Working with the DOM

tl;dr

- The DOM gives your JavaScript access to anything on the page
- You can add elements, remove elements, or change elements
- Accessing form fields should be done with the value property
- Event handling should be done with addEventListener() but there are other ways

The browser knows about an object called 'document'

It is an object which represents the entire HTML document loaded

It has an property called "children"

```
let kids = document.children;
```

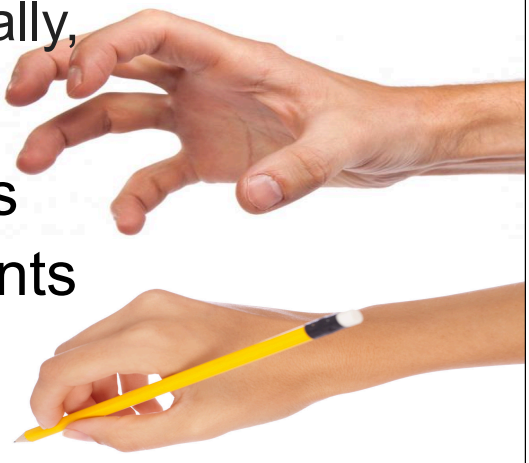
- 'kids' is an object of type *HTMLCollection*

```
let greatGreatGrandChild =
  document.children[0].children[4].children[8].children[1];
```

- You can use this to get any element on the page
- It's a **model** of all of the **objects** in the **document**! It is the ...

MOD DOM!!

To alter content dynamically, we will ...



1. Get existing elements
2. Change those elements
   - Remove them
   - Change them
   - Append content to them

Getting elements

```
let aDiv = document.getElementById("linksDiv");
let allDivs = document.getElementsByTagName("div");
let allAlerts = document.getElementsByClassName('alert');
```

- And the all-new, all-improved ...

```
let firstradio = document.querySelector('[type=radio]');
let allTxtBox = document.querySelectorAll('[type=text]');
```

To add elements, use createElement()

```
const e = document.getElementById("foo");
let p = document.createElement("p");
p.textContent = "Excelsior!"

e.appendChild(p);

let i = document.createElement("img");
i.src = "/imgs/stanLee.jpg";

e.appendChild(i);
```

Create a <p> in memory...

... and put it on the page

Create an <img>

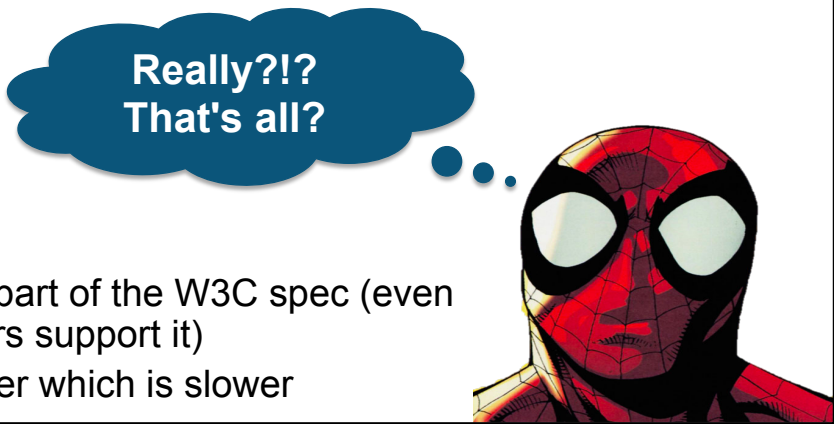
... and put it on the page

Changing elements

```
p.style.background = "yellow";
p.style.height = "50px";
p.setAttribute("title", "Stan says this");
```

So we *could* use innerHTML instead

```
e.innerHTML = `
<img src='/img/stanLee.jpg' />
<p>Excelsior!</p>`;
```



