04. Javascript (Part2)

Group 1	Tingwei	Liu
	Haoyu	Li
	Shaolong	Li
Group 2	Sijun	Hua
	Weihang	Guo
	Chieh Jui	Lee
	Weiren	Feng
Group 3	Chunjingwen	Cui
	Huimin	Mu
	Rongwei	Ji
Group 4	Jiahao	Liang
	Lingyu	Xu
	Di	Wang
Group 5	Ziyue	Fan
	Fangqin	Li
	Ruiqi	Wang
	Ruohan	Wang
Group 6	Haozhong	Xue
	Kanghong	Zhao
	Chia Hsiang jia xiang	Wu
	jingwei	Ma

我 css 遇到两个问题，可能都不算大问题，但是想请教您的意见，等您有空再回答：

1. 公司里面一般 class 的命名习惯是怎么样的
2. 怎么决定一个 container 是多少 px 因为屏幕大小不一样，可能同样的 p 在大屏幕上显示出来占比就和小屏幕不一样

1. 我想问一下在 js 里面的双引号 “” 和单引号 ‘’ 有什么区别吗?
2. js 的 map object 是不是有点像 python 的 dict？在实际的工作应用中多用于什么场景呀?
3. 我这边如果周末学有余力，您觉得我可以自己额外在那个方向 dig deep 一点？是多刷一点前端的题？还是找一下 udemy 上的 html 的课程看一下?
4. 您经常说的前端的一些面试题（类似给你个 data 让你 display 出来），这种题如果我想多练习一些应该去哪里找?

Attention!

This training is **interview-driven**.

To become a qualified developer, *ChatGPT* and *YouTube* will always be your best friends.

Outline

- Strict Mode
- Destructuring
- Type Coercion vs Type Conversion
- == vs ===
- Function
- Dom
- Closure
- Currying

strict mode

- JavaScript in strict mode does not allow variables to be used if they are not declared
  - ES6 modules(import/export) are strict mode **by default**
- Some effects
  - Prevents the use of **undeclared** variables
  - Makes code more **secure** and easier to debug
  - Disallows duplicate parameter names

https://www.w3schools.com/js/js_strict.asp

Destructuring

- Extract values from arrays or properties from objects and assign them to variables in a single, clean statement
  - Array destructuring
    - Values are assigned by **position** (not by name)
    - Skipping Elements
    - Default Values when the value is undefined
    - **Swapping Variables**
  - Object Destructuring
      - Matching by property **name** (not position)
      - Renaming Variables
      - Default values if property is missing
      - Nested Destructuring
  - Destructuring with Rest (...)

Type Coercion vs Type Conversion

- JavaScript is a **weakly-typed** language
- **Type Coercion (Implicit)**: JavaScript automatically changes types when performing operations
  - when we apply **operators** to two different types of values
  - 1 + '1'
- **Type Conversion (Explicit)**: manually specifies to change the type of one value to another specific type
  - to string
  - to boolean
  - to number

== VS ===

- == (Loose equality):
  - Performs type coercion: Converts operands to the same type before comparing
- === (Strict equality):
  - **No type coercion**: **Checks both value and type**
  - **Safer** and more predictable

Function

- A reusable block of code that performs a specific task
- Function Types
  - Named Function (Function Declaration)
    - **function** keyword
    - **Hoisted** (can be called before it's defined)
  - Function Expression
    - A function assigned to a variable
    - **NOT** hoisted (can’t call before it’s defined)
  - Arrow Function (**ES6**)
    - A shorter syntax using =>
    - **NOT** hoisted

Function

- Immediately Invoked Function Expression (**IIFE**)
  - A function that executes immediately after it’s defined
    - Avoid Global Scope Pollution
- Anonymous Function
  - A function without a name
    - Assigned to a variable
    - Passed as a **callback**

Implicit return vs Explicit return

- Explicit Return
  - Uses the return keyword
  - Required when using {} in arrow functions
  - Allows multiple statements
- Implicit Return
  - automatically returns the expression value
  - Only works with arrow functions and without {}

Type	Syntax	`return` keyword	Works with
Explicit return	`{ return value; }`	✅ Yes	All functions
Implicit return	`expression`	❌ No	Arrow functions only

Closure

- A closure is a function that remembers and can access variables from its outer scope, even after the outer function has finished executing
  - Function + its surrounding scope = closure
- How to fix?
  - Using the **IIFE** solution
  - An IIFE creates a new scope by declaring a function and immediately execute it

```
for (var i = 0; i < 3; i++) {
  setTimeout(() => {
    console.log(i);
  }, 1000);
}
```

```
for (let i = 0; i < 3; i++) {
  setTimeout(() => {
    console.log(i);
  }, 1000);
}
```

