CS193X: Web Programming Fundamentals

Spring 2017

Victoria Kirst
(vrk@stanford.edu)

Today's schedule

Today

- JS Events, again
- Finish Tic-Tac-Toe
- DOM revisited
- Hacks and mischief

Wednesday (probably)

- Animations
- Mobile events

Friday (probably)

- Classes and objects in JavaScript
- `this` keyword and `bind`

HW2 deadline extended

HW2 deadline: ~~Wed Apr 26~~ **Fri Apr 28**

- Deadline extended till Friday; late cutoff is Sun Apr 30
- HW3 will go out Friday as well

Reminder: There are **no exceptions** for the late cutoff

- If you do not turn in your HW by the late cutoff, you will get a 0.
- If you think you can't finish in time, please turn in what you have ahead of time! You can update it up till the deadline.

JavaScript events in detail

Events in JavaScript

If you put a "click" event listener on an element, what happens if the user clicks a *child* of that element?

```
<div class="show-details">
  <img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/1083533/forward-arrow.p
  <span>Show details</span>
</div>
```

```
const detailToggle = document.querySelector('.show-details');
detailToggle.addEventListener('click', toggleVisibility);
```

Events in JavaScript

Example: If you click on the `<img>`, will the `toggleVisibility` function fire? (CodePen)

```
<div class="show-details">
  <img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/1083533/forward-arrow.p
  <span>Show details</span>
</div>
```

```
const detailToggle = document.querySelector('.show-details');
detailToggle.addEventListener('click', toggleVisibility);
```

Events in JavaScript

Yes, a click event set on an element will fire if you click on a child of that element

If you put a click event listener on the `div`, and the user clicks on the `img` inside that `div`, then the event listener will still fire.

(CodePen)


```
<div class="show-details">
  <img src="https://s3-us-w
  <span>Show details</span>
</div>
```

Event.currentTarget vs target

```
function toggleVisibility(event) {
  const theElementClicked = event.target;
  const theElementTheEventIsTiedTo = event.currentTarget;
```

You can access either the element clicked or the element to which the event listener was attached:

- ***event*.target**: the element that was clicked / "dispatched the event" (might be a child of the target)
- ***event*.currentTarget**: the element that the original event handler was attached to

(Sorry, I got this wrong earlier)

(Programming note: I got these mixed up the last few lectures and used **`target`** when I meant **`currentTarget`**. Last week's slides and examples have since been fixed.

Whoops, sorry!)

Multiple event listeners

What if you have event listeners set on both an element and a child of that element?

- Do both fire?
- Which fires first?

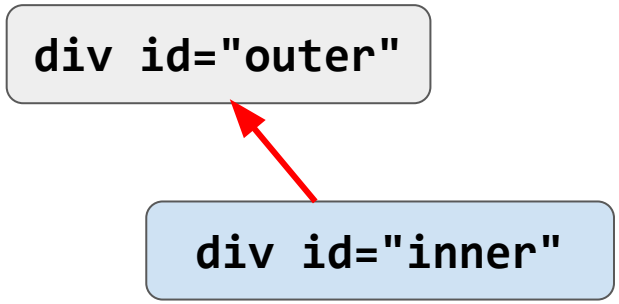
```
<div id="outer">
  Click me!
  <div id="inner">
    No, click me!
  </div>
</div>
```

```
const outer = document.querySelector('#outer');
const inner = document.querySelector('#inner');
outer.addEventListener('click', onOuterClick);
inner.addEventListener('click', onInnerClick);
```

(CodePen)

Event bubbling

- Both events fire if you click the inner element
- By default, the event listener on the inner-most element fires first

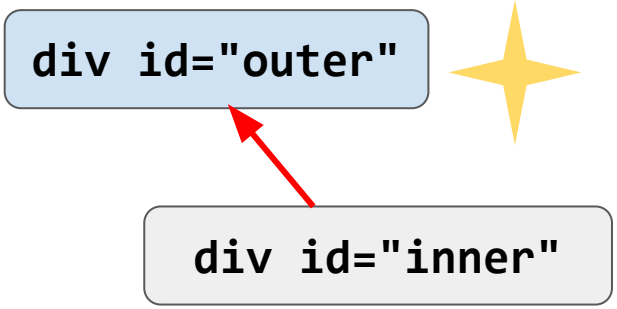



```
<div id="outer">
  Click me!
  <div id="inner">
    No, click me!
  </div>
</div>
```

