Decomposition

Break a problem down into smaller parts.

Pattern

Discover similarities between things.

Abstraction

Ignore irrelevant details to focus on essential features to come up with one solution or classification that works for multiple situations.

Algorithm

Specify a sequence of steps that someone or computer can follow to complete a task.

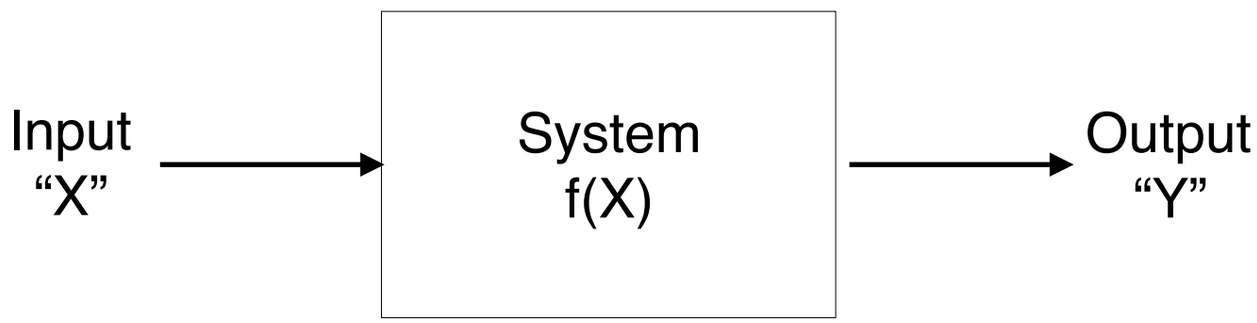
Automation & Testing

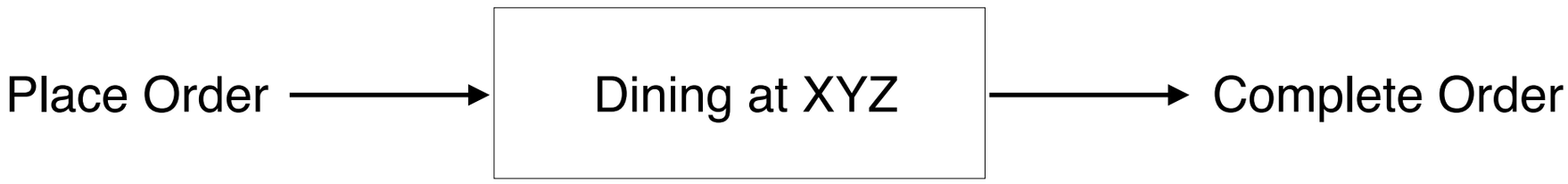
Codify and test the algorithm for automated execution by the computer.

Computational thinking is about system and data.

What is JavaScript? How does it fit into computational thinking?

$$Y = f(X)$$





Example: Cell manipulation in Excel with some cells controlling inputs while others outputs.

JavaScript provides us with the capabilities to build system and transform data.

Data Types in JavaScript

Declaring a variable and its data type:

- String - e.g. var str_var = “This is a string.”;
- Numeric - e.g. var num_var = 3.2;
- Boolean - e.g. var bol_var = true;

Basic Input/Output Commands

- **Entering a variable - e.g. var x = prompt(“Enter x value”);**
- **Displaying a variable - e.g. alert(“x = “ + x_var);**

JavaScript Functions that Transform Input into Output

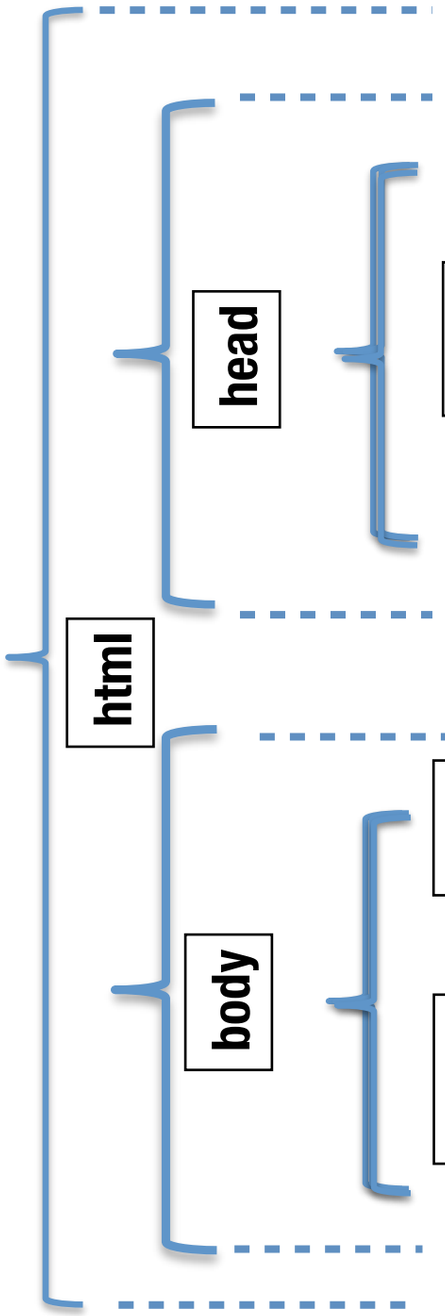
Basic Structure of a JavaScript Function

optional parameters

```
<head>
<script>
function function_name(parameter1, parameter 2…) {
    Embed data type variables, input/output commands
and logical and mathematical operators in the function to
compute and return values.

}
</script>
</head>
```

