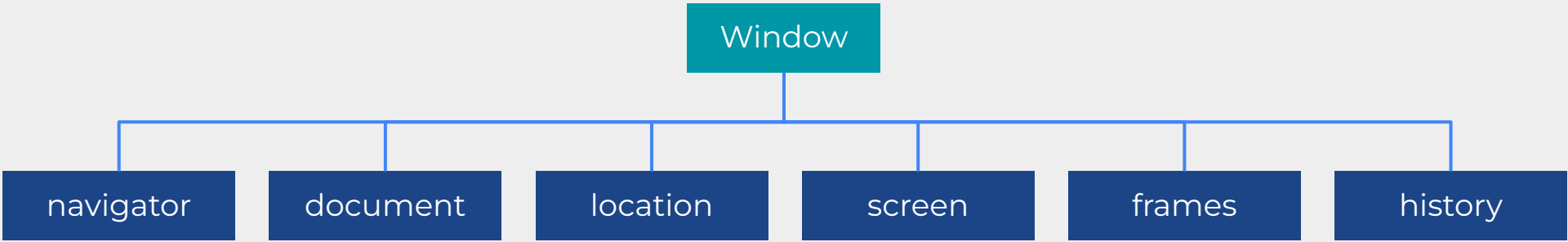
Javascript

Basic Concept of DOM

erb 僱員再培訓局 Employees Retraining Board
SCOPE School of Continuing and Professional Education 專業進修學院 香港城市大學 City University of Hong Kong
KRYSTAL INSTITUTE DIGITAL ECONOMY CORE TECHNOLOGY

The Browser Object Model (BOM)

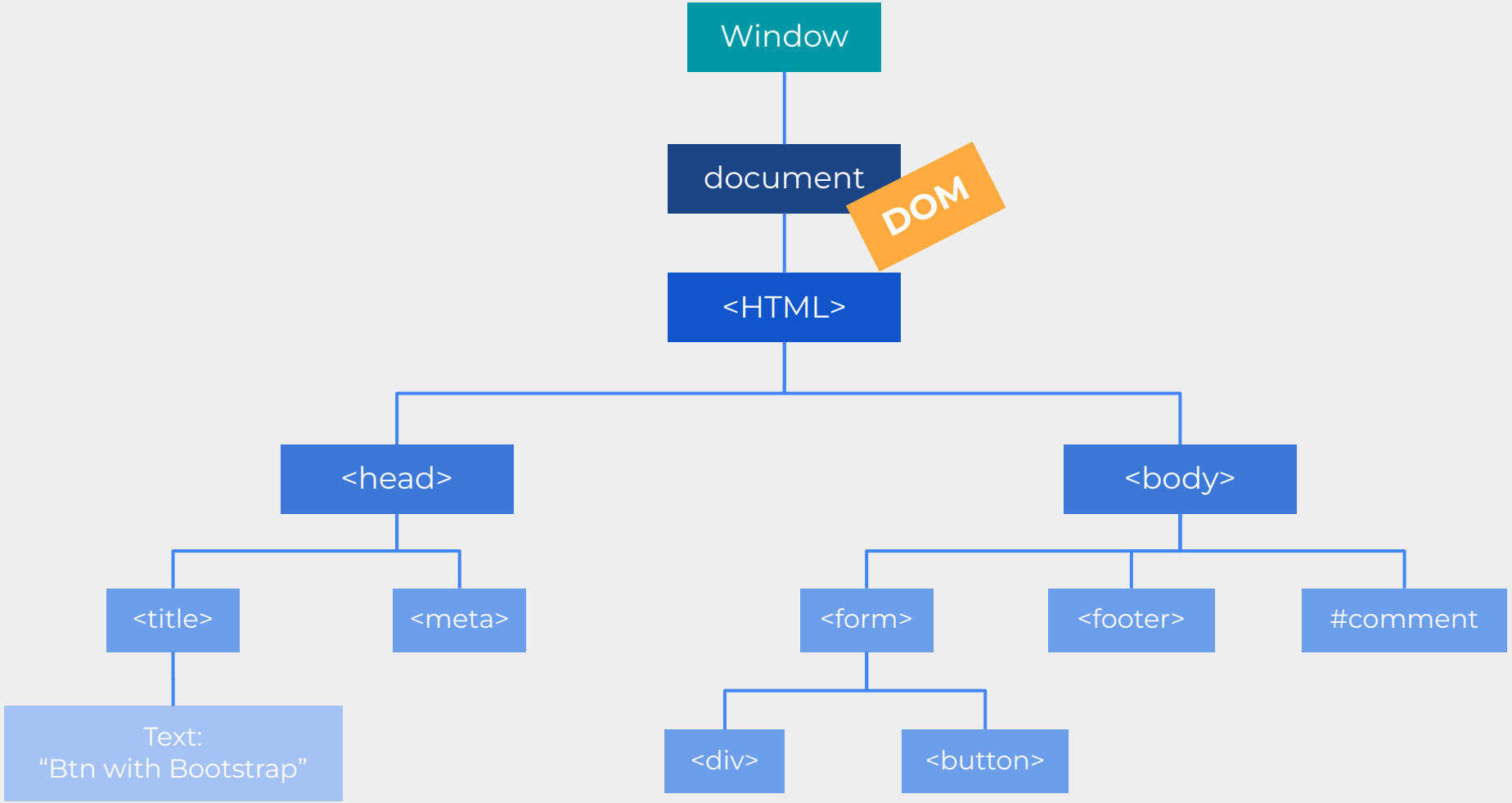
- The browser object model (BOM) is a hierarchy of browser objects that are used to manipulate methods and properties associated with the Web browser.
- All the things inside the BOM are objects which have their own properties and methods.
- The highest level is the window object, which is created once your browser is opened.
- All other browser objects are under window object.



```
> window
< ▼ Window {window: Window, self: Window, document:
  document, name: '', location: Location, …} i
    ▶ $: ƒ (e,t)
    ▶ alert: ƒ alert()
    ▶ atob: ƒ atob()
    ▶ blur: ƒ blur()
    ▶ bootstrap: {Alert: ƒ, Button: ƒ, Carousel: ƒ, Col
    ▶ btoa: ƒ btoa()
    ▶ caches: CacheStorage {}
    ▶ cancelAnimationFrame: ƒ cancelAnimationFrame()
    ▶ cancelIdleCallback: ƒ cancelIdleCallback()
    ▶ captureEvents: ƒ captureEvents()
```

