Lecture 5.1

Traversing The DOM

Topics

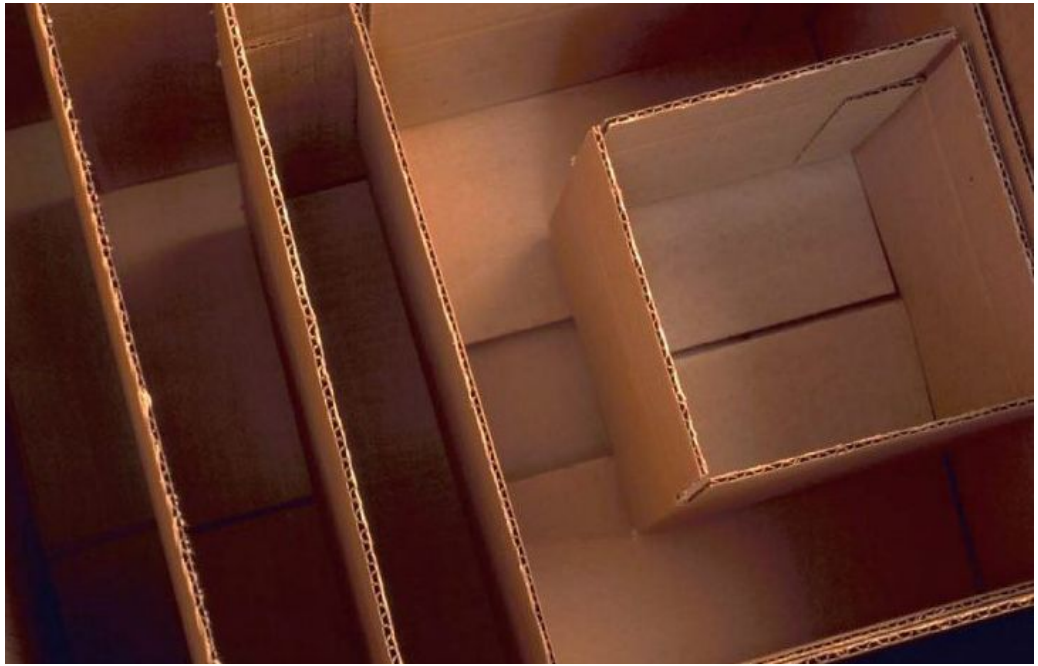
- Traversing DOM Basics
- Parent Node
- Children Nodes
- Sibling Nodes

Traversing Basics

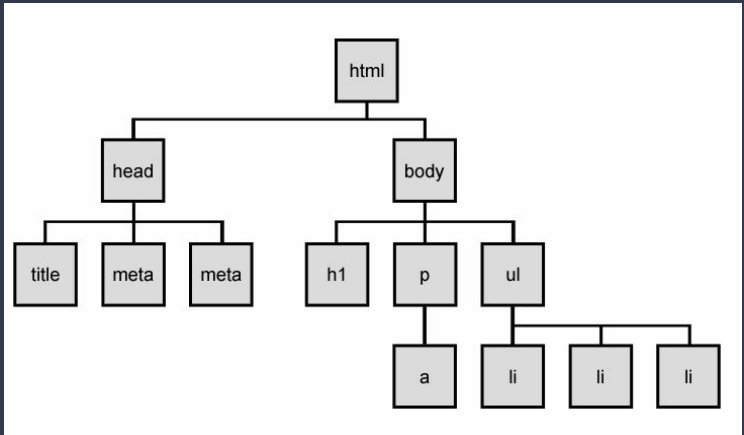
Box Model

We remember CSS, the box model.

We remember, there were boxes...within boxes...within boxes.



DOM Tree



Usually, the DOM is called the **DOM tree.** This sort of works to explain the basic idea since people understand the idea of tree branches and an infinite hierarchy of these branches.

The two key concepts in the DOM:

1. **Containment:** Parent elements contain children elements. And those children contain their children elements.
2. **Order:** DOM elements have a definite order that you can manipulate.

The basics of traversing the DOM

Here is a quick diagram of the first three months of 2018.

In this case, the **year** contains three **months,** which each contain 4 **weeks**.

