Adv Web Dev and Architecture
Lecture 1

review

Amir Amintabar, PhD

Outline

- A question on application design
- OOP review
- Chrome debugging/developer tool
- Coding style rules in this class for this week's lab onward
- HTML review*
- CSS review
- JavaScript review
- Hosting tips
- *The review content in week 1 lecture note is from the pre-requisite courses. It is your responsibility to brush up your knowledge from previous courses. Even if we do not cover all of these slides today, it will be assumed you know these materials.

Why web is so challenging ?

- Compare the challenges of a web app vs a desktop Java/python application development

Web app	Java Desktop app
User can close the page unannounced	The OS signals the Java app before clousure
User can use any browser (even IE or Edge)	Windows, or Mac or Linux that's it
Browser has limited resources (memory, cpu power, GPU power)	Java app has entire resources at hand
User can use any device (phone, tablet, desktop)	Just a desktop computer or thin client
Zillions of frameworks and technologies to learn	Just java and may be Java fx
Security (anyone around the world can access your app)	Its only for you, no one else has access to your computer
DoF service attach, injection,	No one else has access to your computer
Server might go down, network goes down	Its your pc
Remote!	Right in front of you

Q: Suppose you want to develop an application for the following devices, what are the general challenges/ considerations ?

- Mobile device (smartphone)
- Tablet
- Browser
- PC (e.g. Laptop or Desktop)
- IoT nodes (many of which have no screen or a keyboard)

In the following scenarios

- 1- when not connected to any network
- 2. Connected to a local network
- 3. Connected to a global network (internet)
-

Pick a combination of device + scenario and provide your answer in the following fashion

- M.1 refers to mobile not connected to net
- i.3 refers to an IoT node connected to the internet
- So there are a total of 12 scenarios

Tip: What factors would you consider?

- A:
- Security
- Amount of Memory that could be available to the app
- Multitasking
- CPU speed
- GPU speed
- Taking adv of the device Sensors (GEO, Vibration, Camera,

review

OOP in modern JS

Inheritance

- Create a general vehicle class in JavaScript called ***vehicle*** and then extend it to create a specific type of vehicle, a ***car***. This shows the "**is-a**" relatio r "is a" type of vehicle.

What does 'super' do?

```
// Base class Vehicle
class Vehicle {
    constructor(brand) {
        this.brand = brand;
    }

    start() {
        console.log('Starting the vehicle...');
    }
}
```

Extend to

is a

```
// Car class inherits from Vehicle
class Car extends Vehicle {
    constructor(brand, engineType) {
        super(brand);
        this.engineType = engineType;
    }

    displayDetails() {
        console.log(`This is a ${this.brand} car with a ${this.engineType} engine.`);
    }
}
```

Instantiate from

```
let myCar = new Car('BCITMadeCar', 'V1');
myCar.start(); // Starting the vehicle...
myCar.displayDetails(); // This is a BCITMadeCar car with a V1 engine.
```

Composition

- Create separate classes for **Car** and **Engine**, and then compose a car with an engine. This shows the "has-a" relationship where a car "has an" engine.

```
// Engine class
class Engine {
    constructor(type) {
        this.type = type;
    }

    start() {
        console.log(`Starting the ${this.type} engine...`);
    }
}
```

has a

```
// Car class has an Engine
class Car {
    constructor(brand, engine) {
        this.brand = brand;
        this.engine = engine;
    }

    start() {
        this.engine.start();
        console.log(`The ${this.brand} car is now running.`);
    }
```

Instantiate from

```
let myEngine = new Engine('V1');
let myCar = new Car('BCITmadeCar', myEngine);
myCar.start();
// Starting the V1 engine...
// The BCITmadeCar car is now running.
```

Template Strings allow variables in strings

```
let firstName = "John";
let lastName = "Doe";
let text = `Welcome ${firstName}, ${lastName}!`;
```

Automatic replacing of variables with real values is called **string interpolation**.

we do that using **backtick** (`) characters instead of double or single quotes

Source: https://www.w3schools.com/js/js_es6.asp

Inheritance vs composition

- Key Differences
- In the inheritance example, Car **is a** type of Vehicle, **inheriting its properties and methods**.
- In the composition example, Car **has an** Engine as a component.

- The composition approach allows more flexibility.

- For instance, you can easily swap out the Engine for another type without needing to change the Car class. This flexibility is a key advantage of composition over inheritance.

Q:

OOP question: Can we Create Classes in JS?

Somewhere I read, in coding, favor composition over inheritance. What are they? What does this mean?

```
// This code was assisted by ChatGPT, OpenAI.
// Base class
class Car {
    constructor(make) {
        this.make = make;
    }
    drive() {
        return `${this.make} is driving.`;
    }
}
// Inheritance: ElectricCar extends Car
class ElectricCar extends Car {
    constructor(make, batteryLife) {
        super(make);
        this.batteryLife = batteryLife;
    }
    charge() {
        return `${this.make} is charging with
        ${this.batteryLife}% battery remaining.`;
    }
}
```

```
// Composition: Car with an Engine class
class Engine {
    constructor(type) {
        this.type = type;
    }
    startEngine() {
        return `The ${this.type} engine is starting.`;
    }
}
class SportsCar extends Car {
    constructor(make, engine) {
        super(make);
        this.engine = engine;
    }
    accelerate() {
        return `${this.make} is accelerating with a
        ${this.engine.startEngine()}`;
    }
}
```

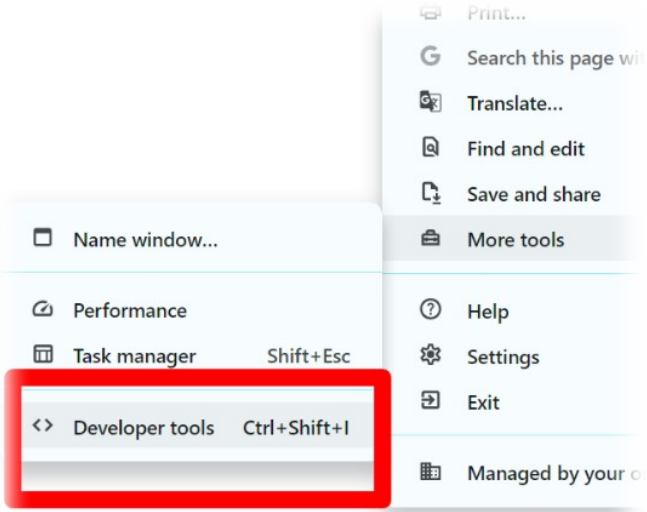
Q1: Now instantiate a Tesla with 80% battery life

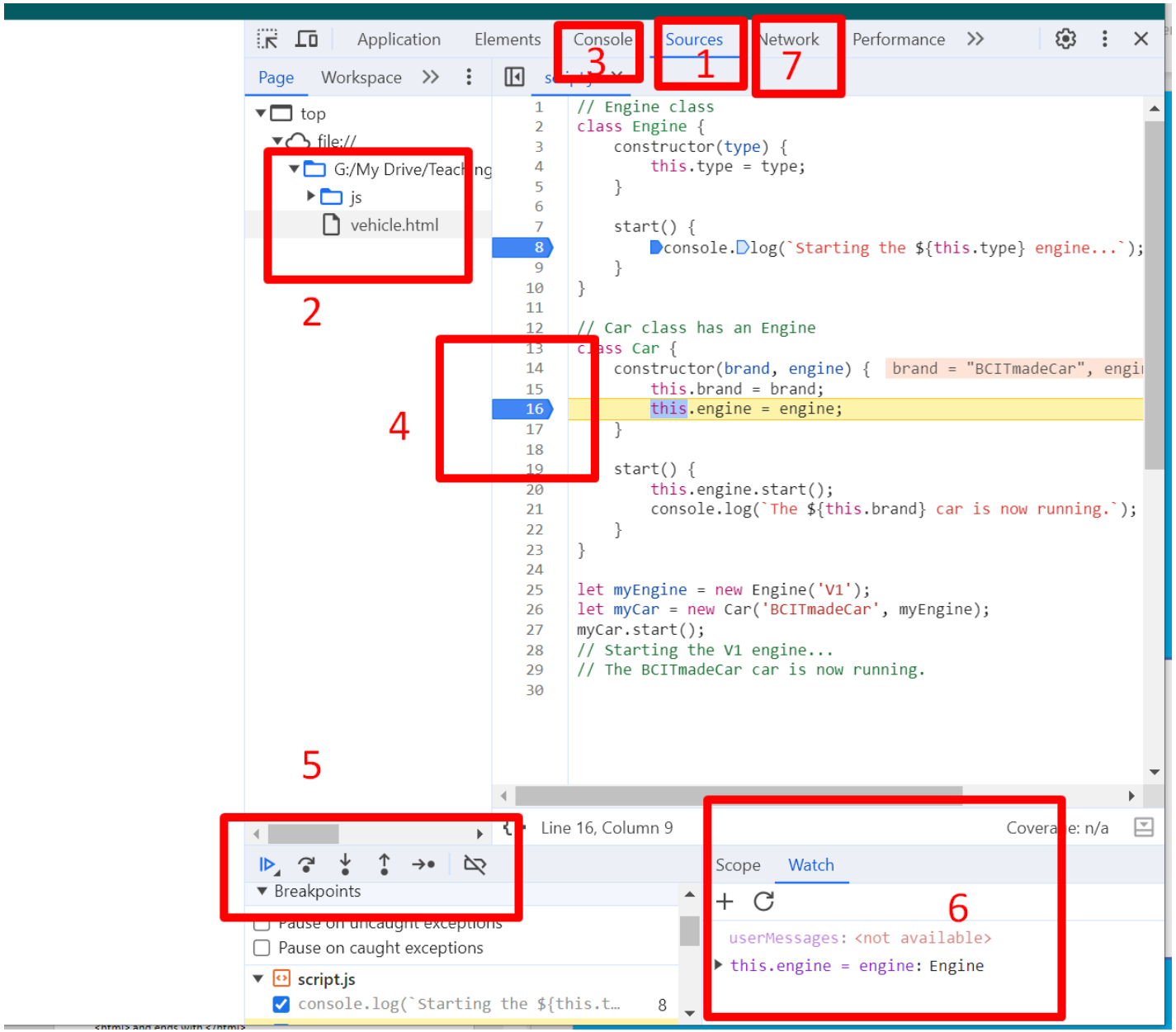
Q2: Create a Ferrari SportsCar with V8 engine

```
const tesla = new ElectricCar('Tesla', 80);
console.log(tesla.drive()); // Tesla is driving.
console.log(tesla.charge()); // Tesla is charging with 80%
battery remaining.
const v8Engine = new Engine('V8');
const ferrari = new SportsCar('Ferrari', v8Engine);
console.log(ferrari.drive()); // Ferrari is driving.
console.log(ferrari.accelerate()); // Ferrari is
accelerating with a The V8 engine is starting.
```

Chrome developer tool

- What does 1, 2, …, 7 do?





In-class activity 2

- A. Using OOP create a class of buttons and display n buttons on screen each with its own color. (define n at the top of your code and give it a value between 3 to 10)

- B. Now add a new method to your class which moves buttons to a random place within the browser window

Coding style for this lab onward

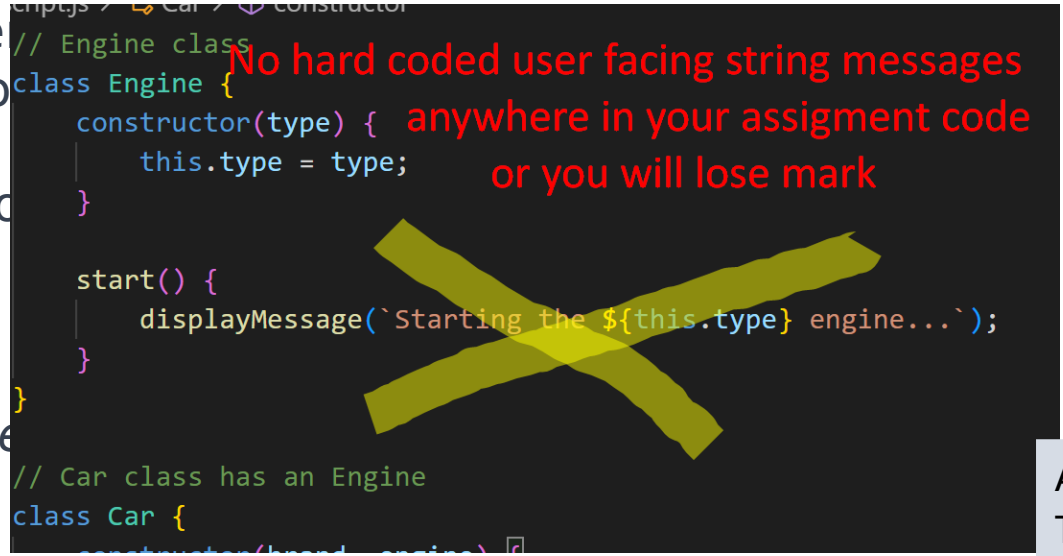
- **Variable Declaration:**
  - Use `const` first and `let` for variable declarations to ensure block-level scoping, with `const` being the default choice for variables that don't change. And never use var
  - Choose clear and descriptive names for variables.
  - If a variable is not meant to be global, it should not be declared as such.

Object-Oriented Practices:

- Use classes and constructor functions to encapsulate and manage related data and behaviors.

- **Function Length:**
  - A function focuses on a single task. Avoid side effects in functions; a function should ideally perform one action or calculate and return a value based on its inputs.
  - Has to be short; a good rule of thumb is that a function should fit on a screen without s

- String and **User Message** Manageme
  - Store user-facing strings, like erro messages and UI labels, in separate files (e.g., JS or JSON) fo easy management and localization.
  - *Absolutely no hard coded user facing string messages anywhere in your assignment code or you will lose mark*
  - Consider implementing a centralized message displaying



Q: How can we store dynamic strings which are formed by joining (concatenating) multiple substrings and value of variables? E.g.

Your score is … for today

A:
Tip:
Store it like
“Your score is %1 for today”

Reference

- chatGPT was used for preparation of the previous contents for proofreading and content generation of this lecture note and may be future ones

HTML, CSS, JavaScript Review* from previous courses

* This week's lecture notes include review material from prerequisite courses. You are responsible for refreshing your understanding of this prior knowledge. Regardless of whether we cover all the slides in today's session, you are expected to be familiar with this content.

What you are expected to know before this class:

- You should have a basic understanding of SQL-based database, JavaScript, HTML and CSS *based on your prior coursework*:

1- Web Development 1
COMP1537: https://www.bcit.ca/outlines/20211087560/

- Create JavaScript classes and objects.
- Create JavaScript code that handles various types of events on th
- Apply the standard three-tiered web architecture (data, applicati
  app with that architecture.
- Utilize various storage mechanisms on both the client-side and se
  requirements.
- Design and implement a Progressive Web Application (PWA).
- Apply core JavaScript concepts such as functions as objects, vario
  control constructs, and how JSON is utilized.

3- Web Development 2 *
COMP2537: https://www.bcit.ca/outlines/20211087580/

Utilize techniques to dynamically serve web content such as image

Perform DOM manipulation via jQuery on the server-side.

Utilize a routing framework on the server-side.

Use asynchronous programming on the client and server

2- Relational Database Design and SQL
COMP1630: https://www.bcit.ca/outlines/20211086178/

- Use SQL - DDL to implement a relational database.
- Use SQL for data manipulation such as the basic Select statement.
- Use SQL for advanced manipulation such as Group BY, Having, Correlated subqueries.
- Create stored procedures and triggers.
- Discuss techniques for transaction management and concurrency control.

Learning Resources

* A few of you might not have taken web dev 2 due to taking co-up

HTML review

HTML Basics

- HTML is a language that browsers (Chrome, Internet Explorer, Microsoft Edge, Safari, Opera) understand.
- HTML is used to form components of a web page

```
1  <html lang="en">
2  <head>
3      <meta charset="UTF-8">
4      <meta name="viewport" content="width=device-width, initial-scale=1.0">
5      <meta http-equiv="X-UA-Compatible" content="ie=edge">
6      <title>Document</title>
7  </head>
8  <body>
9
10 </body>
11 </html>
```

HTML Documents (pages)

- All HTML documents must start with a document type declaration:

  <!DOCTYPE html>.

(to follow standard convention)

- The HTML document itself begins with

  <html> and ends with </html>.

HTML vs CSS vs JavaScript vs DOM

- HTML is used to **create** the **element** of a webpage
- **How** those elements are **logically attached** to each other is determined by a tree graph called **DOM**!

