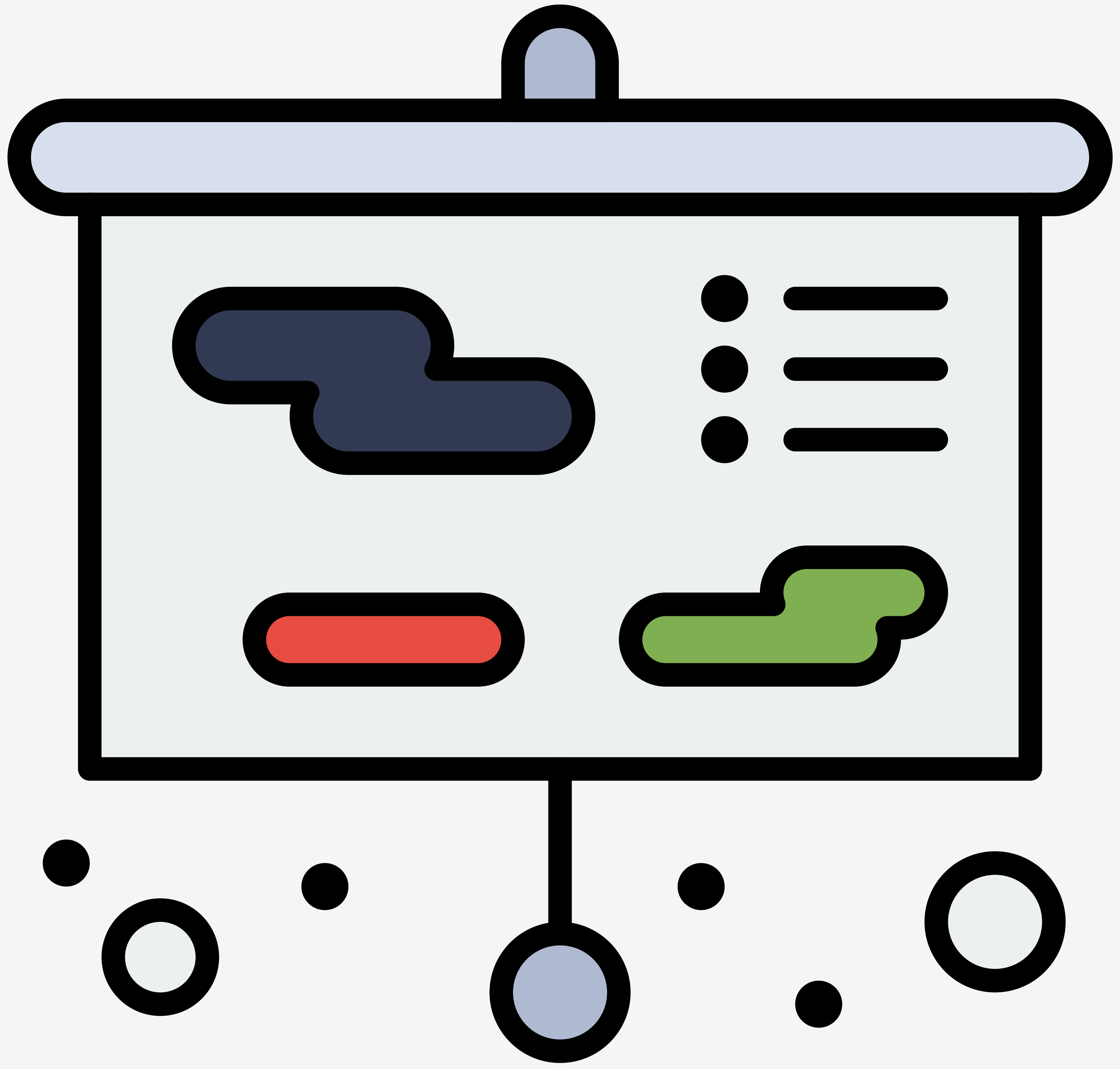
INTRODUCTION TO JAVASCRIPT

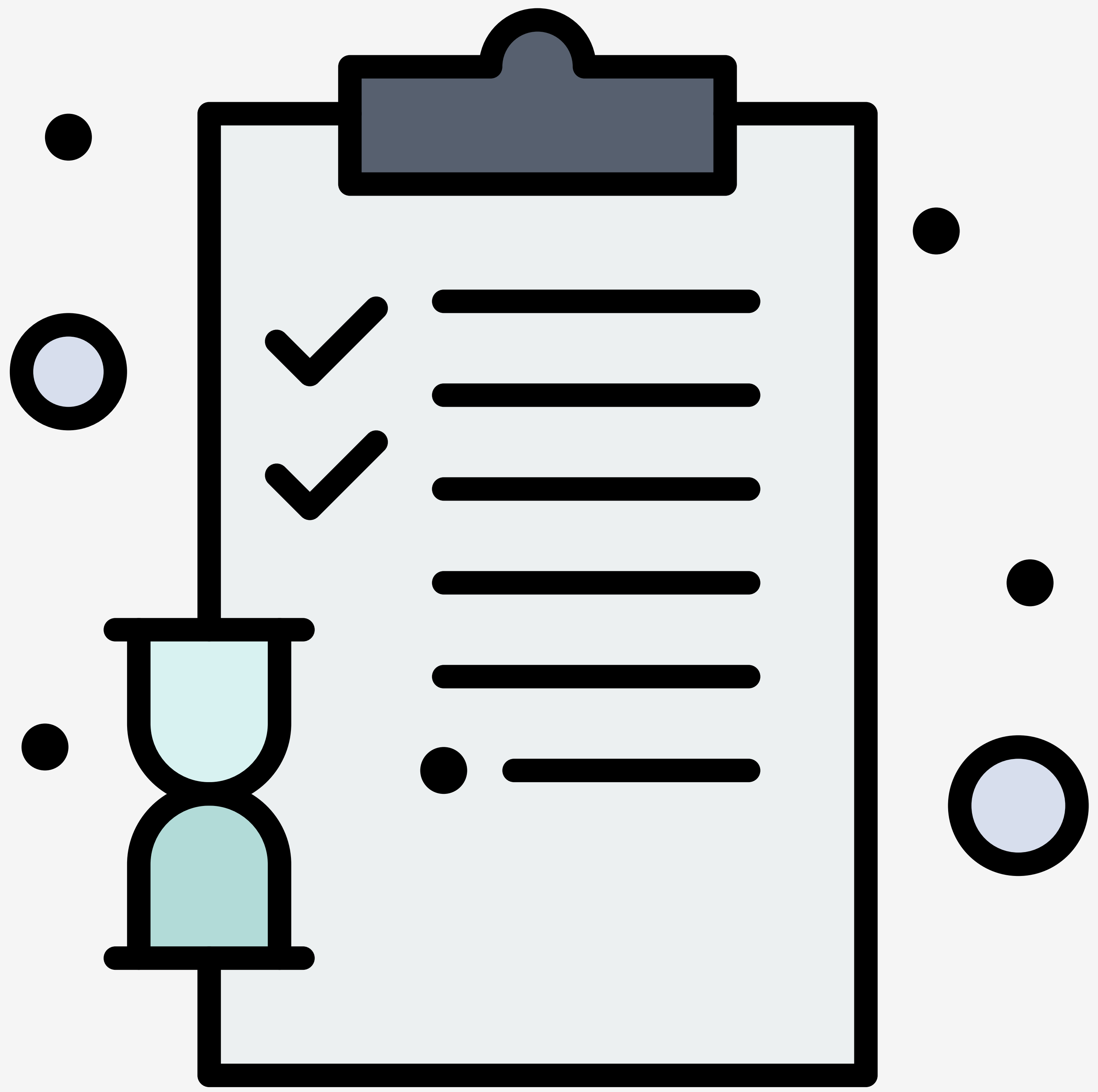
Lecture 12

TODAY'S TOPICS



- Manipulating DOM Elements
- Review: Fun Facts

ANNOUNCEMENTS

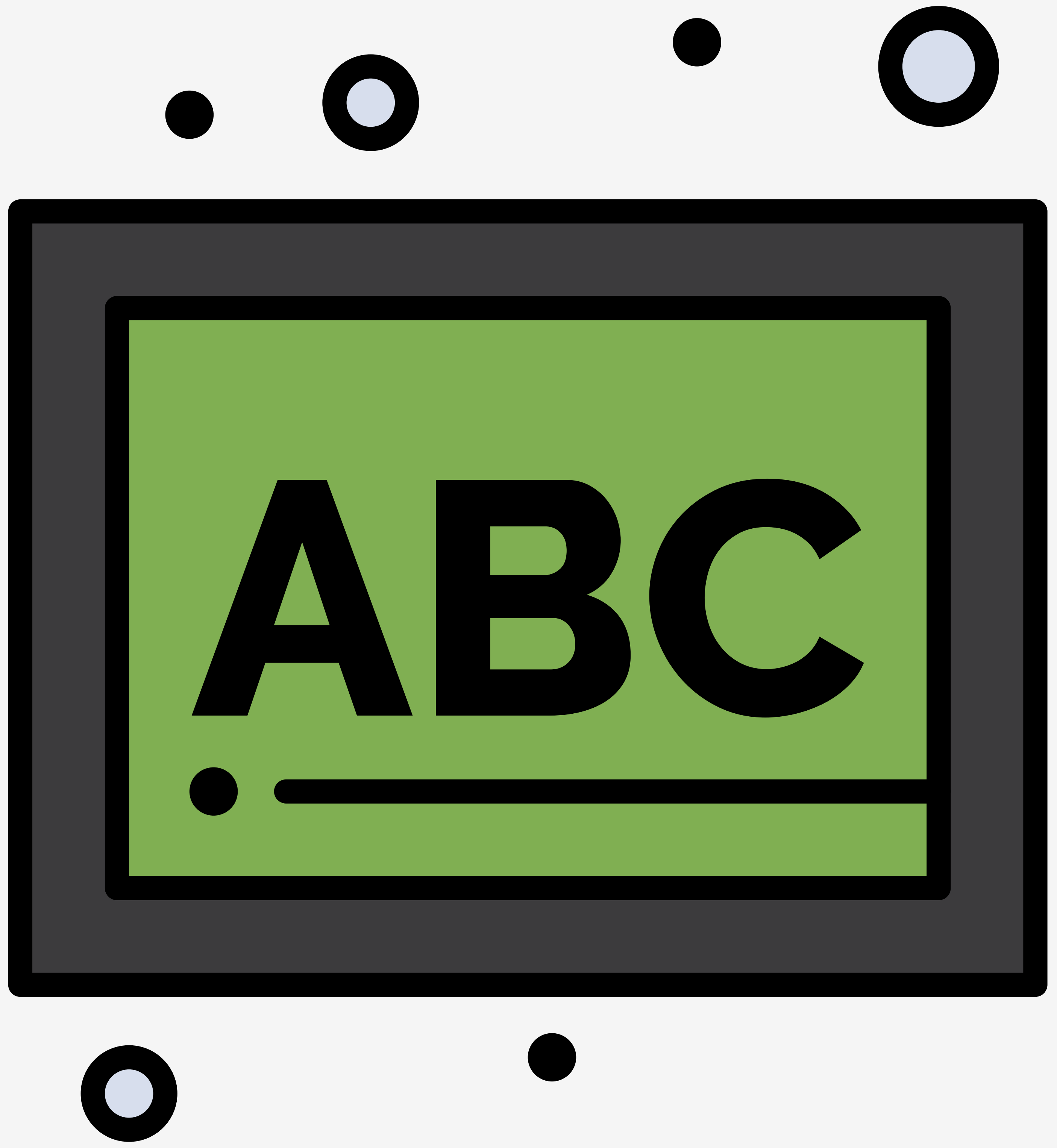


- Sign-in Sheet
- Recordings