The Document Object Model (DOM)

- consists of objects of the document or Web page loaded in the browser window.
- Those objects are formed in a tree structure called **DOM tree.**
- Any type of **object** in the DOM is called a **node.**
- There are 12 node types in total, but only 4 of them are commonly used in practice:
- document – the “entry point” into DOM.
- element nodes – HTML-tags, the tree building blocks.
- text nodes – contain text.
- comments – sometimes we can put information there, it won’t be shown, but JS can read it from the DOM.
- **Everything in HTML becomes part of the DOM tree**, even those are invisible on the screen.



Why is it important to understand the DOM

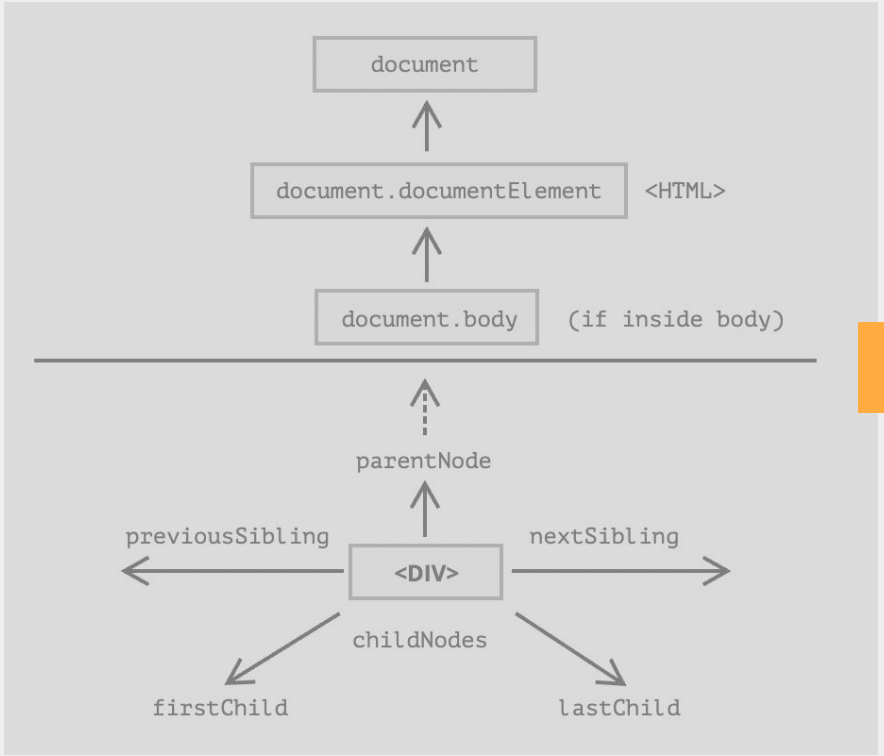
- The browser engine is written by **JavaScript**.
- A dynamic page can **ONLY** made through applying JavaScript processing to the DOM Tree.
- Onec the HTML document is loaded into the browser engine, you are able to interact with the HTML tag (the element node) through modifying their properties and methods.
- **properties** - refer to structural, visual, or content characteristics of the element
- **methods** - refer to actions the object can perform

- The next question would be how can we reach to a specific DOM object?