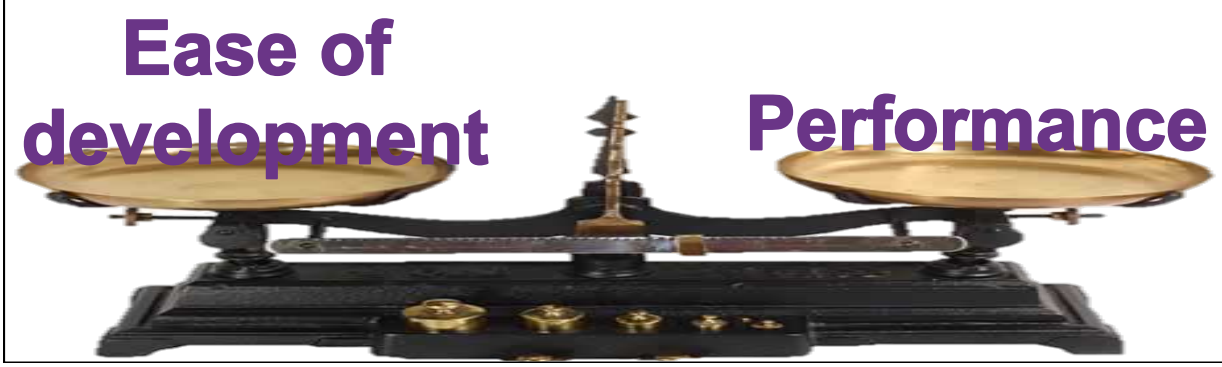
But ...

1. innerHTML is not part of the W3C spec (even though all browsers support it)
2. It forces a re-render which is slower

It's easier to add with innerHTML

When you set innerHTML, the browser ...

1. forces a parsing of a sub-DOM
2. which is then injected into the page DOM
3. which is then re-parsed and
4. which is then re-rendered



Forms and the DOM

Use *value* to get to form fields

- To read:

```
document.querySelector("input[type='text']");
let foo = field.value;
```

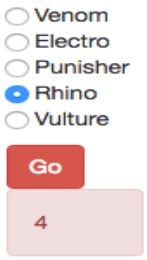
- To write:

```
field.value = "foo";
```

Note: It is not innerHTML nor innerText.

Getting the selected radio button

```
<input type='radio' value="1" name="foe" />Venom
<input type='radio' value="2" name="foe" />Electro
<input type='radio' value="3" name="foe" />Punisher
<input type='radio' value="4" name="foe" />Rhino
<input type='radio' value="5" name="foe" />Vulture
<button>Go</button>
<div id="output"></div>
<script>
document.querySelector("button")
  .addEventListener("click",() => {
  foe = document.querySelector(":checked").value;
});
</script>
```



Select Lists

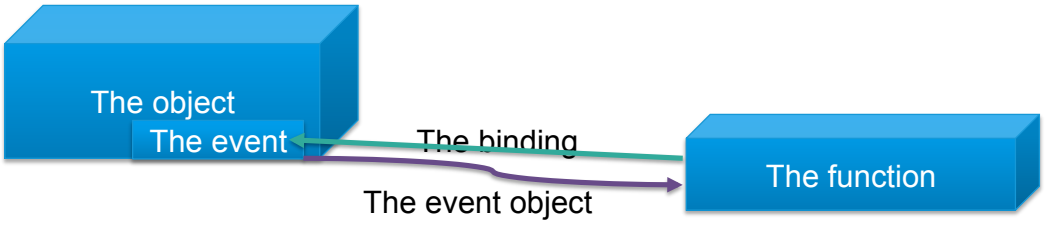
```
<select multiple>
 <option value="1">Venom</option>
 <option value="2">Electro</option>
 <option value="3">Punisher</option>
 <option value="4">Rhino</option>
 <option value="5">Vulture</option>
</select>
<button>Go</button>
<script>
document.querySelector("button")
  .addEventListener("click",() => {
   foes = document.querySelectorAll(":checked");
    .map(o => o.value);  // Array of selected values
});
</script>
```

Event Handling

How to make your page respond to the user

What you need for event handling

1. An object to attach to
2. The event to listen for
3. A function to bind
4. The event object to pass



Four ways to wire them up

Type	Vehicle	Pros	Cons
IE 8 and below	e.attachEvent('onevent', function () {...});	Works in IE 8-	Only works in IE 8-
HTML attribute	<e onevent="f()" />	Simplest	Least flexible
DOM Level 0 style	e.onevent = function () {...};	SOC	Old-school
DOM Level 2 style	e.addEventListener(event, function () {...});	Most capable and flexible	Most typing

IE 8 and below style

- Obviated by addEventListener() in IE9
- Completely removed in IE11

```
var btn = document.getElementById('btn');
btn.attachEvent('onclick', function (e) {
  // Do stuff
});
```

HTML Attributes

- Example:

```
<input type='button' onclick='doSomething()' value='Go' />
```

- Sets the event handler.
- Clobbers all old event handlers
- Can only have one event

DOM Level 0 Style

- Example:

```
var btn = document.getElementById('btn');
btn.onclick = function (e) {
  // Do stuff
}
```

- Once again, not additive.

