# JavaScript– W1S1
# HTML, CSS, and JS first steps

Cyrille Jegourel – Singapore University of Technology and Design

# Outline (Day 1, Session 1)

- **HTML, CSS and JavaScript:** interoperability, HTML tags
- **JavaScript:** First steps
- **Commenting:** in-line, and multiline blocks
- **Variables:** Declaration, assignments, naming, types, strings

# Basic HTML and CSS knowledge

- **No need to be an expert in HTML and CSS to use JavaScript**

However,

- Need to know:
  - what these languages are used for.
  - what a HTML tag is.
  - where to place some elements in a webpage.
  - how to add your own attributes.

# HTML, CSS, JavaScript: overview

- HTML provides the basic structure of sites, which is enhanced and modified by other technologies like CSS and JavaScript.
- CSS is used to control the style of webpages, the presentation, the formatting, and the layout.
- JavaScript is used to add dynamism on a website and to control the behaviour of different elements.

- The wall

- The painting

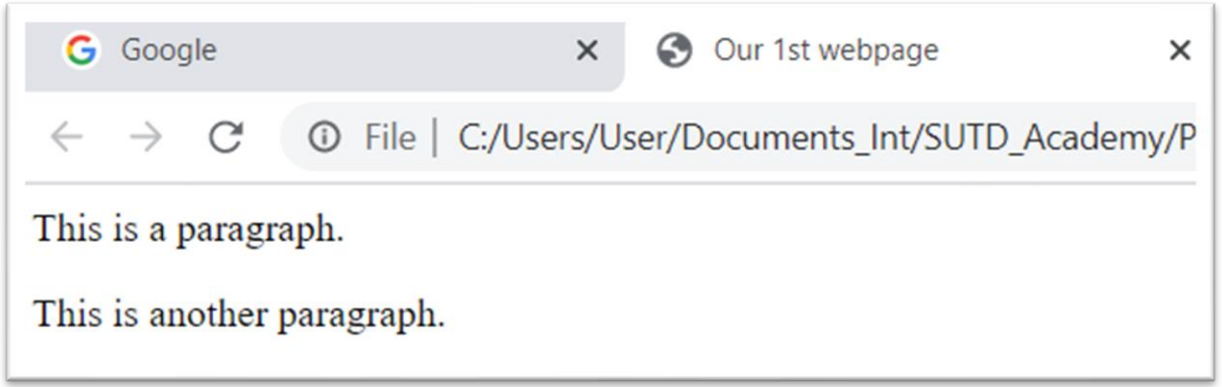
- The aircon

# HTML tags

- **HTML: HyperText Markup Language**.
- Rather than using a programming language to perform functions, use tags to identify different types of content and the purposes they each serve to the webpage.
- tag: keywords which define how web browser format and display some content.
- E.g., to write a paragraph, you need to open a paragraph tag ** ,** then write your content, and finally close your tag ** .**

example.html

```
1  <html>
2    <head>
3      <meta charset="utf-8">
4      <title>Our 1st webpage</title>
5    </head>
6    <body>
7      <p>This is a paragraph.</p>
8      <p>This is another paragraph. </p>
9    </body>
10 </html>
```

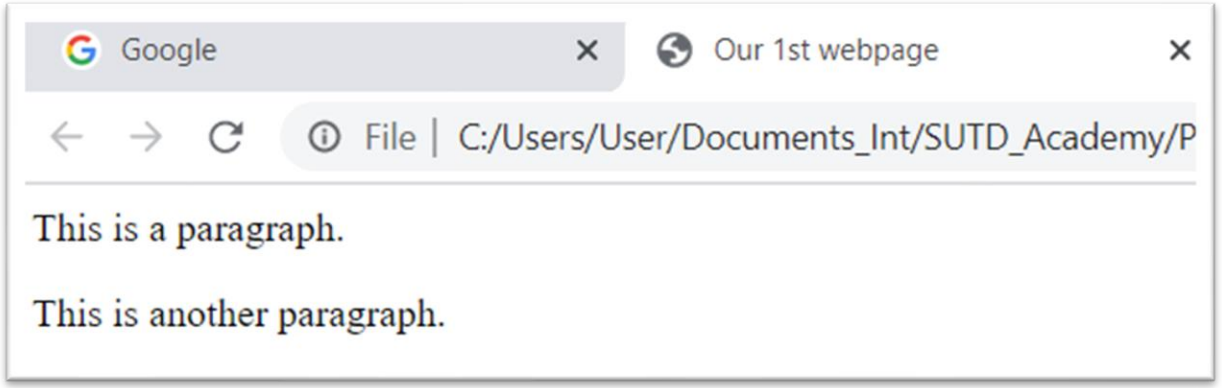


# HTML tags

- In general, tags must be closed:
  - Syntax: <tag> content </tag>
- HTML documents always start and end by the **html** tag and are composed of:
- A header (**head** tag) encapsulating information about the webpage (**title**, references to other files, etc.)
- A **body** encapsulating the elements of the webpage.
- To access the source code of a webpage: Click right + “View page source”.
- To create/edit your webpages: create/open a html document with a text editor.

example.html

```
1  <html>
2    <head>
3      <meta charset="utf-8">
4      <title>Our 1st webpage</title>
5    </head>
6    <body>
7      <p>This is a paragraph.</p>
8      <p>This is another paragraph. </p>
9    </body>
10 </html>
```

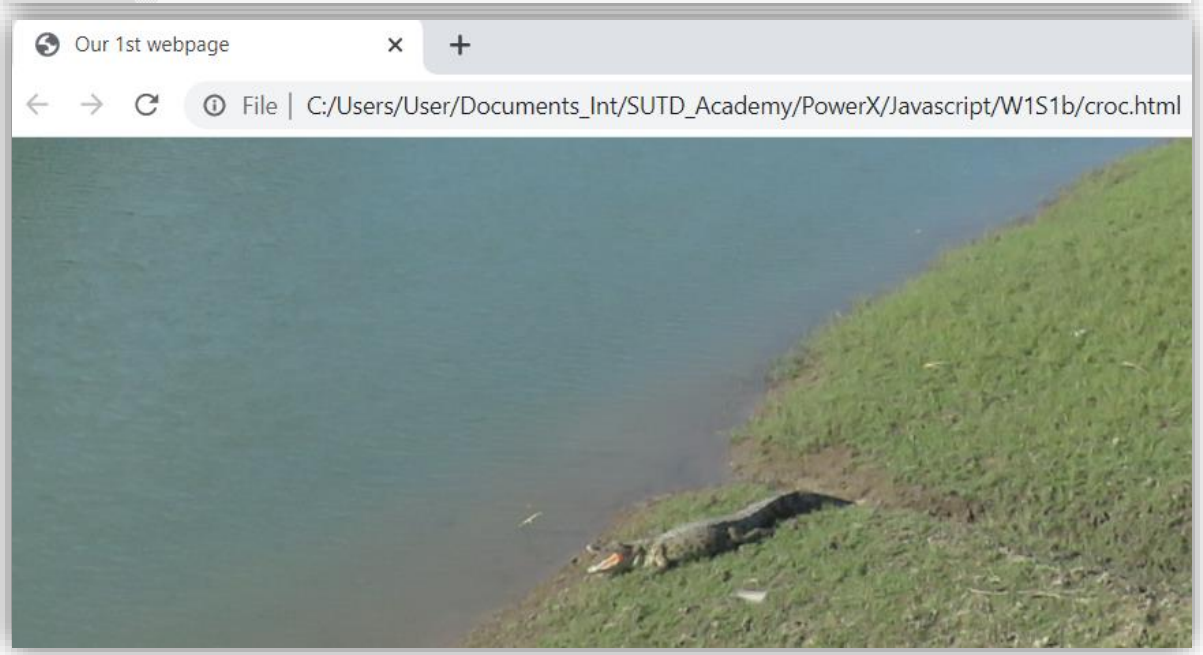


# HTML tag attributes

- Attributes take the form of an opening tag and additional info is placed inside.
- An example of an attribute is:
- <img src="mycrocodile.jpg" alt="A photo of my crocodile.">
- In this instance, the image source (src) and the alt text (alt) are attributes of the <img> tag.