- CSS is used to tell the browser **where** every element is placed and **how it looks**

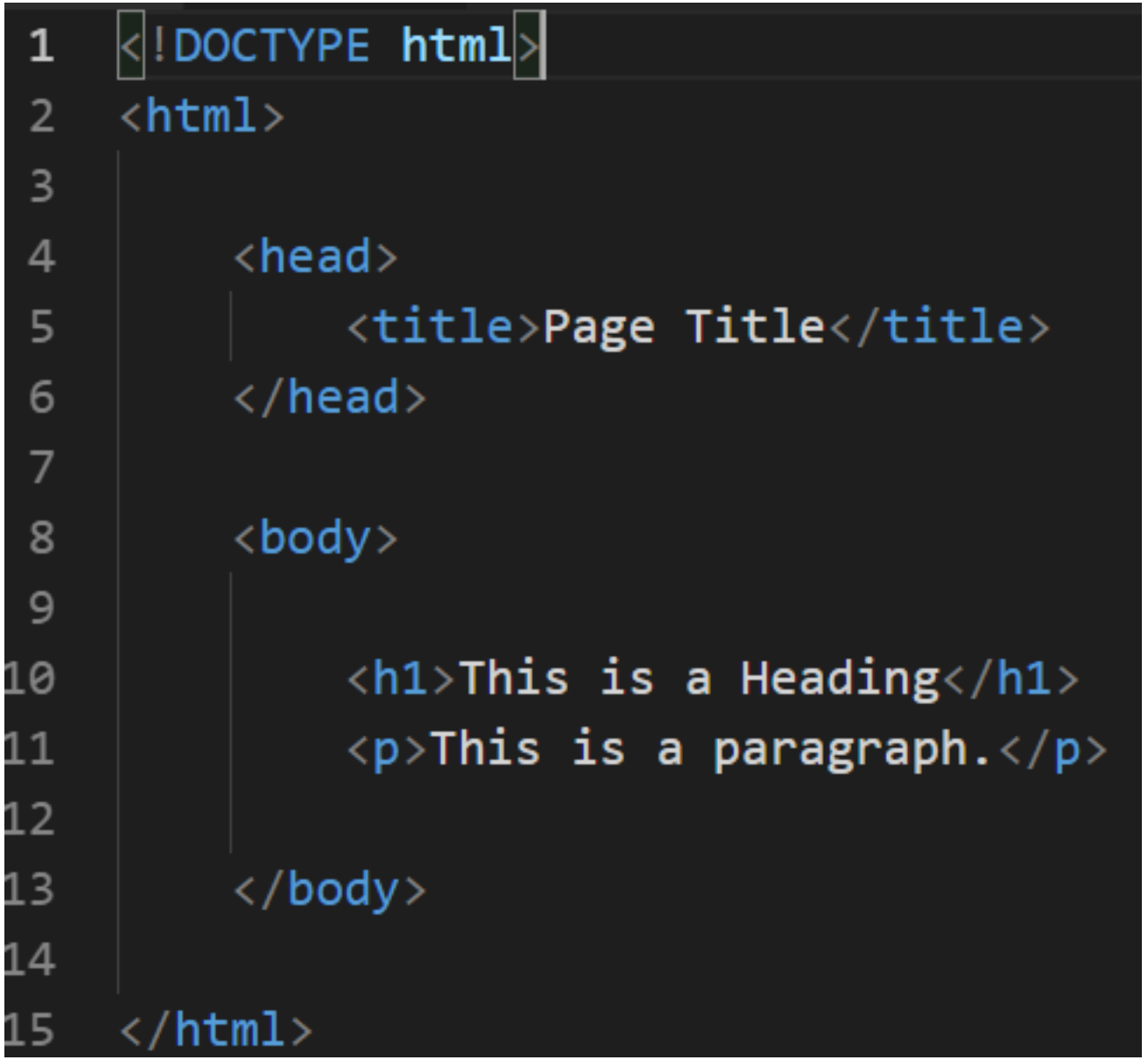
- JavaScript is used to add **interactivity** to elements of a webpage, engage with users by handling events and how to add a new element to the page **dynamically**

- CSS: Cascading Style Sheets
- HTML: Hyper Text Markup Language

Example of an html page:

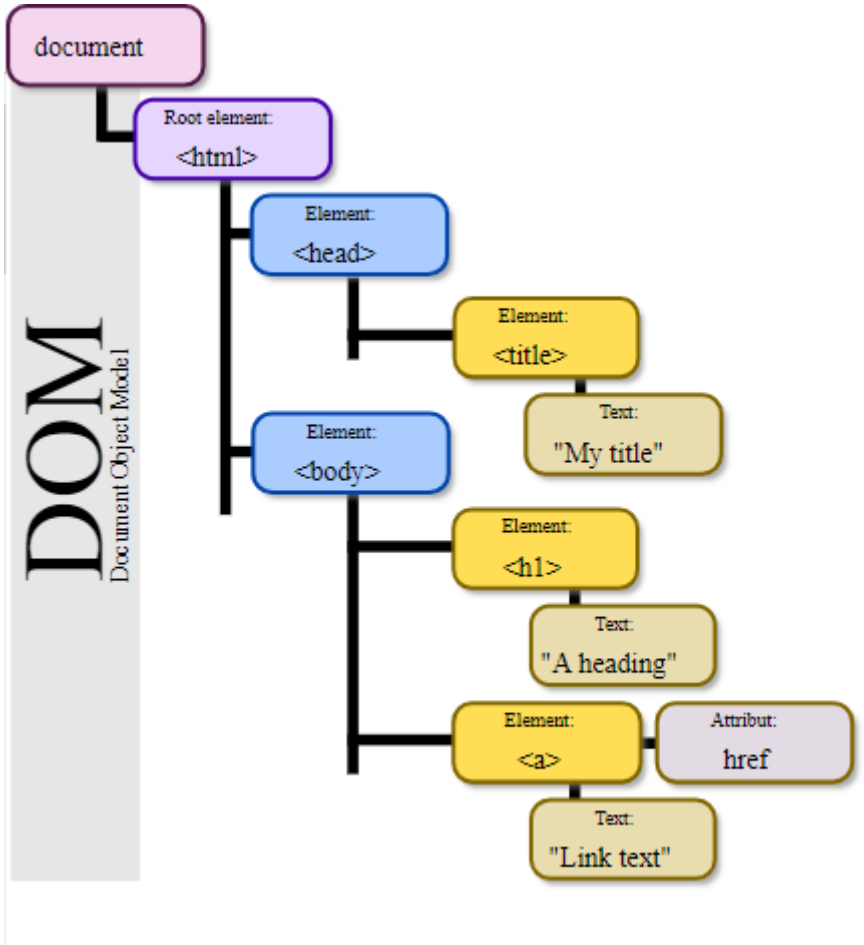
- In this example there is only pure HTML, no css, no JavaScript

- The `<!DOCTYPE html>` declaration defines this document to be HTML5
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the document
- The `<title>` element specifies a title for the document
- The `<body>` element contains the visible page content
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph



HTML Document Object Model (like a tree)

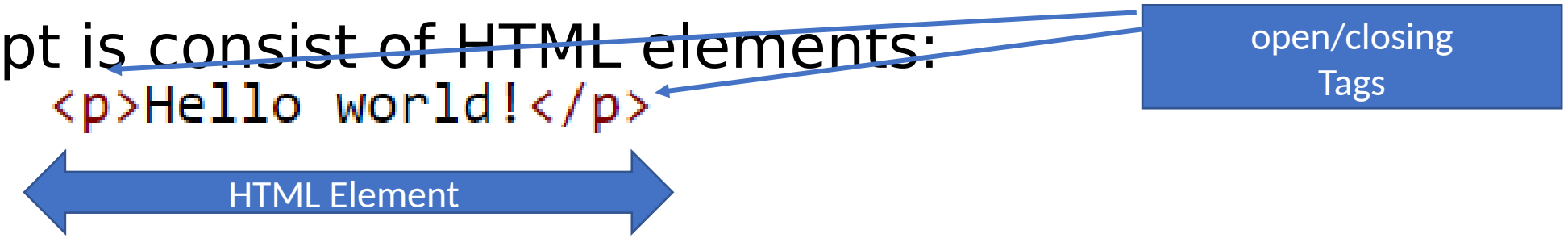
- Notice the tree-like structure of DOM model. Nested tags. We will later use JavaScript to manipulate the DOM model



https://en.wikipedia.org/wiki/Document_Object_Model

HTML Basics

- HTML – Hyper Text Markup Language
- HTML is a ‘markup’ language. It consists of tags: This is bold
- An HTML script is consist of HTML elements:



- Tags generally ‘open’ and ‘close’ – except for single-element tags like
  <img>,
 and <hr>.
- HTML elements are the building blocks of HTML pages
- HTML tags label pieces of content such as "paragraph", "image", and so on

HTML elements and their attributes

HTML Links <a>

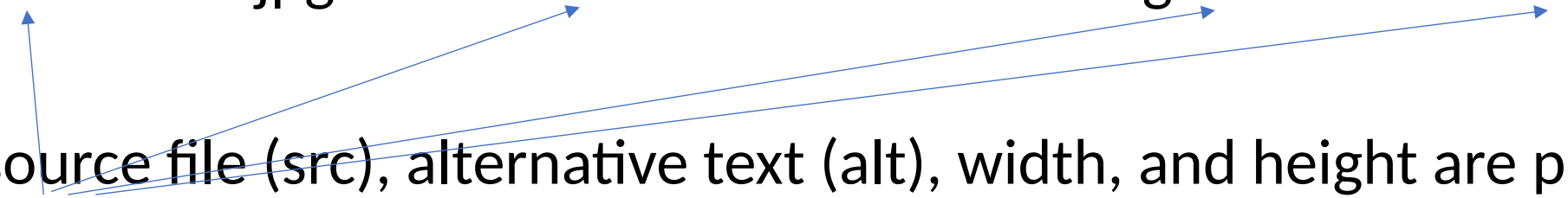
- This is a link

- The link's destination is specified in the href **attribute**.
- Attributes are used to provide additional information about HTML elements.

HTML elements and their attributes

HTML Images <img … >

- <img src="bcit.jpg" alt="bcit.ca" width="104" height="142">



- The source file (src), alternative text (alt), width, and height are provided as **attributes**.(pair:value attributes)
- **Q**: what does the attribute “alt” do?
- **Q**: Why do you think we need the attribute “alt”?

What if we invent an HTML tag the browsers don’t know

- Browsers do not display the HTML tags themselves, but use them to render the content between or inside them on the page.
- E.g. when the browser sees hello it does not show the or

- Everything is inside a tag! **Q**: What does this one mean?
- What happens if some text is NOT inside a tag?

- **Q**: what if we invent new tag?
- **Q**: What happens if the browser does not understand a tag? Say <HelloWorld>Some text </ HelloWorld >

Internet technical terms

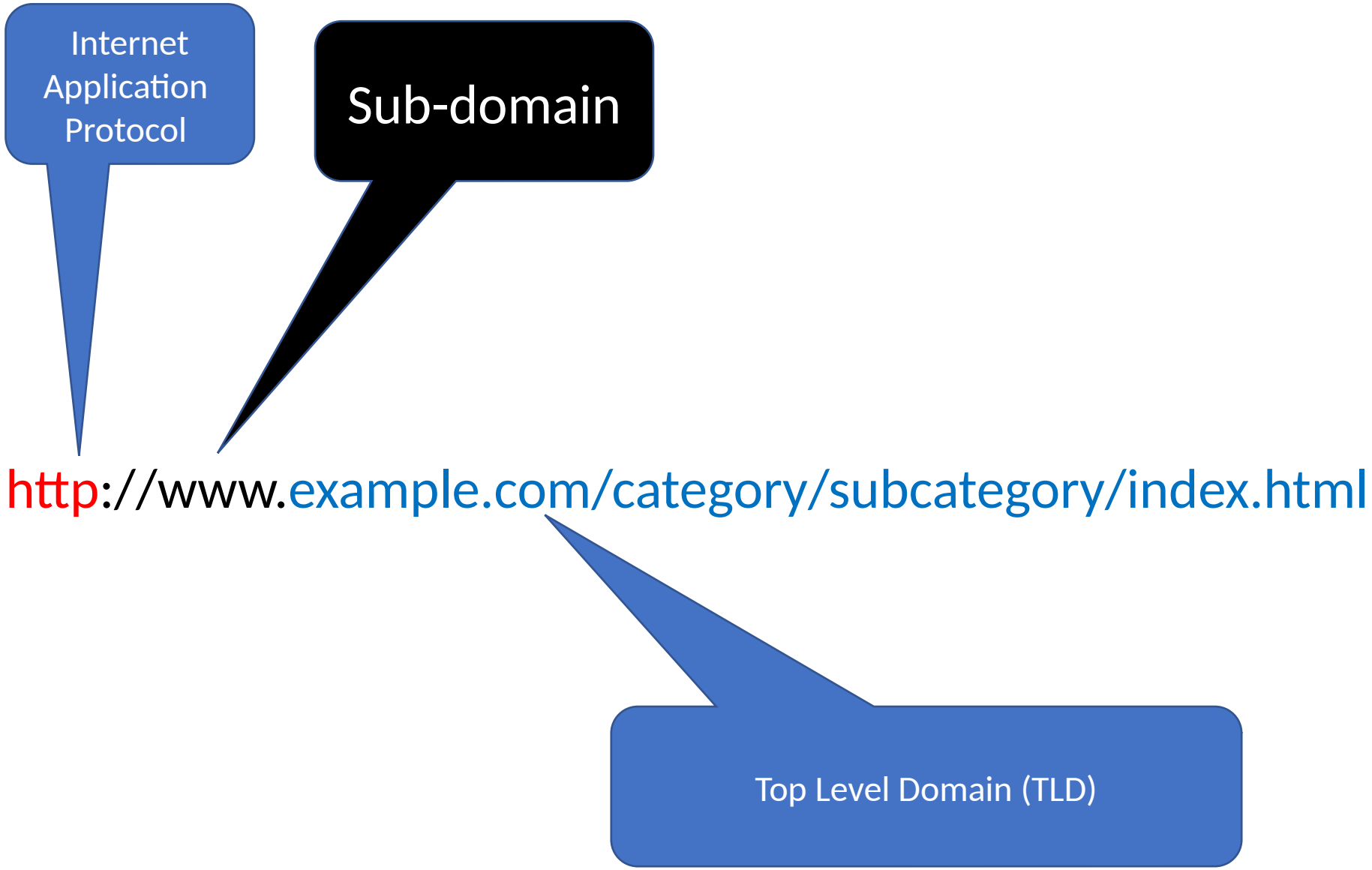
HTTP Hypertext Transfer Protocol

- The transfer protocol is the set of rules that the computers use to move files from one computer to another on the Internet.
- The most common transfer protocol used on the Internet is the Hypertext Transfer Protocol (HTTP).
- HTTPS stands for Hypertext Transfer Protocol Secure (it uses separate protocols called SSL)
- Two other protocols that you can use on the Internet are the File Transfer Protocol (FTP) and the Telnet Protocol (TP)

Uniform Resource Locators

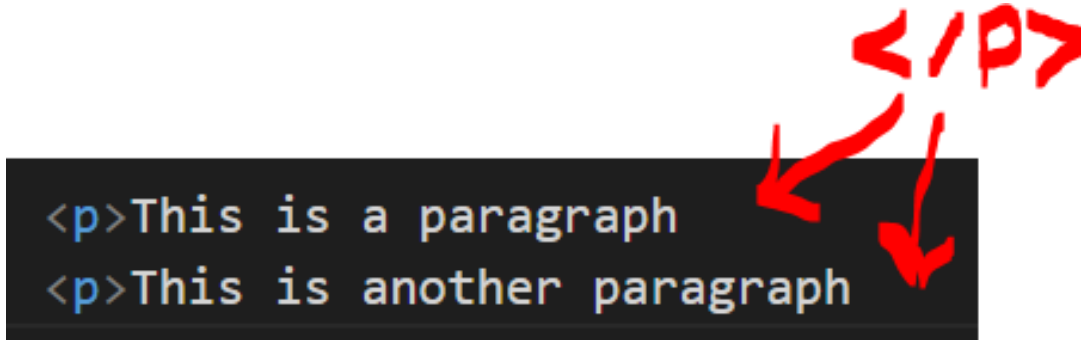
- The IP address and the domain name each identify a particular computer on the Internet.
- However, they do not indicate where a Web page's HTML document resides on that computer.
- To identify a Web pages exact location, Web browsers rely on Uniform Resource Locator (URL).
- URL is a four-part addressing scheme that tells the Web browser:
  - What transfer protocol to use for transporting the file
  - The domain name of the computer on which the file resides
  - The pathname of the folder or directory on the computer on which the file resides
  - The name of the file

Structure of URL (Uniform Resource Locators)



Tip1: Do not forget the end tag

- Sometimes entire page can go wrong,
- However some HTML elements will display correctly, even if you forget the end tag:
- Example:



Tips …

- Tip 2: Remember most of HTML tags are paired tags (open and close tags), and there are few single tags like
. It's a good practice to close single tags in the opening tags:

```
<br/>
```

- Tip3: Single tags are also referred as empty tags as there is nothing between open and close tags)

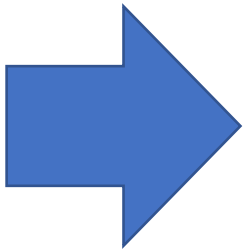
- Tip 4: HTML tags are not case sensitive: <P> means the same as <p>. (we use lower case only)

HTML Lists

- HTML lists are defined with the <ul> (unordered/bullet list) or the <ol> (ordered/numbered list) tag, followed by <li> tags (list items):

```
<h2>An Unordered HTML List</h2>
<ul>
  <li>Alpha</li>
  <li>theta</li>
  <li>Delta</li>
</ul>

<h2>An Ordered HTML List</h2>
<ol>
  <li>Alpha</li>
  <li>theta</li>
  <li>Delta</li>
</ol>
```



An Unordered HTML List

- Alpha
- theta
- Delta

An Ordered HTML List

1. Alpha
2. theta
3. Delta

HTML input elements

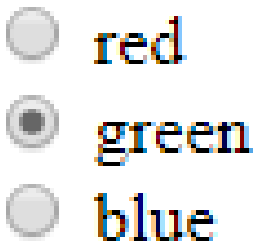
Input buttons

```
<input type="button" value="Try it">
```

- **Q:** How many attributes are these in the code snipped above?
- **Q:** Now change the "Try it" caption to "Click me"
- **Q:** Now make the button linked to this image "https://media.giphy.com/media/Wsx8SB3gOyWZ2/giphy.gif
"

Radio buttons

```
<input type="radio" name="color" value="red"> red      <br>
<input type="radio" name="color" value="green"> green  <br>
<input type="radio" name="color" value="blue"> blue
```



The browser rendered them in same group as all had the name

Text formatting elements

- Elements like and make text **bold** or *italic* .
- - Bold text
- - Important text
- - Italic text
- - Emphasized text
- <mark> - Marked text
- <small> - Small text
- <del> - Deleted text
- <ins> - Inserted text
- <sub> - Subscript text
- <sup> - Superscript text

[--] Try all of these tags and see the results

This text is normal.

