**เฉลยข้อสอบกลางภาค วิชา ENGCE306 Web Programming**

**ข้อ 1. ความเข้าใจพื้นฐานเกี่ยวกับ HTML Tags**

1. **การสร้างหัวข้อเรื่อง (Heading):**
   * **แท็ก: <h1> ถึง <h6>**
   * **หน้าที่: ใช้สำหรับกำหนดหัวข้อเรื่อง โดย <h1> มีความสำคัญที่สุดและขนาดใหญ่ที่สุด ไล่ลำดับความสำคัญและขนาดเล็กลงไปจนถึง <h6>**
2. **การสร้างย่อหน้า (Paragraph):**
   * **แท็ก: <p>**
   * **หน้าที่: ใช้สำหรับกำหนดกลุ่มข้อความให้เป็นหนึ่งย่อหน้า เบราว์เซอร์จะแสดงผลโดยมีการเว้นวรรคก่อนและหลังย่อหน้าโดยอัตโนมัติ**
3. **การสร้างลิงก์ (Link):**
   * **แท็ก: <a> (Anchor Tag)**
   * **หน้าที่: ใช้สำหรับสร้างลิงก์เพื่อเชื่อมโยงไปยังหน้าเว็บอื่น, ไฟล์, หรือส่วนต่างๆ ภายในหน้าเดียวกัน โดยใช้แอตทริบิวต์ href เพื่อระบุ URL ปลายทาง**

**ข้อ 2. การวิเคราะห์ CSS Box Model และ Layout**

* **ความกว้างทั้งหมด (Total Width): ความกว้างทั้งหมดที่ element นี้ใช้ไปบนหน้าจอคือ 330px คำนวณจาก: width (300px) + margin-left (15px) + margin-right (15px) = 330px**
* **องค์ประกอบของความสูงทั้งหมด (Total Height): ความสูงทั้งหมดจะคำนวณจากผลรวมของ: height (auto, ขึ้นอยู่กับความสูงของเนื้อหา) + padding-top (20px) + padding-bottom (20px) + border-top (5px) + border-bottom (5px) + margin-top (15px) + margin-bottom (15px)**
* **ผลของ box-sizing: border-box;: ค่า box-sizing: border-box; ทำให้ค่า width (300px) ที่เรากำหนด เป็นความกว้างที่รวม padding (20px*2) และ border (5px*2) เข้าไว้ด้วยกันแล้ว ซึ่งหมายความว่าพื้นที่สำหรับเนื้อหา (content) จริงๆ จะถูกลดขนาดลงเหลือ 250px (300 - 40 - 10) วิธีนี้ช่วยให้นักพัฒนาสามารถควบคุมขนาดของกล่องได้ง่ายขึ้น เพราะขนาดของกล่องจะไม่ขยายใหญ่ขึ้นเมื่อเราเพิ่ม padding หรือ border ซึ่งแตกต่างจากค่า default (content-box) ที่ width จะหมายถึงความกว้างของ content เท่านั้น**

**ข้อ 3. ความเข้าใจเรื่อง JavaScript Closure และ Function Factory**

* **หลักการของ "Closure": Closure คือคุณสมบัติของฟังก์ชันใน JavaScript ที่ทำให้ฟังก์ชันภายใน (inner function) สามารถ "จดจำ" และ "เข้าถึง" ตัวแปรและสภาพแวดล้อม (scope) ของฟังก์ชันภายนอก (outer function) ที่สร้างมันขึ้นมาได้ แม้ว่าฟังก์ชันภายนอกนั้นจะทำงานเสร็จสิ้นไปแล้วก็ตาม ในตัวอย่างนี้ เมื่อเรียก createGreeter('Hello') ฟังก์ชันภายในที่ถูก return ออกมาจะ "ปิดล้อม" (close over) ตัวแปร greeting ที่มีค่าเป็น 'Hello' ไว้ ทำให้เมื่อเราเรียก helloGreeter('Alice') ในภายหลัง มันยังคงจำได้ว่าต้องใช้คำทักทายว่า 'Hello'**
* **ผลลัพธ์ที่จะแสดงใน Console:**