croc.html

```
1  <html>
2    <head>
3      <meta charset="utf-8">
4      <title>Our 1st webpage</title>
5    </head>
6    <body>
7      <img src="mycrocodile.jpg"
8      alt="A photo of my crocodile.">
9    </body>
10 </html>
```



# HTML rules not to forget

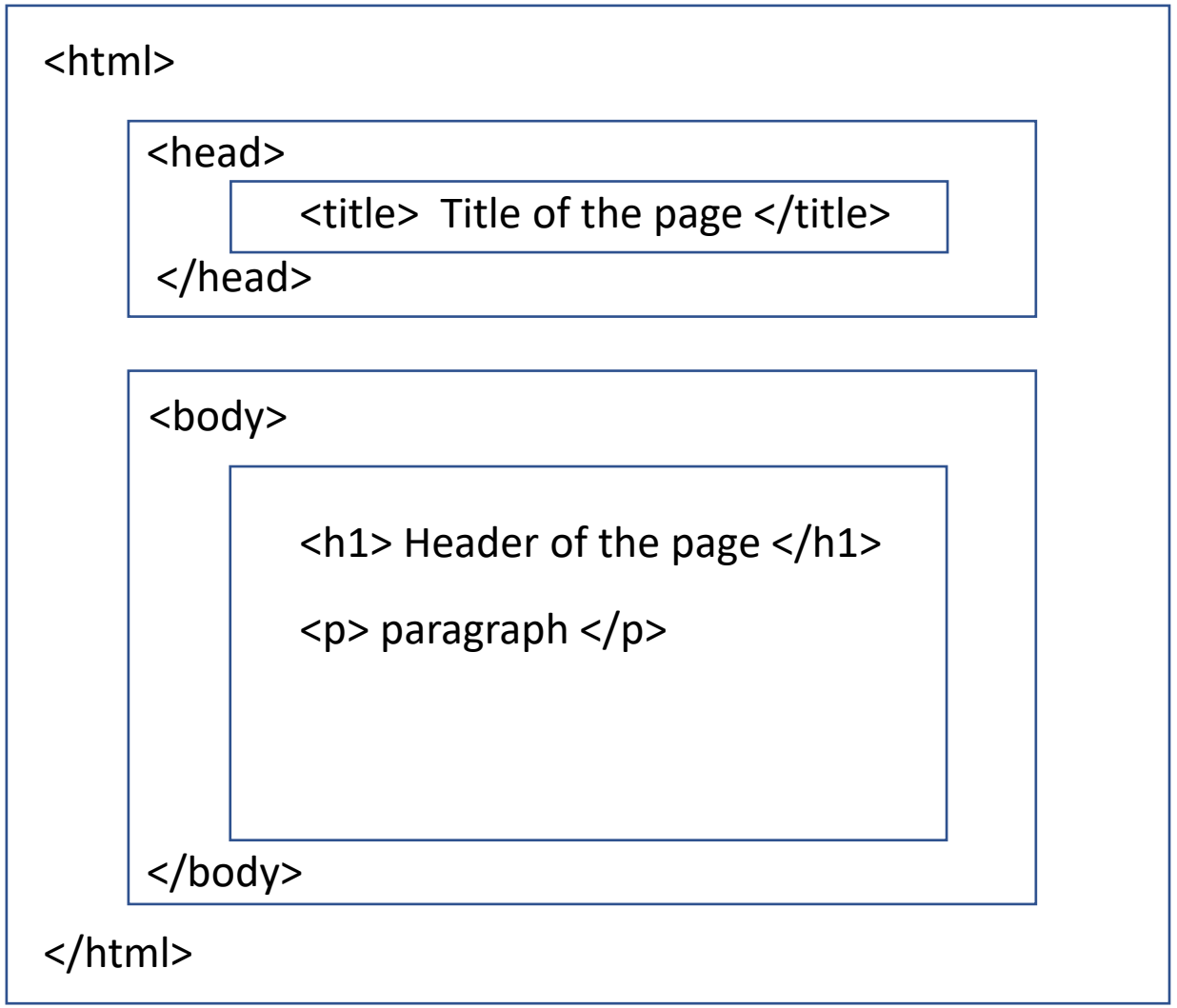
- Most of the tags must be opened (<tag>) and closed (</tag>) with the element information such as a text placed between the tags.
- When nesting multiple tags, the tags must be closed in the order in which they were opened.

example.html

```
1  <html>
2    <head>
3      <meta charset="utf-8">
4      <title>Our 1st webpage</title>
5    </head>
6    <body>
7      <p>This is a paragraph.</p>
8      <p>This is another paragraph. </p>
9    </body>
10 </html>
```

# Basic Construction of an HTML page

- The following tags should be placed underneath each other at the top of every HTML page that you create.
- **<!DOCTYPE html>** — This tag specifies the language you will write on the page.
- **<html>** — This tag signals that from here on we are going to write in HTML code.
- **<head>** — This is where all the metadata for the page goes — stuff mostly meant for search engines and other computer programs.
- **<body>** — This is where the content of the page goes.



# A basic <head> section

Inside the **<head>** tag:

- **<title>** (always included): where we insert the page name as it will appear at the top of the browser window or tab.
- **<meta>** : where information about the document is stored: character encoding, name (page context), description.
- Most standard attribute: **charset="UTF-8"**
- Many others: name followed by content:
  - <meta name="author" content="Cyrille">
  - <meta name="keywords" content="chess, wine">
- More here: https://www.w3schools.com/tags/tag_meta.asp

```
<head>
    <title>My chess and wine blog</title>
    <meta charset="UTF-8">
    <meta name="description" content="This
    field contains information about the
    page. It is usually around two
    sentences long.">.
    <meta name="author" content="Cyrille">
    <meta name="keywords" content="chess,
    wine">
</head>
```

# Adding content

Inside the <body> tag, you can include headings, text, links, lists, images, tables, forms, multimedia:

- **To add headings** from the most important to the less important: **<h1>, <h2>, …, <h6>**.
- **To add text**, use **<p>**. You can control how the text or some words are displayed: **** or **<i>** for italic, **<b>** or **** for bold, **<u>** for underline.
- **To add a link**, use the anchor tag <a> as follows:
  - **<a href=**"address_of_another_wepbage"**>**Text to display to a visitor in order to entice them to click on that link.**</a>**
- These tags must be opened and closed around the text in question.

bodytags.html

```
1  <html>
2    <head>
3      <meta charset="utf-8">
4      <title>Our 1st webpage</title>
5    </head>
6    <body>
7      <h1> Main Heading </h1>
8      <h2> Sub Heading </h2>
9      <p> I am in <em> italic</em>.</p>
10     <p> I am <b><i><u>bold, italic and
       underlined</u></i></b>.</p>
11     <a href="https://www.straitstimes.com/">
       Click on me. </a>
12   </body>
13 </html>
```

