Client-Side Web Development

Class 11.1

Today’s Topics

- Objects
- Event Delegation
- **Exercise:** Dice

Announcements

Web Portfolio

Any Questions?

Objects

An *object* is a collection of properties and methods

Each property of an object is made up of a *key/value pair*.

A *key* is like the variable name.

A *value* can be of any data type and an object can have many different data types.

```
// Create an Object for a Little Pig
const littlePig = {
  name: 'Barnaby'
}
```

Just like a variable, a *property* can be accessed or changed.

Two method can be used, dot or bracket notation

```
// Create an Object for a Little Pig
const littlePig = {
  name: 'Barnaby'
}

// Using dot notation
console.log(littlePig.name) // Barnaby

// Using bracket notation
console.log(littlePig['name']) // Barnaby
```

The assignment operator (=) is used to reassign a property

```
// Create an Object for a Little Pig
const littlePig = {
  name: 'Barnaby'
}

// Using dot notation
littlePig.name = 'Hamlet'

// Using bracket notation
littlePig['name'] = 'Wilbur'
```

The same syntax is used to add new properties.

```
// Create an object for the first little pig
const firstLittlePig = {
  name: 'Barnaby'
}

// Add work ethic using dot notation
firstLittlePig.workEthic = 'very lazy'

// Add house using bracket notation
firstLittlePig['house'] = 'straw'
```

Use the `delete` keyword to remove a property

```
// Create an object for the first little pig
const firstLittlePig = {
  name: 'Barnaby',
  workEthic: 'very lazy',
  house: 'straw'
}

// Remove house property
delete firstLittlePig.house
```

Nested Arrays and Objects

It is common to have nested arrays and objects inside arrays and objects.

```
// An array of objects
const littlePigs = [
  {name: 'Barnaby', workEthic: 'very lazy'},
  {name: 'Hamlet', workEthic: 'lazy'},
  {name: 'Wilbur', workEthic: 'hard'}
]

// An nested array in an object
const wolf = {
  name: 'Midas',
  appearance: ['Big', 'Bad'],
  favoriteFood: 'Pork Chops'
}
```

To access values in a nested object or array requires an extra level of dot or bracket notation.

```
// An array of objects
const littlePigs = [
  {name: 'Barnaby', workEthic: 'very lazy'},
  {name: 'Hamlet', workEthic: 'lazy'},
  {name: 'Wilbur', workEthic: 'hard'}
]

// The second little pig's name
console.log(littlePigs[1].name) // Hamlet

// The first little pig's work ethic
console.log(littlePigs[0]['workEthic']) // very lazy
```

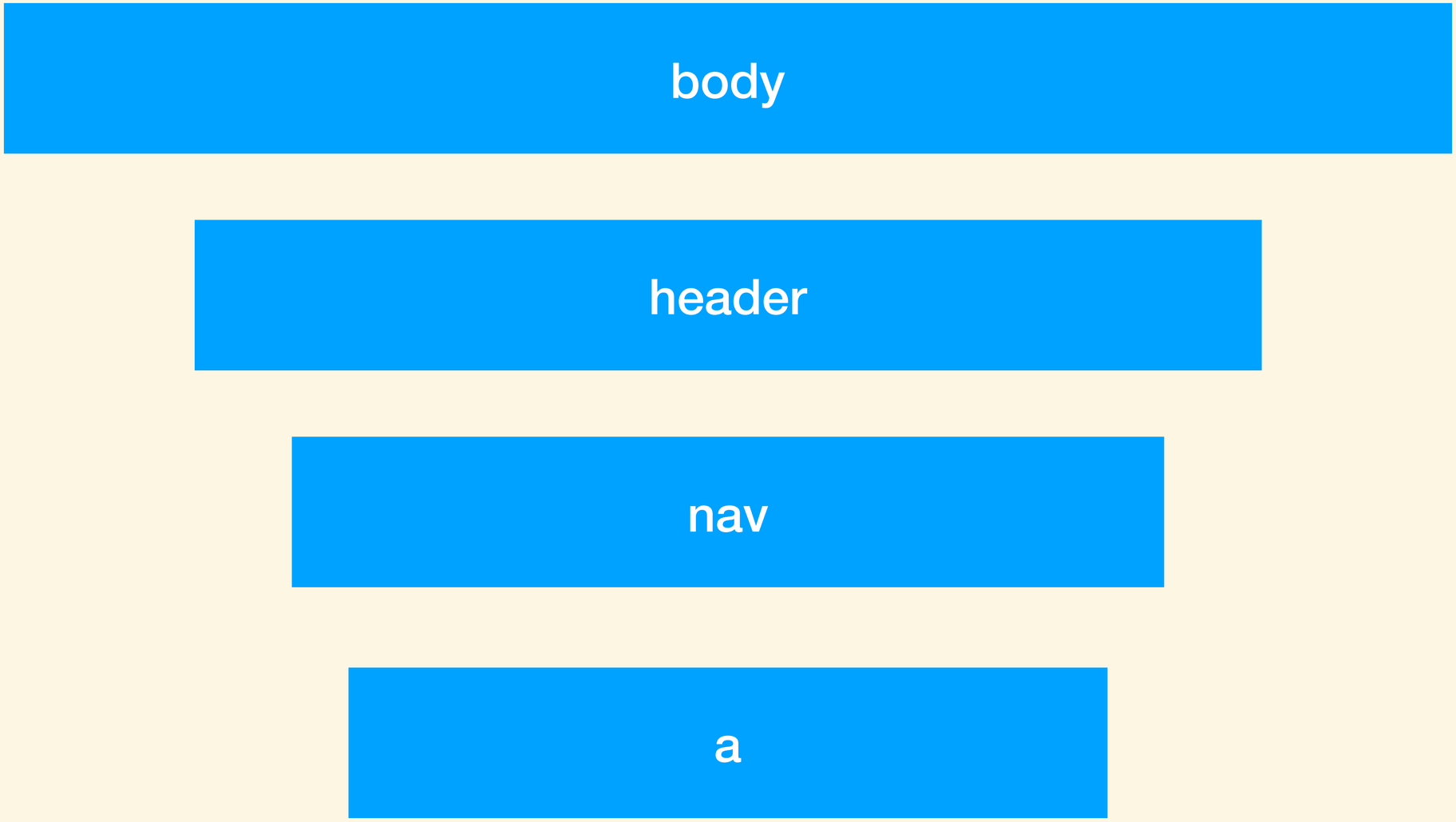
```
// An nested array in an object
const wolf = {
  name: 'Midas',
  appearance: ['Big', 'Bad'],
  favoriteFood: 'Pork Chops'
}

// The Big Bad Wolf
console.log(`The ${wolf.appearance[0]}
  ${wolf.appearance[1]} Wolf`)
```

Event Delegation

Event Delegation **is a coding technique for adding event listeners to multiple elements**

Event Delegation **utilizes a process in JavaScript called Event Propagation**

Event Propagation **is the process of moving up the DOM tree when event occurs**


body
header
nav
a

In *Event Delegation* a listener is added to a parent of all the elements which the event will occur

Then uses the `target` property of the `event` object to determine which element received the event

```
<ul id="list">
  <li class="item">Clean the car</li>
  <li class="item completed">Feed the cat</li>
  <li class="item">Buy milk</a>
</ul>
```

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

$list.addEventListener('click', function (e) {
  if (e.target.classList.contains('item')) {
    e.target.classList.toggle('completed')
  }
})
```

Examples

Project: Interactive Gallery

Exercise: Dice

For next class...

- **Review:** Domino's
- **Lab:** Deck of Cards