QUESTIONS?

DOCUMENT OBJECT MODEL

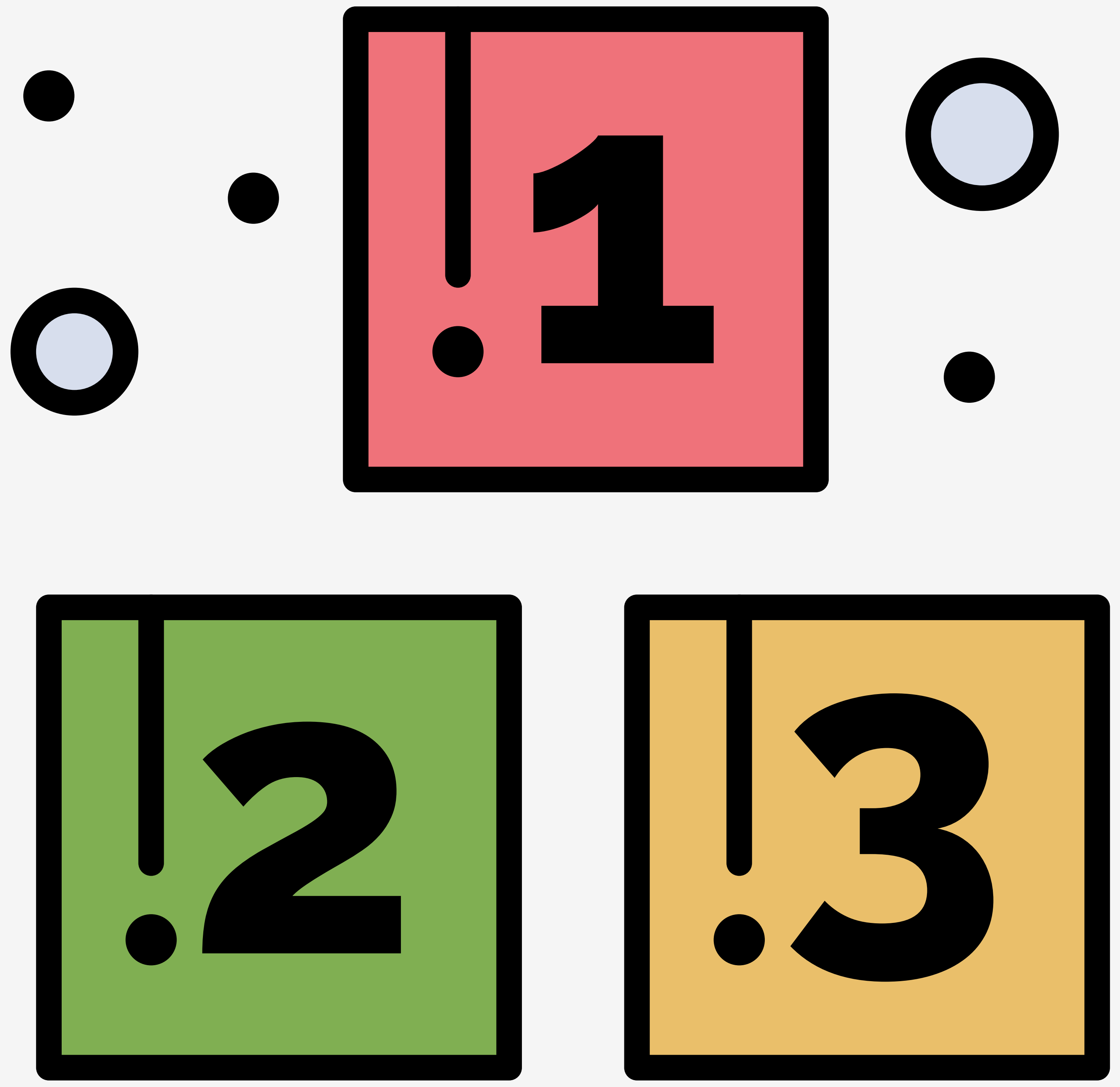
DOCUMENT OBJECT MODEL



- A programming interface for HTML documents
- The DOM describes the HTML as a tree like structure with parents and children
- JavaScript interacts with HTML through the DOM