# Adding content

Inside the <body> tag, you can include headings, text, links, lists, images, tables, forms, multimedia:

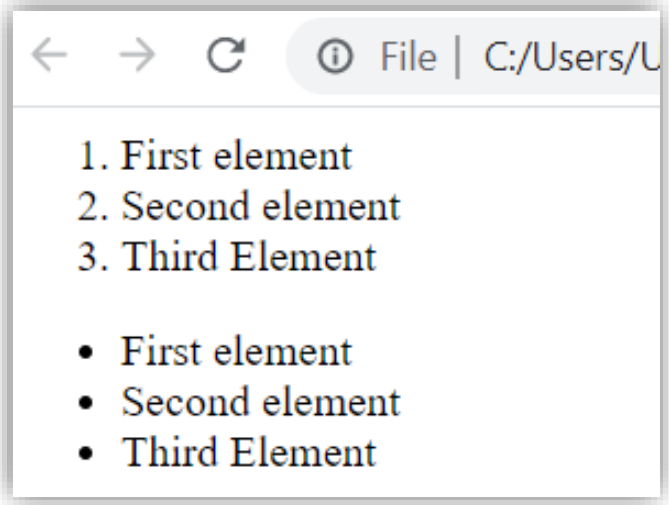
- **To add headings** from the most important to the less important: **<h1>, <h2>, …, <h6>**.
- **To add text**, use **<p>**. You can control how the text or some words are displayed: **** or **<i>** for italic, **<b>** or **** for bold, **<u>** for underline.
- **To add a link**, use the anchor tag <a> as follows:
  - **<a href=**"address_of_another_wepbage"**>**Text to display to a visitor in order to entice them to click on that link.**</a>**
- These tags must be opened and closed around the text in question.

# Adding content (II)

Inside the **<body>** tag, you can include headings, text, links, lists, images, tables, forms, multimedia:

- **To add a list**, use **<ol>** tag if you want the list elements to be ordered, otherwise **<ul>**.
- Inside the list tag, each element must be enclosed into a <li> tag.
- **To add an image**, use the **<img>** tag seen earlier:
  - **<img** src="mycrocodile.jpg" alt="A photo of my crocodile.">
- You can control the width and height with the corresponding attributes
- **<img** src="mycrocodile.jpg" alt="A photo of my crocodile.“ width=768 height=576>
- More on that another day.

```
<body>
  <ol>
    <li>First element</li>
    <li>Second element</li>
    <li>Third Element</li>
  </ol>
  <ul>
    <li>First element</li>
    <li>Second element</li>
    <li>Third Element</li>
  </ul>
</body>
```



# Adding block and inline elements

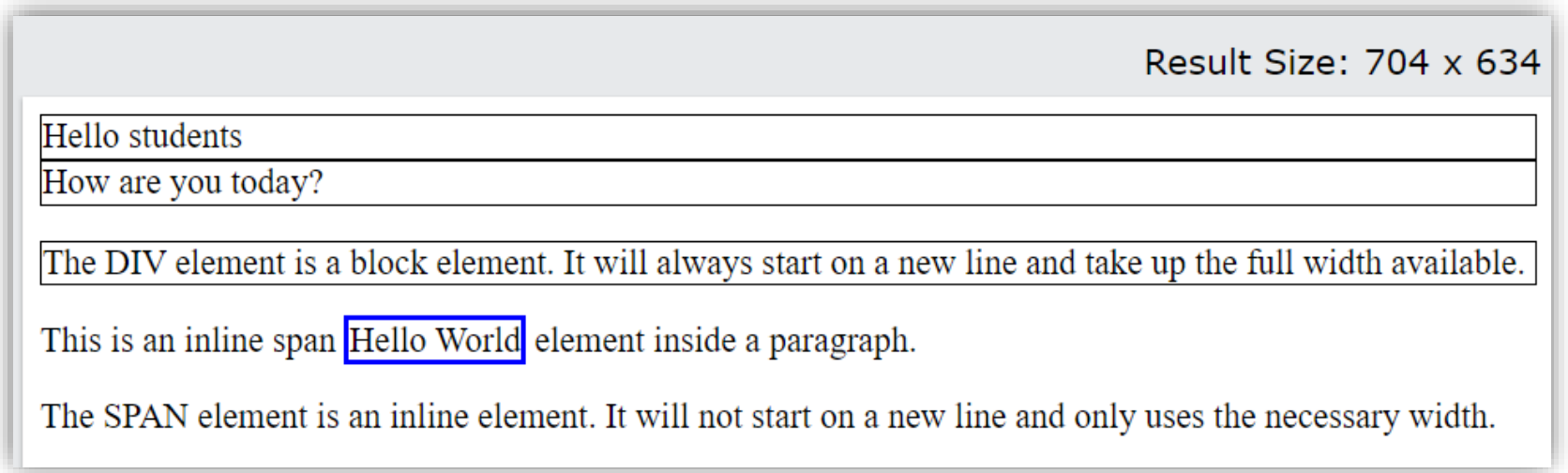
- HTML can also display default block and inline values, respectively using the tags ** ** and **.**
- ** ** elements always start on a new line.
- ** ** elements always take the full width available on your screen.
- ** ** elements have a top and a bottom margin.
- **** elements do not start on a new line.

```
<!DOCTYPE html>
<html>
<body>

<div style="border: 1px solid black">Hello students</div>
<div style="border: 1px solid black">How are you today?</div>
<p style="border: 1px solid black">The DIV element is a block element. It will
always start on a new line and uses the full width available.</p>
<p>This is an inline span <span style="border: solid blue">Hello World</span>
element inside a paragraph.</p>

<p>The SPAN element is an inline element. It will not start on a new line and
only uses the necessary width.</p>
</body>
</html>
```

- An inline element only takes up as much width as necessary and they do not have top and bottom margins.

# Adding block and inline elements

- HTML can also display default block and inline values, respectively using the tags <**div**> and <**span**>.
- <**div**> elements always start on a new line.
- <**div**> elements always take the full width available on your screen.
- <**div**> elements have a top and a bottom margin.
- <**span**> elements do not start on a new line.




- An inline element only takes up as much width as necessary and they do not have top and bottom margins.

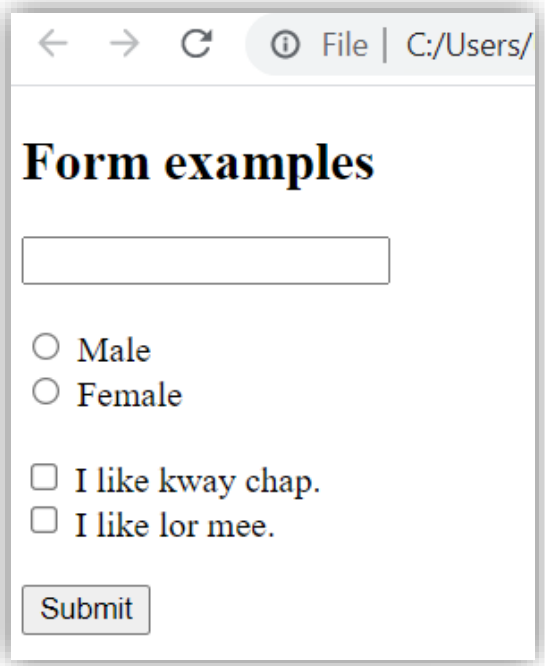
# Adding form (in short)

