Course: ____________ Name: _____________________ Date: ___________

Part I: DOM Basics (7)

0. What is the `document` object?

[ ] The elements tab in chrome developer tools

[ ] An HTML file

[ ] A javascript object representing the content of a page

[ ] A javascript data type for very long strings

1. The `document` object, like all other global variables, is a property of which object?

[ ] body

[ ] window

[ ] location

[ ] process

2. What object does the `getElementById` method exist on?

[ ] Any javascript object

[ ] window

[ ] document

[ ] addEventListener

Example A:

```
const shoppingList1 = document.querySelector('#shopping-list');
const shoppingList2 = document.getElementById('shopping-list');
```

3. In the example above: shoppingList1 and shoppingList2 are the same thing

[ ] true [ ] false

4. What does Element.querySelector() return?

[ ] HTMLElement Object

[ ] NodeList Object

[ ] HTML String

[ ] Boolean

5. What is a difference between the querySelector and querySelectorAll methods?

[ ] There is no difference between them

[ ] querySelector returns all matches in the document querySelectorAll returns the first match

[ ] querySelector returns the first match in the document querySelectorAll returns all matches

[ ] querySelector returns a CSS Selector in the document querySelectorAll returns a list of CSS selectors

Example B:

```
<body>
  <ul id="shopping-list">
    <li class="shopping-item">Milk</li>
    <li class="shopping-item">Eggs</li>
    <li class="shopping-item">Butter</li>
    <li class="shopping-item">Flour</li>
  </ul>
</body>
```

6. How would you select all list items in the list from the example above?

[ ] document.body.querySelectorAll('#shopping-list li')

[ ] document.body.querySelector('#shopping-list li')

[ ] document.body.querySelectorAll('#shopping-list .shopping-item')

[ ] Both (0) and (2)

Part II: DOM Manipulation (8)

0. Creating an element with the createElement method makes that element appear on the page

[ ] true [ ] false

1. What Element object property can you use to change the entire value of an element's class attribute?

[ ] class

[ ] [✓ className]

[ ] [✓ classList]

[ ] None of the above

2. The Node.insertBefore() method takes only **one argument:**

[ ] true [ ] [✓ false]

3. Which method can be used to remove an element from the DOM?

[ ] [✓ Node.removeChild()]

[ ] Node.removeElement()

[ ] Node.deleteElement()

[ ] Node.removeNode()

4. To retrieve user input from a form control (input, textarea, select) you can use:

[ ] [✓ The `.value` property]

[ ] An element’s dataset

[ ] The `.textContent` property

[ ] There is no way to retrieve the user input without sending the fform to a server

Example C:

```
const fibonnaci = [0, 1, 1, 2, 3, 5, 8, 13, 21];

const div = document.createElement('div');

div.style['background-color'] = 'black';
div.style['color'] = 'white';
div.style['font-family'] = 'monospace';

fibonnaci.forEach(number => {
  let span = document.createElement('span');
  let text = document.createTextNode(number + ' ');

  span.style['font-size'] = `${number}px`;

  span.appendChild(text);
  div.appendChild(span);
});

document.body.appendChild(div);
```

5. What HTML will be added to the page as the result of example C?

[ ]
```
<div>
  <span>0 </span>
  <span>1 </span>
  <span>1 </span>
  <span>2 </span>
  <span>3 </span>
  <span>5 </span>
  <span>8 </span>
  <span>13 </span>
  <span>21 </span>
</div>
```

[ ]
```
<div style="background-color: black; color: white; font-family: monospace;
font-size: 21px;">
  <span>0 </span>
  <span>1 </span>
  <span>1 </span>
  <span>2 </span>
  <span>3 </span>
  <span>5 </span>
  <span>8 </span>
  <span>13 </span>
  <span>21 </span>
</div>
```

[ ]
```
<div style="background-color: black; color: white; font-family: monospace;">
  <span style="font-size: 0px;">0 </span>
  <span style="font-size: 1px;">1 </span>
  <span style="font-size: 1px;">1 </span>
  <span style="font-size: 2px;">2 </span>
  <span style="font-size: 3px;">3 </span>
  <span style="font-size: 5px;">5 </span>
  <span style="font-size: 8px;">8 </span>
  <span style="font-size: 13px;">13 </span>
  <span style="font-size: 21px;">21 </span>
</div>
```