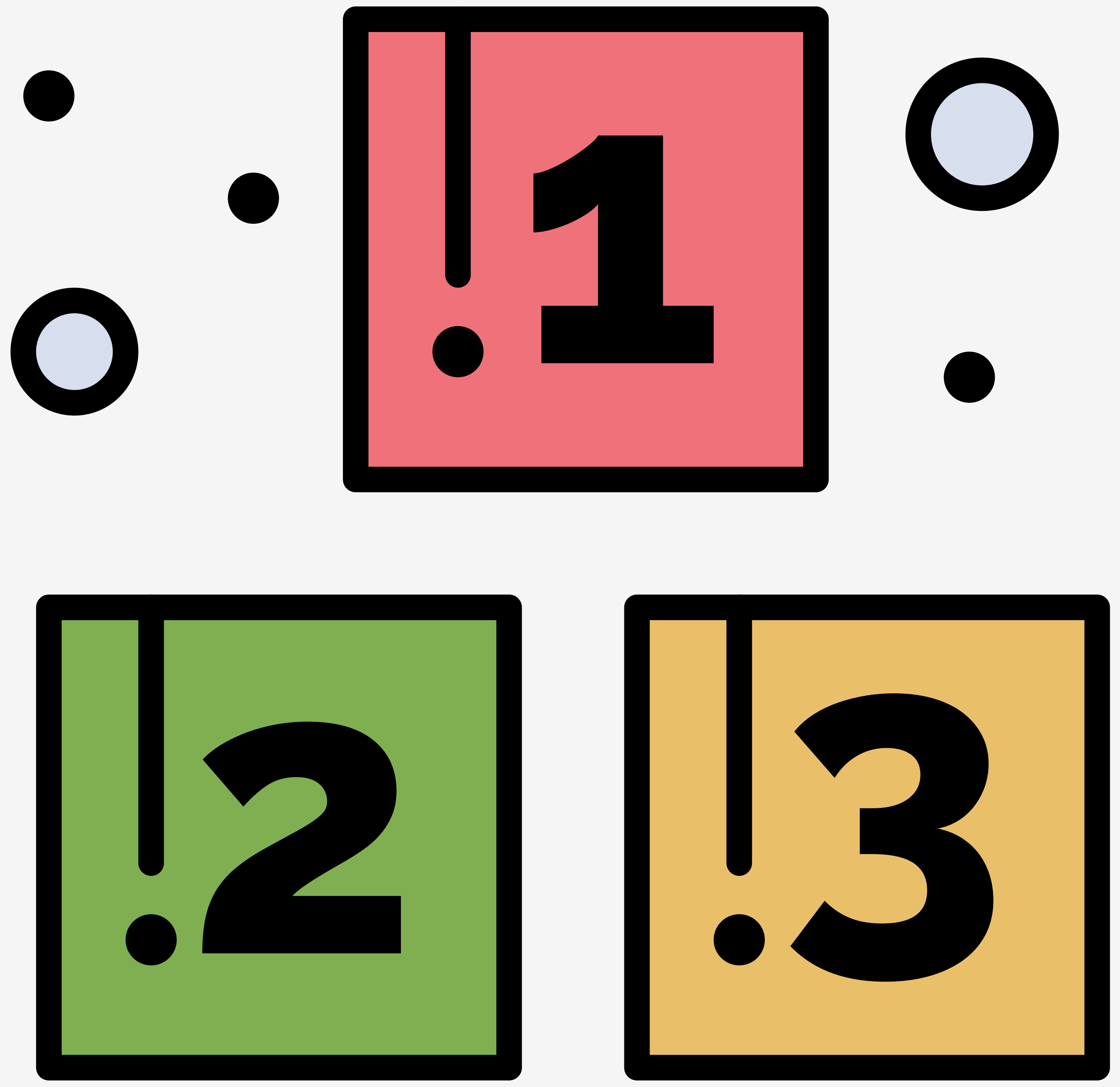
FINDING
DOM ELEMENTS

FINDING DOM ELEMENTS



- JavaScript has many methods for retrieving elements from the DOM
- The most common methods are:
  - `getElementById()`
  - `querySelector()`
  - `querySelectorAll()`

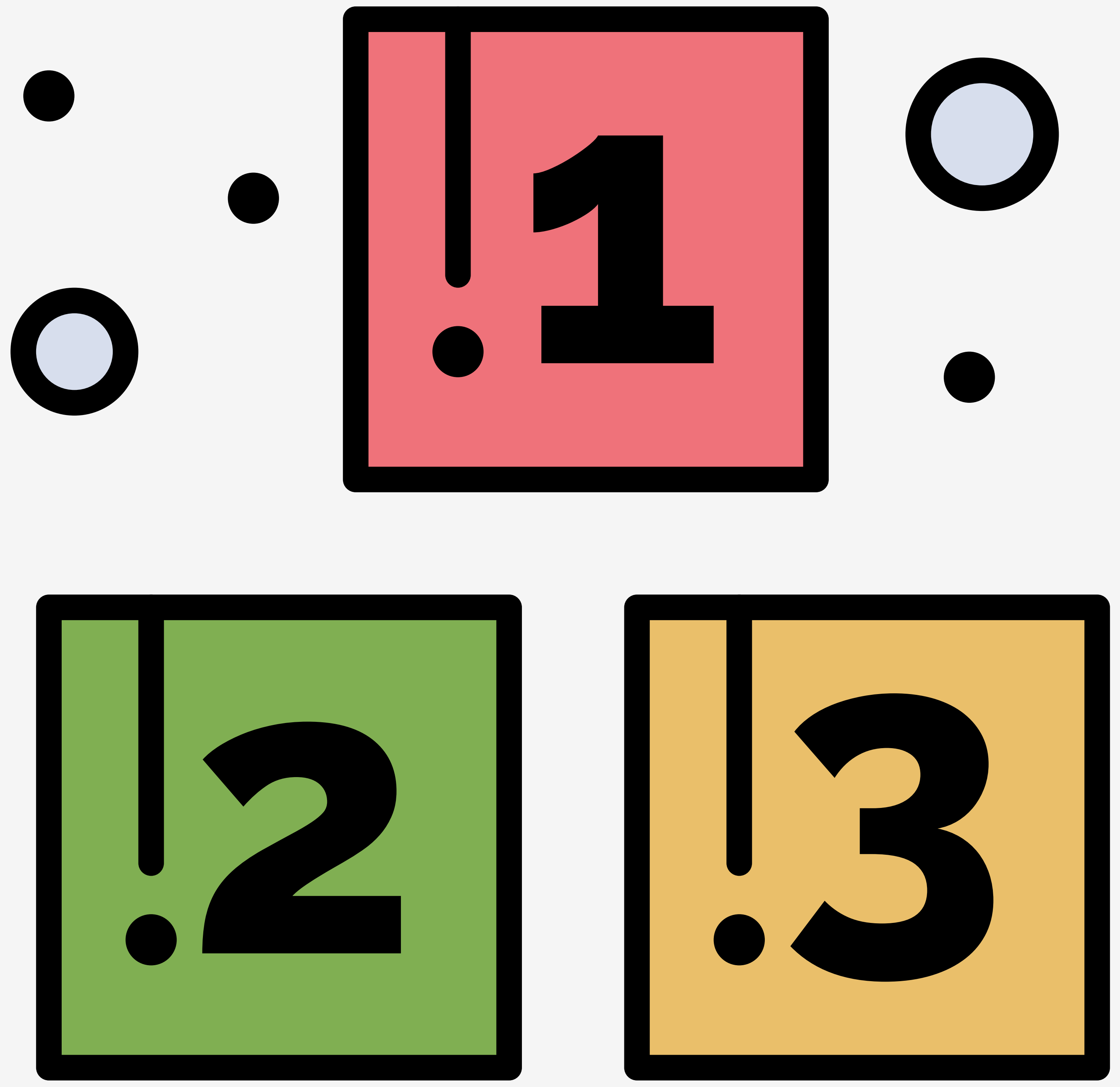
FINDING DOM ELEMENTS



- The `getElementById()` method must be used with the `document` object
- It will find the first element with a matching `id`
- The `getElementById()` will return an Element object or `null`
- The `getElementById()` method is the preferred way to find elements

```
// finds first element with an id of box
const $box = document.getElementById('box')

console.log($box) // Element Object
```