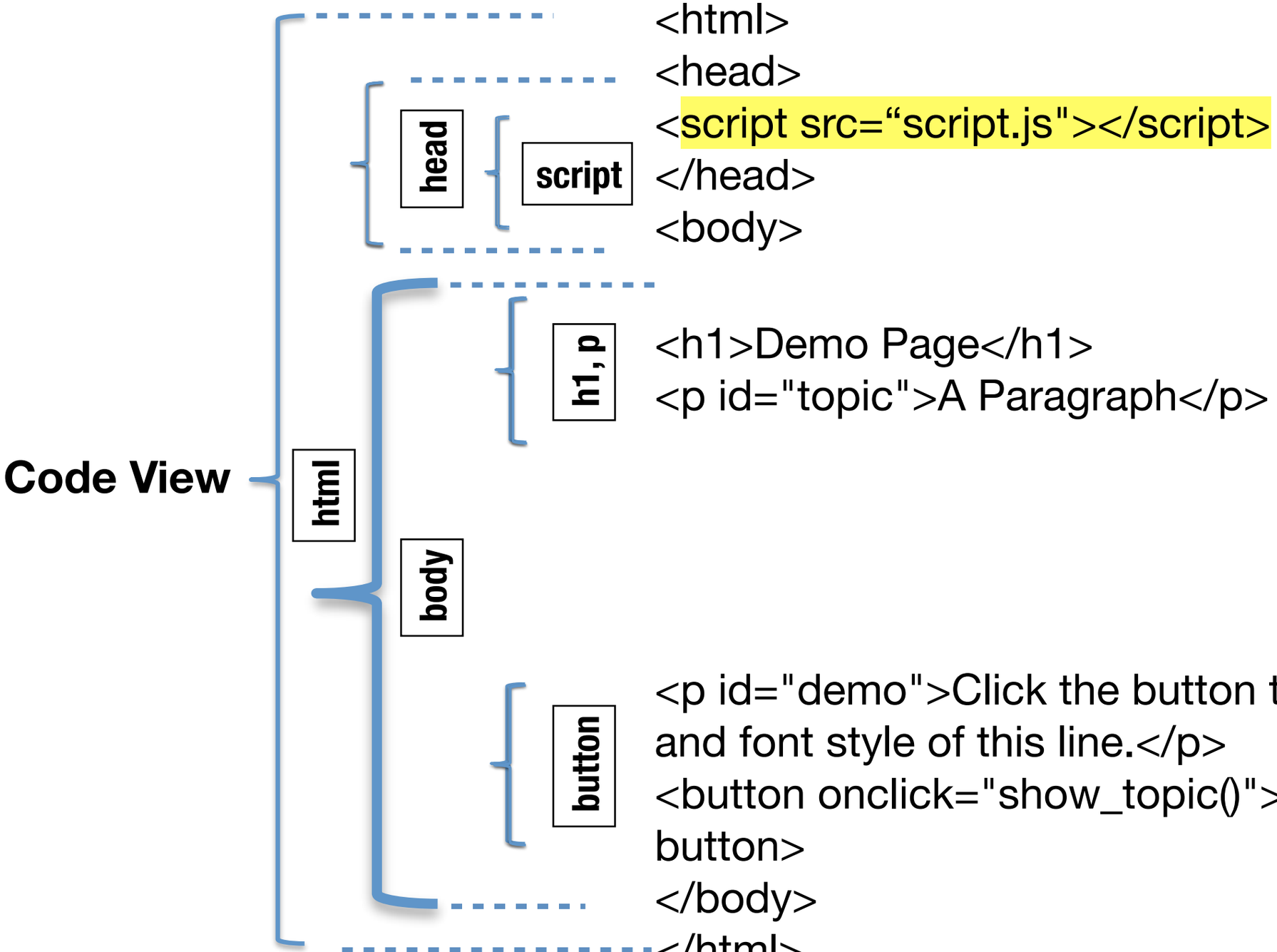
Code View




```
<!DOCTYPE html>
<html>
<head>
<script>
function show_topic() {
    var x = document.getElementById("demo");
    x.style.fontSize = "25px";
    x.style.color = "red";
}
</script>
</head>
<body>
<h1>Demo Page</h1>
<p id="demo">Click the button to change the color and
font style of this line.</p>
<button onclick="show_topic()">Click this button!</
button>

</body>
</html>
```