- You can use the **<form>** container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.
- Within the **<form>** container, the **input** element is used to display a type of form which depends on the value assigned to a **type** attribute.
- The general syntax is **<input type="Some_value">** where "Some_value" is:
  - "**text**" which displays a single-line text input field.
  - "**radio**" which displays a radio button (for selecting one of many choices)
  - "**checkbox**" which displays a checkbox (for selecting zero or more of many choices)
  - "**submit**" which displays a submit button (for submitting the form)
  - "**button**" which displays a clickable button.

```
<!DOCTYPE html>
<html>
<body>

<h2>Form examples</h2>

<form>
  <input type="text"><br><br>

  <input type="radio" id="male" name="gender">
  <label for="male">Male</label><br>
  <input type="radio" id="female" name="gender">
  <label for="female">Female</label><br> <br>

  <input type="checkbox" name="kwaychap">
  <label for="kwaychap"> I like kway chap.</label>
  <br>
  <input type="checkbox" name="lormee">
  <label for="lormee"> I like lor mee.</label>
  <br><br>

  <input type="submit" value="Submit">
</form>
</body>
</html>
```

# Adding form (in short)

```html
<!DOCTYPE html>
<html>
<body>

<h2>Form examples</h2>

<form>
  <input type="text"><br><br>

  <input type="radio" id="male" name="gender">
  <label for="male">Male</label><br>
  <input type="radio" id="female" name="gender">
  <label for="female">Female</label><br> <br>

  <input type="checkbox" name="kwaychap">
  <label for="kwaychap"> I like kway chap.</label>
  <br>
  <input type="checkbox" name="lormee">
  <label for="lormee"> I like lor mee.</label>
  <br><br>

  <input type="submit" value="Submit">
</form>
</body>
</html>
```



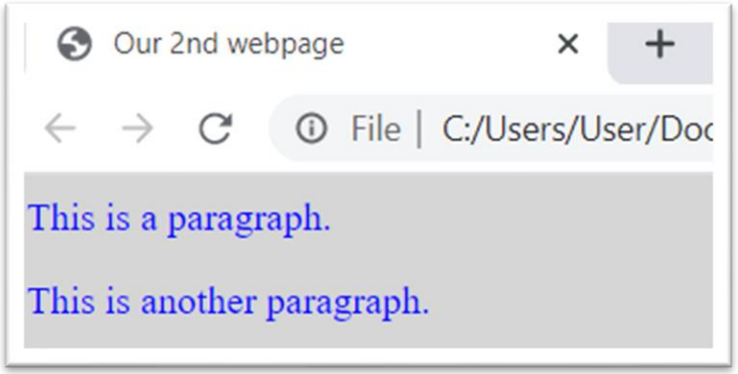

# Adding form (in short)

- The **<label>** tag defines a label for many form elements. It displays an (explicit) text close to the form elements.
- The for attribute of the **<label>** tag should be equal to the id attribute (or even the name for checkbox) of the **<input>** element to bind them together.
- Radio buttons let a user select ONE of a limited number of choices whereas checkboxes let a user select zero or more options of a limited number of choices.
- Notice that each input field must have a name attribute to be submitted. If the name attribute is omitted, the value of the input field will not be sent at all.
- Right now, the submit button does not work because the action to trigger has not be defined.

```
<!DOCTYPE html>
<html>
<body>

<h2>Form examples</h2>

<form>
  <input type="text"><br><br>

  <input type="radio" id="male" name="gender">
  <label for="male">Male</label><br>
  <input type="radio" id="female" name="gender">
  <label for="female">Female</label><br> <br>

  <input type="checkbox" name="kwaychap">
  <label for="kwaychap"> I like kway chap.</label>
  <br>
  <input type="checkbox" name="lormee">
  <label for="lormee"> I like lor mee.</label>
  <br><br>

  <input type="submit" value="Submit">
</form>
</body>
</html>
```

# HTML style tag

- In the header, use the **style** tag to define how the elements of the page should be formatted or displayed.
- In the example, I am setting the background colour of the body to be light grey and the colour of its text to be blue.

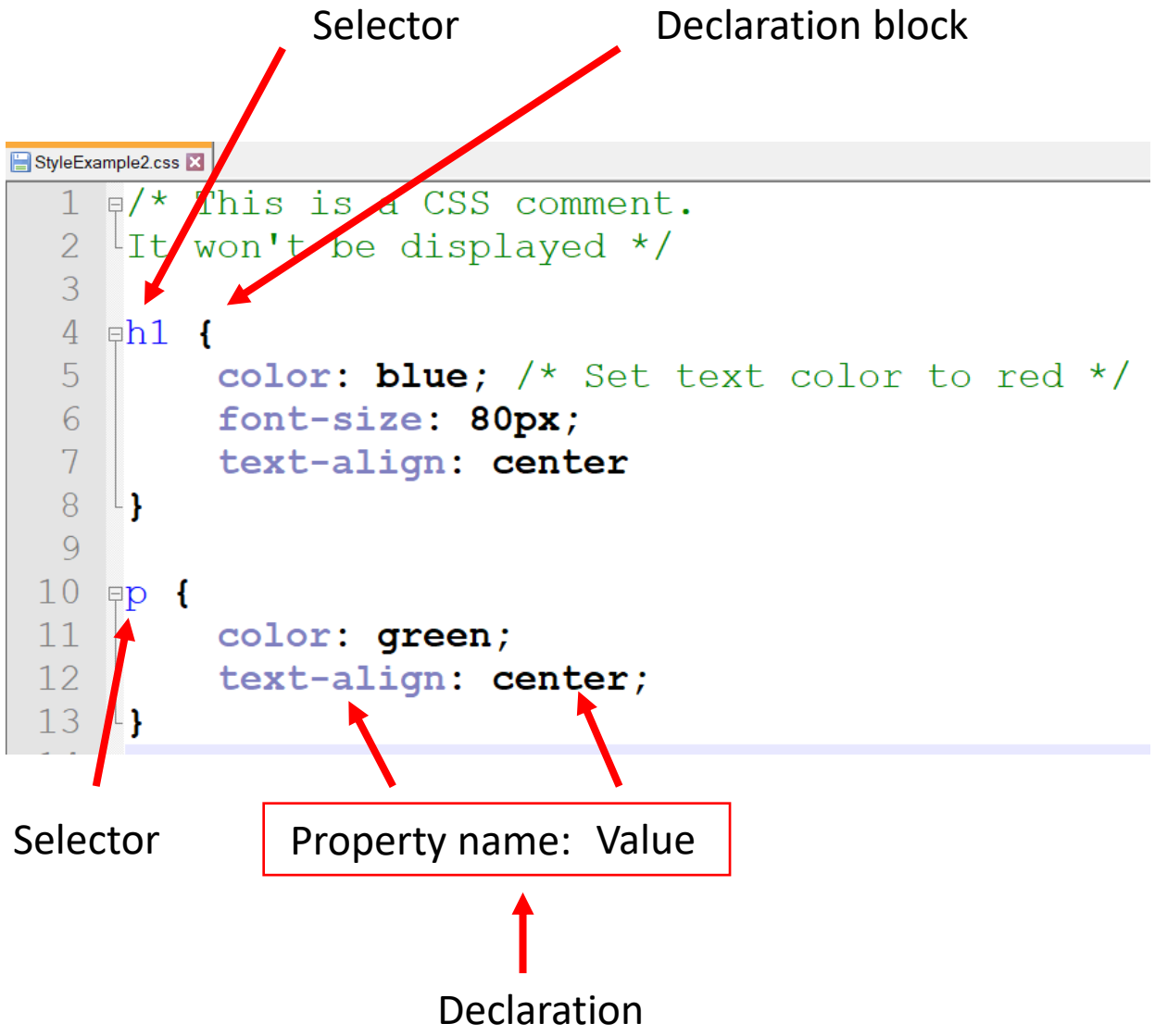



example2.html

```
1  <html>
2    <head>
3      <meta charset="utf-8">
4      <title>Our 2nd webpage</title>
5      <style>
6      body {
7        background-color: lightgrey;
8        color: blue;
9      }
10     </style>
11   </head>
12   <body>
13     <p>This is a paragraph.</p>
14     <p>This is another paragraph. </p>
15   </body>
16 </html>
```