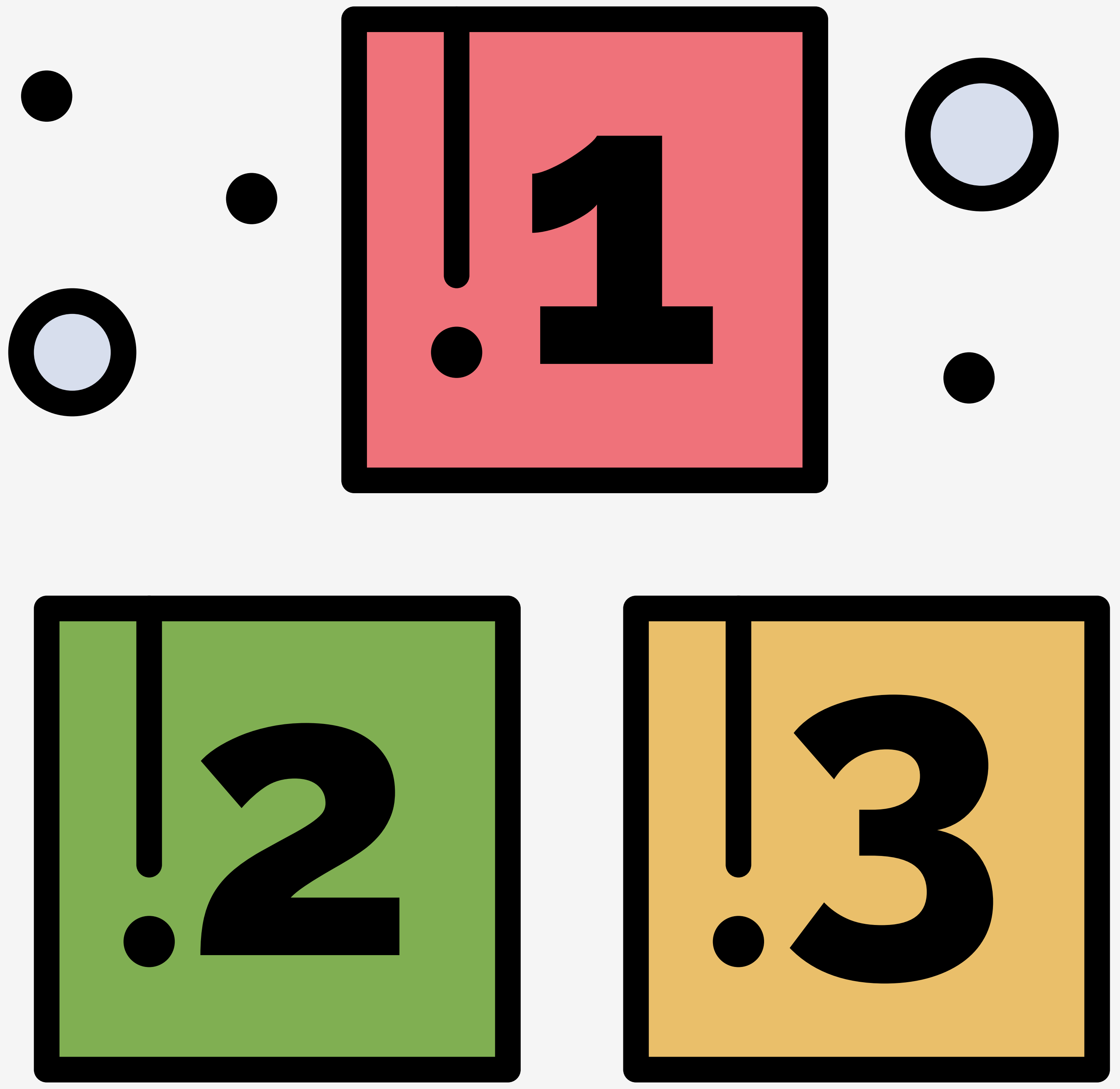
FINDING DOM ELEMENTS



- The `querySelector()` method can be used with the `document` object or an element object
- It will find the first element with a matching CSS Selector
- The `querySelector()` will return an Element object or `null`

```
// Find element by tag name
const $header = document.querySelector('header')
console.log($header) // Element Object

// Find element by class name
const $button = document.querySelector('.button')
console.log($button) // Element Object

// Find element by id
const $box = document.querySelector('#box')  Bad!
console.log($button) // Element Object
```

FINDING DOM ELEMENTS

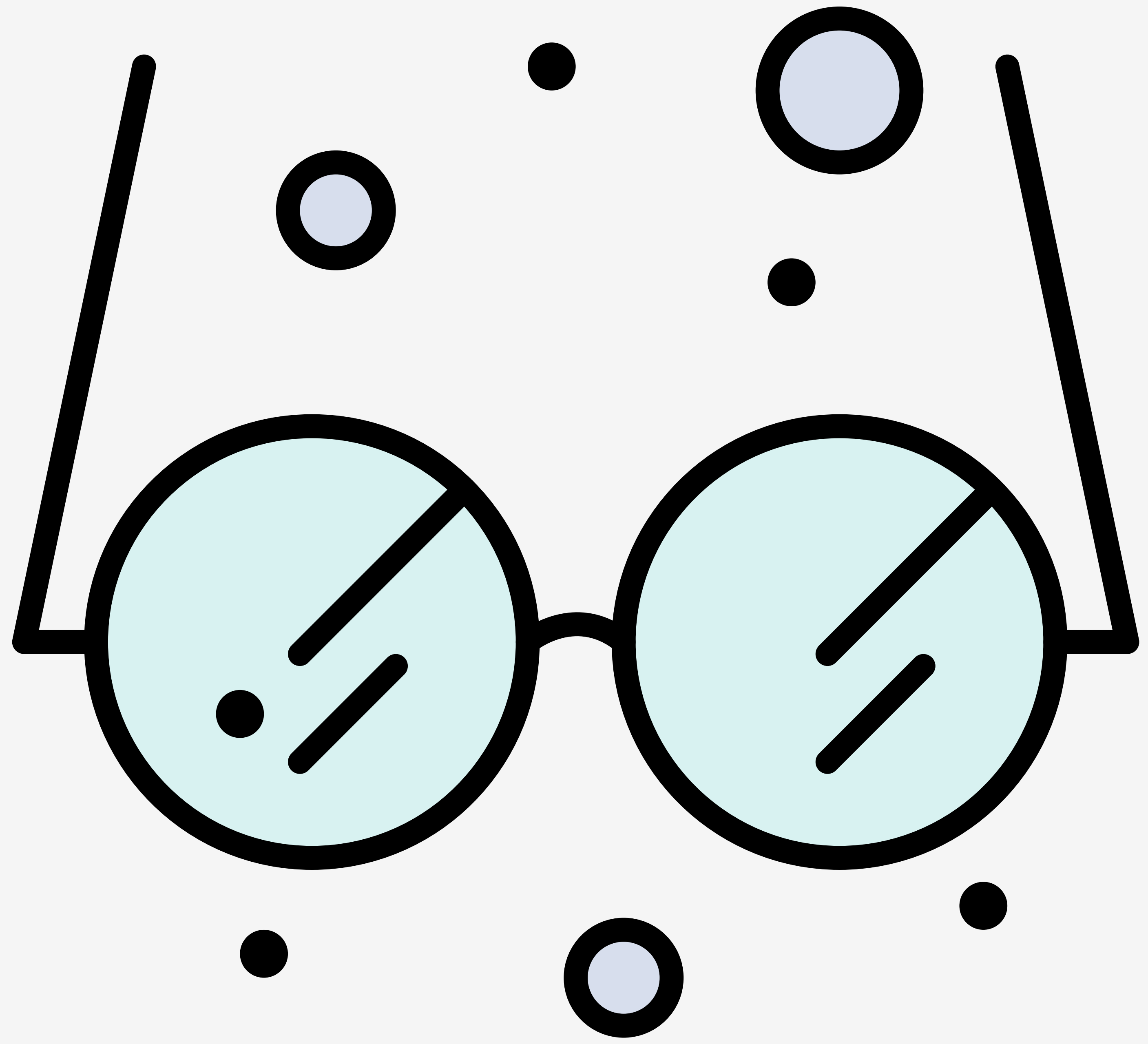


- The `querySelectorAll()` method can be used with the `document` object or an element object
- It will find the *ALL* elements with a matching CSS Selector
- The `querySelectorAll()` will return an NodeList
- A NodeList is an array like object containing element objects

```javascript
// Find elements by class name
const $buttons = document.querySelectorAll('.button')
console.log($buttons) // NodeList

for (const button of $buttons) {
  console.log(button.textContent)
}
```