Similar to CSS, JS can be placed in an External File


Code View
html
head
script
body
h1, p
button
<!DOCTYPE html>
<html>
<head>
<script src="script.js"></script>
</head>
<body>
<h1>Demo Page</h1>
A Paragraph
Click the button to change the color and font style of this line.
<button onclick="show_topic()">Click this button!</button>
</body>
</html>

JavaScript Operations and Commands that Can Enrich the Transformation Process

Basic Logical and Mathematical Operations

- == equal (comparing string and boolean)
- != not equal (comparing string and bollean)
- = equal (comparing numerical values)
- >= greater than or equal to (comparing numerical values)
- <= smaller or equal to (comparing numerical values)
- +,-,*,/,%, &&, ||,! (addition, subtraction, multiplication, division, modular, and, or, not)

Basic Structure of a JavaScript Function

```
<!DOCTYPE html>
<html>
<head><script>
function addition(a, b) {
    a = parseInt(a); b = parseInt(b);
    c = a + b;
    return c;
}
function get_values() {
    var a = prompt("Enter first number:");
    var b = prompt("Enter second number:");
    var z = addition(a,b); alert("The answer is:" + z);
}
</script></head>
<body>
    <button onclick="get_values();">Click here</button>
</body>
</html>
```

Basic Logical and Mathematical Operations

if (condition) {action} else {action}
Examples:

- if (boolean_var == true) {alert(“That is correct”);} else {alert(“That is incorrect”);}
- if (string_var != “David”) {alert(“Not Peter”);}
- if (num_var >= 8) {alert(“The number is greater than or equal to eight.”);} else {alert(“The number is smaller than eight.”);}

Input/Output Commands without Pop-up

- **Entering a variable values through HTML form - e.g.**

```
<script>
function guessInteger() {
        guess = document. forms[‘guessForm’][‘guessValue'].value;
        if (guess == ‘’) {
              document.getElementById(‘demo’).innerHTML = “Empty!”;
              return;
        } else {
              guess_int = parseInt(guess);
          if (guess_int) == 20)
              {document.getElementById(‘demo’).innerHTML = “Right!”;} else
              {document.getElementById(‘demo’).innerHTML = “Wrong!”;}
       return;
 }
</script>
<body>
<form name=‘guessForm’>
        <input name = “guessValue” class=“inputField”>
</form>
<button class=‘button’ onclick=‘guessInteger()'>Guess an Integer</button>
<div id=‘demo’></div>
</body>
```

More Advanced JS Data Structures: Array and Object

- **Array - a list of elements e.g.**
  var fruits = [“apple”,”grape”,”pear”];)

- **Object - a collection of properties represented in name:values pairs e.g.**

```
var student {
        student_id: 1155115511;
        student_fname: “Bernard”;
        student_lname:”Suen”;
        student_major: “EPIN”;
      }
```

Loop

Loop is an iterative programming construct suitable for handling JavaScript array and object.

```
for (initialization; condition; increment) {
    JavaScript statements
}
```

Try the following steps:

```
1) var fruits = [];
2) for (i=1; i< 10 ;i++) {
        fruits[i] =
   prompt("Enter
   fruit:");
   }
3) alert("fruits contain"
   + fruits);
```

Can you write a web application using HTML/CSS/ JavaScript to compute the addition of a consecutive sequence of integers (e.g. 1+2+3..200)?

Extra points will be given to those students whose web application can handle both odd and even numbers and input and output without pop-up prompt and alert.

$$Y = f(a,b)$$

“Ending Number” e.g.
b= 200

“Starting Number” e.g.
a=1

Compute next lower number from the top and next higher number from the bottom.
200->199->198->197….
1->2->3->4….
Sum the results

Sum of
a + b where
a = next lower number from the top and b = next higher from the bottom

“You may need a loop to complete this function.“

Functions in JavaScript Programming

- You can look at a function as a mini-system.
- A function is designed to transform input into output.
- You can execute a function within another function.
- A program can be viewed as a collections of functions decomposed into hierarchy of functions to get things done.
- Good programmer looks for patterns in job to be done and abstract common parameters, algorithms, and outcomes to be placed inside a function for code reuse.

The Grammar of JavaScript

- JavaScript is a programming language that can be used to write functions placed inside html or an external file.
- JavaScript can be placed between the <script> and </script> tags inside the <head> section or link to an external file through the script src link.
- JavaScript codes can be understood as a collection of functions that respond to events triggered by internal browser activities and external user interactions.
- JavaScript can be used to manipulate HTML elements and CSS styles.

Problem Set #2

Write a JavaScript program to create a puzzle (feel free to design a math or word puzzle as you wish). Ask the player to input values through HTML form and display the result inside a box (with an ID selector) positioned inside the browser.

Problem Set #3

- Scrape a website with ParseHub and download the CSV.
- Pick a open dataset and download it for cleaning with Refine.

OR

- Scrape a website with ParseHub and bring it into Refine for cleaning.

Thank You!