This event ordering (inner-most to outer-most) is known as **bubbling**. (CodePen)

Event bubbling

- Both events fire if you click the inner element
- By default, the event listener on the inner-most element fires first




```
<div id="outer">
  Click me!
  <div id="inner">
    No, click me!
  </div>
</div>
```

This event ordering (inner-most to outer-most) is known as **bubbling**. (CodePen)

`stopPropagation()`

We can stop the event from bubbling up the chain of ancestors by using ***event***`.stopPropagation()`:

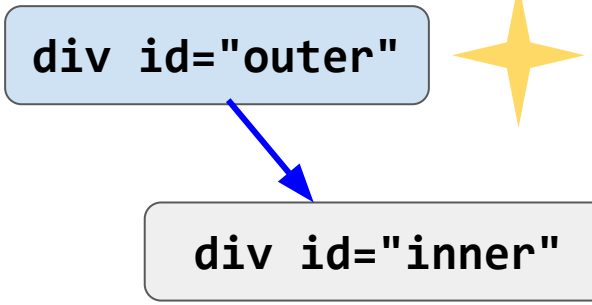
```
function onInnerClick(event) {
  inner.classList.add('selected');
  console.log('Inner clicked!');
  event.stopPropagation();
}
```

See default behavior vs with `stopPropagation`

Event capturing

To make event propagation go the opposite direction, add a 3rd parameter to `addEventListener`:

```
event.addEventListener(
    'click', onClick, { capture: true} );
```



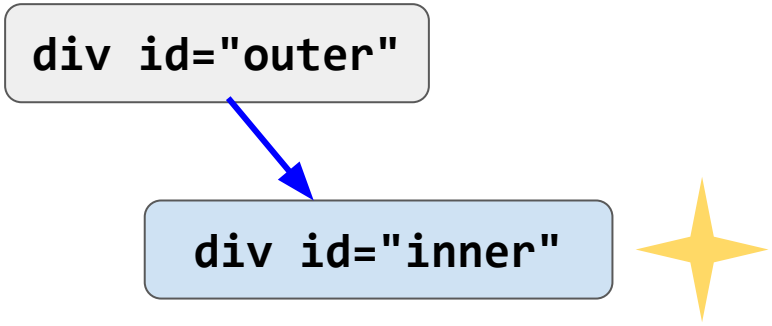

```
<div id="outer">
  Click me!
  <div id="inner">
    No, click me!
  </div>
</div>
```

This event ordering (outer-most to inner-most) is known as **capturing**. (CodePen)

Event capturing

To make event propagation go the opposite direction, add a 3rd parameter to `addEventListener`:

```
event.addEventListener(
    'click', onClick, { capture: true} );
```




```
<div id="outer">
  Click me!
  <div id="inner">
    No, click me!
  </div>
</div>
```

This event ordering (outer-most to inner-most) is known as **capturing**. (CodePen)

`stopPropagation()`

We can also use ***event***`.stopPropagation()` in capture-order:

```
function onOuterClick(event) {
  outer.classList.add('selected');
  console.log('Outer clicked!');
  event.stopPropagation();
}
```

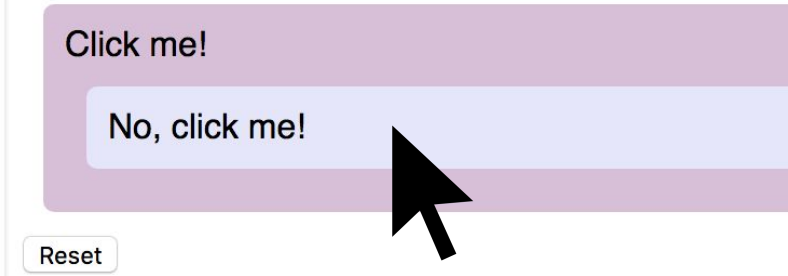
See default behavior vs with `stopPropagation`

Some technical details...