****

**ข้อ 4. การประยุกต์ใช้ TypeScript Interface**

**รูปภาพประกอบด้วย ข้อความ, เมนู, ตัวอักษร, ภาพหน้าจอ

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง**

**ข้อ 5. การใช้ตัวดำเนินการใน JavaScript**

1. **ความแตกต่างระหว่าง == และ ===:**
   * **== (Loose Equality): เป็นการเปรียบเทียบที่ ตรวจสอบเฉพาะค่า (value) โดยจะมีการแปลงชนิดข้อมูลอัตโนมัติ (Type Coercion) หากชนิดข้อมูลของทั้งสองฝั่งไม่ตรงกัน**
   * **=== (Strict Equality): เป็นการเปรียบเทียบที่ ตรวจสอบทั้งค่า (value) และชนิดข้อมูล (type) จะคืนค่าเป็น true ก็ต่อเมื่อทั้งค่าและชนิดข้อมูลเหมือนกันทุกประการ**
   * **ตัวอย่าง:**

****

**การทำงานของ && (AND) และ || (OR):**

* **&& (AND): จะให้ผลลัพธ์เป็น true ก็ต่อเมื่อ เงื่อนไขทุกตัวเป็น true ทั้งหมด หากมีเงื่อนไขใดเงื่อนไขหนึ่งเป็น false ผลลัพธ์โดยรวมจะเป็น false ทันที**
* **|| (OR): จะให้ผลลัพธ์เป็น true เมื่อ มีเงื่อนไขใดเงื่อนไขหนึ่งเป็น true จะเป็น false ได้ในกรณีเดียวคือทุกเงื่อนไขต้องเป็น false ทั้งหมด**

**ข้อ 6. การใช้ Semantic HTML**

* **โค้ดโครงสร้าง Blog:**

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, เมนู, ตัวอักษร

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง**

**เหตุผลที่ Semantic HTML ดีกว่า <div>:**

1. **ดีต่อ SEO (Search Engine Optimization): Search engines เช่น Google สามารถเข้าใจโครงสร้างและความสำคัญของเนื้อหาในหน้าเว็บได้ดีขึ้น เช่น รู้ว่า <header> คือส่วนหัว และ <main> คือเนื้อหาหลัก ทำให้จัดอันดับการค้นหาได้แม่นยำขึ้น**
2. **ดีต่อการเข้าถึง (Accessibility): โปรแกรมอ่านหน้าจอ (Screen Reader) สำหรับผู้พิการทางสายตาสามารถใช้แท็กเหล่านี้ในการนำทางได้ เช่น สามารถสั่งให้ข้ามไปอ่านเนื้อหาหลักในแท็ก <main> ได้ทันที**

**ข้อ 7. การจัดการ DOM ด้วย JavaScript**

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร, เมนู

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง**

**ข้อ 8. การสร้าง Class ใน TypeScript**

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร, เอกสาร

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง**

**ข้อ 9. การประยุกต์ใช้ CSS Selectors**

1. **เลือก <h2> ภายใน .card:**

**รูปภาพประกอบด้วย ตัวอักษร, ข้อความ, ภาพหน้าจอ, วิชาการพิมพ์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง**

1. **เลือก div ที่มีทั้ง class .card และ .featured:**

**รูปภาพประกอบด้วย ตัวอักษร, ข้อความ, ลายมือ, วิชาการพิมพ์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง**

1. **เลือก <a> ที่มี target="\_blank":**

**รูปภาพประกอบด้วย ข้อความ, ตัวอักษร, วิชาการพิมพ์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง**

**ข้อ 10. การวิเคราะห์และประยุกต์ใช้ Array Methods**

**รูปภาพประกอบด้วย ข้อความ, ตัวอักษร, ภาพหน้าจอ

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง**

**คำอธิบายการทำงาน:**

1. **.filter(emp => emp.title === 'Senior' && emp.isActive): ขั้นตอนนี้จะกรอง array employees และสร้าง array ใหม่ที่บรรจุเฉพาะพนักงานที่มีคุณสมบัติ title เป็น 'Senior' และ isActive เป็น true ซึ่งจะได้ผลลัพธ์เป็น array ที่มี object ของ Alex และ Dana**
2. **.map(emp => emp.salary): ขั้นตอนนี้จะนำ array ที่ได้จากการ filter มาแปลงค่า โดยจะดึงเอาเฉพาะค่าของ salary จากแต่ละ object ออกมาสร้างเป็น array ใหม่ ซึ่งจะได้ผลลัพธ์เป็น [90000, 120000]**
3. **.reduce((sum, currentSalary) => sum + currentSalary, 0): ขั้นตอนสุดท้ายจะทำการคำนวณผลรวมของตัวเลขทั้งหมดใน array ที่ได้จาก .map() โดยเริ่มต้นค่า sum ที่ 0 แล้วบวกค่าเงินเดือน (currentSalary) ของแต่ละคนเข้าไปจนครบ ผลลัพธ์สุดท้ายคือ 210000**