This text is bold.

This text is italic.

…

<b> vs , <i> vs

- Both looks just the same however:

- The HTML <b> element defines bold text, without any extra importance.
- adds *semantic* "strong" importance too. **Q**: where is this helpful?

- The HTML <i> element defines italic text, without any extra importance. However added semantic importance.

Quotation and Citation semantic Elements

Tag	Description
<abbr>	Defines an abbreviation or acronym
<address>	Defines contact information for the author/owner of a document
<bdo>	Defines the text direction
<blockquote>	Defines a section that is quoted from another source
<cite>	Defines the title of a work
<q>	Defines a short inline quotation

- **Q:** Give one example where these semantic elements are important

HTML Styles

1- Styling HTML elements using the style attribute

Remember what the attributes of an HTML element were

```
<a href="https://bcit.ca">Visit bcit!</a>
```

Attribute

```
<tagname style="property:value;">
```

```
<img src="life.gif" alt="life is good!" height="42">
```


Q: Did you notice any difference when comparing “style“ with other attributes ?

- Setting The HTML style attribute follows the same syntax as other attributes However there are two differences:
- 1- “style” is the default attribute of all elements versus “src” which is only the attribute of few elements such as img and video.
  - So **style** is a **global** attribute.
- 2- The value of the style attribute is not a single scalar number, its an object with various property-value pairs)

```
<tagname style="property:value;">
```

- List of all html attributes: https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes

Background color

- Example: changing the background of the body.
- Tip: remember all HTML elements, including body, has a an attribute named style

```
<body style="background-color: powderblue;">

    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>

</body>
```

- [--] Now try it on your laptop!

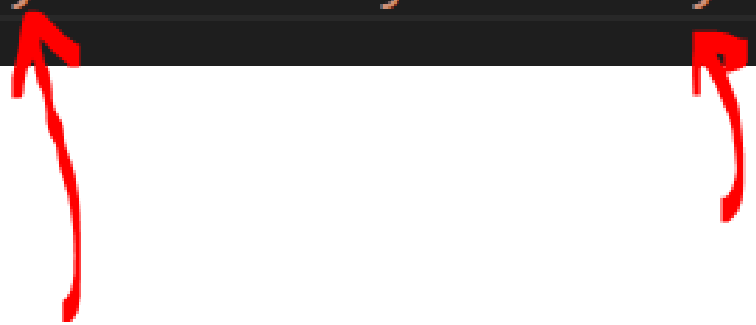
Text color, Font type

- Q: can we set both the color and the font of an element all at once in same line?

```
<h1 style="color: blue;">This is a heading</h1>
<p style="color: red;">This is a paragraph.</p>
```

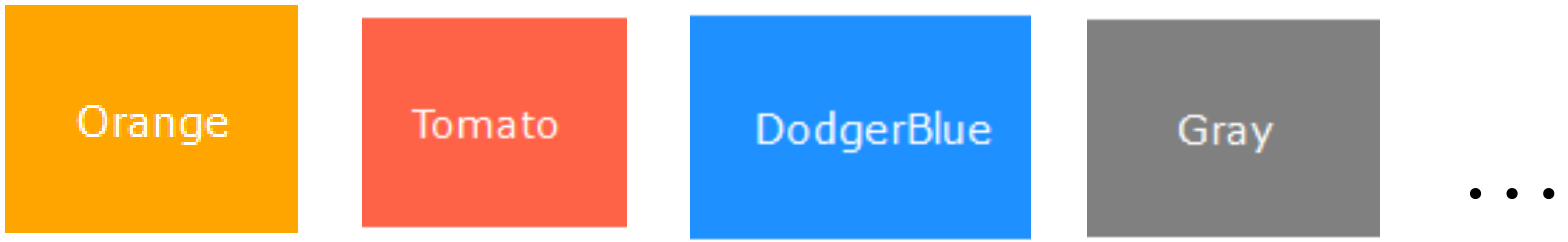
```
<h1 style="font-family:verdana;">This is a heading</h1>
<p style="font-family:courier;">This is a paragraph.</p>
```

```
<h1 style="color: blue;font-family:verdana;">This is a heading</h1
```



Specifying colors in HTML

1- using a predefined color names such as



- **Q:** Guess how many predefined color names are there in HTML?
- check: https://www.w3schools.com/colors/colors_names.asp

2- RGB, HEX, HSL, RGBA, HSLA values

Red
0 to 255

Green
0 to 255

Blue
0 to 255

Same as color name "Tomato":

Hello

Same color but described in Hexadecimal:

Hello

Same as color name "Tomato", but 50% transparent:

Hello

```
<p>Same as color name "Tomato":</p>
<h1 style="background-color: rgb(255, 99, 71);">Hello</h1

<p>Same color but described in Hexadecimal:</p>
<h1 style="background-color: #ff6347;">Hello</h1>

<p>Same as color name "Tomato", but 50% transparent:</p>
<h1 style="background-color: rgba(255, 99, 71, 0.5);">Hello</h1>
```

Alpha value
0 to 1
1: Opaque
0.5: 50% Opaque
0: Transparent

What's this?

https://www.w3schools.com/html/html_colors.asp

Q: Which way would you prefer to refer to a color in your page?

- A) Using color names
- B) using RGB
- C) using hexadecimal representation

Font size, Text Alignment

- style="font-size:300%;“

- style="text-align:center;"

- center , Left, right, justify

Border Color

```
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

Hello World

Hello World

- **Q:** What if we wanna style a group of elements?

- **Q:** What if we wanna style a all the hyperlinked texts (the ones in <a> tag in the page ?
-

There are 3 ways of styling HTML elements

- 1- Styling HTML elements using the style attribute

  This method is called **Inline** styling which we just learned today.

- 2- Using <style> tags in the <head> section of same HTML file

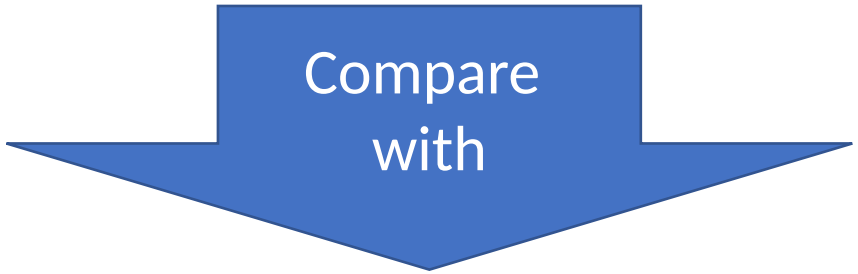
  This method is called **Internal** styling

- 3- Using an external CSS file

  This method is called **External** styling

- **CSS** stands for **C**ascading **S**tyle **S**heets.
- CSS describes **where** every element is placed and **how** it looks on screen, paper, or in other media (remember this from last week).

1- Styling HTML elements (inline method) using the style **attribute**



2- Styling HTML elements (internal method) using the <style> **elements** in <head>

Styling (internal method) using the <style> tags in <head>

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        h1 {
            color: blue;
        }
    </style>
</head>

<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <h1>This is also a heading</h1>
</body>

</html>
```

Q: Compare with same inline method:
What are the differences (in terms of usage and scope)

```html
<h1 style="color: blue;">This is a heading</h1>

<h1 style="color: blue;">This is another heading</h1>
```

A:
1- In internal method, Style acts as a tag (an element)
2- In internal methods all the h1 elements get affected
vs
in the inline method we need to set the style attribute for every single element separately

More HTML elements

The Table Element <table>

Key elements in forming a table:

- <table> Element
  Contains the table

- <**tr**> Element
  Contains a table **row**

- <**td**> Element
  Contains a table **data** cell

- <**th**> Element
  Contains a table **header** cell

HTML Tag

- It creates a section in an HTML document
- Try this
- ```

    <h3>This is a heading</h3>
    This is a nice short paragraph.

  ```

- Q: Add another div tag to same page and place two images inside that div tag
- Q: Now by using only one a tag (anchor tag) link both images to BCIT
  ```

Two more global attributes:

- id
- class

id, a global attribute

- Remember what was an element
- Remember what was an attribute . Attribute like src of an image element. Or href of an a element
- Some attributes where global like “style”.
- When we say “style” is a global attribute that means we can set the value of style attribute for every single HTML element
- There is another attribute which is global too. It is called ”id”.
- **Its value has to be unique**. Just like your student id, there is only one students in this class with your student id. We can gi
- There has to be only one element in that pa
- ello World!</h1>

As seen all of these elements have different id values

id

```
<h1 id="myHeader">Hello World!</h1>
<p id="id1">Hello World!</p>
<p id="id2">Hello World!</p>
```

class, a global attribute

- There is also another attribute which is global too. It is called ”class”.
- **Its value dose not have to be unique**. That means, multiple elements in the page can have same class value
- There has to be only one element in that page with same id
- Example
- <h1 class= "greenPeople" id="myHeader">Hello World!</h1>

```
<h1 class = "bb" id = myHeader"> Welcome to BCIT </h1>
<p  class = "bb" id = id1"> BCIT is good </p>
<p  class = "green" id = id2"> Welcome to Vancouver </p>
<p  class = "green" id = myVancouver"> Vancouver is rainy! </p>
```

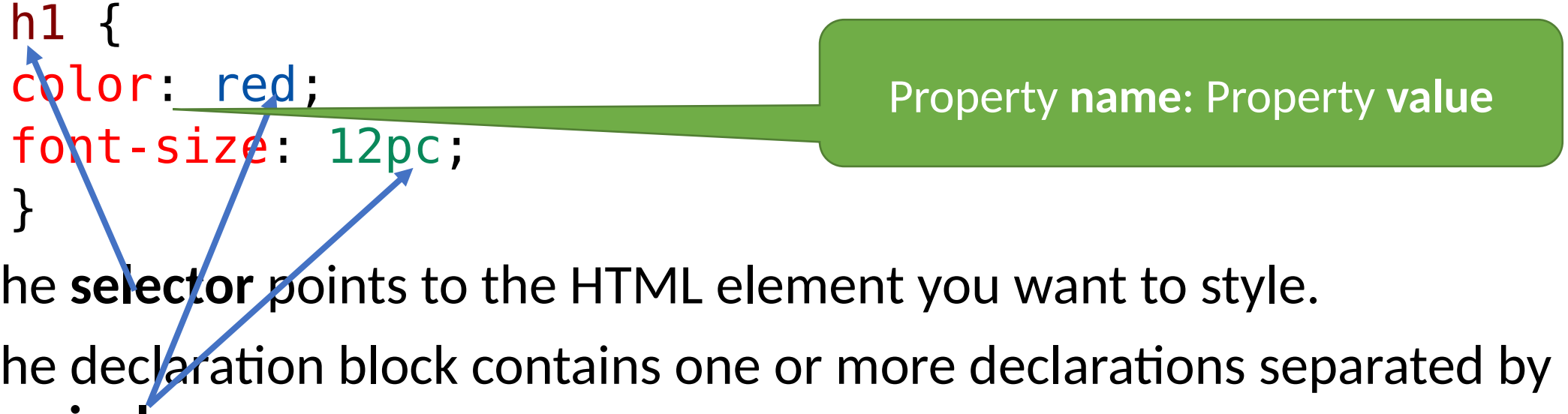
Its ok to give same class value to multiple elements

CSS language for styling

Selectors in CSS

- CSS is used to define styles, design, layout and variations in display for different devices and screen sizes.
- A CSS **rule-set** consists of a **selector** and a **declaration block**:

```
h1 {
color: red;
font-size: 12pc;
}
```



- The **selector** points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by **semicolons**.
- Each declaration includes a CSS property **name** **:** **value** pair.
- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

CSS **selectors**
to **select** a group of element
for styling

Selectors

- A CSS **rule-set** consists of a selector and a declaration block:

```
h1 {
color: red;
font-size: 12pc;
}
```

- CSS **selectors** are used to "find" (or select) HTML elements based on their
- element name,
- id,
- class,
- attribute,
- and more.

element selector

- The element selector selects elements based on the element name.

```
<head>
  <style>
    p {
      text-align: center;
      color: blue;
    }
  </style>
</head>

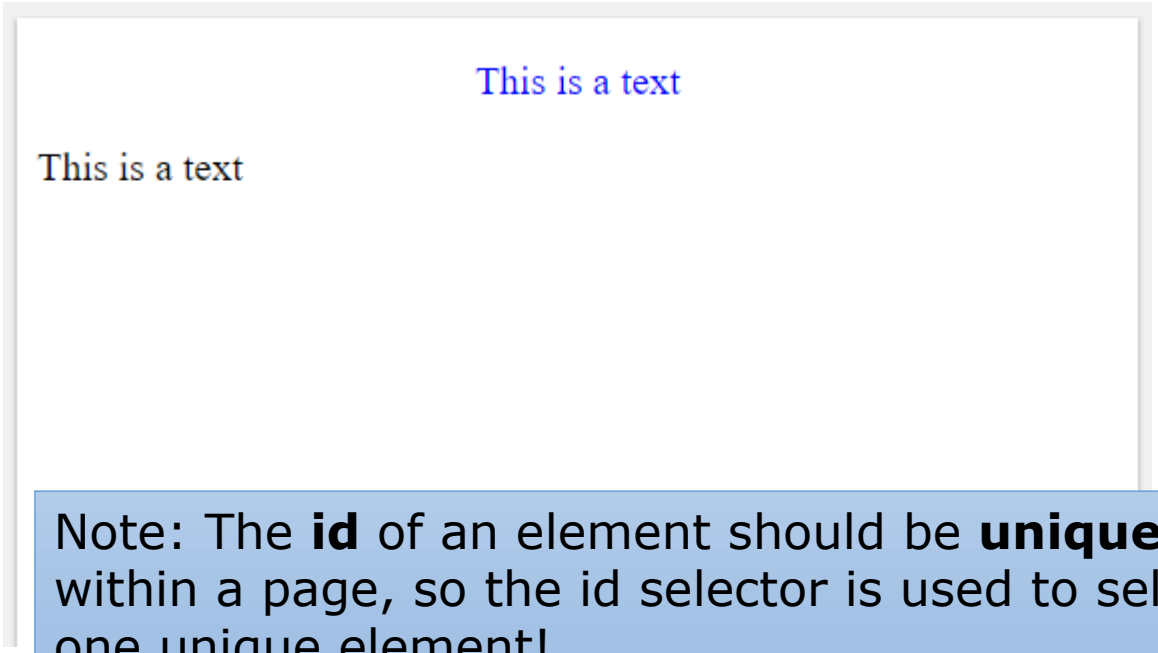
<body>
  <h3>This is a text</h3>
  <p>This is a text</p>
  <p>This is a text</p>
</body>
```

id selector

- The id selector uses the id **attribute** of an HTML **element** to select a specific element.