Behind the scenes

Technically, the browser will go through **both** a capture phase and a bubbling phase when an event occurs:

```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Events: Two event listeners</title>
  </head>
  <body>
    <div id="outer">
      Click me!
      <div id="inner">
        No, click me!
      </div>
    </div>
    <button>Reset</button>
  </body>
</html>
```

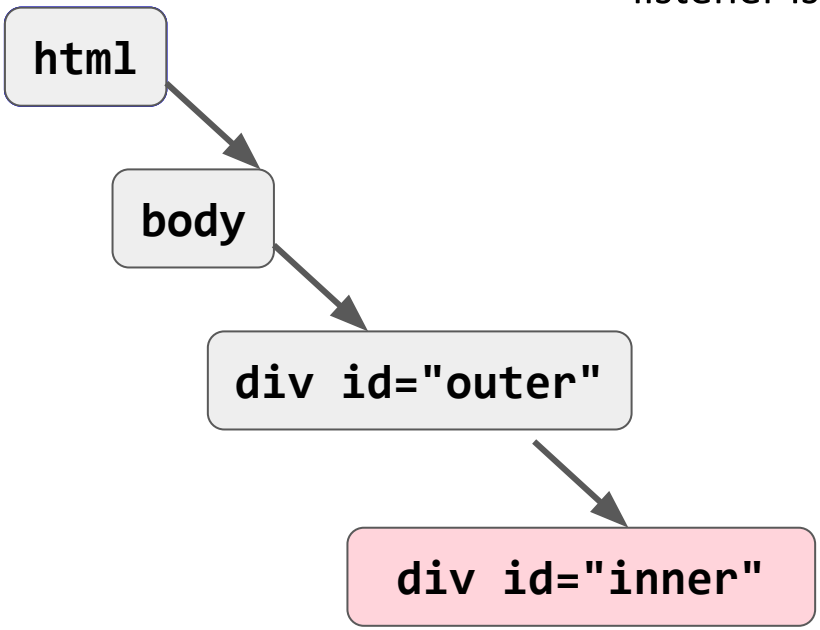


If we click on the div with id="inner"...

Behind the scenes

The browser creates the `target`'s"**propagation path**," or the list of its ancestors up to root (w3c)

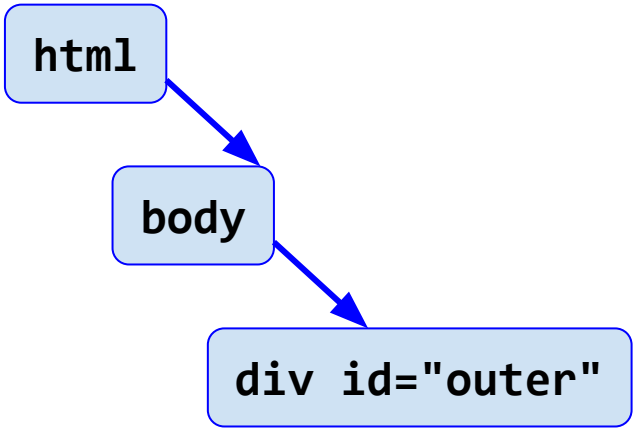
(`target` meaning the thing you clicked; not necessarily the element the event listener is attached to)

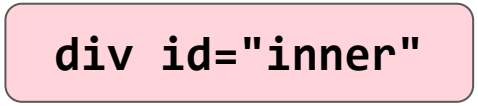



```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Events: Two event list
  </head>
  <body>
    <div id="outer">
      Click me!
      <div id="inner">
        No, click me!
      </div>
    </div>
    <button>Reset</button>
  </body>
```

"Capture phase"

The browser begins at the top of the propagation path and invokes any event listeners that have `capture="true"`, in path order until it gets to the target. This is the "**capture phase**" (w3c)