DOM Level 2 Style

- Example

```
var btn = document.getElementById('btn');
btn.addEventListener('click', function (e) {
  // Do stuff
}, false);
```

- This is additive. It doesn't clobber!
- The third argument, a bool says if we should capture
  - true = capture
  - false = bubble (the default)

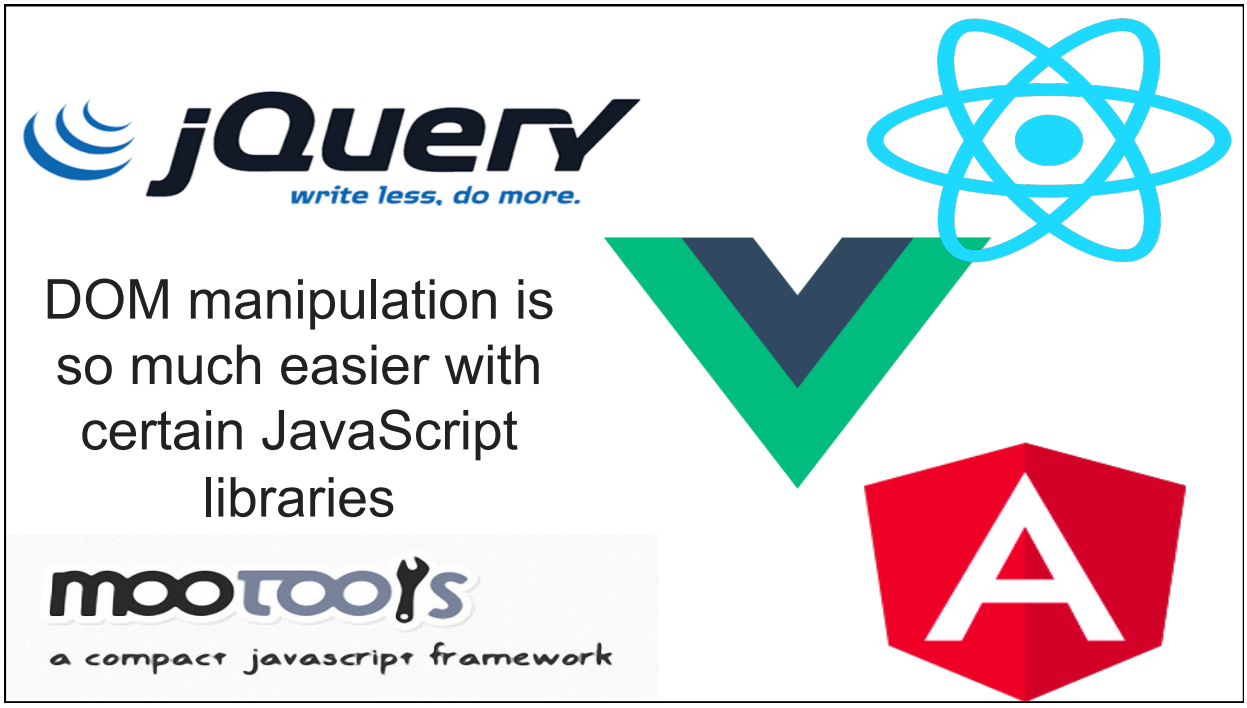
So many choices! Which one to use?

- Fortunately this is an easy decision.

Use addEventListener()

- If you must support IE8-, use something like this:

```
if (x.addEventListener)
  x.addEventListener(...)
else
  x.attachEvent(...)
```



tl;dr

- The DOM gives your JavaScript access to anything on the page
- You can add elements, remove elements, or change elements
- Accessing form fields should be done with the value property
- Event handling should be done with addEventListener() but there are other ways