[ ] **None of the above, there is no text added to the `<span>` elements**

Example D:

```
const students = [
  {
    name: 'Inkar Haber',
    age: '27',
    location: 'Tehran'
  },
  {
    name: 'Joakim MacDermott',
    age: '31',
    location: 'Birmingham'
  },
  {
    name: 'Eilert Schwartz',
    age: '52',
    location: 'Saint Petersburg'
  }
]

const listElem = document.createElement('ul');

students
 .sort((studentA, studentB) => studentA.name.localeCompare(studentB.name))
 .map(student => {
   let listItem = document.createElement('li');

   listItem.innerText = `${student.name} is ${student.age} years old from
${student.location}`;
   listItem.classList.add('student-item');

   return listItem;
 })
 .forEach(listItem => listElem.appendChild(listItem));

document.body.appendChild(listElem);
```

6. What HTML will be added to the page as the result of example D?

[ ]
```
<ul>
  <li class="student-item">
    Inkar Haber is 27 years old from Tehran
  </li>
  <li class="student-item">
    Joakim MacDermott is 31 years old from Birmingham
  </li>
  <li class="student-item">
    Eilert Schwartz is 52 years old from Saint Petersburg
  </li>
</ul>
```

[ ]
```
<ul>
  <li class="student-item">
    Eilert Schwartz is 52 years old from Saint Petersburg
  </li>
  <li class="student-item">
    Inkar Haber is 27 years old from Tehran
  </li>
  <li class="student-item">
    Joakim MacDermott is 31 years old from Birmingham
  </li>
</ul>
```

[ ]
```
<ul class="student-item">
  <li>
    Eilert Schwartz is 52 years old from Saint Petersburg
  </li>
  <li>
    Inkar Haber is 27 years old from Tehran
  </li>
  <li>
    Joakim MacDermott is 31 years old from Birmingham
  </li>
</ul>
```

[ ] **Nothing, this is not valid javascript**

Example E:

```
let animals = ['chicken', 'cow', 'pigeon', 'duck', 'pig', 'dog', 'sheep', 'goat',
'horse'];

animals.filter(animal => animal.includes('o'))
  .map(animal => {
    let image = document.createElement('img');
    image.src = `images/${animal}.jpg`;
    image.data.animal = animal;

    return image;
  })
  .forEach(image => {
    let container = document.createElement('div');
    container.className = 'animal-container';
    container.appendChild(image);

    document.body.appendChild(container);
  });
```

7. What HTML will be added to the page as the result of example E?

[ ]
```
<div class="animal-container">
  <img src="images/cow.jpg" data-animal="cow">
</div>
<div class="animal-container">
  <img src="images/pigeon.jpg" data-animal="pigeon">
</div>
<div class="animal-container">
  <img src="images/dog.jpg" data-animal="dog">
</div>
<div class="animal-container">
  <img src="images/goat.jpg" data-animal="goat">
</div>
<div class="animal-container">
  <img src="images/horse.jpg" data-animal="horse">
</div>
```

[ ]
```
<img src="images/cow.jpg" data-animal="cow">
<img src="images/pigeon.jpg" data-animal="pigeon">
<img src="images/duck.jpg" data-animal="duck">
<img src="images/pig.jpg" data-animal="pig">
<img src="images/dog.jpg" data-animal="dog">
<img src="images/sheep.jpg" data-animal="sheep">
<img src="images/goat.jpg" data-animal="goat">
<img src="images/horse.jpg" data-animal="horse">
```

[ ]
```
<div class="animal-container" data-animal="cow">
  <img src="images/cow.jpg">
</div>
<div class="animal-container" data-animal="pigeon">
  <img src="images/pigeon.jpg">
</div>
<div class="animal-container" data-animal="dog">
  <img src="images/dog.jpg">
</div>
<div class="animal-container" data-animal="goat">
  <img src="images/goat.jpg">
</div>
<div class="animal-container" data-animal="horse">
  <img src="images/horse.jpg">
</div>
```

[ ] None of the above, some of the animals are missing

Part III: DOM Traversal (8)

0. What is DOM Traversal?

[ ] Adding and Removing elements from the DOM

[ ] Changing elements in the DOM

[ ] Selecting Elements based on their relationship with a specific element

[ ] Both (0) and (2)

1. A Node and an Element are the exact same thing

[ ] true [ ] false

Example F:

```
<body>
<main class="recipe-container">
  <header id="recipe-header">
    <h1 class="recipe-title">Shakshuka</h1>
    <p class="additional-info">Great algorithm for an easy meal. Feeds 2 adult
humans.</p>
  </header>

  <section id="recipe-ingredients">
    <h2>Prepare Before:</h2>
    <p>Some kind of sentence...</p>
    <ul class="ingredients-list">
      <li class="ingredient-item">Item 1</li>
      <li class="ingredient-item">Item 2</li>
      <li class="ingredient-item">Item 3</li>
      <li class="ingredient-item">Item 4</li>
      <li class="ingredient-item">Item 5</li>
    </ul>
  </section>

  <section id="recipe-instructions">
    <h2>Cooking Time...</h2>
    <ol class="instructions-list">
      <li class="instruction-item">Instruction 1</li>
      <li class="instruction-item">Instruction 2</li>
      <li class="instruction-item">Instruction 3</li>
      <li class="instruction-item">Instruction 4</li>
    </ol>
  </section>
</main>
</body>
```