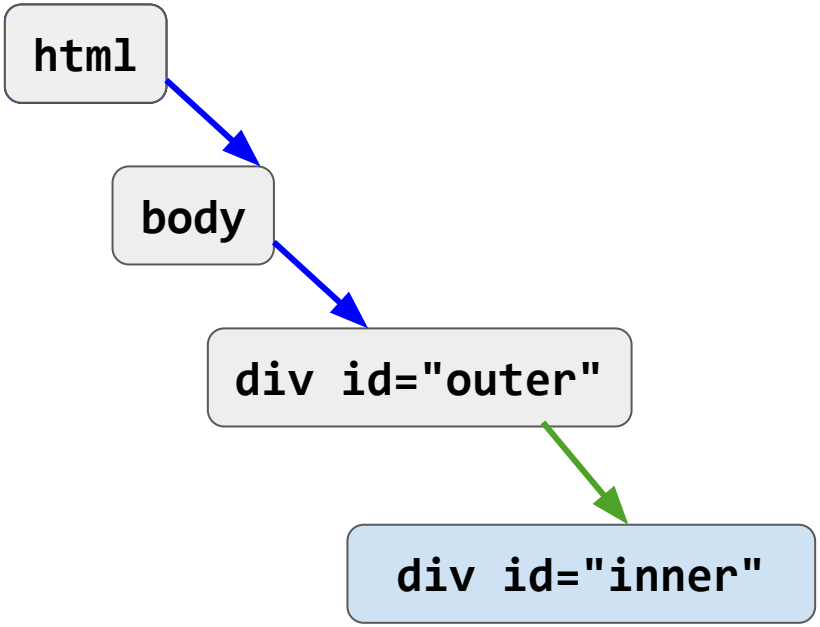





```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Events: Two event list
  </head>
  <body>
    <div id="outer">
      Click me!
      <div id="inner">
        No, click me!
      </div>
    </div>
    <button>Reset</button>
  </body>
```

"Target phase"

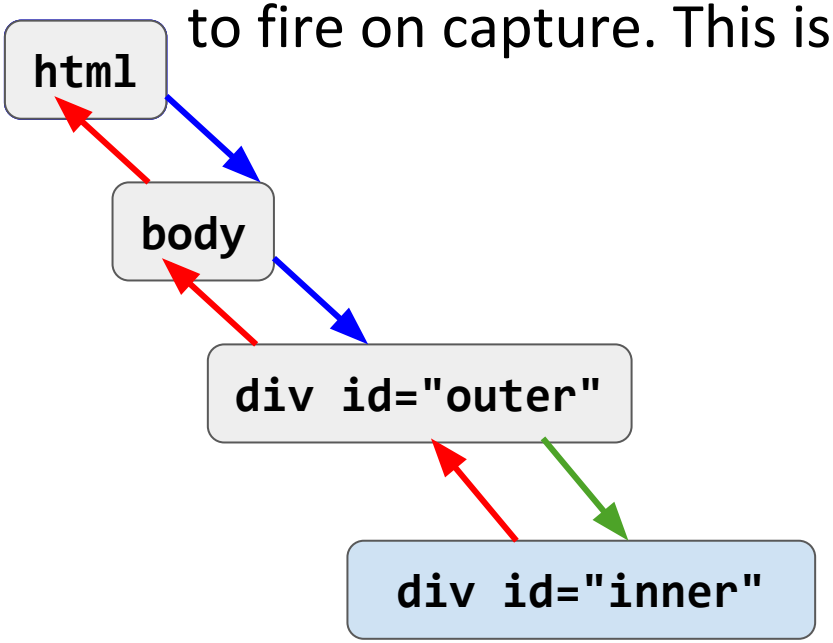
Then the browser invokes any event listener that was set on the target itself. This is the "target phase" (w3c)




```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Events: Two event list
  </head>
  <body>
    <div id="outer">
      Click me!
      <div id="inner">
        No, click me!
      </div>
    </div>
    <button>Reset</button>
  </body>
```

"Bubble phase"