```
<head>
  <style>
    #firstId {
      text-align: center;
      color: blue;
    }
  </style>
</head>

<body>
  <p id="firstId">This is a text</p>
  <p>This is a text</p>
</body>
```



Note: The **id** of an element should be **unique** within a page, so the id selector is used to select one unique element!

Q: what would happen if we set the id attribute of the second p element to firstId too?

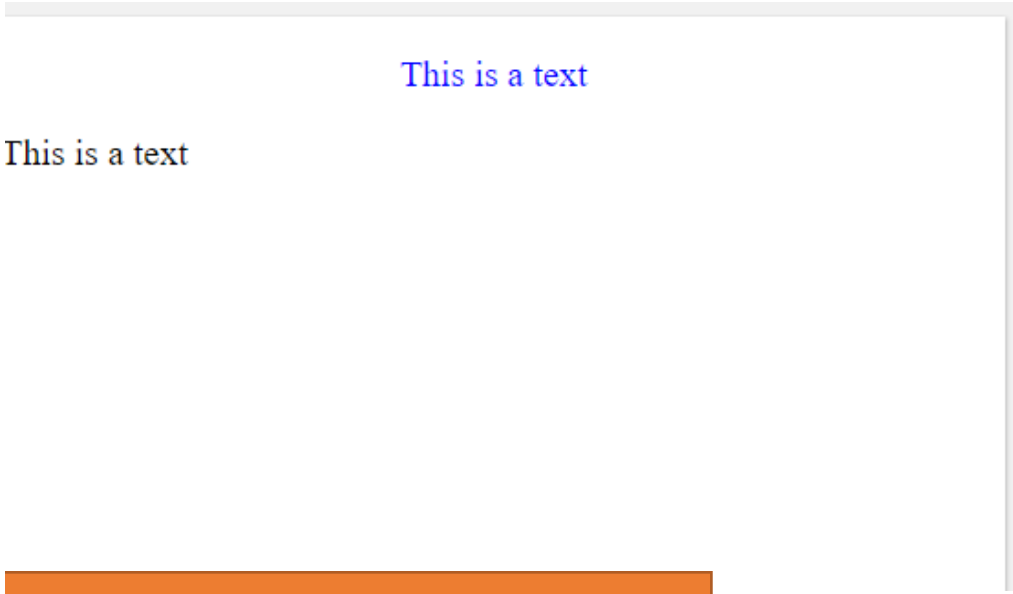
A: bad practice

class selector

- Remember “class” was also one of the global attributes of all HTML elements
- The class selector selects elements with a **specific class attribute**.
- **Q:** Now style the previous example using the class attribute

```
<style>
  .firstClass {
    text-align: center;
    color: blue;
  }
</style>
</head>

<body>
  <p class="firstClass">This is a text</p>
  <p>This is a text</p>
</body>
```



Q: what would happen if we set the class attribute of the second p element to firstClass too?

mix and match selectors

- **Q:** In the HTML code below, there are three elements of type p and h3. Two of the elements have both the same class attribute “myClass”. Without using id attribute and without using inline styling how can you change the color of first p tag tag to blue?

```
<body>
  <h3 class="myClass">Th
  <p class="myClass">Thi
  <p>This is text 3</p>
</body>
```

This is text 1

This is text 2

This is text 3

```
<head>
  <style>
    p.myClass {
    color:  blue;
    }
  </style>
</head>

<body>
  <h3 class="myClass">This is text 1</h3>
  <p class="myClass">This is text 2</p>
  <p>This is text 3</p>
</body>
```

More than one class for an element

- HTML elements can refer to more than one class
- Here the <p> element will be styled according to class="center" and to class="large"

```
<head>
  <style>
    .center {
      text-align: center;
    }

    .large {
      font-size: 200%;
    }
  </style>
</head>
<body>
  <p class="center">Center-aligned.</p>
  <p class="center large">Center-aligned and large</p>
</body>
```

Grouping selectors (separate each selector with a comma)

```
<style>
  h1 {
    text-align: center;
    color: green;
  }

  h2 {
    text-align: center;
    color: green;
  }

  p {
    text-align: center;
    color: green;
  }
</style>
```

```
<style>
  h1, h2, p {
    text-align: center;
    color: green;
  }
</style>
```

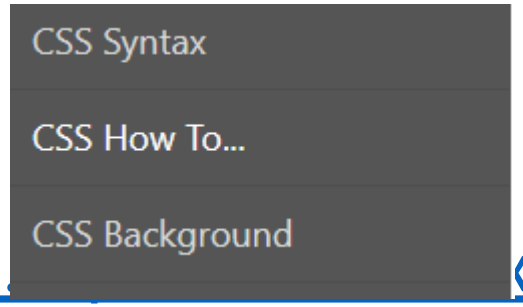
Better practice to place selectors underneath each other

```
<style>
  h1,
  h2,
  p {
    text-align: center;
    color: green;
  }
</style>
```

CSS comments

```
<style>
  p {
    color: yellow;
    /* This is a single-line comment */
  }
  /* This is
    a multi-line
    comment */
</style>
```

- **Q:** complete these three sections of exercises



- here : https://www.w3schools.com/css/exercise_syntax1

Introduction to
JavaScript

Let’s see
few example what **JavaScript**
can do

HTML vs CSS vs JavaScript vs DOM

- HTML is used to **create** the **element** of a webpage
- **How** those elements are **logically attached** to each other is determined by a tree graph called **DOM**!

- CSS is used to tell the browser **where** every element is placed and **how it looks**

- JavaScript is used to add **interactivity** to elements of a webpage, engage with users by handling events and how to add a new element to the page **dynamically**

- CSS: Cascading Style Sheets
- HTML: Hyper Text Markup Language

Example 1 Document write

- Document write to write some HTML contents on screen (browser's screen)

```
1  <!DOCTYPE html>
2  <html>
3
4  <body>
5
6      <script>
7          document.write("Hello World!");
8      </script>
9
10 </body>
11
12 </html>
```

- Note 1: to place JavaScript in our HTML page we use the element <script> </script>

- Note2: we can place JavaScript code in <head> or in the <body>

Try document.wirte("Hello ****world****!")

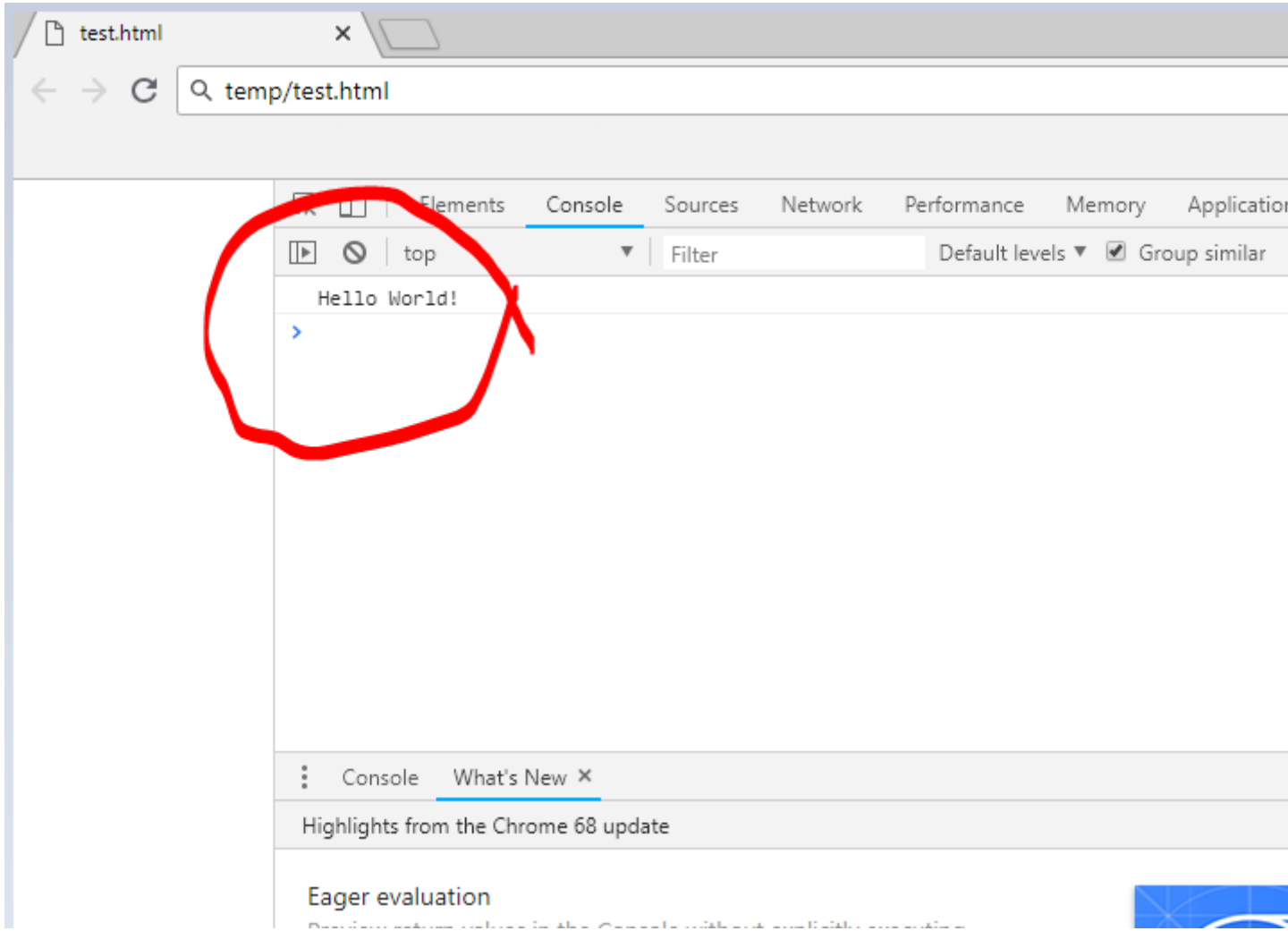
Note! JavaScript runs in the browser (not outside it)

- **Note**: When we say document.write write on screen; by screen we always mean **browser's screen** as JavaScript only runs within the browser; it cannot do anything outside browser.
- JS cannot delete your files for example. (you cannot write a JavaScript code to delete a file from your local hard-drive or server hard-drive!)

Example 2 Console log

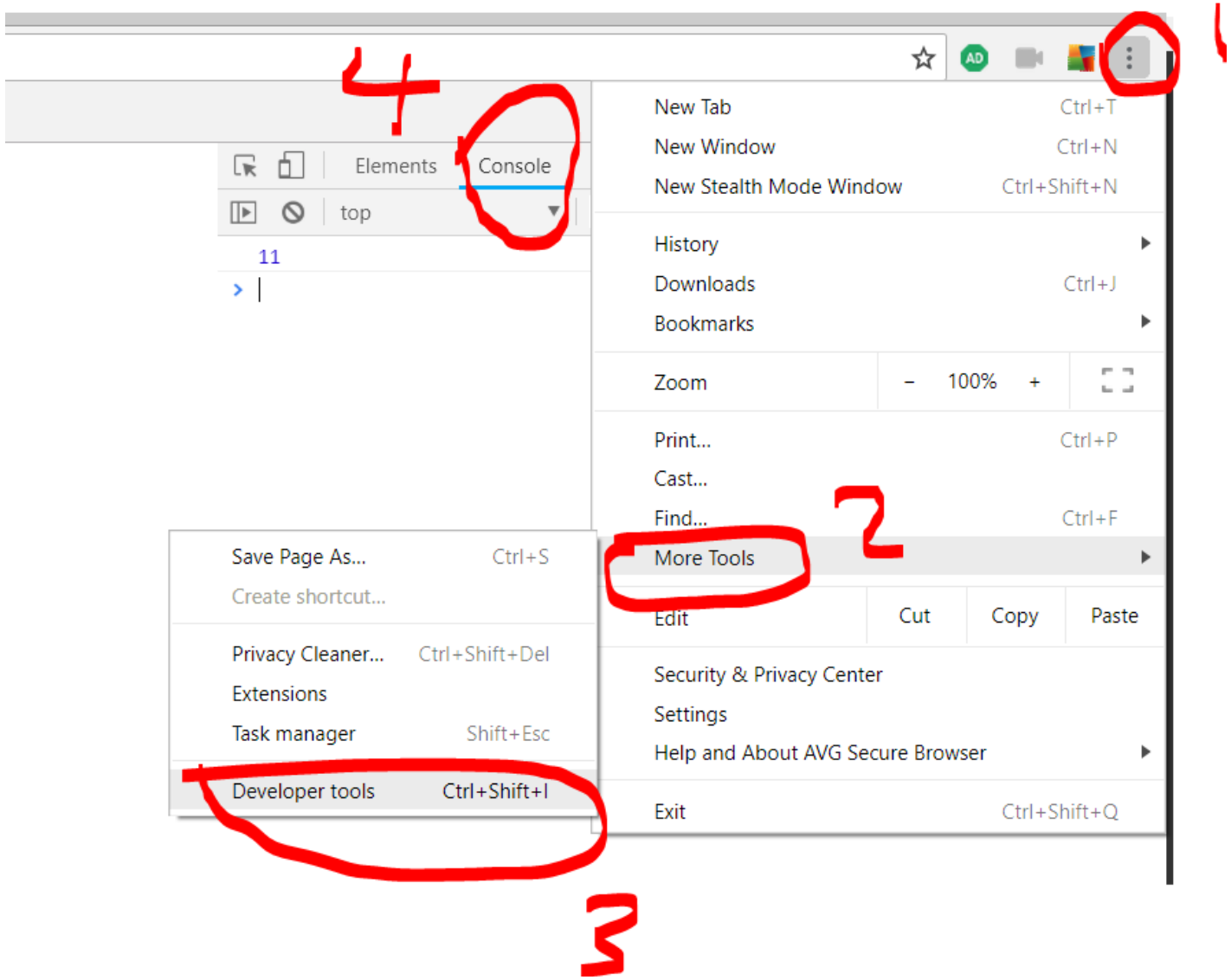
- Writing on console

```
1  <!DOCTYPE html>
2  <html>
3
4  <body>
5
6      <script>
7          console.log("Hello World!");
8      </script>
9
10 </body>
11
12 </html>
```



Here is how to open console window in Chrome

Note the debugging tools in there

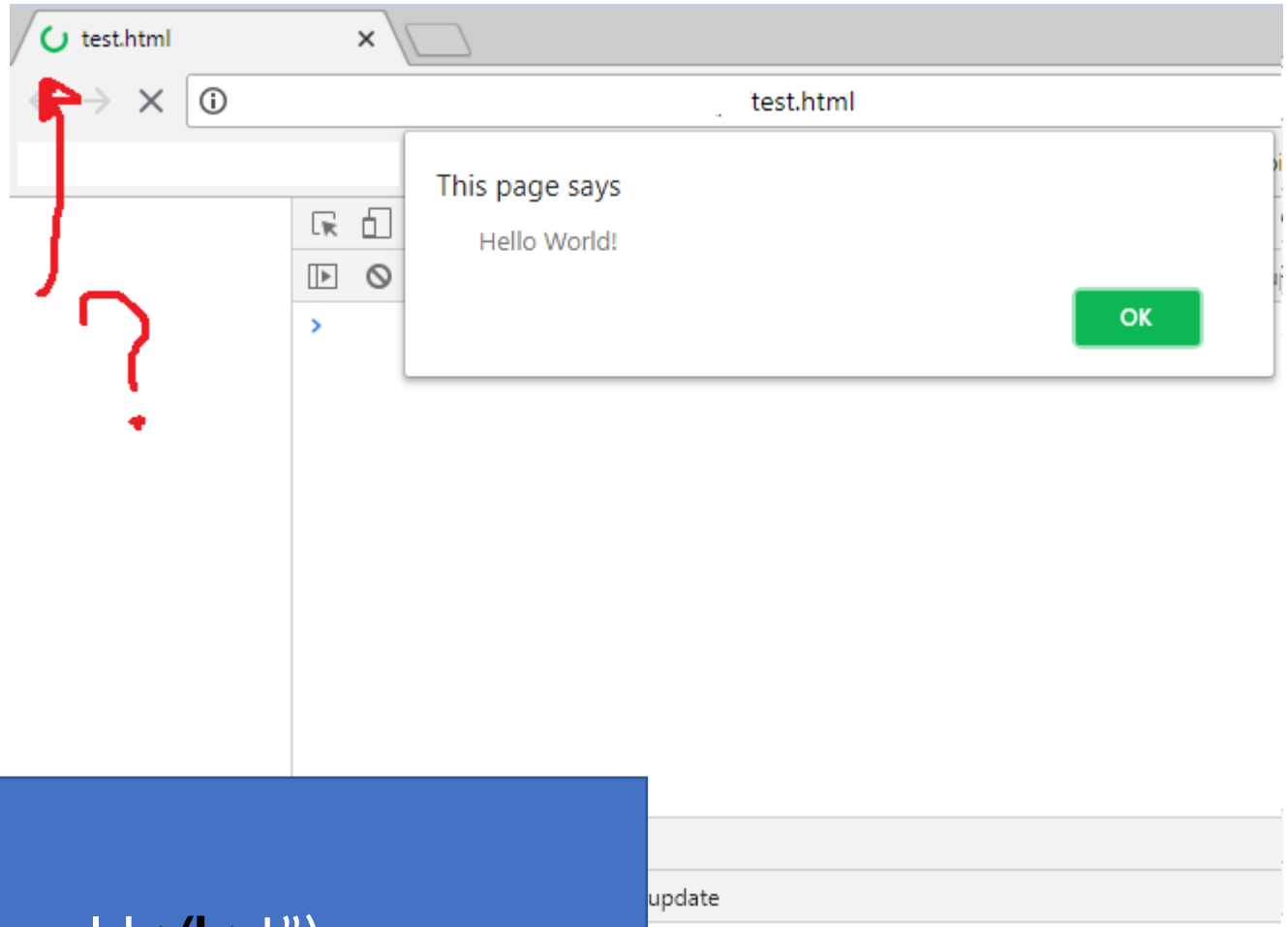


Example 3 pop up alert message