Currying

- Currying: break down a function that takes **multiple** arguments into a sequence of functions that each take a **single** argument
  - Currying always takes one argument at a time
  - f(a, b, c) → f(a)(b)(c)
  - reusable, configurable

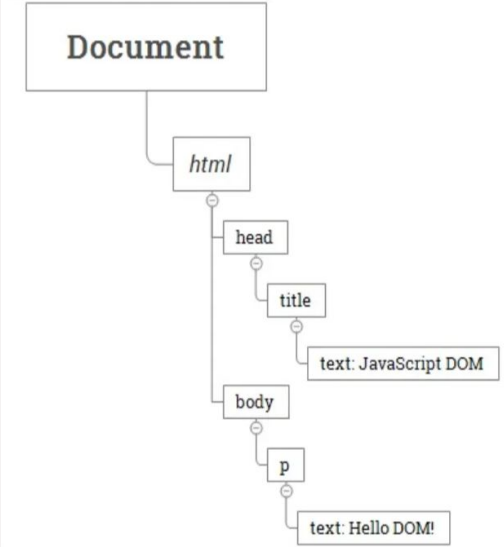
```
function add(a, b, c) {
  return a + b + c;
}
```

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

Document Object Model (DOM)

- a **tree-like** structure that represents the HTML elements of a webpage
- Provides API that allows you to add/remove/modify parts of the document effectively

```
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>JavaScript DOM</title>
    <link>
  </head>
  <body>
    <p>Hello DOM!</p>
    <script src="DOM.js"></script>
  </body>
</html>
```



DOM Element Selectors

- Once you selected an element, you can add styles to the element, manipulate its attributes. (add, remove, modify)
  - document.getElementsBy_
    - document.getElementBy**Id**(idName)
    - getElementsBy**Name**
    - document.getElementsBy**Tag**Name(tag)
    - document.getElementsBy**Class**Name(className)
  - querySelector
    - class(.), id(#), tag…
    - document.querySelector(selector)
      - Returns the first matching element using CSS selector syntax
    - document.querySelectorAll(selector)
      - Returns all matching elements as a NodeList

Create and Remove Elements

- Creating Elements
  - document.createElement(tagName): Creates an element by tagName
  - Set Properties / Content / Style: Customize it
  - .appendChild(): Adds one node to the end
  - .append(): Adds nodes or strings
  - .prepend(): Adds to the start of a node
- Removing Elements
  - .remove(): remove the targeted element
  - .parentNode.removeChild(child): removes a specific child node from the selected element

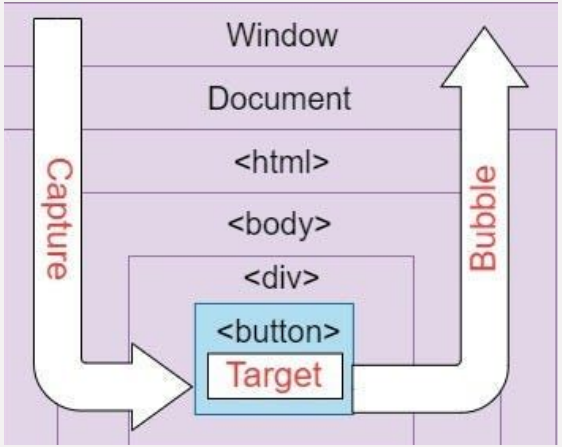
Dom Events

- A DOM event is a signal that something has happened in the document
  - element.**addEventListener**(eventType, function)
  - element.**removeEventListener**(eventType, function)
- https://www.w3schools.com/jsref/dom_obj_event.asp

Event Type	Triggered When...
`click`	Element is clicked
`input`	User types in an input field
`submit`	A form is submitted
`load`	Page or image finishes loading
`keydown`	A key is pressed down
`mousemove`	Mouse moves over an element
`change`	Value of a select/input changes

Event Propagation

- Event propagation is the order in which events travel through the DOM tree when an event occurs on an element
- 3 phases
  - **Capturing**: Top → Target, event travels from the root down into the target element
  - **Target**: event has reached the target element
  - **Bubbling**: Target → Top, event “bubbles up” and travels outward

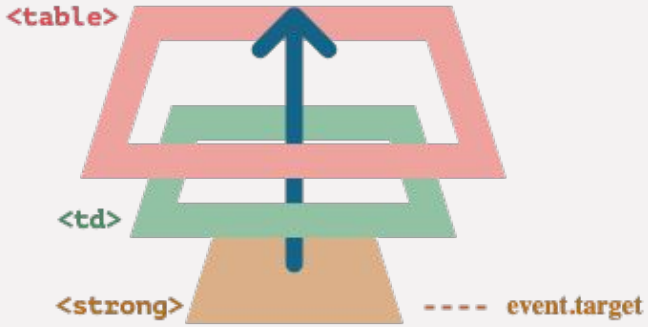


stopPropagation() vs preventDefault()