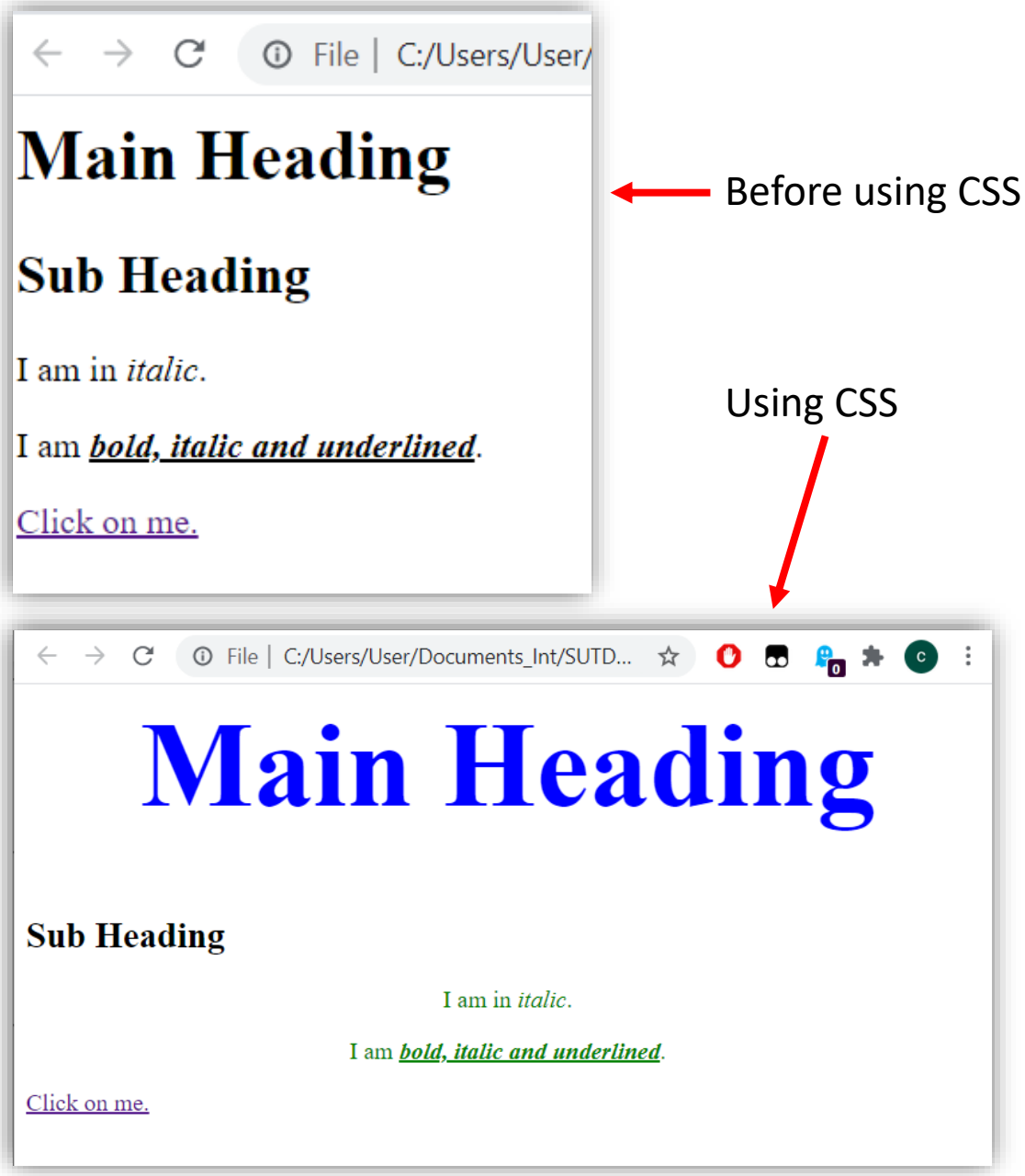
# The CSS syntax

- CSS stands for Cascading Style Sheets.
- A CSS rule consists of a selector and a declaration block.
- 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 and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.



# The CSS syntax

- A CSS rule consists of a selector and a declaration block.
- 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 and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.



# The CSS **id** selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.
- The CSS rule on the right will be applied to the HTML element with **id="para1".**

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>

</body>
</html>
```

# CSS file

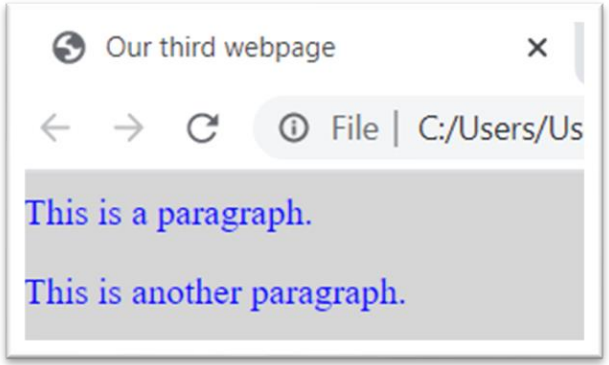
- All these style rules becomes quickly messy if left in an HTML document. Usually, we write them in a separate CSS file.
- Dictates how the HTML elements of a website should appear on the frontend of the page.
- In the HTML file, we replace the style tags by a pointer to the CSS file using the **link** tag.

example3.html | StyleExample.css

```
1  body {
2    background-color: lightgrey;
3    color: blue;
4  }
```

example3.html | StyleExample.css