```
<body>

    <script>
        window.alert("Hello World!");
    </script>

</body>
```

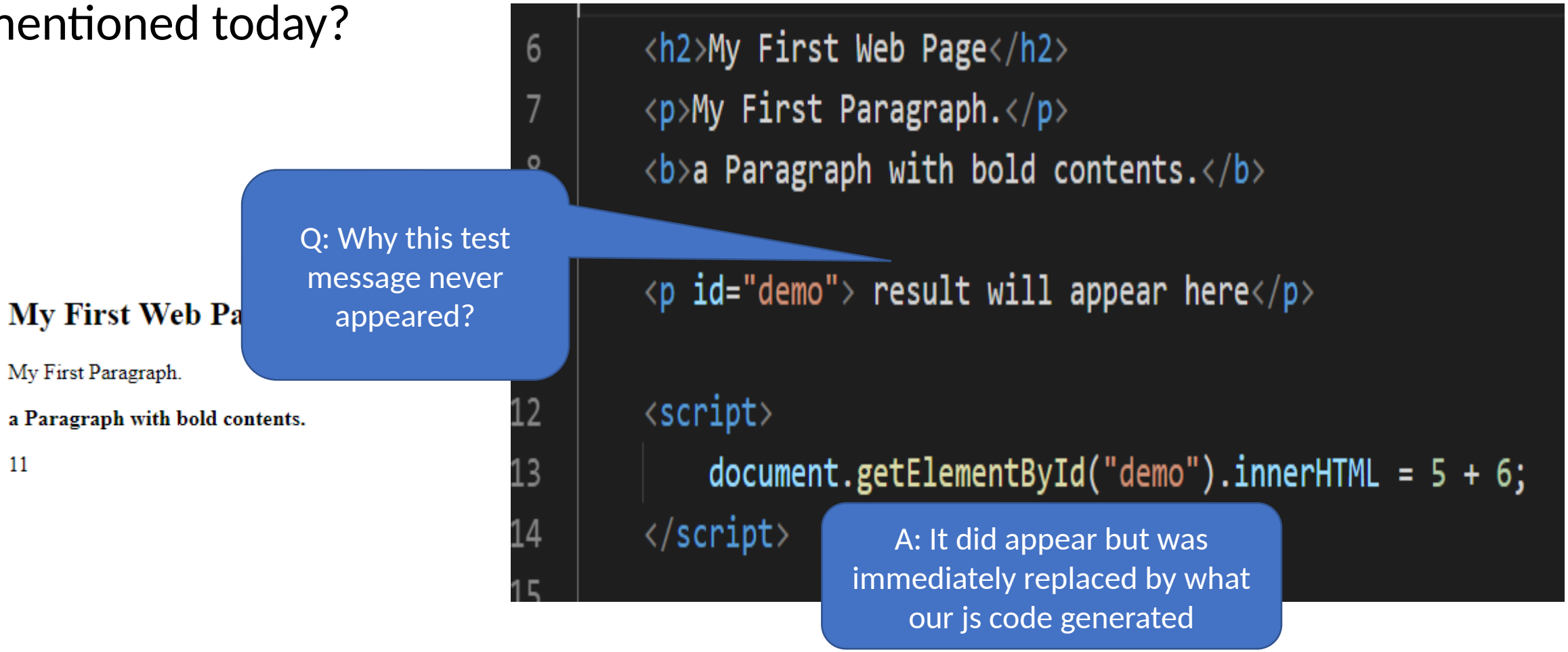


- Note: use of alert is not recommended.
- Q: why?
- A: due to bad user experience

Try alert("Hello ****world****!")

Example 4. Modifying HTML contents of an element

- This one is very interesting!
- Remember the attribute “id” we mentioned today?



Summary of examples so far

- Writing into the HTML output using **document.write()**.
- Writing into the browser console, using **console.log()**.
- Writing into an alert box, using **window.alert()**.
- Writing into an HTML element, using **innerHTML**.

In class activity 1

- Correct the displayed message (whatever day today is).
- You are only allowed to change the content of the <script> element

```
<body>
    <h2>Display correct statement</h2>
    <b id="blabla"> Topday is Monday!</b>

    <script>

    </script>

</body>
```

Example 5

- With JavaScript we can set the src attribute of an img element

```
<body>

    <b> This img tag(element) had nothing a millisecond ago!!!</b>
    <img id="wazzapp" />
     <script>
         document.getElementById("wazzapp").src = "./images/elf.png";
     </script>
</body>
```

In class activity 2

- Fill up the following empty img tags with 3 different images
- Note 1! You are only allowed to change the content of the <script> element
- Note 2! Store images in a separate folder named “imgs”

```
<body>
    <img id="img1" />
    <img id="img2" />
    <img id="img3" />
    <script>
        // something goes here to fill up images
    </script>
</body>
```

BTW it’s a comment! Means it is for us, developers.
Browser does not execute it

Variables in JavaScript

- With the keyword “let” we declare variables in Javascript.
- We can apply arithmetic operations like +,* … on variables

```
<body>
    <script>
        let x = 5;
        let y = 6;
        console.log(x + y)
    </script>
</body>
```

- Test it for yourself
  - We can also use “var” to declare variables in JavaScript
  - There are differences that we discuss later

```
var age = 32;
var myName = "John";
```

Example 6

- We can update variables in JavaScript

```
<script>
    let x = 5;
    let y = 6;
    y = y + 9;
    console.log(y)
</script>
```

Example 7 String variables

- We can store text in variables. They are called “string “ variable by comp sci community

```
<script>

    let name = "John McDonalds";
    console.log(name);
</script>
```

Example 8 String variables cont.

- We can do strange stuff with JavaScript strings. Try to see what happens if we do this :
- Console.log(“John is” + “attending school”);
-

Example 9 comments in JavaScript (vs comments in HTML, CSS

```
    <style>
        h1 {
            color: blue;
            /* This is a comment in CSS */
        }
    </style>
</head>
<body>
    <h1 id = "someId" > Welcome! </h1>
    <!-- this is a comment in HTML -->
    <script>
        let myVar = 900;
        // This line will not be executed.
        var age = 32;
        /* this is a multiple
        line comment */
        var myName = "John";
```

Decision Making: Equality and Relational Operators

- Decision based on the truth or falsity of a condition
  - Equality operators
  - Relational operators

Decision Making: Equality and Relational Operators

Standard algebraic equality operator or relational operator	JavaScript equality or relational operator	Sample JavaScript condition	Meaning of JavaScript condition
Equality operators			
=	==	x == y	x is equal to y
=	===	x === y	x is equal to y and both objects are the same type
≥	!=	x != y	x is not equal to y
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
	>=	x >= y	x is greater than or equal to y
	<=	x <= y	x is less than or equal to y

Equality and relational operators.

Example 10 decision making

```
Syntax:
if (condition) {
block of code to be executed if the condition is
true
}
```

```
5     <script>
6         let age = 23;
7         if (age > 18) {
8             document.write = "you are an adult!";
9         }
10    </script>
```

In class activity

Modify this script to show “you are an adult ” in green if your age is >18.

```
<p>Hello there!</p>
<p id="message"></p>
<script>
    let age = 3;
    if (age > 18) {
        //print "you are an adult" in green
    }
</script>
```

Tip: you need to do it in two lines
First line to change the innerHTML
Second line to change style of tag p

You are not allowed to touch anything outside script tag

Example 11 loop through variables

- In JavaScript we can create a loop. Each iteration picks the next value of i.
- i++ means, increment i by one on every iteration
- That means everything between {} will be first executed for i being 0
- Then goes back to for and executes everything between {} for i being 1
- …
- and lastly for i being 9

```
var i;
for (i = 0; i < 10; i++) {
    console.log(i);
}
```

review
Part 2

Outline

- Review
- JavaScript Object
- Handling events
- var vs let vs const
- Variable hoisting in var
- setTimeout and order of executions (due to being asynchronous event loop)
- CSS review part 2
- HTML Web Storage
- Json: JavaScript Object Notation
- Using same event handler for multiple buttons
- Object constructor with methods

Datatypes in JavaScript

Tip: There are six primitive data types and one object type in Javascript ECMA6

- Six primitive datatypes
  1. Boolean. `true` and `false`.
  2. null. denoting a null value.
     Because JavaScript is case-sensitive, `null` is not the same as `Null`, `NULL`.
  3. undefined. Basically means uninitialized.
  4. Number. 42 or `3.14159`.
  5. String. "Hello BCIT"
  6. Symbol (new in ECMAScript 2015).
     A data type whose instances are unique and immutable.
- Objects
  - `let x = { firstName: "John", lastName: "Doe" };`
  - `Note: Arrays are also of type objects in JavaScript.`

JavaScript === vs ==

- The == (or !=) operator performs an automatic type conversion if needed.
- The === (or !==) operator will not perform any type conversion. It first compares if both sides have the same type, then compares the values
- QB : which of the following statements is true ?

```
[10] === 10    // is false
[10]  == 10    // is true
'10' == 10     // is true
'10' === 10    // is false
 []   == 0     // is true
 [] ===  0     // is false
 '' == false   // is true but true == "a" is false
 '' ===   false // is false
```

Tip:
undefined,
null,
0,
false,
NaN,
'' (empty string)

are all falsy.

Means in the condition part of if statement
all of them mean false

Variable Declarations and variable Scopes

- **var**: Declares a **function-scoped** variable.
- **let**: Declares a **block-scoped**, variable.
- **const**: Declares a **block-scoped** const. Will make only **primitive** variables constant (cannot be changed later once declared);

```
const Pi = 3.14;

Pi = 90;    // Uncaught TypeError: Assignment to constant variable.
```

variable Pi was once initialized at the time of declaration using const and cannot be initialized/modified again.

Declaration using const …

- Q: which one issues an error message?

- Declaring a constant object and modifying it
- Declaring a constant primitive variable and then modifying it

Scope of Declarations by var, let, const:

```
{
  var a = 10;
}
console.log(a);
```

```
{
  let a = 10;
}
console.log(a);
```

```
{
  const a = 10;
}
console.log(a);
```

What does console.log(a) print on console ?

10	**Uncaught ReferenceError: a is not defined**	**Uncaught ReferenceError: a is not defined**
Function scoped	Block Scoped	Block Scoped

JavaScript Null vs Undefined

- JavaScript is a dynamic language
- If we don't initialize it by default its value is **undefined** (because its type is undefined)
- We use **null** in the situation when *we explicitly want to clear the value of the variable.*
- For example you wanna present the list of color to the user to select from, but the user has not selected any color yet,
- you can set that variable selectedColor = null. Or user has selected the color but we wanna clear the selection of color
- remember in MVC a variable might be bound to view so if you wanna clear that view null is a good idea

Arrays

```
var name = []; // empty array
var name = [value, value, ..., value]; // pre-filled
name[index] = value; // store element
```

```
14          var ducks = ["Huey", "Dewey", "Louie"];
15          var stooges = []; // stooges.length is 0
16          stooges[0] = "Larry"; // stooges.length is 1
17          stooges[1] = "Moe"; // stooges.length is 2
18          stooges[4] = "Curly"; // stooges.length is 5
19          stooges[4] = "Shemp"; // stooges.length is 5
```

- stooges.length() returns the size (length) of the array

- **Q:** Write a JavaScript program to display all elements of the array stooges on console

Arrays cont..

- We can add to the end of an array, or we can remove the last element of an array

```
14      var ducks = ["Huey", "Dewey", "Louie"];
15      var stooges = []; // stooges.length is 0
16      stooges[0] = "Larry"; // stooges.length is 1
17      stooges[1] = "Moe"; // stooges.length is 2
18      stooges[4] = "Curly"; // stooges.length is 5
19      stooges[4] = "Shemp"; // stooges.length is 5
20      // adding Terry to list of stooges
21      // observe it in google chrome development tool
22      stooges.push("Terry");
23      /* removing last element
24       observe it in google chrome development tool
25      */
26      stooges.pop();
```

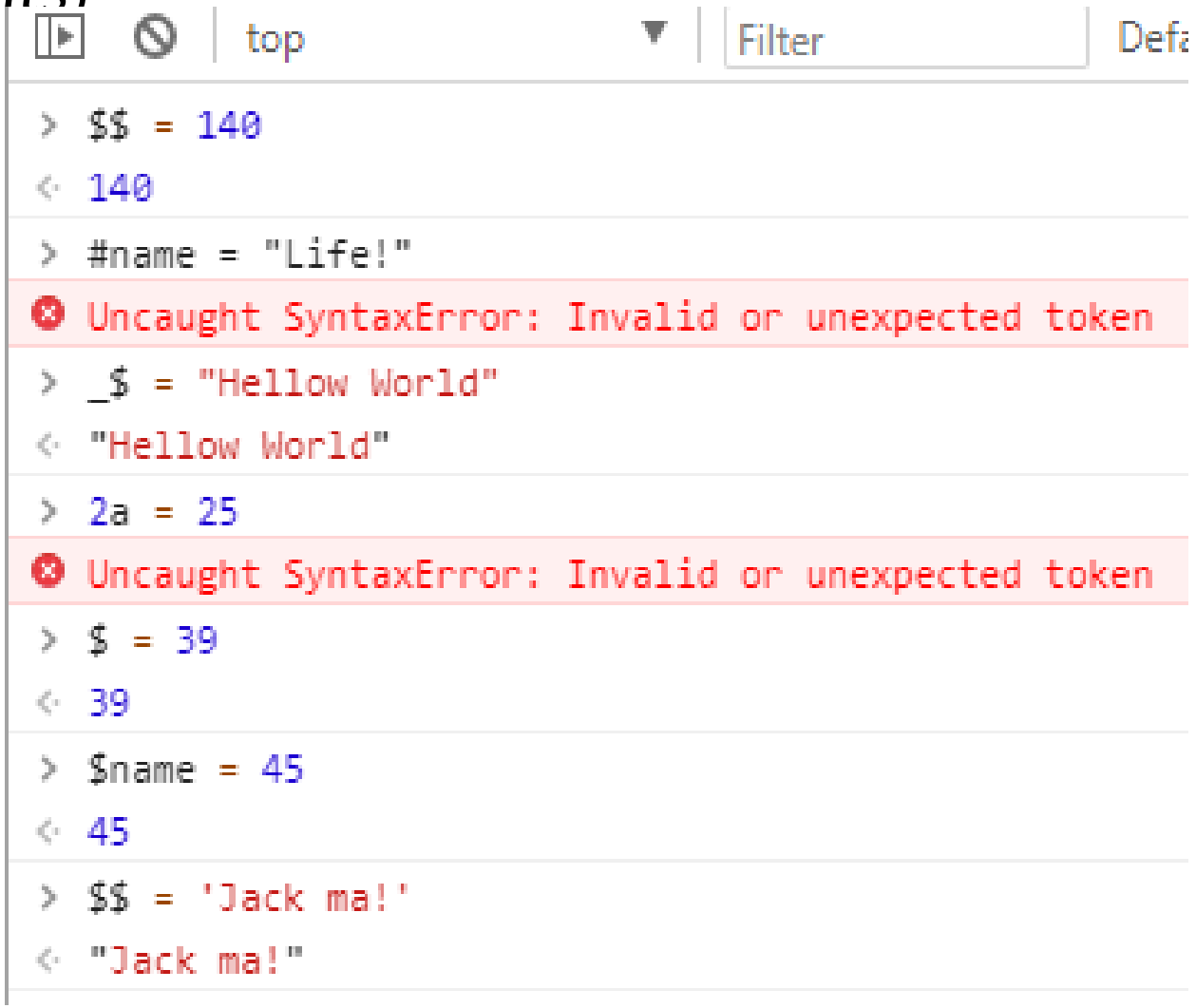
Correction:
The indices have to be
[0]
[1]
[2]
[3]

Accessing Array Elements

- Variable identifier rules
- A JavaScript variable (identifier) must start with a letter, underscore (_), or dollar sign ($); subsequent characters can also be digits (0-9) (*but a variable name cannot start with digits*)
- Q: which one of these can be a valid variable name?

1. $$ = 140;
2. #name = “life!”;
3. _$ = “Hellow World”;
4. 2a = 25;
5. $ = 39;
6. $name = 45;

Now you can tell what is Jquery.
Jquery is just an object named $



Practice:

- Considering the two functions below, what will be printed on console by each of the statements below?