2 - 6. In example F: Assuming that `current` is the list item with the text "Instruction 3", match the following descriptions with the Element properties that will select them in JS

a.	ul.ingredients-list	[ ]	current.parentElement
b.	li.instruction-item with the text "Instruction 4"	[ ]	current.parentElement .previousElementSibling
c.	h2 with the text "Cooking Time…"	[ ]	current.closest('section') .previousElementSibling .children[2]
d.	p.additional-info	[ ]	current.closest('.recipe-container') .firstElementChild .children[1]
e.	ol.instructions-list	[ ]	current.nextElementSibling

7. Why is previousElementSibling preferred to previousSibling when traversing the DOM?

[ ] previousSibling has a larger performance impact than previousElementSibling

[ ] previousSibling and previousElementSibling are exactly the same

[ ] previousSibling is not part of the DOM API

[ ] previousElementSibling will always contain a DOM Element, while previousSibling might contain other things

Part IV: Events (7)

0. Which arguments does the EventTarget.addEventListener() method take?

[ ] Event Name String, Element to listen to, Function expression to call when the event occurs

[ ] Element to listen to, Function expression to call when the event occurs

[ ] Event Name String, Function expression to call when the event occurs

[ ] Function expression to call when the event occurs

1. The function that you pass as an argument to the EventTarget.addEventListener() method is often called:

[ ] A callback

[ ] An event listener

[ ] An event handler

[ ] Both (0) and (2)

2. Event.preventDefault() is a method that:

[ ] Stops the event from bubbling up the DOM

[ ] Has to be called in order to attach and event handler to an element

[ ] Prevents any default browser behaviour from executing when the event handler is called on the element it is attached to

[ ] Both (0) and (2)

3. What is the difference between `event.target` and `event.currentTarget`

[ ] `event.target` is the event name that was triggered, `event.currentTarget` is the element that triggered the event

[ ] `event.target` is the element that triggered the event, `event.currentTarget` is the event name that was triggered

[ ] `event.target` is the element the event listener is attached to, `event.currentTarget` is the element that triggered the event

[ ] `event.target` is the element that triggered the event , `event.currentTarget` is the element the event listener is attached to

4. A parent will respond to the same event a child does because of event bubbling.

[ ] true [ ] false

```
const triggers = document.querySelectorAll('#trigger-container .trigger');

triggers.forEach(trigger => trigger.addEventListener(‘click’, event => {
  event.preventDefault();

  let currentTrigger = event.target;
  let triggerParent = currentTrigger.parentElem;

  for(let i = 0; i < 100; i++){
    let paragraph = document.createElement(‘p’);
    paragraph.innerText = ‘Haha, you clicked the trigger!’;
    triggerParent.appendChild(paragraph);
  }
}));
```

5. In example G: What will happen when a user clicks on an element with the class trigger?

[ ] Nothing will happen

[ ] 100 paragraphs will be added inside the ‘.trigger’ element

[ ] 100 paragraphs will be added to all parents of all trigger elements

[ ] 100 paragraphs will be added to the parent of the trigger element that was clicked

6. In example G: What code will we write to **delegate the click event?**

[ ]
```
const triggers = document.querySelectorAll('#trigger-container .trigger');

triggers.forEach(trigger => {
  trigger.delegate('click', trigger.parentElem, event => {
    // ... Same code as in example
  })
});
```

[ ]
```
const triggerContainer = document.querySelector('#trigger-container');

triggerContainer.delegate('click', '.trigger', event => {
  event.preventDefault();

  let currentTrigger = event.target;
  let triggerParent = event.currentTarget;

  // ... Same code as in example
});
```

[ ]
```
const triggerContainer = document.querySelector('#trigger-container');

triggerContainer.addEventListener('click', event => {
  if(!event.target.classList.contains('trigger')) return;

  event.preventDefault();

  let currentTrigger = event.target;
  let triggerParent = event.currentTarget;

  // ... Same code as in example
});
```

[ ]
```
const triggers = document.querySelectorAll('#trigger-container .trigger');

triggers.forEach(trigger => {
  trigger.addEventListener('click', event => {
    event.preventDefault();

    let currentTrigger = event.target;
    let triggerParent = event.currentTarget;

    // ... Same code as in example
  })
});
```