```
1  <html>
2    <head>
3      <meta charset="utf-8">
4      <title>Our third webpage</title>
5      <link href="StyleExample.css" rel="stylesheet">
6    </head>
7    <body>
8      <p>This is a paragraph.</p>
9      <p>This is another paragraph. </p>
10   </body>
11 </html>
```



# About the link tag

- The **link** tag does not need to be closed.
- Immediately followed by a reference of the location to the CSS file (in general, in the same folder), **href=** “some_address” and a **rel** attribute that specifies the relationship between the current document and the linked document/resource.
- For now, we set **rel** to “stylesheet”.
- More on that later. Meanwhile:

  https://www.w3schools.com/tags/att_link_rel.asp

example3.html | StyleExample.css

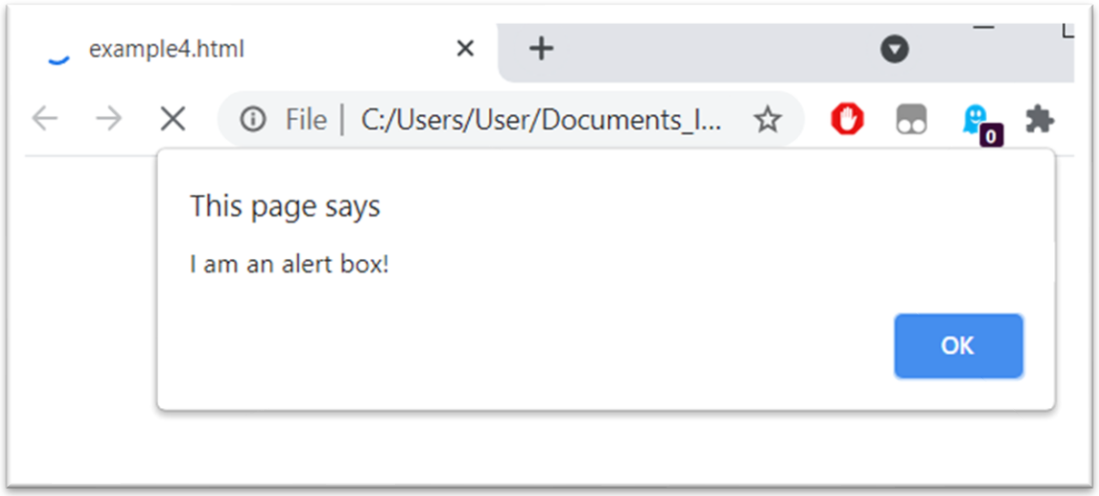
```
1  <html>
2    <head>
3      <meta charset="utf-8">
4      <title>Our third webpage</title>
5      <link href="StyleExample.css" rel="stylesheet">
6    </head>
7    <body>
8      <p>This is a paragraph.</p>
9      <p>This is another paragraph. </p>
10   </body>
11 </html>
```

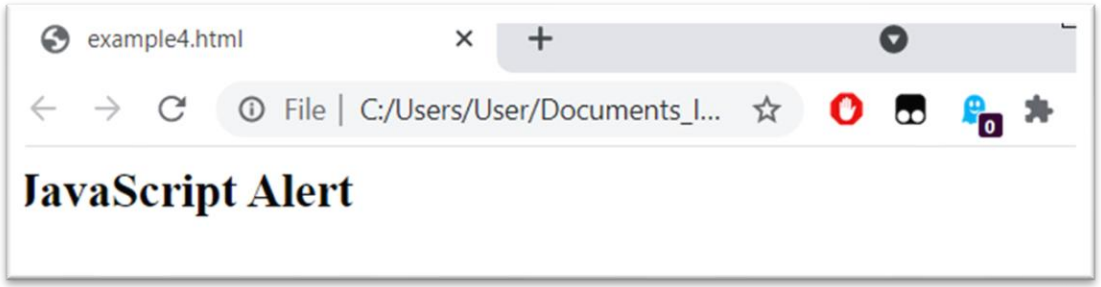
# What about JavaScript?

- **How to place JS elements in a HTML document?**
- Same than with CSS elements. Wherever you want to insert JS elements, use the **script** tag and write JS code in between.
- **alert** is a JS function which creates a dynamic pop-up window.

example4.html

```
1  <html>
2  <body>
3  <h2>JavaScript Alert</h2>
4  <script>
5    alert("I am an alert box!");
6  </script>
7  </body>
8  </html>
```



# HTML document using external JS files

- Like for CSS files, in what follows, we will mainly write JS files and will make a pointer in the HTML document to them if necessary.
- The **src** attribute in the HTML document points to the location of the JS file.
- You can now open the HTML document into your web browser. The result should be similar to example4.html.
- Note that you can reuse the JS file in other HTML documents.

firstjsfile.js

```
1  alert("I am an alert box!");
```

example5.html | firstjsfile.js

```
1  <html>
2  <body>
3  <script src="firstjsfile.js"></script>
4  <h2>JavaScript Alert</h2>
5  </body>
6  </html>
7
```

# Activity 1

- Create a webpage as similar as possible on the surface than a basic Google page.
- Requirements:
  - There should be at least one anchor in the page that brings the user to another webpage.
  - When the user opens your webpage, an alert box should be triggered to inform him/her that it is not the original Google page.
  - Upload your files on the personal students Google Drive folder before Wednesday 30 June, 11:59pm. You should have received a link by email. There should be at least one HTML file, one CSS file, and one JS file in the folder.
  - Feel free to add any features, any anchors to other HTML files if you want.

# Developing JavaScript programs

- It might be troublesome to juggle between JS files and HTML documents every time we develop a JS program.
- In order to save time, the developers often use proper JS tools and functions to visualize and debug their programs.
- I will use the interactive online JS console at www.scrimba.com.
- **console.log()** is a JS function that displays its content in a JS console.

# Comments in JavaScript

- Comments are ignored by the interpreter but are necessary to make a program easier to read.
- To insert a comment on a single line, place a pair of forward slashes **//** before the text.
- For multi-line comments, use **//** at the beginning of each line
- Or wrap the whole text between **/*** and ***/**

```
5  // I am a comment.
6  console.log("Hello SUTD!")
7  // We are
8  // multi-line comments
9  console.log("Hello Singapore!")
10 /*
11 We are nicer
12 multi-line comments
13 */
14 console.log("Hello World!")
```

```
CONSOLE
› Hello SUTD!
› Hello Singapore!
› Hello World!
```

# Declaring variables and assigning values

- Use the **var** keyword to declare a variable followed by its name.
- A variable can be declared without being initialized.
- Use the **assignment** operator, **=**, to assign a value to the variable.
- To end the statement, use a semi-colon **;**

```
 9  var x;
10  var y = 10;
11  console.log(x)
12  console.log(y)
13
```

CONSOLE

```
undefined
10
```

# Great advice #1

**Great Advice #1: Initialize your variables and end your statements.**

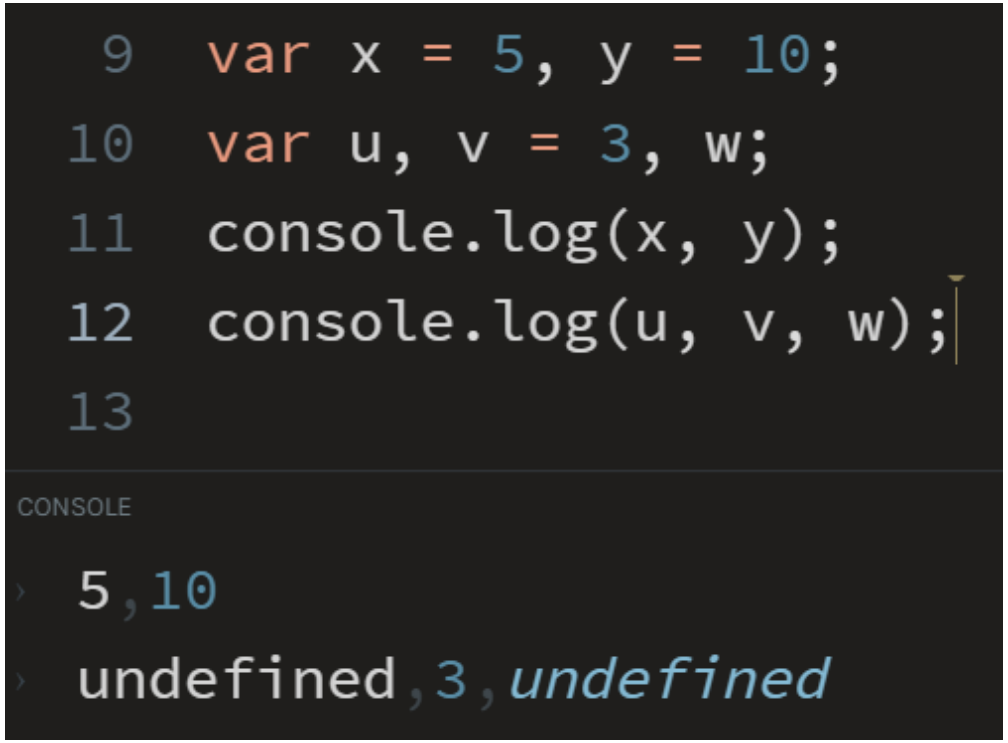
JS is sometimes a bit too permissive, and the semi-colon can often be omitted. It is however considered as good practice to end your statements with a semi-colon.

Another good practice is to assign a value during the declaration.

# Multiple assignments

JS also allows **multiple declarations and assignments**

- by having several variables names and values separated by commas on both sides the values are respectively assigned to the variable names.
- The statement must end by a semi-colon rather than a comma.



# About variable names

- **Variables names** can include any combination of **letters** (both lowercase and uppercase), **digits, underscore** (_) and dollar **symbols ($)**.
- You can use the **underscore** symbol as a separator to make variables names a bit more explicit.