The relationship of the DOM tree

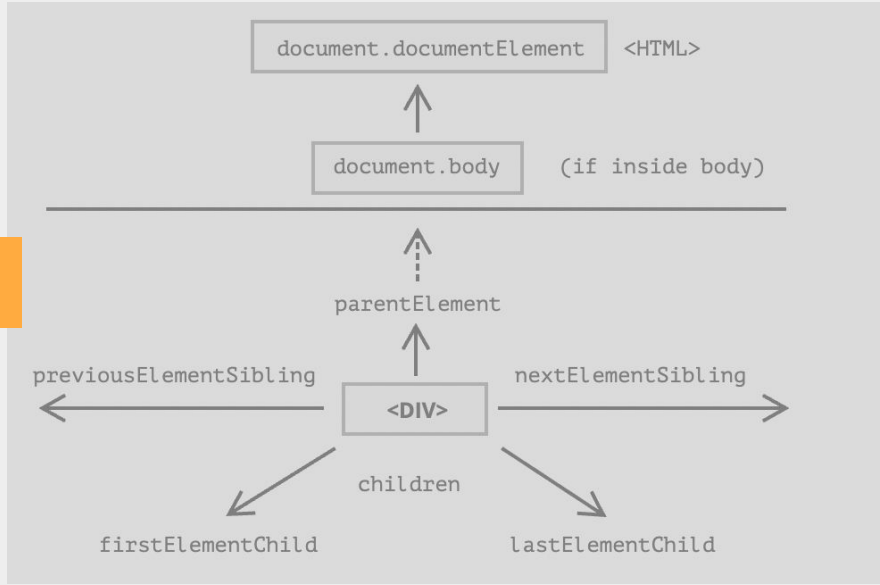
- All operations on the DOM start with the **document object**. That's the main "entry point" to DOM. From it we can access any node.
- You can access to both nodes and elements inside the DOM tree.
- But note that the result of accessing the nodes in the DOM is **differ** from the result of accessing the elements in the DOM
- Although they look similar.

DOM Nodes



V.S

Element Only



DOM Nodes ❖ Try in the console: let children = document. body.childNodes;

```
> let children = document.body.childNodes;
<· undefined
> children
<· ▼NodeList(11) [text, form.p-5, text, footer, text, script, text, script,
   text, script, text] i
    ▶0: text
    ▶1: form.p-5
    ▶2: text
    ▶3: footer
    ▶4: text
    ▶5: script
    ▶6: text
    ▶7: script
    ▶8: text
    ▶9: script
    ▶10: text
      length: 11
    ▶[[Prototype]]: NodeList
```

Element Only ❖ Try in the console: let children = document.body.children;

```
> let children = document.body.children;
<- undefined
> children
<- HTMLCollection(5) [form.p-5, footer, script, script, script] i
    0: form.p-5
    1: footer
    2: script
    3: script
    4: script
    length: 5
    [[Prototype]]: HTMLCollection
```

Iteration of DOM collections

- Although **childNodes** and **children** looks like array, they are actually **NOT** an array.
- They are **collection** – a special array-like iterable object.
- So Array methods are not able to use for iteration of the collection.
- for..of can be used for the iteration:

```
for (let node of document.body.childNodes) {
    console.log(node);
}
```

- The same method can be used for the element iteration too.

```
for (let elem of document.body.children) {
    console.log(elem);
}
```

Node Navigation

- parentNode - show the parent node of the current node
- childNodes - show all the direct child in a Node list
- firstChild / lastChild - fast access to the first or the last children
- previousSibling / nextSibling: show the pervidou or next node of the same parent

Element Navigation

- parentElement - show the parent element of the current node
- children - show only the children that are element node
- firstElementChild / lastElementChild - fast access to the first or the last element children
- previousElementSibling / nextElementSibling - show the neighbor elements

Searching inside DOM

QuerySelector Methods:

- querySelector() Method
- querySelectorAll() Method

GetElement Methods:

- getElementById() Method
- getElementsByTagName() Method
- getElementsByClassName() Method

document.getElementById

- If an element has the id attribute, we can get the element using the method **document.getElementById(id)**, no matter where it is.
- The method **getElementById** can be called **ONLY** on document object. It looks for the given id in the whole document.

  →Only **document.getElementById**, **NOT** **anyElem.getElementById**
- The id must be **unique,** with only **one** element in the document with the given id.

  →Otherwise, the behavior of methods that use it is unpredictable

  →may return any elements at random

querySelectorAll(css)

- A **NodeList** object with the elements that matches the CSS selector(s).
- Since it returns a NodeList, the following assignment is not execuable:

  →document.querySelectorAll("p").style.backgroundColor = "red";
- Instead, you need to run:

  →document.querySelectorAll("p")[0].style.backgroundColor = "red";

querySelector(css)

- returns the **first element** for the given CSS selector.

getElementsByClassName() / getElementsByTagName()

- They return a **collection**, not an element! So the following is not working:

  →document.getElementsByTagName('input').value = 5;

- should either **iterate** over the collection or get an element by its **index** like:

  →document.getElementsByTagName('input')[0].value = 5;

- Do not forget the letter **“s”** in the name of the method.