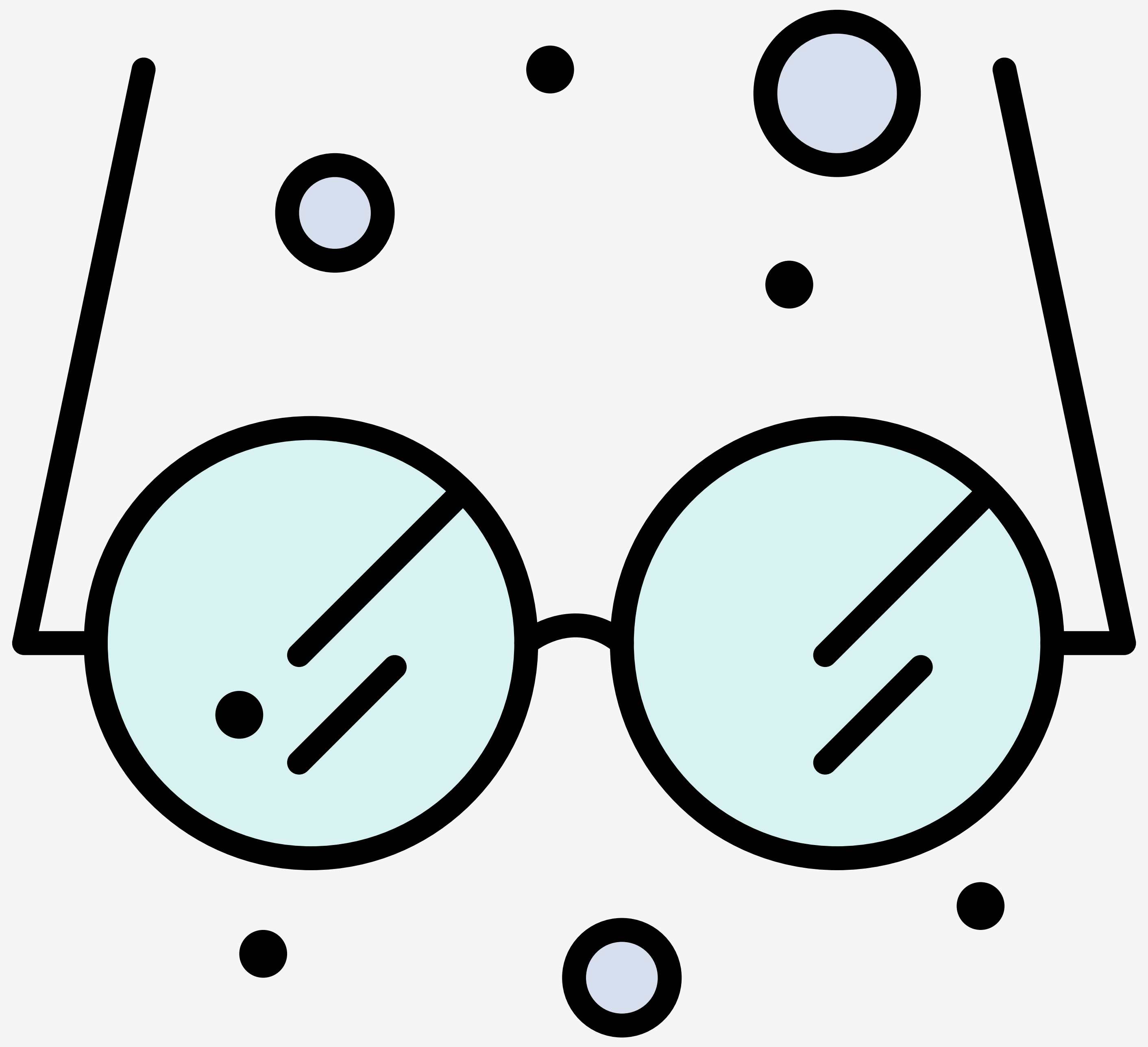
TRAVERSING THE DOM

TRAVERSING THE DOM



- It is possible to move through the DOM from a specific Element
- The following methods and properties allow for the traverse the DOM
  - `children`
  - `firstElementChild` / `lastElementChild`
  - `nextElementSibling` / `previousElementSibling`
  - `closest()`

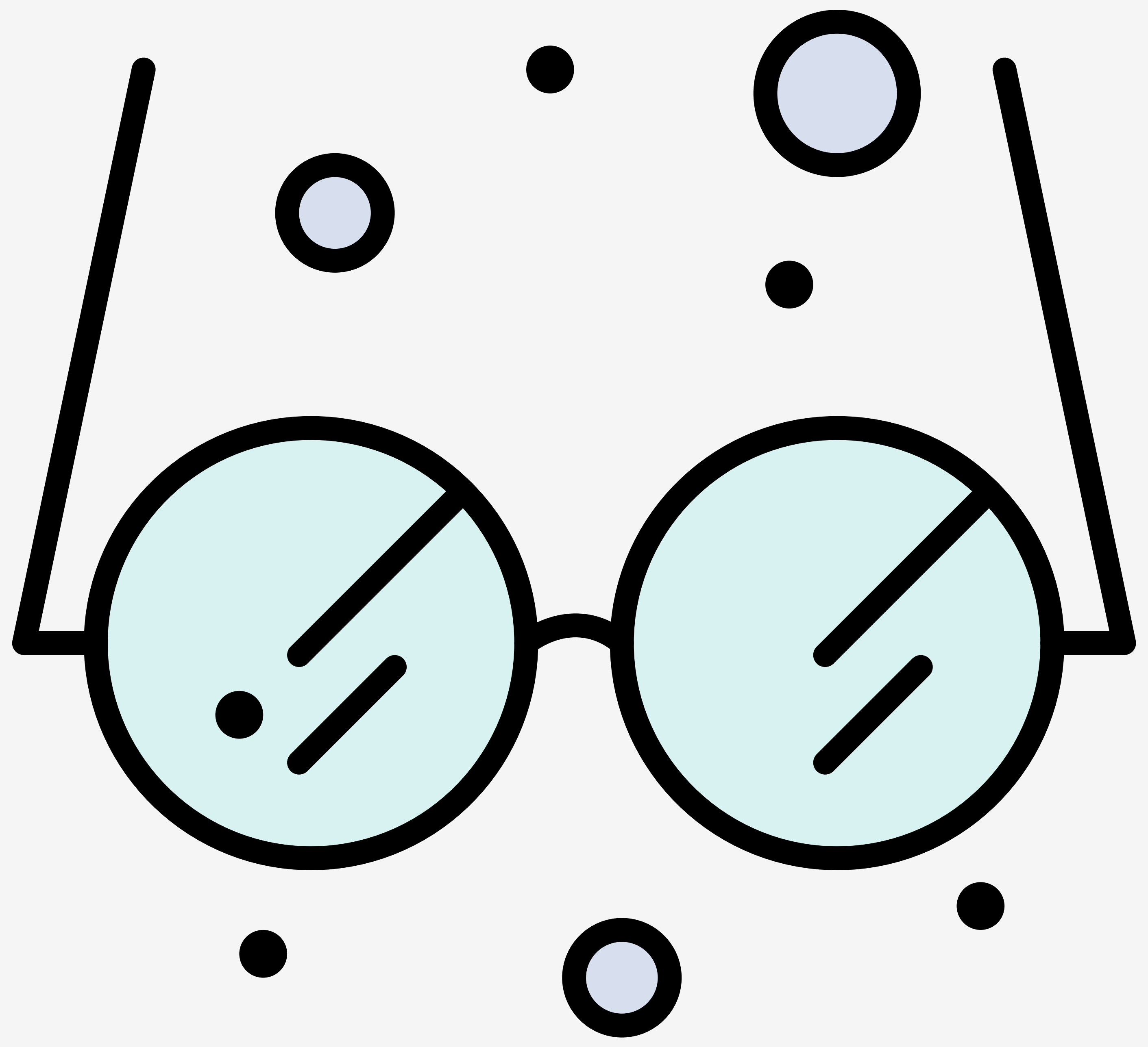
TRAVERSING THE DOM



- The `children` property contains all the child elements of a target element
- The `children` property is read-only
- The `children` property will return a HTMLCollection
- An HTMLCollection is an array-like object of Element objects

```javascript
const $list = document.getElementById('#list')
const items = $list.children // HTMLCollection

for (const item of items) {
  console.log(item.textContent)
}
```

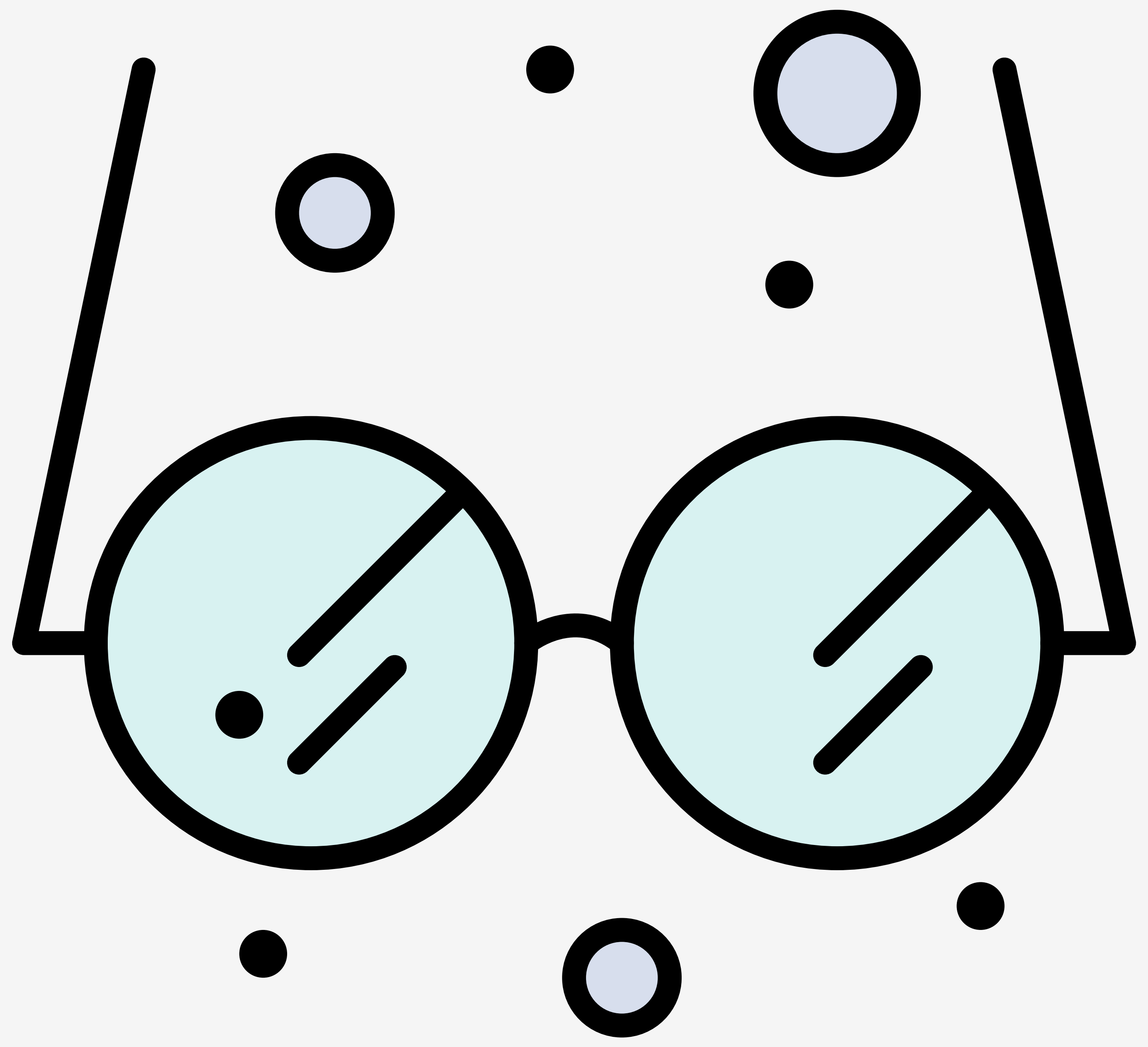
TRAVERSING THE DOM



- The `firstElementChild` property returns the first element of the target element; `null` if no elements
- The `lastElementChild` property returns the last element of the target element; `null` if no elements
- The `nextElementSibling` property returns the next sibling of the target element; `null` if last element
- The `previousElementSibling` property returns the previous sibling of the target element; `null` if first element

```
const $list = document.getElementById('#list')

const firstItem = $list.firstElementChild
const lastItem = $list.lastElementChild
const secondItem = firstItem.nextElementSibling
```