```
function mul(a, b) {
    return (a * b);
}

function add(a, b) {
    return (a + b);
}
```

- `console.log(mul(2, add(2, 3)));`
- `console.log(add(2, mul(2, 3)));`
- `console.log(mul(2, mul(2, 3)));`
- `console.log(add(2, add(2, 3)));`
- `console.log(mul(2, 3), add(2, 3));`

remarks

- var number;
- console.log(number); // what will be logged on console?

- let num;
- console.log(num); // what will be logged on console?

var vs let vs const, more depth

- Execute the function below and see what happens

```
function example() {
    // I just created a block
    {
        var n1 = 10;
        let n2 = 20;
    }
    console.log(n1);
    console.log(n2);
}
```

- Why n2 is not accessible outside the block (outside curly brackets) ?

Variable Declarations and variable Scopes

var num = 10;

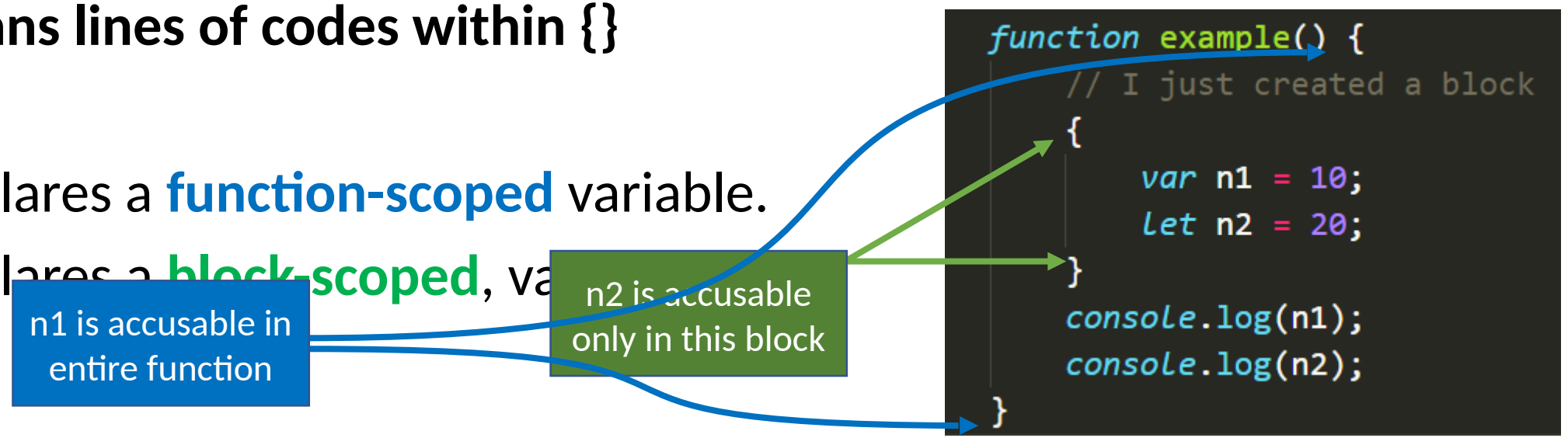
declares a variable which can be accusable within the function it was declared.

let num = 10;

declares a variable which is only accusable within the block it was declared.

a *block* means lines of codes within {}

- **var** Declares a **function-scoped** variable.
- **let** Declares a **block-scoped**, va



Variable Declarations and variable Scopes

- **var**: Declares a **function-scoped** variable.
- **let**: Declares a **block-scoped**, variable.
- **const**: Declares a **block-scoped** const. Will make only **primitive** variables constant (cannot be changed later once declared);

```
const Pi = 3.14;

Pi = 90;    // Uncaught TypeError: Assignment to constant variable.
```

variable Pi was once initialized at the time of declaration using const and cannot be initialized/modified again.

Why do you think it says "Type" error? What does it have to do with type?

What if we declare an object using const?

- Can we still modify properties that object after we initialized it?
- Can we reassign that object (that variable)
- Remember we could declare an object literal like this:

```
const  x = { firstName: "John", lastName: "Doe" };
```

- Q: which one issues an error message (and terminates the script) ?
  - 1- `const  x = { firstName: "John", lastName: "Doe" };`
  - `x = { firstName: “Tom", lastName: " Hanks" };`

  - 2- `const  x = { firstName: "John", lastName: "Doe" };`
  - `x.firstName = "Tom ";`
  - `x.lastName = " Hanks" ;`

Tips

- If a variable is declared outside function, it becomes global
- That means anything after variable declaration will have access to it. Example:

```
10      <script>
11          let test = 123;
12          function area(r) {
13              var result = 0;
14              if (r === 0) {
15                  result = 0;
16              } else {
17                  let pi = 3.14;
18                  result = pi * r * r;
19              }
20              // this line has no access to variable pi cuz it is outside the block pi was declared at line 17
21              return result;
22          }
23          // this line has no access to variable result which was defined in function area at line 13
24
25          var myArea = 0;
26          console.log(area(1));
27  // varibles test and myArea are accessable everywhere, Q: why?
28      </script>
```

variable hoisting

variable hoisting (only for variables declared by “var”)

- The JavaScript engine treats all variable declarations using “var” as if they are declared at the top of a functional scope (if declared inside a function, if not as if they are declared at top of the script)

```
<script>

    console.log(myVar);// will not issue error
    var myVar;
    myVar = 140;
</script>
```

JS engine moves declarations using **var** to top

Will not issue error, even though myVar was declared next line
It will simply print undefined

```
<script>

    console.log(myLet);// will issue error
    let myLet;
    myLet = 140;
</script>
```

Will issue error and terminate execution.
JS engine will NOT hoist variables declared by “let”

var attaches global variables to global window

```
var a = 10;
console.log (window.a); // prints 10 as a is attached to global window object
let b = 10;
console.log (window.b); // will not print 10, as b is not added as property of window
```

let does not allow re-declaration of variables in strict mode

```
'use strict';
var me = 'foo';
var me = 'bar'; // No problem, `me` is replaced.
```

```
'use strict';
let myLet = 'foo';
let myLet = 'bar'; // SyntaxError: Identifier 'me' has already been declared
```

let vs var summary

	var	let
Scope	Function scope	Block scope
Hoisting	Will hoist declaration to top of function	
Adding to global object (window)	Global variable declared by var add to window	let does not add property to global window
strict mode*	Re-Declaration is ok	No declaration

* Likely not enough time in this course to cover strict mode, but your are encouraged to develop in strict mode to prevent annoying silent errors

Handling events

like when user click on an element !

Events in JavaScript

- HTML events are "things" that happen to HTML elements.
- Examples:
- **An HTML web page has finished loading**
- **An HTML input field was changed**
- **An HTML button was clicked**
- you may want to do something when events happened.
  For example display of a message when a button is clicked!
- JavaScript lets you execute code or a function when events are detected.
- HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.
- Q: what were attributes? Give us some example ..

Source: https://www.w3schools.com

- HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

```
<button id = 'me' onclick="document.getElementById('me').innerHTML = 'Clicked me!' ">click here :-) </button>
```

Did u notice? Value of the attribute “onclick” could be JavaScript code

So there are two methods to handle events:
Method 1- inline (inside HTML element) ,
Metod 2- in JavaScript block

A better way to handle events it to use a function!

```
<body>

    <button id='me'> click here :-) </button>

    <script>

        function clickHandler() {
            document.getElementById('me').innerHTML = 'Clicked me!';
        }

        document.getElementById('me').onclick = clickHandler;
```

In class activity

- **Q1:** why const a;
- Will throw error messages
- (Uncaught SyntaxError: Missing initializer in const declaration)

- **Q2:**create a textarea using textarea tag and place a button underneath it. Clicking the button has to clear the content of the text area

JavaScript Objects

Review :JavaScript variable type Number, String, …,Objects

- You learned JavaScript variables are of 7 different types
- There are **six** primitive data types and **one** object type in Javascript ECMA6

1. Number. `let length = 16;`
2. Boolean. `let condition = true;`
3. String. `let lastName = "John";`
4. undefined. Basically means the **type of variable** is **undefined** to browser.
   - `let x`
   - `console.log(x); // x is undefined!`

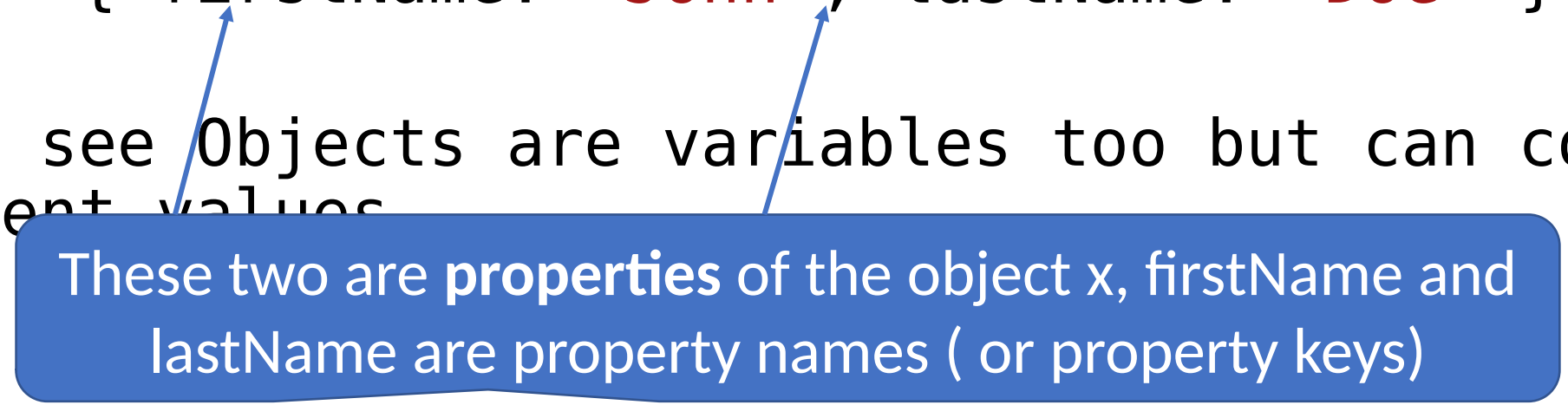
- Variable x has been defined, but its **type** has **not been defined.**
- JavaScript determines the type of a variable by looking at its current value, that is why JavaScript is called ***dynamically typed***

5. null. denoting a null value. *Because JavaScript is case-sensitive, `null` is not the same as `Null`, `NULL`.*
6. Symbol (new in ECMAScript 2015). We will not use in this course

7.Objects

JavaScript Objects

- Example:
- `let x = { firstName: "John", lastName: "Doe" }; // Object`
- As you see Objects are variables too but can contain different values



- Accessing properties of an object. Two ways:
- *objectName.propertyName or*
- *objectName["propertyName"]*
- `Console.log(x.firstName); // "John" will be printed`
- `Console.log(x["firstName"]); // "John" will be printed`

Object constructor

- Object contractor is the same at function declaration:

```
function Person(firstName, lastName){
    this.firstName =  firstName;
    this.lastName = lastName;
}
let person = new Person("Elon", "Musk");
```

Of course you could also declare it in form of object literals but obviously cannot be instantiated:

```
let person = { firstName: 'Elon', lastName: "musk" }
```

Using JavaScript built in objects

- **Example: Date contractor**

```
let date1 = new Date('January 15, 2018 09:00:00');
        let date2 = new Date(); // gets you current date

        let timeDiff = date2 - date1; //QQ what does it do?
        console.log(timeDiff);// milliseconds passed
```

Built-in (means existing) JavaScript objects

- Date is a built-in JavaScript object
- Here is how we create a new Date object:
- `let d = new Date();`
- Console.log(d.getTime());

Typeof

- typeof is a JavaScript built-in function that receives a single parameter input and returns the type of that input. Try these:

```
console.log(typeof (42));
// expected output: "number"

console.log(typeof ('blubber') );
// expected output: "string"

console.log(typeof (true));
// expected output: "boolean"

let x
console.log(typeof x);
// expected output: "undefined";
```

Typeof …

- Let myObj = { firstName: "John", lastName: "Doe" };
- onsole.log(typeof (myObj));
- // expected output: "object"
- Let myObj = { firstName: "John", lastName: "Doe" };
- console.log(typeof (myObj));
- // expected output: "object"
  - let person = {
  - firstName: "John",
  - lastName: "Brown",
  - age: 50,
  - eyeColor: "green"
  - };
- onsole.log(typeof (person));
- Q: onsole.log(typeof (person.age));
- Q: onsole.log(typeof (person.eyeColor));

Typeof ...

- **Q:** Same way declare an array and see its type by using typeof

Objects and functions inside Object

```
let MainObj = {

    prop1: "prop1MainObj",

    Obj1: {
        prop1: "prop1Obj1",
        prop2: "prop2Obj1",
        Obj2: {
            prop1: "hey you",
            prop2: "prop2Obj2"
        }
    },

    Obj3: {
        prop1: true,
        prop2: "prop2Obj3"
    }
};
```

- **Q**: How can we access the prop2Obj3 property?
- **Q**: plot the tree structure of this object

Object Contractor with a method

```
function Person(first, last, age, eyecolor) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
  this.name = function() {return this.firstName + "
" + this.lastName;};
}
```

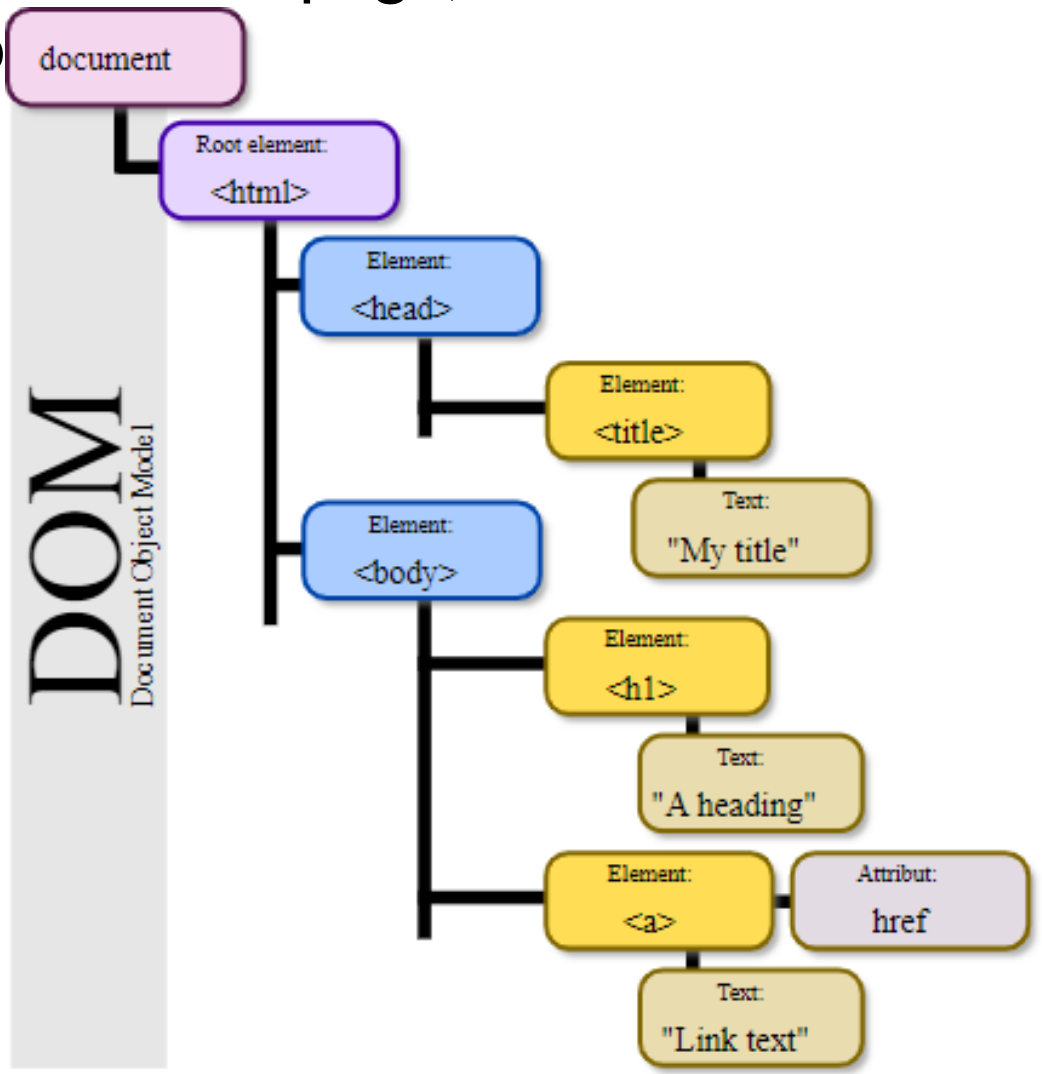
In class activity: Review

- Q: Do you remember what was the difference between undefined and not defined error?
- `let x`
- `console.log(x); // x is undefined will be printed!(means the variable is defined but its type is undefined) it happens when the variable has not ben initialized`
- `console.log(y); // y is not defined reference error!(means the variable y is not defined yet)`

The HTML **DOM** (Document Object Model)

The HTML DOM (Document Object Model) 1

- Review: When a browser renders a web page, it forms the **D**ocument **O**bject **M**odel o

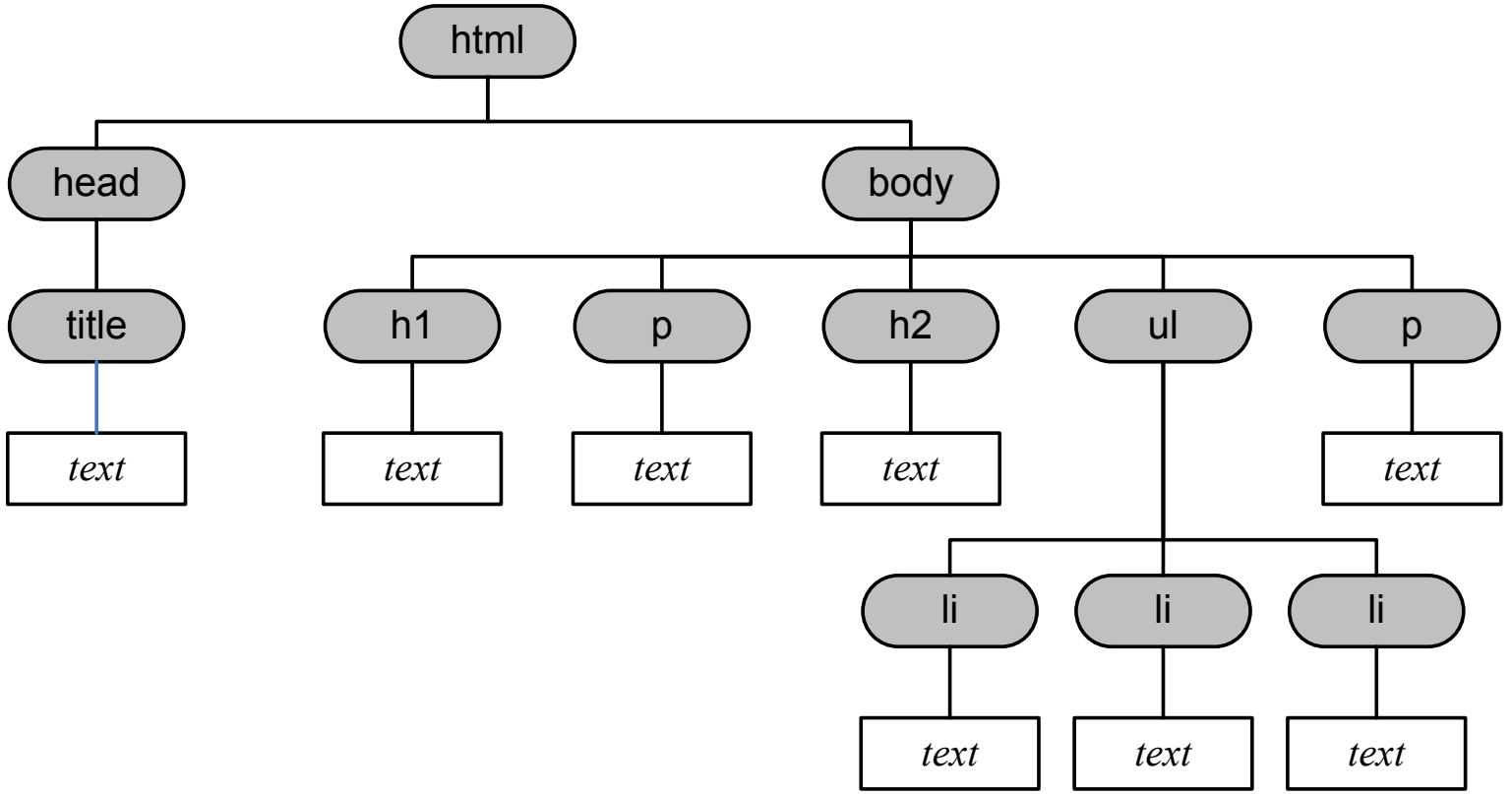


Source: https://www.w3schools.com/js/js_htm
wikipedia

The HTML DOM (Document Object Model) 2

- Here is an example: for this html page, the browser creates this object

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <title>page title</title>
5  </head>
6  <body>
7      <h1>hello page</h1>
8      <p>Welcome home.</p>
9      <h2>Our team </h2>
10     <ul>
11         <li>Jeff Brown</li>
12         <li>Jack Ma</li>
13         <li>Amy Alam</li>
14     </ul>
15     <p>Contact us</p>
16 </body>
17 </html>
```