HTML equivalent

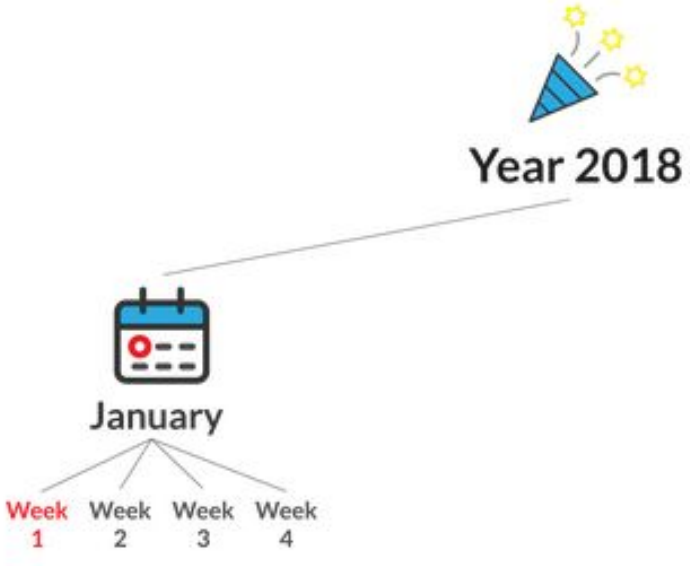
Each month is a div that not only has the class of month, but also a class with a specific name of the month. The same structure is used on the div with class year on line 1.

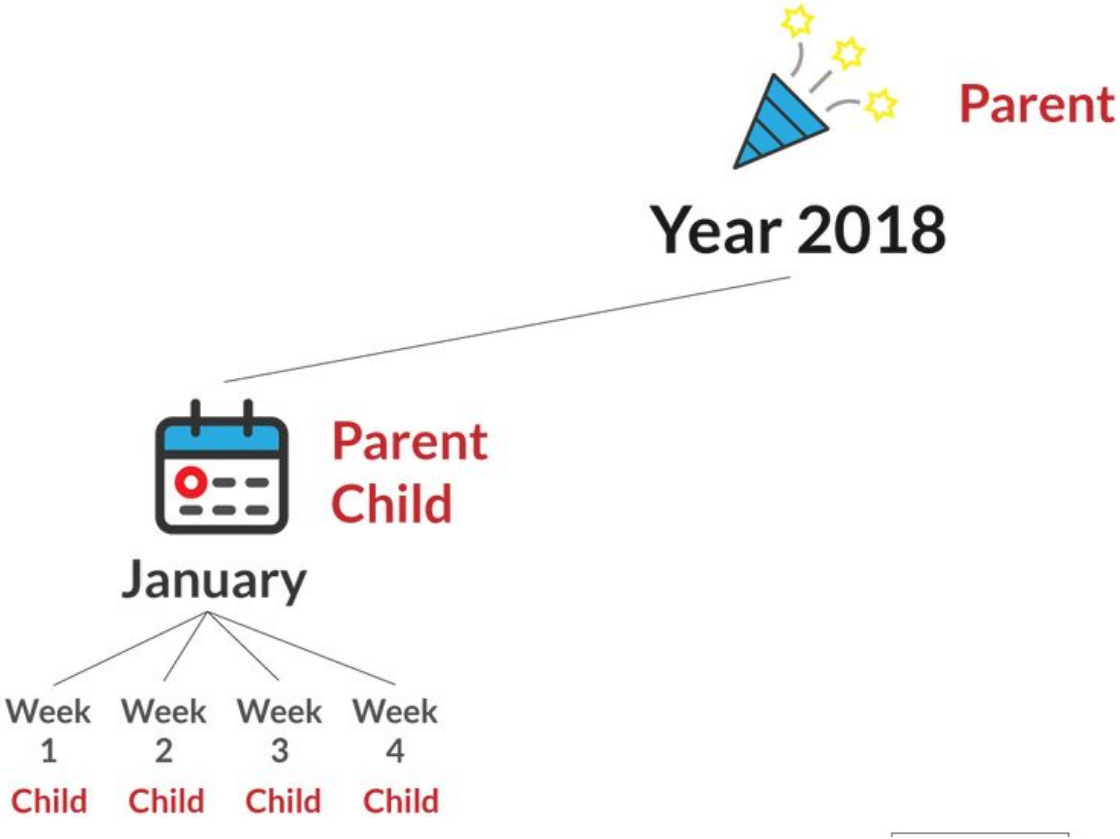
Also, the weeks do not have any particular identifier beyond the week class. You will see why in a moment.

Besides using classes and IDs to access elements via the DOM, we can also **use the relationships between elements**. There are three relationship we know — child, parent, and sibling.