TRAVERSING THE DOM

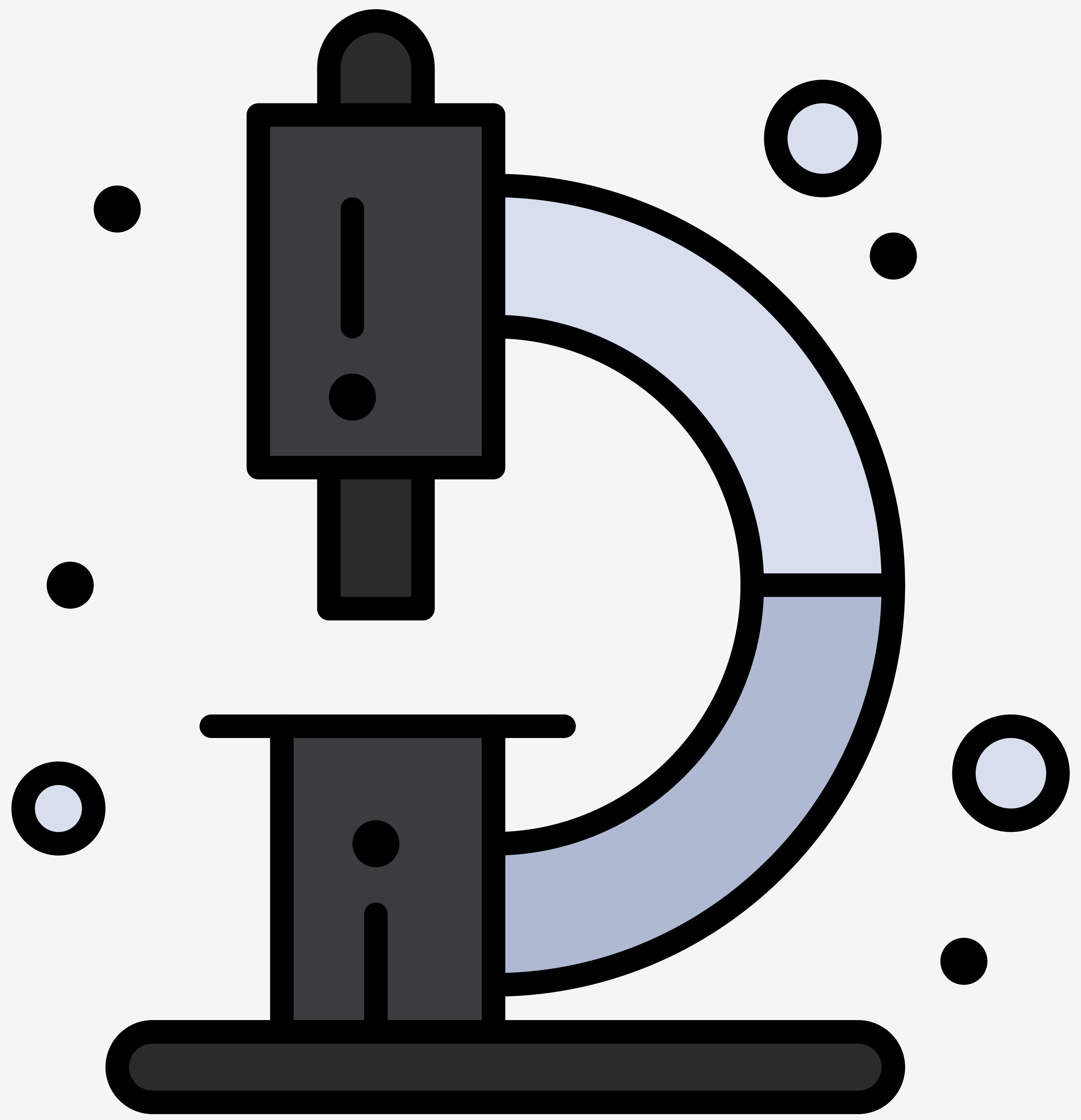


- The `closest()` methods traverses parents of an element
- Will return the first element that matches the CSS Selector, including the target element
- If not element is found `null` is returned
- The `closest()` method is useful when working with events

DEMO

MANIPULATING ATTRIBUTES

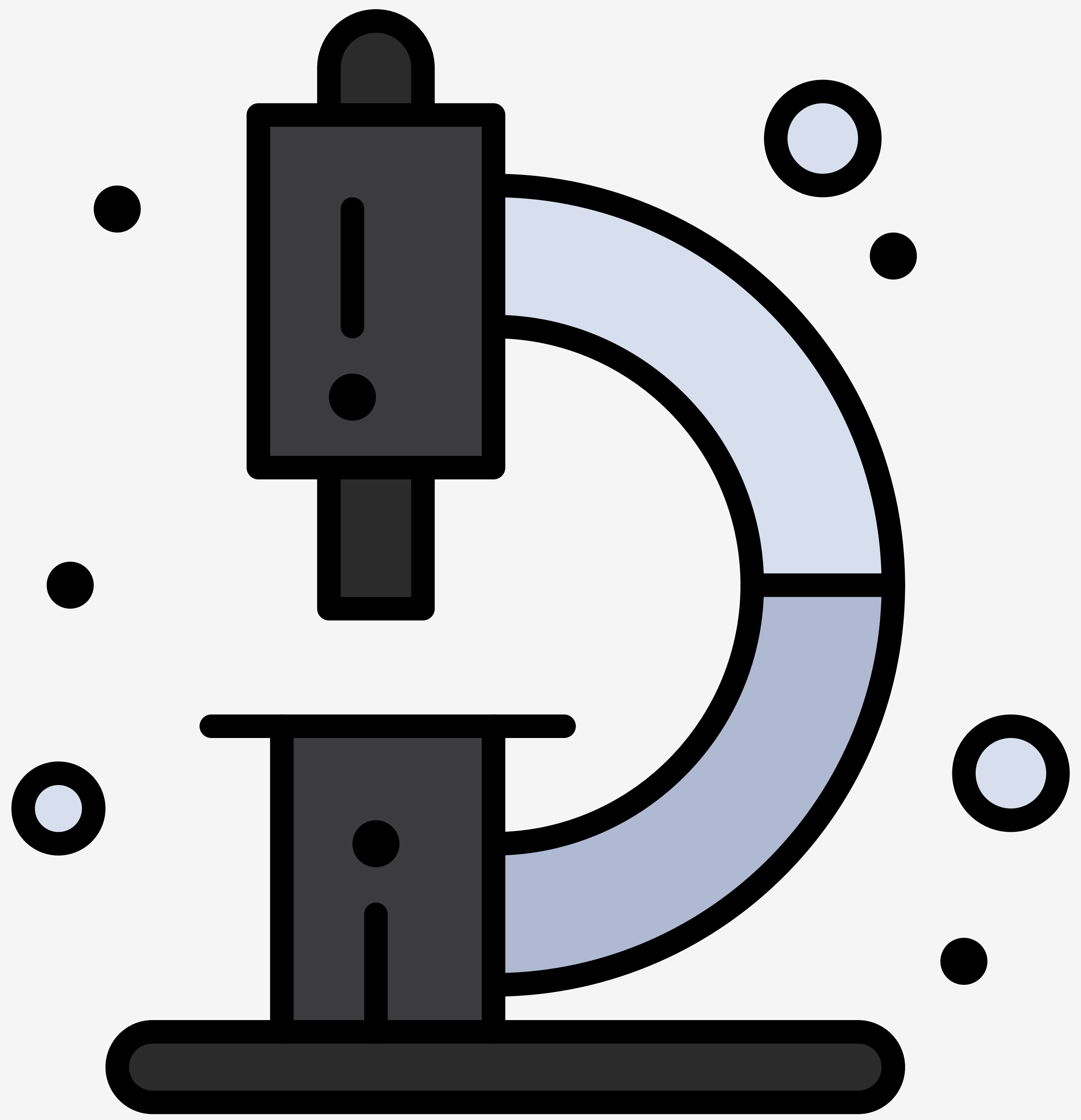
MANIPULATING ATTRIBUTES



- After retrieving an DOM Element, it is possible to read, add, remove or change the element's attributes, classes, or text

- Manipulating Attributes can be done using methods or properties.