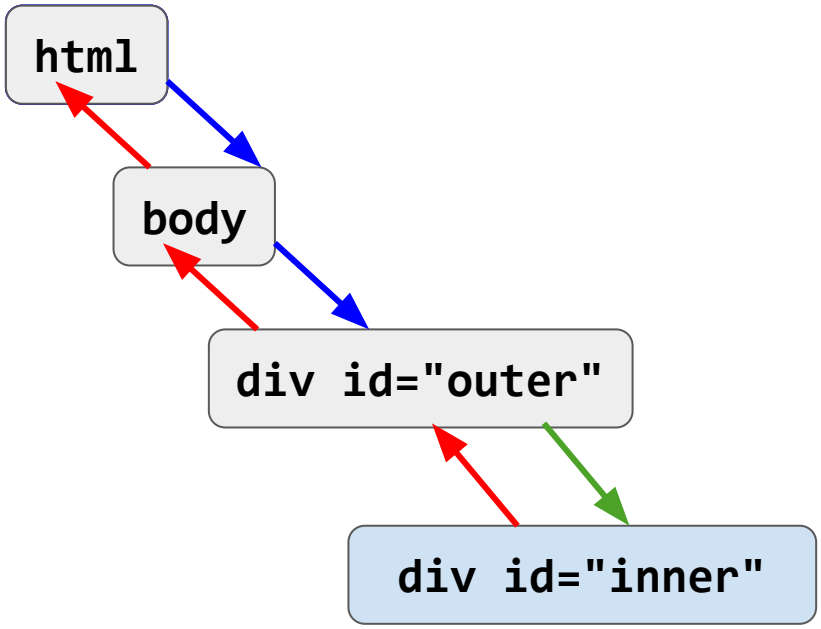
If the event type has `bubbles=true` (see `click`, e.g.) the browser goes back up the propagation path in reverse order and invokes any event listener that wasn't supposed to fire on capture. This is the "**bubble phase**" (w3c)




```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Events: Two event list
  </head>
  <body>
    <div id="outer">
      Click me!
      <div id="inner">
        No, click me!
      </div>
    </div>
    <button>Reset</button>
  </body>
```

stopPropagation()

Therefore stopPropagation() actually stops the rest of the 3-phase dispatch from executing



In Practice

Don't worry about:

- You never need to use capture order - you can always use bubbling
- You don't really need to know how the browser goes through "capture phase", "target phase", then "bubble phase"

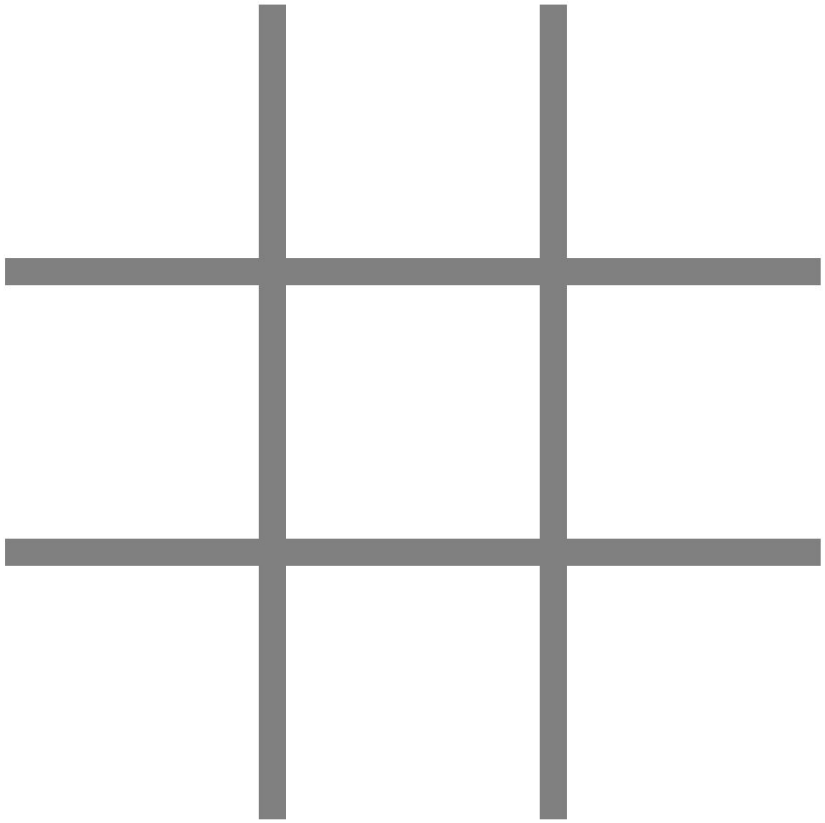
Do worry about:

- **You do need to understand bubbling, though**
- `stopPropagation()` also comes in handy

Case Study: Tic-Tac-Toe

Example: Tic Tac Toe

Let's try to implement a game of Tic-Tac-Toe. (finished)



Tic Tac Toe plan

1. ~~Every time we click on an empty space, change the empty space in an "X" by adding an image of an "x" into the empty `<div>`~~
2. ~~After our turn, the computer puts an "O" in a random empty space~~
3. **When there are 3 Xs or 3 Os in a row, declare a winner**

Step 2 complete

Distinguishing boxes

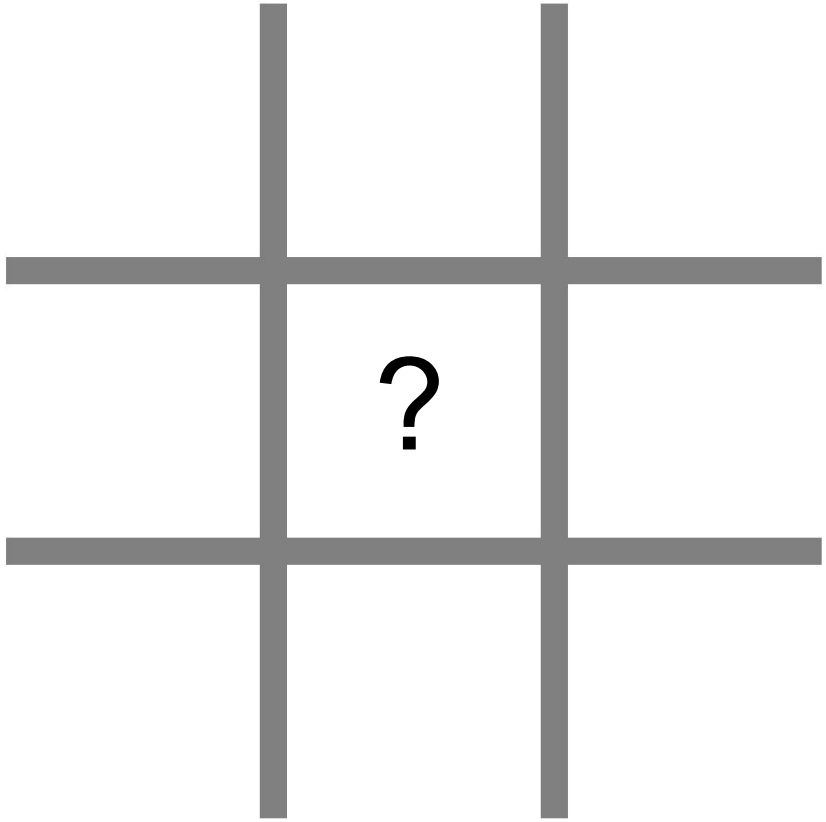
```
const freeBoxes = [];
const boxes = document.querySelectorAll('#grid div');
for (const box of boxes) {
  box.addEventListener('click', changeToX);
  freeBoxes.push(box);
}
```

```
function changeToX(event) {
  const container = event.currentTarget;
  const image = document.createElement('img');
  image.src = X_IMAGE_URL;
  container.appendChild(image);
  container.removeEventListener('click', changeToX);

  // Also remove |container| from |freeBoxes|
  const indexToRemove = freeBoxes.indexOf(container);
  freeBoxes.splice(indexToRemove, 1);
  computerChooseO();
}
```

The same event handler is called for each element...