Child Elements

Child element: An element that is contained within another element
Example — The **january** div is a **child** of the year **2018**.



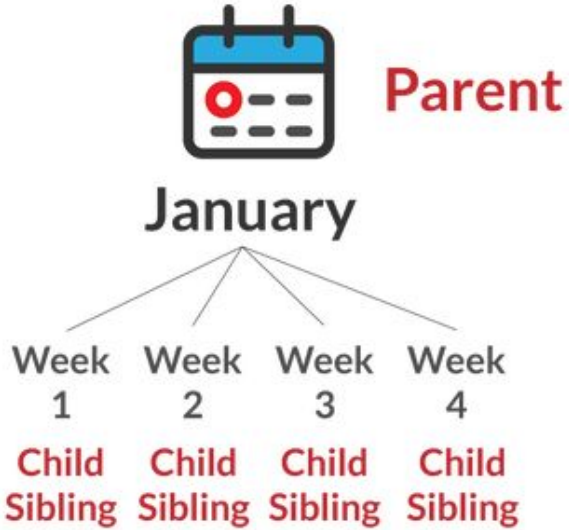


Parent element: An element that contains other elements.

Sibling Elements

Sibling Element: an element that has the same direct parent as other elements.

Example: All **4 weeks** within **January** are siblings of each other.



Difference between HTML and DOM

Well, think of the way you use a personal calendar. It is just a record of the things that you do during your days. It is not the actual activity! In other words, it is a model of the stuff that goes on during the day.

The HTML is the actual content of your day. HTML elements make up the webpage, while the DOM is an accessible interface to direct changes.

Parent Node

node.parentNode

To get the parent node of a specified node in the DOM tree, you use the parentNode property:

```
let parent = node.parentNode;
```

The parentNode is read-only.

The Document and DocumentFragment nodes do not have a parent, therefore the parentNode will always be null.

If you create a new node but haven't attached it to the DOM tree, the parentNode of that node will also be null.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>JavaScript parentNode</title>
</head>
<body>
    <div id="main">
        <p class="note">This is a note!</p>
    </div>

    <script>
        let note = document.querySelector('.note');
        console.log(note.parentNode);
    </script>
</body>
</html>
```

Child Nodes

Child Nodes

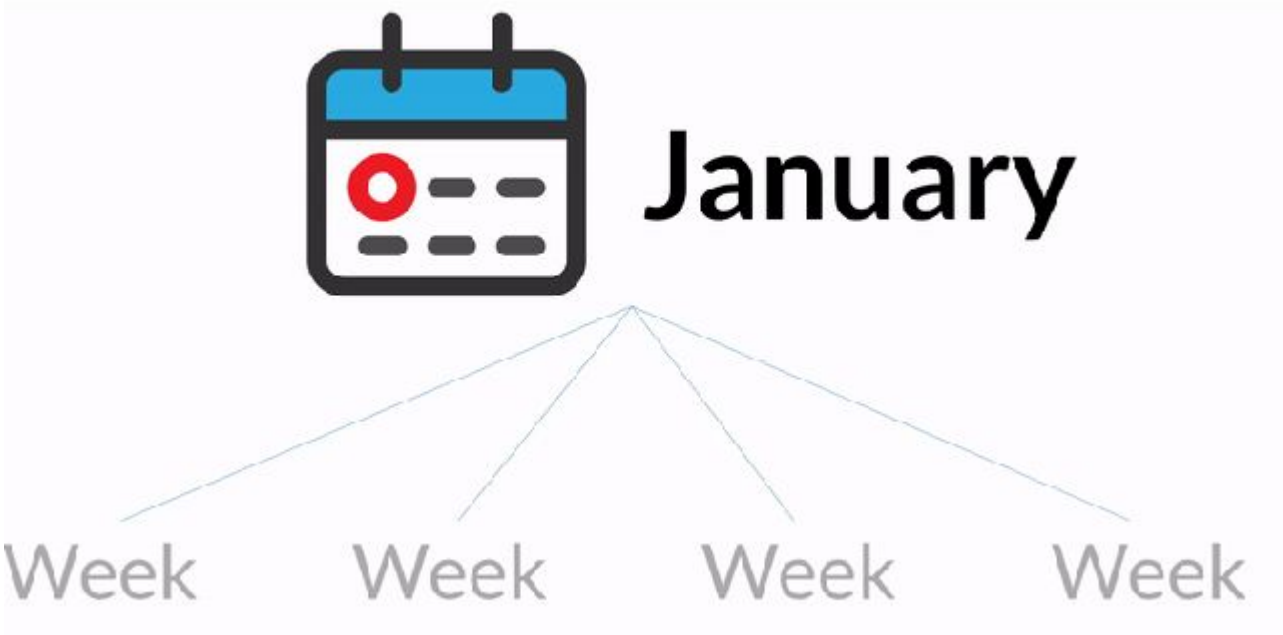
node.childNodes

To get a live NodeList of child elements of a specified element, you use the childNodes property:

```
let children = parentElement.childNodes;
```

The childNodes property returns all child elements with any node type. To get the child element with only the element node type, you use the children property:

```
let children = parentElement.children;
```



Child Nodes (Weeks) of the parent (Month) Node

node.firstChild

To get the first child element of a specified element, you use the firstChild property of the element:

If the parentElement does not have any child element, the firstChild returns null.

```
let firstChild = parentElement.firstChild;
```

Or to get the first child with the Element node only, you can use the firstElementChild property:

```
let firstElementChild = parentElement.firstElementChild;
```

node.firstChild

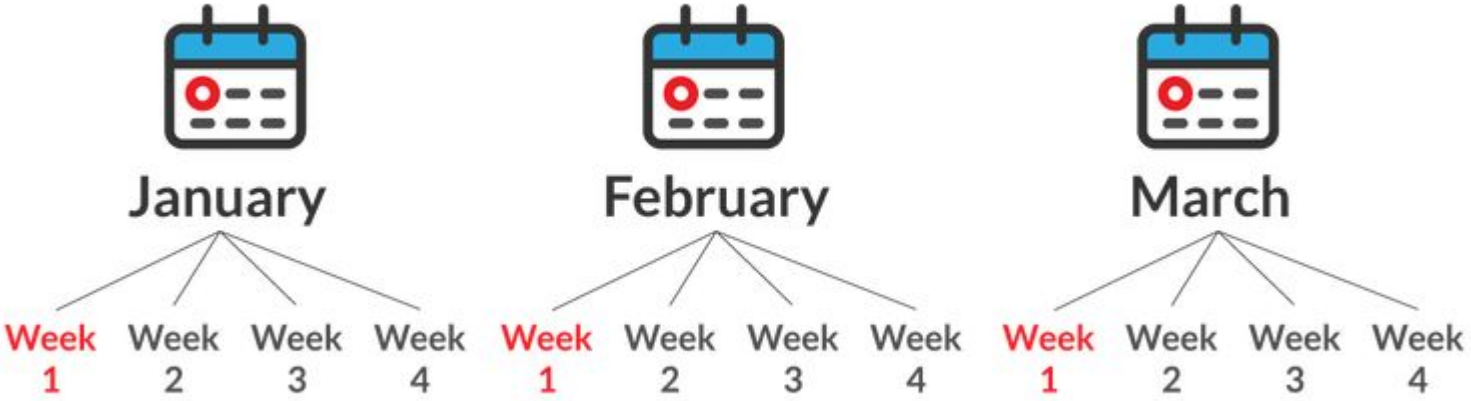
To get the first child element of a specified element, you use the firstChild property of the element:

If the parentElement does not have any child element, the firstChild returns null.

```
let firstChild = parentElement.firstChild;
```

Or to get the first child with the Element node only, you can use the firstElementChild property:

```
let firstElementChild = parentElement.firstElementChild;
```



First Child Node

node.lastChild

To get the last child element of a specified element, you use the lastChild property of the element:

If the parentElement does not have any child element, the lastChild returns null.

```
let lastChild = parentElement.lastChild;
```

Or to get the last child with the Element node only, you can use the lastElementChild property:

```
let lastChild = parentElement.lastElementChild;
```

Recap: Child Nodes

- The firstChild and lastChild return the first and last child of a node, which can be any node type including text node, comment node, and element node.
- The firstElementChild and lastElementChild return the first and last child Element node.
- The childNodes returns a live NodeList of all child nodes of any node type of a specified node. The children return all child Element nodes of a specified node.

Sibling Nodes

Node siblings

To get the next sibling of an element, you use the nextElementSibling attribute:

```
let nextSibling = currentNode.nextElementSibling;
```

To get the previous siblings of an element, you use the previousElementSibling attribute:

```
let prevSibling = currentNode.previousElementSibling;
```