MANIPULATING ATTRIBUTES

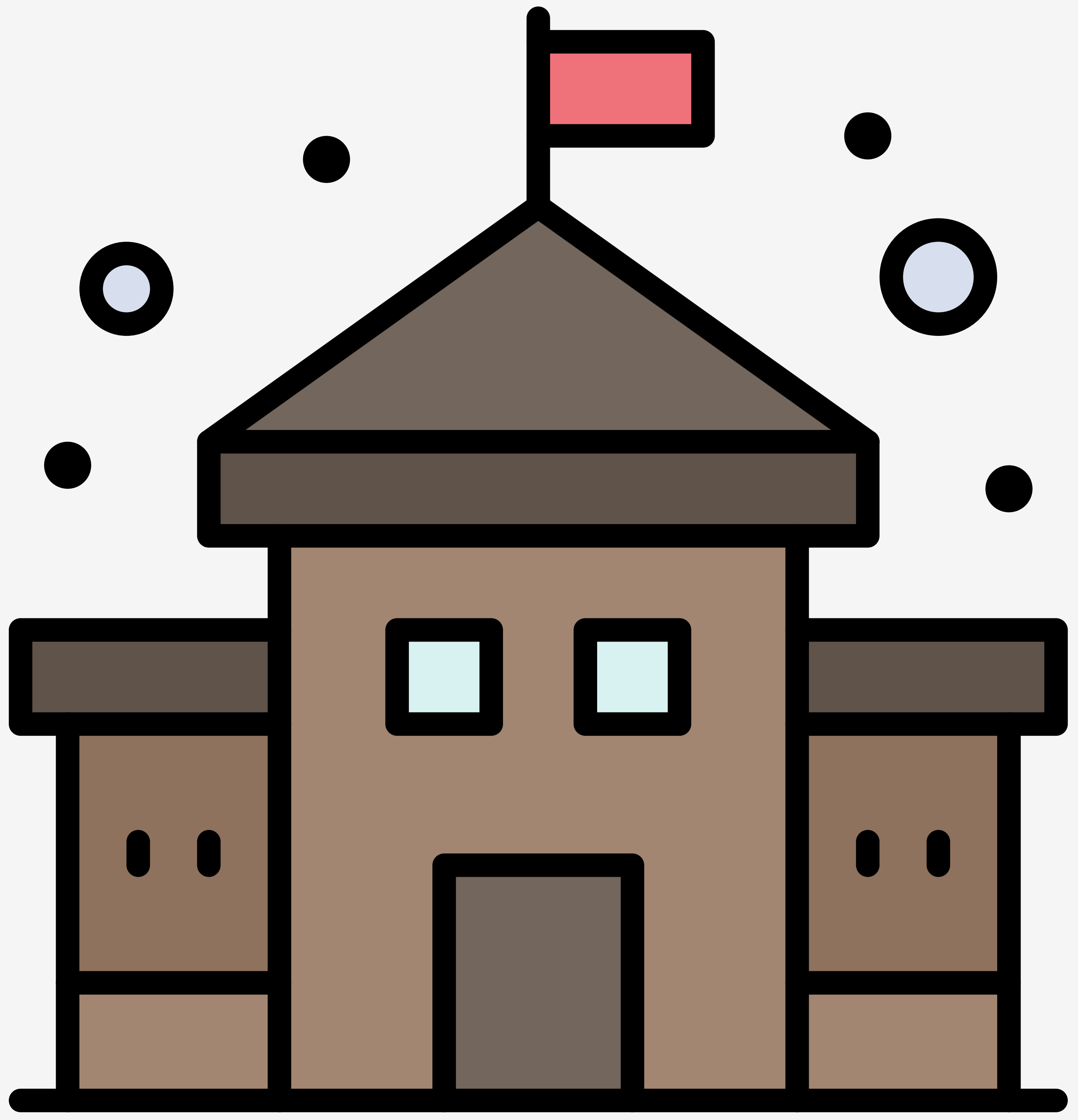


- Most standard attributes have a corresponding property in the Element object
- These properties can be access using dot or bracket notation.
- The methods include:
  - `getAttribute()`
  - `setAttribute()`
  - `removeAttribute()`

```javascript
const $link = document.getElementById('link')

// Reading the id attribute
console.log($link.getAttribute('id')) // link
console.log($link.id) // link

// Setting the href attribute
$link.setAttribute('href', 'https://google.ca')
$link.href = 'https://google.ca'

// Removing the target attribute
$link.removeAttribute('target')
```

MANIPULATING CLASSES

MANIPULATING CLASSES



- Manipulating classes is different than other attributes
- There are two properties for manipulating classes `className` and `classList`
- The classList property contains the following methods:
  - `add()`
  - `remove()`
  - `toggle()`

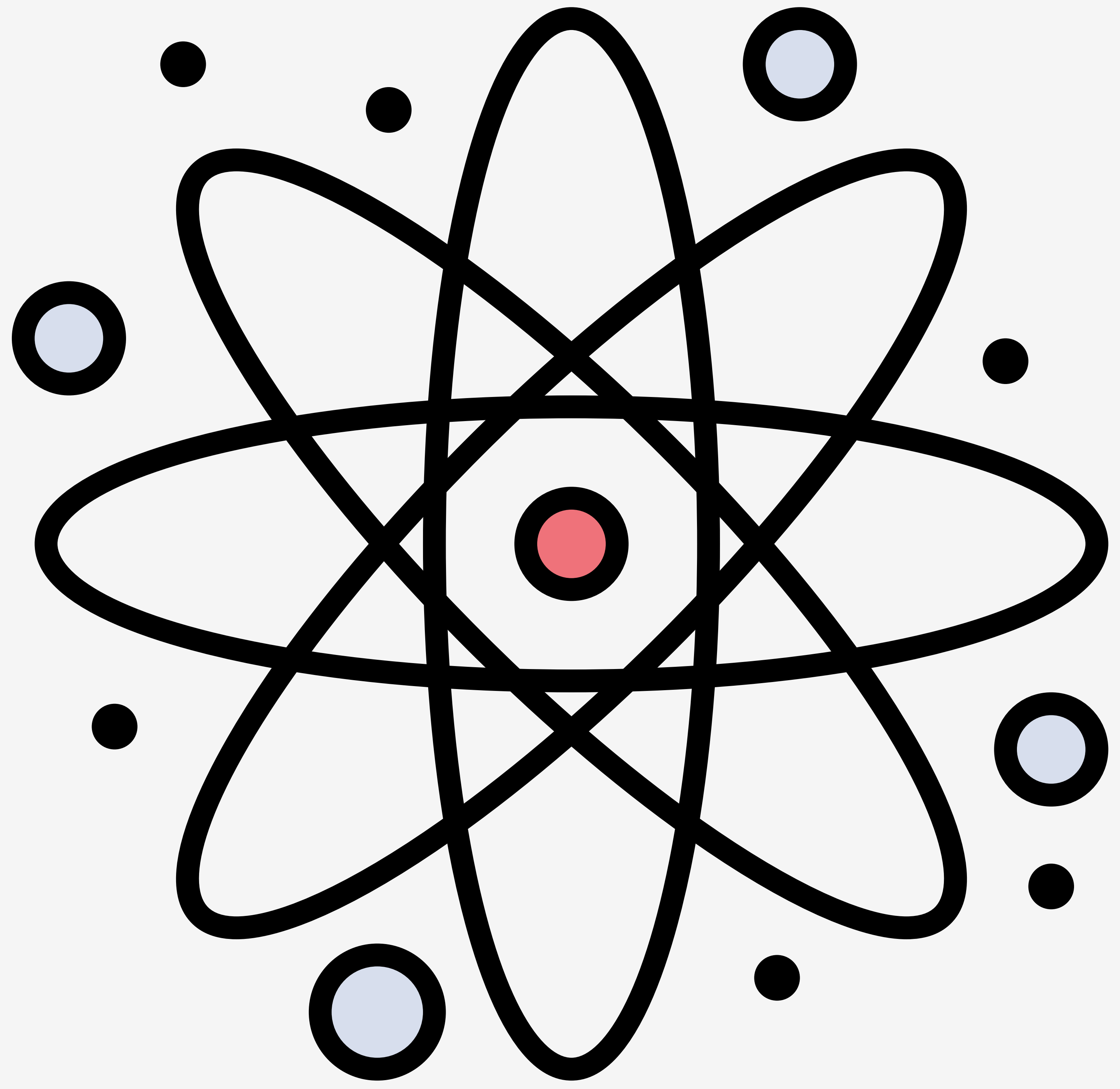
```javascript
const $link = document.getElementById('link')

// Reading the class attribute
console.log($link.className) // link

// Updating the class attribute
$link.className = 'red active'
console.log($link.className) // red active
```

```js
const $link = document.getElementById('link')

// Reading the class attribute
console.log($link.className) // link

// Adding classes
$link.classList.add('red', 'active')
console.log($link.className) // link red active

// Removing classes
$link.classList.remove('active')
console.log($link.className) // link red

// Toggling classes
$link.classList.toggle('active')
console.log($link.className) // link red active
```