```
//A non-explicit variable name
var x = 10
//An explicit variable name
var radius = 10
//Too explicit maybe?
var number_of_students_in_this_class = 15
```

# About variable names

- **Variables names** can include any combination of **letters** (both lowercase and uppercase), **digits, underscore** and **dollar** symbols.

However,

```
16  //A valid (but ugly) name
17  var a_1st_VaRiaBle = 2;
18
19  //Variables cannot start with a digit
20  var 2nd_variable = 0;
21
CONSOLE
› Error: SyntaxError: unknown: Identifier d
```

```
22  //But they can start with _ or $
23  var _credit = 90, $debit = 50;
24  console.log(_credit, $debit);
CONSOLE
› 90,50
```

```
26  //Avoid keywords and reserved words
27  var import = 0;
CONSOLE
› Error: SyntaxError: unknown: Unexpected keyword
```

# About variable names

- **Variables names** can include any combination of **letters** (both lowercase and uppercase), **digits, underscore** and dollar **symbols**.

However,

- They cannot **start with a digit** or use **special characters other than underscores and dollars.**
- They may also not use some **reserved keywords** (e.g. import, var, etc.)

```
16  //A valid (but ugly) name
17  var a_1st_VaRiaBle = 2;
18
19  //Variables cannot start with a digit
20  var 2nd_variable = 0;
21
```

CONSOLE

```
Error: SyntaxError: unknown: Identifier d
```

```
22  //But they can start with _ or $
23  var _credit = 90, $debit = 50;
24  console.log(_credit, $debit);
```

CONSOLE

```
90,50
```

```
26  //Avoid keywords and reserved words
27  var import = 0;
```

CONSOLE

```
Error: SyntaxError: unknown: Unexpected keyword
```

# Great advice #2

**Great Advice #2: make your variables names explicit.**

It is a good idea to make your variables **explicit**, rather than using meaningless ones that leave your reader guessing (x, y, a, b1, c2, etc.).

You can **use underscores** if you find it helpful (e.g. var current_year = 2021).

Your future self, and colleagues, will thank you, when they work on your code later on.

# Variable types

- **Definition (variable type/class):** Everything is stored in memory as 0s and 1s. A **variable type/class** defines how the sequence of 0s and 1s should be interpreted.

# Variable types

- **Definition (variable type/class):** Everything is stored in memory as 0s and 1s. A **variable type/class** defines how the sequence of 0s and 1s should be interpreted.

Let us start with two basic types

- **Number:** JS does not require numbers to be declared as integers or floats. You can use exponential notation for long numbers (put the letter e in place of “times 10 to the power of”)
- **String:** a string of text.

```
29  var x = 2;
30  var y = 2.5e4;
31  var z = "I am hungry!";
32  console.log(typeof(x));
33  console.log(typeof(y));
34  console.log(typeof(z));
```

CONSOLE

```
number
number
string
```

# Strings: quotes and special characters

- You can place all sorts of text and characters inside a string.
- JS allows the use of single or double quotes to define a string.

However, to consider:

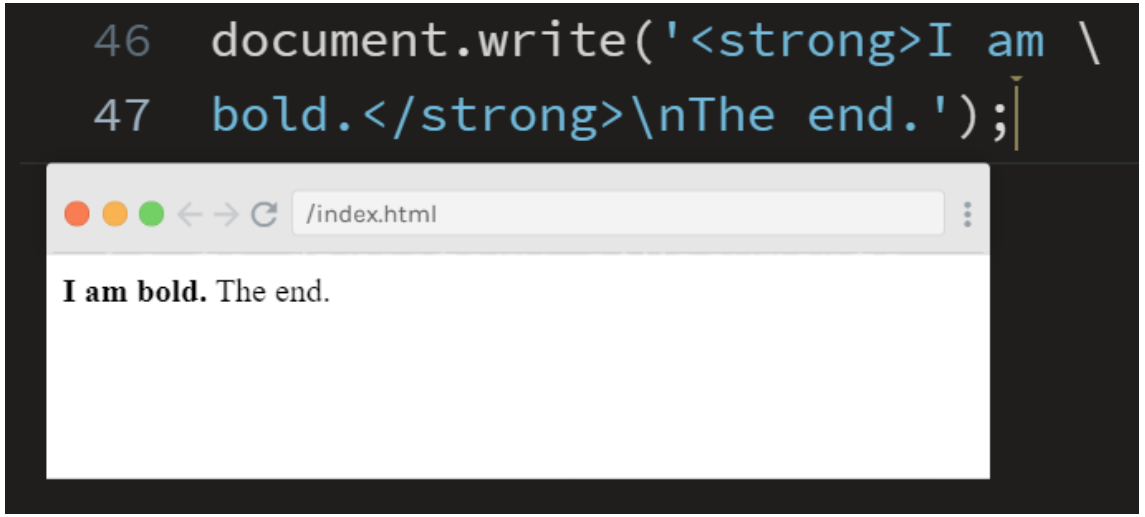
- The starting and ending quote must match.
- Strings are case sensitive.
- Some characters require a special code to be correctly interpreted by JS: (\n  for newline, \t for tab, \\ for backslash \, \' and \" for single and double quotes respectively, etc.)

```
40  //The backslash will not be displayed
41  console.log("Go to directory c:\Documents");
42
43  //The backslash will be displayed
44  console.log('Go to directory c:\\Documents');
```

```
CONSOLE
› Go to directory c:Documents
› Go to directory c:\Documents
```

# Strings: document.write()

- The **document.write()** method is used to insert JS text in a HTML document.
- This method supports some HTML tags (e.g., strong tag to bold the text)
- Note: a backslash \ at the end of a line in a string allows you to finish the string at the next line.
- \n has not been interpreted. Why?
- \n is only a newline in JS, not in HTML. It only affects the appearance of the source code.
- To add a new line in the browser display, use the **<br>** tag.




# Strings: escaping characters

- Caution: watch for single, double quotes and apostrophes in a string.
- You need to escape them with \ to display them in the console and not to cause errors.
- You can also use single quotes inside double quotes (or the reverse) to avoid escaping the quotes.