Tree model of objects

```
let MainObj = {

    prop1: "prop1MainObj",

    Obj1: {
        prop1: "prop1Obj1",
        prop2: "prop2Obj1",
        Obj2: {
            prop1: "hey you",
            prop2: "prop2Obj2"
        }
    },

    Obj3: {
        prop1: true,
        prop2: "prop2Obj3"
    }
};
```

- **Q**: You have seen this object in previous slides. plot the tree model of this object

Dynamically changing DOM using JavaScript 1

- Imagine in an empty html page, we want to add a new *element* (e.g. a button) dynamically using JavaScript.
- You need to do two things
- 1: create that element in memory
- `let btn = document.createElement("BUTTON");`
- `2:place that element in DOM tree`
- `document.body.appendChild(btn);`
- `3: do something to make that button visible ! (e.g. give it a value)`

Dynamically changing DOM using JavaScript 2

- Add two buttons dynamically to your DOM model
- Clicking on first button shall display a message using alert, e.g. button 1 was clicked
- Clicking on second button to print something on console (button 2 was clicked)

getElementsByTagName

- getElementsByTagName returns an **array** of elements of the given type

```
<body>
<p>Hello ..... World!</p>
<p>Hello World!</p>
<p>Hello ..... World!</p>
<script>
let pElements = document.getElementsByTagName("p");
pElements[1].innerHTML = "just Hello!";
</script>
</body>
```

Using input box to get input from user

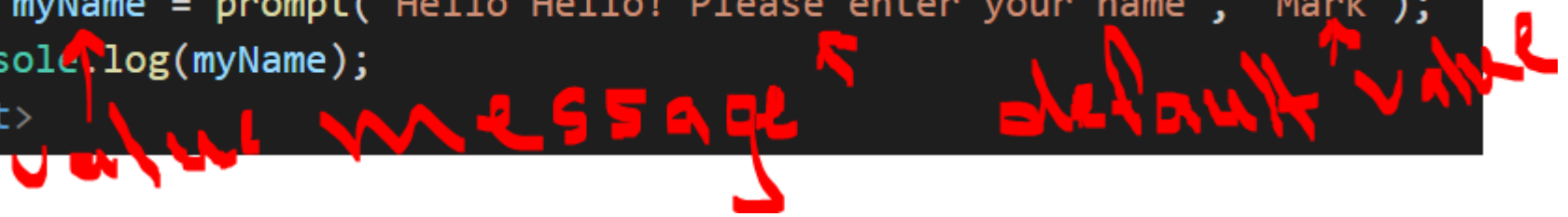
- What does this code snippet do?

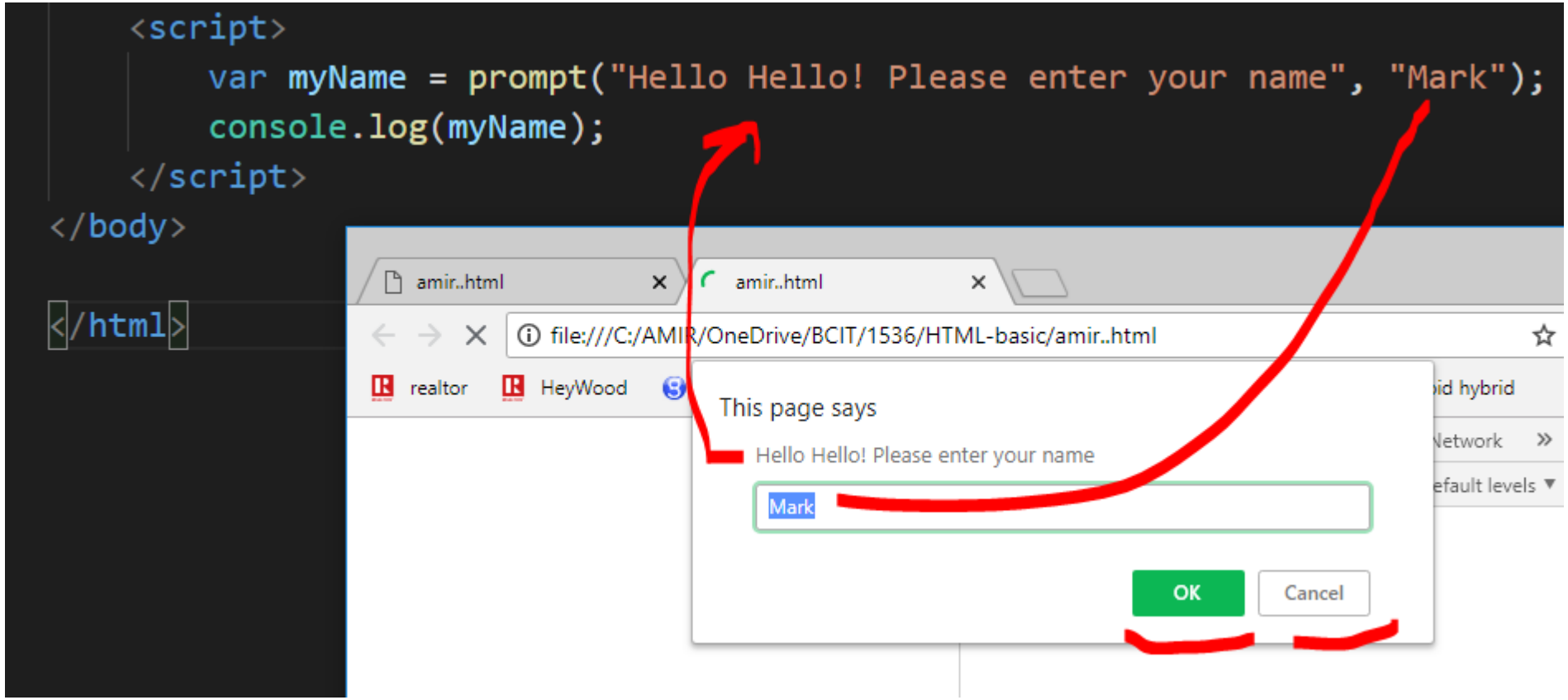
```
What is 2+3? <input type="text" id="myText" value="">
<p>Click the button to change the value of the text field.</p>
<input type="button" onclick="myFunction()" value="Try it"></input>
<script>
    function myFunction() {
        alert("You said 2+3 is:"
        + parseInt(document.getElementById("myText").value));
    }
</script>
```

Getting input from user

- **Q:** Write a JavaScript program to request user's name and then display that name onto console.
- Tip: you can use prompt:
- var myName = prompt(*textMessage*, *defaultText*)

```
 9      <script>
10          var myName = prompt("Hello Hello! Please enter your name", "Mark");
11          console.log(myName);
12      </script>
```





The prompt() method returns the input value if the user clicks "OK". If the user clicks "cancel" the method returns null.

setTimeout vs setInteral

- `setTimeout (function, t) // ( remember Tim.. is uppercase)`
- setTimeout executes the function after the time t (in millisecond) is elapsed
- Example:
- `setTimeout(function(){ alert("Hello"); }, 3000);`

- `setInterval (function, t)`
- Does the same thing at setTimeout but repeats it
- `setInterval(function(){ alert("Hello"); }, 3000);`

Order of execution … Event Queue

- What would be the output of this script?

```
<script>
    console.log(1);
    setTimeout(function () {
        console.log(2);
    }, 1000);
    console.log(3);
</script>
```

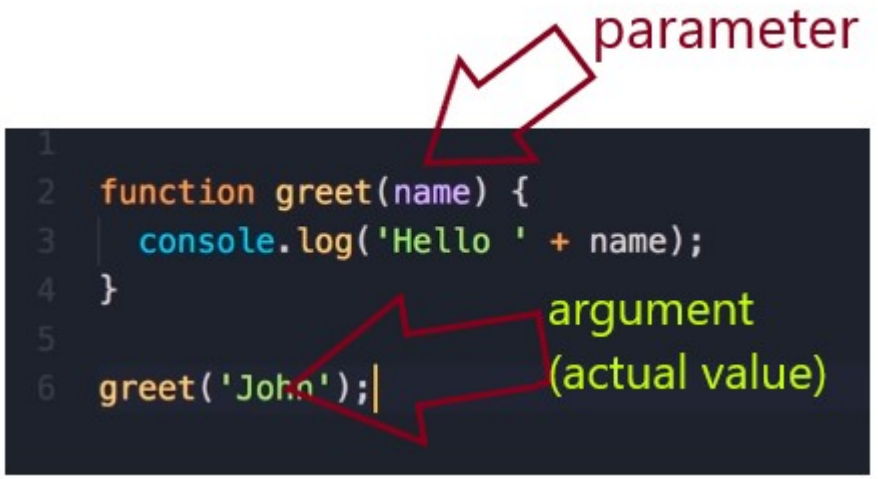
1
3
2

- What if we replace 1000 with 0 ?

1
3
2

Study: https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop

Argument vs Parameter



Coding style tips for your assignments (part one)

- Chose **let and const**, **over var**. In fact you do not need var, so your code has to be “var” free!
- Avoid use of global variables when you don’t have to
- Do not use hard coded strings in your code. Move them all in a separate file and define const string variables and use those variables in your code
- Separation of concerns: each function is responsible to deliver one single simple task, not more. (more in next page)

Separation of concerns: Each function has to do one thing only

The function calcSum can be Brocken into two smaller functions ➔

```
//this function does two things:
// 1- calculates the sum
// 2- displays the sum
function calcSum(k) {
    sum = 0;
    for (let i = 1; i <= k; i++) {
        sum = sum + i;
    }
    console.log(sum);
}

calcSum(k); // invocation
```

Mode modular, more rideability

```
//this function does one thing:
// calculates the sum
function calcSum(k) {
    sum = 0;
    for (let i = 1; i <= k; i++) {
        sum = sum + i;
    }
    return (sum);
}
// this function does one thing
// display of the parameter d on concole
function display(d) {
    console.log("result is: " + d);
}
// invocations
let result = calcSum(k); // invocation
display(result);
```

Things to remember

- Declaring a variable without use of var or let or const will make it global, regardless where it is defined.
- JavaScript is a dynamic language; meaning the type (string, boolean …) of a function return value is not pre-defined. If a function does not use a return statement or an empty return statement with no value, JavaScript automatically returns undefined.
- That means that ***in JavaScript every function returns something***, at least undefined. (all of these code snippets will display “undefined” at the end)

```
function foo (){
  k = 100;
}

console.log(foo());
```

```
function foo (){
  k = 100;
  return ;
}

console.log(foo());
```

```
console.log(console.log('Hi!'));
```

ECMA6 (ESC6 javascript) tips

ESC6 tips of the week/ Arrow functions to replace old ones

- regular functions

```
setTimeout(function() {

}, timeout);
```

- arrow functions

```
setTimeout(() => {

}, timeout);
```

```
date = function() {
  return new Date();
}
```

```
date = () => {
  return new Date();
}
```

```
// one parameter
hello = (val) => "Hello " + val;

hello = val => "Hello " + val;

// two or more parameters
sum = (a,b) => a+b;
```

CSS review part 2

(remember) There are 3 ways of styling HTML elements

- 1- Styling HTML elements using the style attribute

  This method is called **Inline** styling which we just learned today.