- stopPropagation()
  - Stop event from bubbling up
    - Prevent parent-level listeners
- preventDefault()
  - Override (Do not perform) the **default** behavior associated with this event
  - Default behaviors:
    - Clicking a link: navigates to another page
    - Submitting a form: reloads the page

Event Delegation

- A technique that add a single event listener to a **parent** element instead of **individual child elements**
  - Attach a listener to a parent
  - Inside that listener, you check **event.target** to determine which child actually triggered the event
- Scenarios: lists, tables, menus, buttons in groups, etc.



Additional materials

- https://www.w3schools.com/Js/

Homework - Requirement

- 全程recording: 八股闭眼 & 摘耳机, 无AI辅助
- 全程recording: Coding可适当AI辅助, 但需写完(边解释边写), 模拟真实面试情景
- 录音提交地址: https://drive.google.com/drive/folders/1Mrm341uOIS8c2Ty6NwYvvSybHa6hESem?usp=drive_link
- 提交格式: 小组+日期, 例如Group1_12/07/2025
- 提交截止时间: due第二天5PM (周四作业递延到下周一)

Homework - Class Code

- Write the code as taught in class, take a screenshot of your code, and post it in the Wechat group. **It's due the same night.**

Homework - Short Answer Questions

1. What is the difference between == and ===?
2. What is type coercion, and how does it differ from type conversion?
3. What are the different types of functions in JavaScript? Can you explain their syntax
4. What is an IIFE (Immediately Invoked Function Expression), and when would you use one?
5. Can you explain how hoisting affects function declarations and function expressions?
6. What is destructuring in JavaScript? Explain what is array destructuring and object destructuring
7. Can you name the new ES6 features by now?
8. What is the DOM?
9. How can you select an HTML element using JS?
10. What is a DOM event?

Homework - Short Answer Questions

11. How do we register event handlers for a selected element?
12. Explain event propagation. How many phases are there? In what order does it occur? What do you do to prevent propagation?
13. Explain event delegation. Why is it important?
14. What is event bubbling and What is event capturing?
15. What is the use of preventDefault method?
16. Are functions hoisted?
17. What are closures? Can you give me an example?
18. What is currying?
19. Explain what is spread operator and what is rest operator
20. What is “use strict”? What are the major effects that it has?

Homework - Coding Questions

1. Destructure this object to extract name and age

const person = { name: 'John', age: 30, city: 'NYC' };

// Your code here

2. Consider the following code snippet. Assume that it works and was imported into a .html file with the proper button IDs.
    a. What is the console output when the user clicks on “Button 3” and why?
    b. How would we fix the issue before ES6? How do we fix it after ES6?

```
for (var i = 1; i <= 3; i++) {
  document.getElementById(`btn${i}`).addEventListener('click', function () {
  console.log(`you just clicked #${i} button`);
}) }
```

Homework - Coding Questions

3. Given the sample UI, implement the following product management page with styling that matches as closely as possible.

- The user can fill out the fields and click “Add New” to create a new product in the table.
- However, if any of the input fields are empty, no product should be created.
- The user can delete existing products in the table by clicking the delete button.
- By default, when the user loads the page for the first time, there should be these 3 items in the table as shown below

Homework - Coding Questions

Product Name	Product Category	Product Price	Action
M&M	Snacks	$1.99	Delete
Table	Furniture	$199	Delete
Kale	Vegetables	$2.49	Delete

Add Product

Product Name: Product Category: Product Price: Add New

Homework - Coding Questions

4. Given the following UI and HTML code, implement the following features, while matching the styling as closely as possible.

- By default, the London tab is selected and only its content is displayed.
- When the user clicks on a tab, it is the only one that becomes highlighted, and only the corresponding content will be displayed.
- Ex: The user clicks ‘Paris’, so only the ‘Paris’ content is displayed.
- Hint:
    - hint1: event delegation is preferred
    - hint2: ele.classList.toggle specific class you’d like to add

Homework - Coding Questions

HTML Part: Feel free to add class and id

```
<div>
  <button>London</button>
  <button>Paris</button>
  <button>Tokyo</button>
</div>
<div>
  <h2>London</h2>
  <p>London is the capital city of England</p>
</div>
<div>
  <h2>Paris</h2>
  <p>Paris is the capital of France.</p>
</div>
<div>
  <h2>Tokyo</h2>
  <p>Tokyo is the capital of Japan.</p>
</div>
```

Homework - Coding Questions