```
49  console.log('Arun said "Hi Bilal!".');
50  console.log("Bilal replied \"How are you?\".");
51  console.log("\"I'm fine.\" answered Arun.");
```

CONSOLE

```
Arun said "Hi Bilal!".
Bilal replied "How are you?".
"I'm fine." answered Arun.
```

# Other basic types

- **Boolean:** can only have two values, **true** or **false**. Useful in branching programs or for event handlers.
- **Null:** used to indicate an empty *object*. A null data type can only have one value: null.
- **Undefined:** occurs when a variable has not been assigned an initial value. An undefined data type can only have one value: undefined.
- There exists other types (**Symbol**, **Object**) but they are more advanced. More on that later.

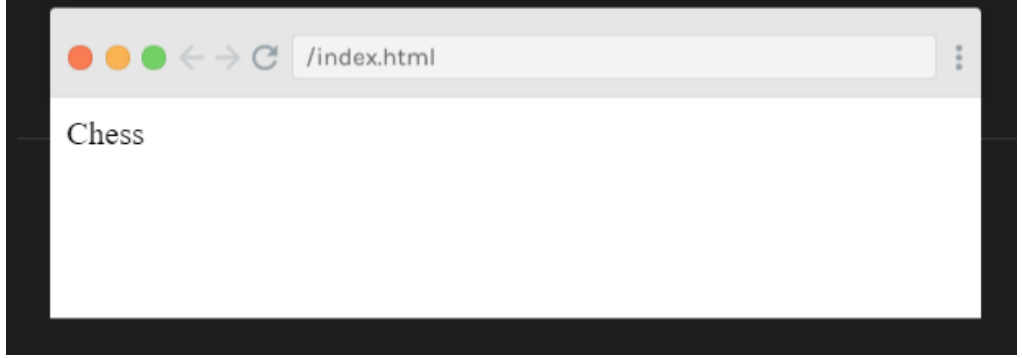
```
53  var covid_is_annoying = true;
54  var laksa_is_meh = false;
55  var empty = null;
56  var whoami;
57  console.log(typeof(covid_is_annoying));
58  console.log(typeof(laksa_is_meh));
59  console.log(typeof(empty));
60  console.log(typeof(whoami));
```

CONSOLE

```
boolean
boolean
object
undefined
```

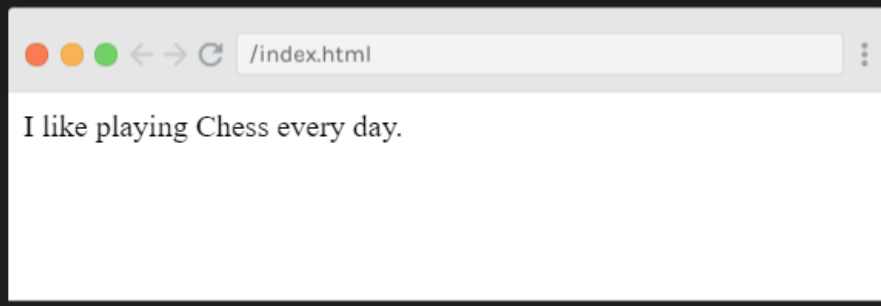
# Using variables in scripts

```
63  var my_hobby = "Chess";
64  document.write(my_hobby);
```




- You can make a call to a variable which has been declared in the program. In the example, **document.write(my_hobby)** prints the value of my_hobby in the browser.
- You can add text strings to the variable to print using the **+** operator.

```
var my_hobby = "Chess";
document.write("I like playing " + my_hobby + " every day.")
```

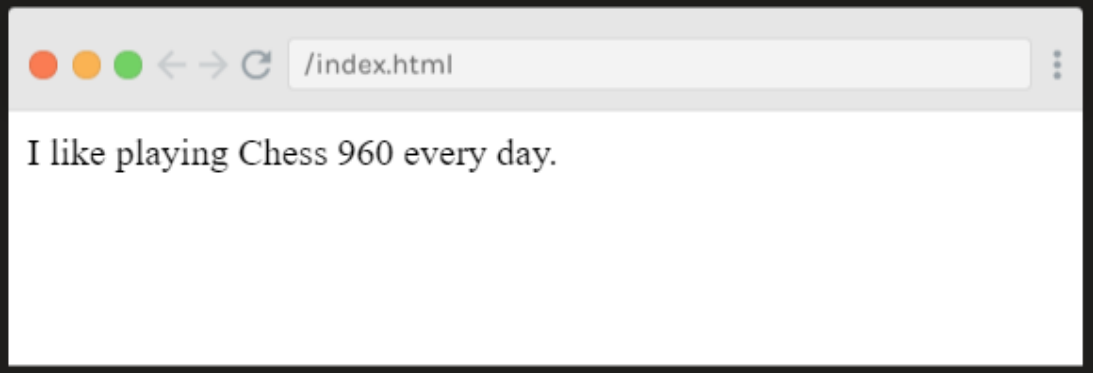



- Note the space at the end of the 1st string and at the beginning of the 2nd string. This ensures that a space appears before and after the variable.

# String concatenation with variables

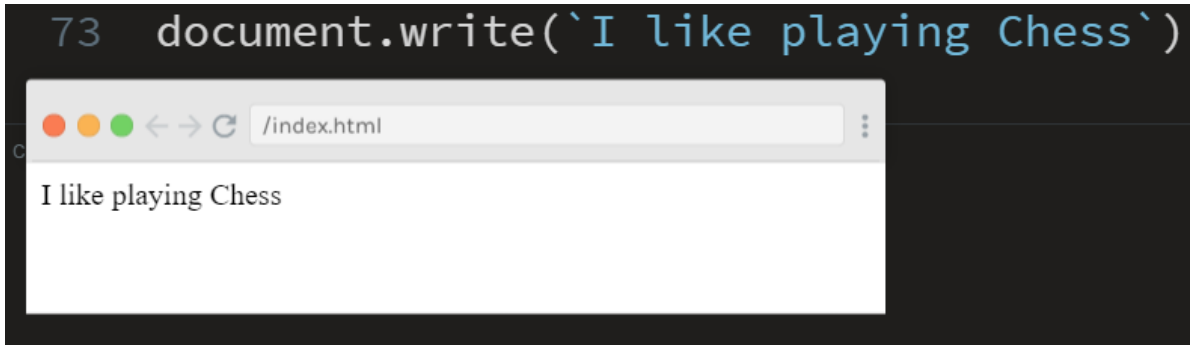
- It might be easier to place every string involved into one concatenated variable to avoid dealing with the quotes.
- Note that it is possible to concatenate strings with numbers.
- JS converts the number into a string before the concatenation.

```
var my_hobby = "Chess ";
var variation = 960;
var first = "I like playing ";
var second = " every day."
document.write(first + my_hobby + variation + second)
```



# Template literals

- You can also enclose a string (called templated literals here) with backticks (`).
- Main advantage: you can use special placeholder syntax to insert variables without using concatenation.
- Enclosing your variables within **${}** concatenates the values of the variables in the string. Note that the variable values can be any type of strings (with quotes or backticks) and also numbers.
- Template literals allow a string to be on multiple lines whereas strings within quotes cannot.
- Much more convenient!




# Conclusion

- **HTML, CSS, JS:** basic webpage structure and elements
- **JS Commenting:** in-line, block
- **Variables:** Naming conventions, types, assignments
- **Strings**: Concatenation, automatic conversion, and template literals