DEMO

MANIPULATING MULTIPLE ELEMENTS

MULTIPLE ELEMENTS



- The `querySelectorAll()` method returns a NodeList
- A NodeList does not have access to properties or methods
- A loop must be used to retrieve each element in a NodeList in order to manipulate

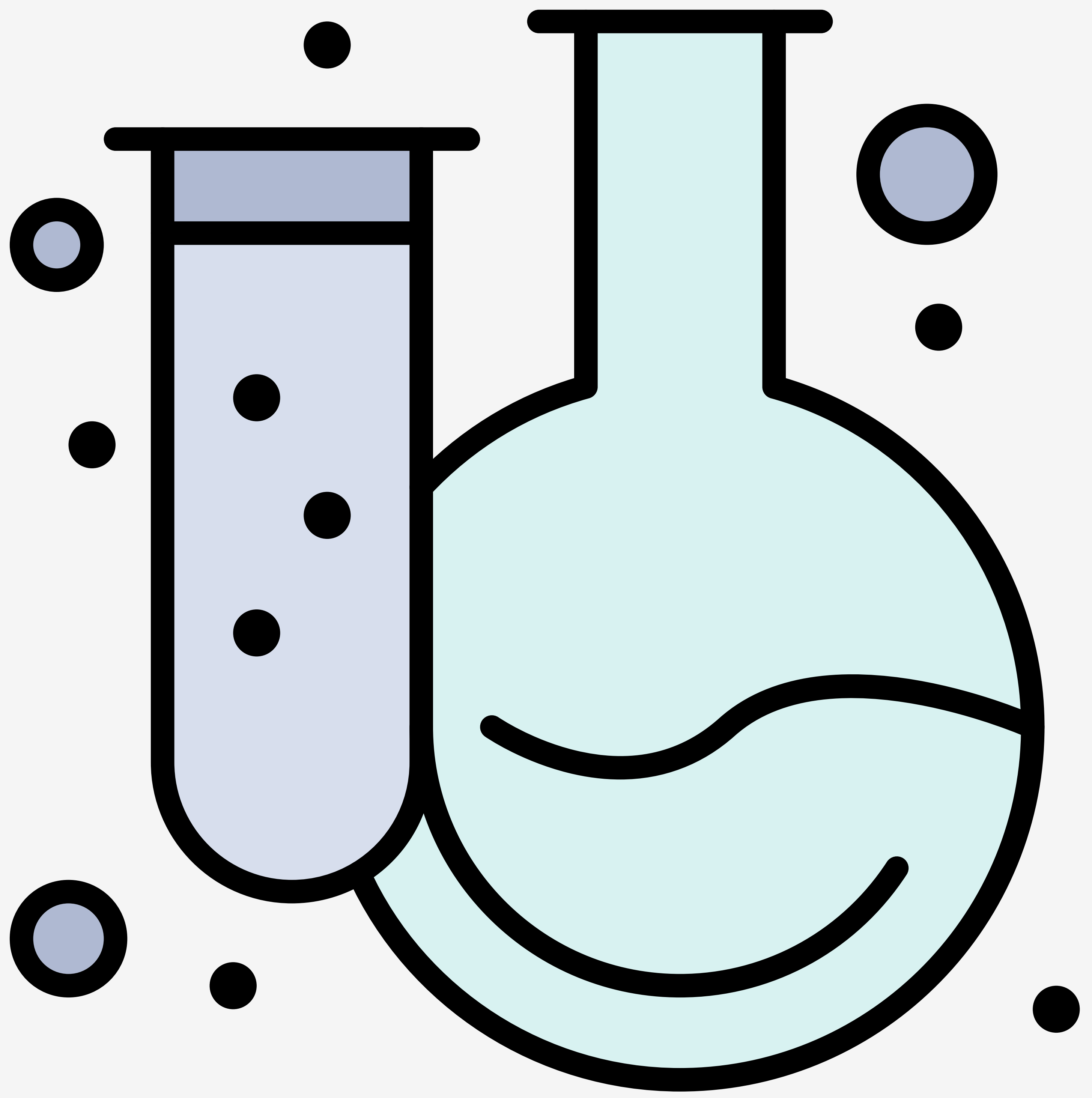
```
const links = document.querySelectorAll('.link')

links.target = '_blank'
links.classList.add('red')

for (const link of links) {
  link.target = '_blank'
  link.classList.add('red')
}
```

Error!

DEMO

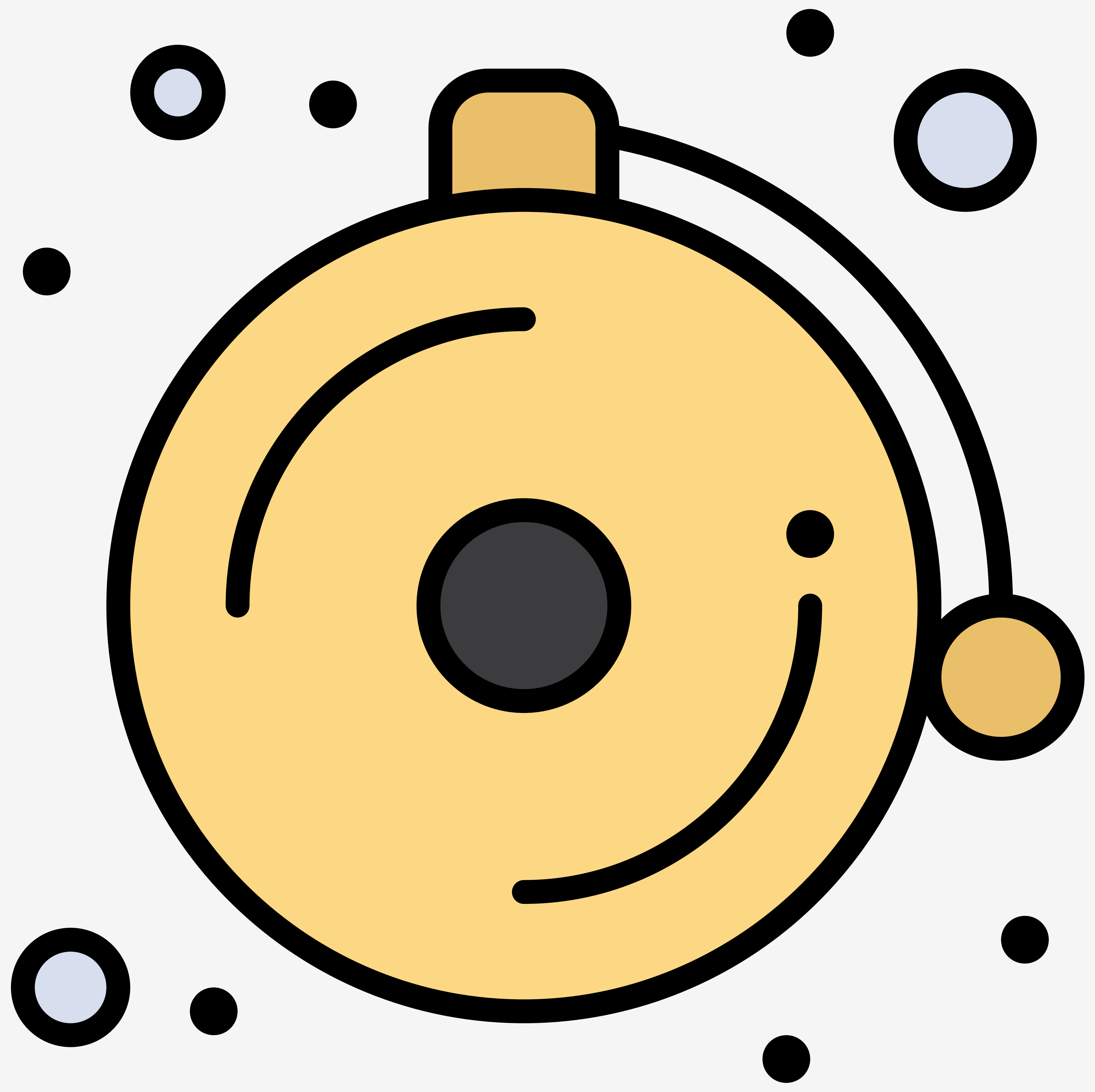
PRACTICE

FUN FACTS - NOT GRADED



- Practice DOM Manipulation by reviewing Fun Facts

NEXT TIME...



- Creating DOM Elements
- Functional Fishing and Complete Autocomplete due *TONIGHT*