- 2- Using <style> tags in the <head> section of same HTML file

  This method is called **Internal** styling

- 3- Using an external CSS file

  This method is called **External** styling

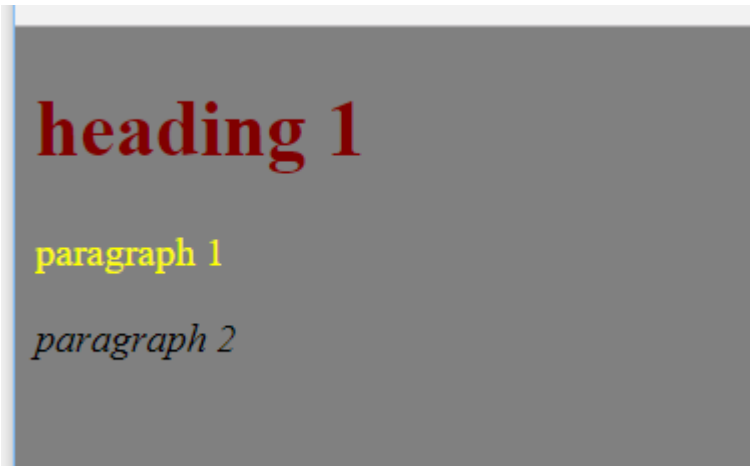
- **CSS** stands for **C**ascading **S**tyle **S**heets.
- CSS describes **where** every element is placed and **how** it looks on screen, paper, or in other media (remember this from last week).

inline vs internal

```
<head>
</head>

<body style="background-color: gray;">
    <h1 style="color: maroon;">heading 1</h1>
    <p style="color: yellow;">paragraph 1</p>
    <p style="font-style: italic;">paragraph 2</p>
</body>
```

```
<head>
    <style>
        body {
            background-color: gray;
        }
        h1 {
            color: maroon;
        }
        .pClass {
            color: yellow;
        }
        #pId {
            font-style: italic;
        }
    </style>
</head>
<body>
    <h1>heading 1</h1>
    <p class="pClass">paragraph 1</p>
    <p id="pId">paragraph 2</p>
</body>
```



internal
vs
External
CSS

External Style sheet

- An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

```
<html>
<head>
    <link rel="stylesheet"
          type="text/css"
          href="myStyle.css" />
</head>
<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```

myStyle.cc file contains the CSS rules

Internal vs External

```
<head>
  <style>
    body {
        background-color: gray;
    }
    h1 {
        color: maroon;
    }
    .pClass {
        color: yellow;
    }
    #pId {
        font-style: italic;
    }
  </style>
</head>
<body>
    <h1>heading 1</h1>
    <p class="pClass">paragraph 1</p>
    <p id="pId">paragraph 2</p>
</body>
```

myStyle.css

```
body {
  background-color: gray;
}
h1 {
  color: maroon;
}
.pClass {
  color: yellow;
}
#pId {
  font-style: italic;
}
```

my.html

```
<head>
  <link rel="stylesheet"
        type="text/css"
        href="myStyle.css" />
</head>
<body>
  <h1>heading 1</h1>
  <p class="pClass">paragraph 1</p>
  <p id="pId">paragraph 2</p>
</body>
```

- You have noticed that with internal and external methods you used CSS language to apply styling on multiple HTML elements at the same type. This is more efficient.

- You have noticed that the only difference between internal and external is that in external we need a separate file for CSS contents

- For internal and external styling we need to know the CSS language. You have noticed so far that the terms are exactly the same as used in inline styling.

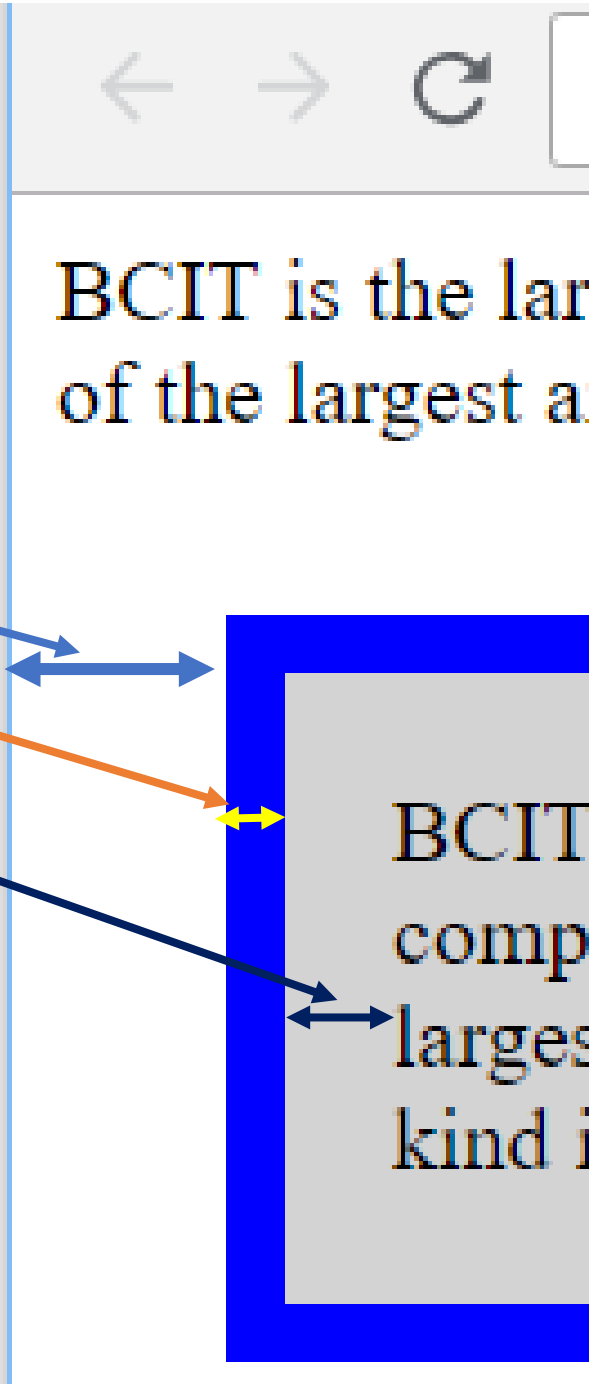
CSS layout
padding …

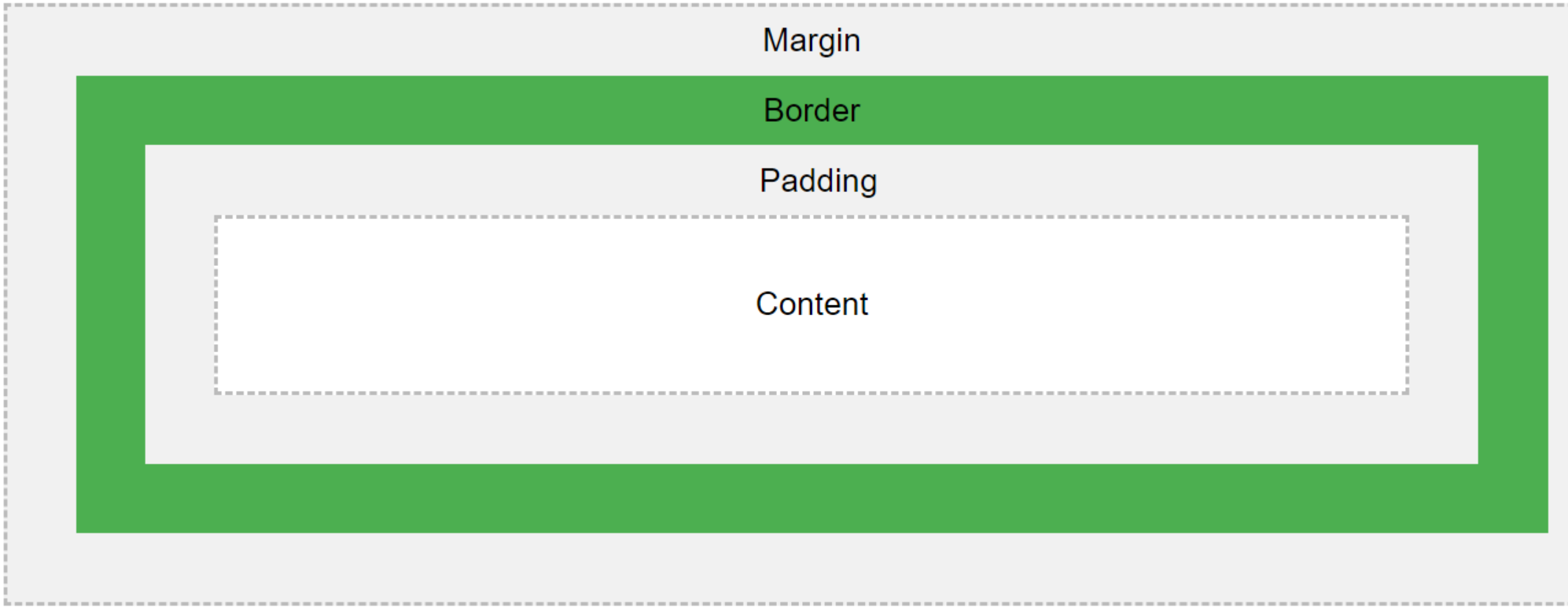
The CSS Box Model

- All HTML elements can be considered as boxes.
- In CSS, the term "box model" is used when talking about design and layout.
- A box that wraps around HTML element
- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

```
  <style>
    #div2 {
      background-color: lightgrey;
      width: 300px;
      margin: 30px;
      border: 10px solid blue;
      padding: 20px;
    }
  </style>
</head>
<body>
  <div>
    BCIT is the largest provincial provider
    of computing graduates. It has
    some of the largest and most extensive
    programs of their kind in BC and  beyond.
  </div>

  <div id="div2">
    BCIT is the largest provincial provider
    of computing graduates. It has
    some of the largest and most extensive
    programs of their kind in BC and  beyond.
  </div>
</body>
```





Source https://www.w3schools.com/css/css_boxmodel.asp

CSS layout , display property

- Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block, inline, none
- **Block**
- A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The element is a block-level element.

- Default display value of the following elements is block
- <h1> - <h6> <p> <form> <header> <footer> <section>
- **Inline**
- An inline element does not start on a new line and only takes up as much width as necessary. This is an inline element inside a paragraph.
- Display property of span> <a> <img> is inline by default

CSS layout , display property cont.

- **none**
- “display: none” is used to remove elements from screen without deleting them.

CSS layout visibility property

- Specifies whether or not an element should be visible
- visibility:hidden or visibility:show

- Note:
- We can also hide an element using visibility property:
- visibility:hidden

- display: none vs visibility:hidden
- visibility:hidden; also hides an element. However, the element will still take up the same space as before.
- In both cases the element will **not** be removed from DOM tree
- **Q:**Try https://www.w3schools.com/css/tryit.asp?filename=trycss_display

CSS layout properties cont.

- The following CSS layout properties will be tried during the lab as part of your lab assignment
- width and max-width
- position
- overflow
- float and clear
- Horizontal & Vertical Align
- **Note**: due to use of external css styling we can change the feel and look of entire website by just changing one line of code. Can you guess how?
- Check out this link https://www.w3schools.com/css/css_intro.asp

Position property of Style attribute

- The position property of Style attribute specifies the type of positioning method used for an element (static, relative, absolute, fixed, or sticky).
- static: Default value. Elements render in order, as they appear in the document flow
- **fixed**: The element is positioned relative to the browser window
- `style="left:20px ; position: absolute;"`
- There are other possible values for the property position such as relative, sticky, initial, inherit.
- Refer to https://www.w3schools.com/cssref/pr_class_position.asp
- and https://www.w3schools.com/cssref/playit.asp?filename=playcss_position&preval=fixed
  for more

CSS Combinators

- Descendant Selector
- Example: selects all <p> elements inside elements:

```
div p {
    background-color: yellow;
}
```

- There are more Combinators such as child selector (>), adjacent sibling selector (+), general sibling selector (~)

CSS Specificity

- If there are two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific and therefore wins out.
- Example: all the three sets or rules apply to the div, but which one wins?

```
<head>
    <style>
        #myId {background-color: blueviolet;}
        div#myId {background-color: pink;}
        div[id=myId] {background-color: greenyellow;}
    </style>
</head>
<body>
    <div id="myId">This is a div</div>
</body>
```

CSS Units (Absolute)

- Absolute Lengths
- Absolute length units are not recommended for use on screen, because screen sizes vary so much. They are mostly used for print layouts
- cm(centimeter) used for printer devices
- px (pixels) Pixels are fixed-size units that are used in screen media (i.e. to be read on the computer screen).* For low-dpi devices such as monitors, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.
- pt (points)
- Points are traditionally used in print media (anything that is to be printed on paper, etc.). One point is equal to 1/72 of an inch. Points are much like pixels, in that they are fixed-size units and cannot scale in size*
- Good Source to visit (https://kyleschaeffer.com/development/css-font-size-em-vs-px-vs-pt-vs/)

- Relative Lengths
- em Relative to the font-size of the element (2em means 2 times the size of the current font)
- em" is the width of the capital letter M for a given font, which is also the size of the font in points. (https://www.quora.com/What-does-the-em-stands-for-with-respect-to-CSS)
- % relative to parent element

Hosting

Tips on how to choose the right web hosting provider

HTML, JavaScript, CSS
nodejs files
MySQL databases
SSL

Hosting

- In order to make your website available on the net to visit by anyone around the world, you need to (remotely) host your website.
- That means there is a computer, referred as server (web server, DB server, data server ..), on which you store your HTML, CSS, JS, image, video files, databases, etc
- That computer is kept switched on, 24/7
- That computer is connected to the internet
- That computer has a unique address
- So when you type. e.g. https://myDomain.com, the server side script runs in that computer and generates (renders) HTML files to return to your browser.
- The HTML files from that web server are fetched and made available to your browser. They actually store in your browser's storage (yes the HTML files, images and CSS files are copied to your local machine to be used by your browser. The video files or other streaming content don’t entirely store in your browser's local storge
- There are various types of remote servers, webservers, database servers, file servers etc

Hosting provider

- There are companies that maintain such computers, or datacenters. Those companies are called Hosting providers.
- Every **website** you've ever visited **is hosted** on **a server** (or on a bunch of servers such as cloud services).

Choosing the right hosting plan!

- For your term projects/ assignments you need a web hosting account that provides
- 1) **relational** database server
  - For your assignment, you may need to use **relational** databases such as mySQL, Postgres, Marina DB, MariaDB
- 2) Hosting multiple nodejs projects at the same time
  - You will need to use NodeJS runtime environment to create the backend of your project
  - No all web hosting plans allow you to host nodejs applications
- 3) SSL certificate
  - So that you can host http**s**:// instead of only http://
  - Many third party APIs require you to use https when you access them
  - Your web hosting provider can install SSL certificate for you or you may have to do it yourself depending on the service you obtain for hosting your assignments. Making sure the SSL certificate is installed is part of your assignments.
  - From former students we heard that some free web hosting services also offer SSL certificate as well.

- Hosting plans
- There are various types of hosting plans,
- Shared vs VPS vs Dedicated vs Cloud Hosting. ***It is your responsibility to find the right hosting service provider***

Choosing the right hosting plan! Part 1

- There are various types of hosting plans:
- **Shared**
- When you choose a shared hosting, the hosting company will put your website on a server along with hundreds of other websites (or may be thousand other websites) . All of those websites are sharing resources such as memory, disk space, CPU, Database server, mail serve etc
- Shared plans are the cheapest but, depending on number of websites hosted in same machine, might be the slowest. For your projects in this course a right shared plan would suffice
- **Shared VPS (Virtual prove server)**
- is similar to shared hosting in that your website shares a server with other sites, you get your own virtual private server, but there are fewer sites sharing space and resources and the server is likely fasted.

Choosing the right hosting plan! Part 2

- **Dedicated server.**
- Your website is stored on a single physical machine dedicated to your website. All the resources, CPU, RAM, Hard disk are being controlled by you.
- **Cloud Hosting**
- It is based on the concept of cloud computing technologies that allow multiple machines to act as one system. This allows multiple servers to work together to handle high traffic levels or spikes for any particular
- Shared hosting services are generally ran by two types of operating systems, Linux and Windows.
- Generally Linux servers are cheaper. However you need to make sure they offer what you need for your assigments.

Choosing the right hosting provider and a disclaimer

- There are many companies that provide web hosting services
- you can also host your web applications on **free services**, **however you need to make sure their service is up and running during the entire semester, and meet the assignment requirements such as offering SSL when needed, mySQl hosting, and hosting multiple nodejs apps at the same time.**
- You can choose, free or paid cloud, dedicated, VPS or shared hosting services, its up to you but you need to remember its your responsibility to manage your server as every company offers different services. If you have any issues with your web hosting provider you need to contact their customer service.
- Managing shared hosting servers is easier as they are already set up and ready for you to use via some panels such as cPanel.
- Note: It is your responsibility to chose the hosting provider to meet the assignment requirements. Even if you see in my videos or class that I am using a web hosting service from a particular company for my own projects, it does not mean that would be the best option and you don’t need to do your due diligence and research.
- Finding the right hosting web service provider to host multiple app and all of your assignment at the same time is what you need to practice and learn as part of your assignment

In case you wanted to obtain a domain name

- If the web hosting you are using is providing you with a unique folder address such as
- https://myAdvWebDev.herokuapp.com
- https://juetwkl43532f2a2.herokuapp.com
- You would probably be fine, and you would not need a domain name to point to your home page for this course, however if you want to look more professional to the recruiters you can get your own domain name and link it to your webhosting account
- You can get your domain name from any domain name registrars such as.
- GoDaddy, Register.com, networksolutions.com, … and link them to your web hosting provider (they have special steps that their customer service has to provide to you)

- For assignments which need hosting, you will need to follow some conventions and best practices as to where to store files for that particular assignment.
- E.g.
- https://myAdvWebDev.herokuapp.com/comp4537/labs/1/...
- or
- https://juetwkl43532f2a2.herokuapp.com/comp4537/labs/1/...

- Depending on the host provider you obtain your service from, you will need to figure out how to host your assignments in various folders.

- I only show you how it is done in a cPanel environment as an example

Tip 1

- Is it possible to get the web hosting service from a company and the domain name from another company ?
- Again, you don’t need a domain name but Yes its possible.
- For example you can get the web hosting account from glowhost.com and the buy the domain name from goDaddy.ca

- Note:
- It is **your responsibility** to make sure your assignments and projects are properly hosted. If you have any issues you need to contact the costumer service of the web hosting or your domain registrar

Tip 2 index.html is special!

- By default all web browsers will automatically 'look' for an index.html page located in the url directory
- That means, if the html server does not find the html file in the url, it returns the index.html page (if exists)
- Example
- Suppose the in the root of your domain there is a folder named project2 with the following files:

MyDomain.com/project2/

/index.html

/course.html

- Opening the url http://myDomain.com/project2 (note that HTML file name is not provided) will open index.html by default as no file names is included

References

Note: There are more tips for you posted on the course learning hub

- https://medium.com/@agavitalis/how-to-deploy-a-simple-static-html-website-on-heroku-492697238e48