Distinguishing boxes



When the click event fires, how do we know where in the board the event's `currentTarget` is positioned?

Terrible idea: 9 event handlers

```
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div id="one"></div>
    <div id="two"></div>
    <div id="three"></div>

    <div id="four"></div>
    <div id="five"></div>
    <div id="six"></div>

    <div id="seven"></div>
    <div id="eight"></div>
    <div id="nine"></div>
  </div>
</body>
```

```
function changeToXRow1Column1(event) {
  //...
}
function changeToXRow1Column2(event) {
  //...
}
function changeToXRow1Column3(event) {
  //...
}
```

```
const first = document.querySelector('#one');
first.addEventListener('click', changeToXRow1Column1);
const second = document.querySelector('#two');
second.addEventListener('click', changeToXRow1Column2);
```

Uniquely identifying items

```
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div id="one"></div>
    <div id="two"></div>
    <div id="three"></div>

    <div id="four"></div>
    <div id="five"></div>
    <div id="six"></div>

    <div id="seven"></div>
    <div id="eight"></div>
    <div id="nine"></div>
  </div>
</body>
```

But this idea of uniquely identifying squares is a good one!

Solution

```
const freeBoxes = [];
// Map of box number -> 'x' or 'o'
const takenBoxes = {};
const boxes = document.querySelectorAll('#grid div');
for (const box of boxes) {
  box.addEventListener('click', changeToX);
  freeBoxes.push(box);
}
```

Add another state variable, `takenBoxes`, that maps box number to who owns the box

```
function changeToX(event) {
  assignSpace(event.currentTarget, 'x');
```

```
function computerChooseO() {
  const allBoxes  = document.querySelectorAll('#grid di
  const index = Math.floor(Math.random() * freeBoxes.le
  const freeSpace = freeBoxes[index];

  assignSpace(freeSpace, 'o');
```

```
function assignSpace(space, owner) {
  const image = document.createElement('img');
  image.src = owner === 'x' ? X_IMAGE_URL : O_IMAGE_URL;
  space.appendChild(image);

  takenBoxes[space.id] = owner;
  const indexToRemove = freeBoxes.indexOf(space);
  freeBoxes.splice(indexToRemove, 1);
  space.removeEventListener('click', changeToX);
}
```

Update `takenBoxes` with the owner each time a space is assigned.

```
// Returns 'x', 'o', or null for no winner yet.
function getWinner() {
  // Check rows
  let rowResult = checkBoxes('one', 'two', 'three') ||
      checkBoxes('four', 'five', 'six') ||
      checkBoxes('seven', 'eight', 'nine');

  // Check columns
  let colResult = checkBoxes('one', 'four', 'seven') ||
      checkBoxes('two', 'five', 'eight') ||
      checkBoxes('three', 'six', 'nine');

  // Check diagonal
  let diagonalResult = checkBoxes('one', 'five', 'nine') ||
      checkBoxes('three', 'five', 'seven');
  return rowResult || colResult || diagonalResult;
}
```

```
function checkBoxes(one, two, three) {
  if (takenBoxes[one] !== undefined &&
      takenBoxes[one] === takenBoxes[two] &&
      takenBoxes[two] === takenBoxes[three]) {
    return takenBoxes[one];
  }
  return null;
}
```

Find winner by checking rows, columns and diagonal spaces

(CodePen)

```
  </div>
  <div id="results"></div>
</body>
```

```
function displayWinner() {
  const winner = getWinner();

  const resultContainer = document.querySelector('#results');
  const header = document.createElement('h1');
  if (winner === 'x') {
    header.textContent = 'You win!';
  } else if (winner === 'o'){
    header.textContent = 'Computer wins';
  } else {
    header.textContent = 'Tie';
  }
  resultContainer.appendChild(header);
```

Create a results div and add results to the div (CodePen)

Attach "data" to divs?

```
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div id="one"></div>
    <div id="two"></div>
    <div id="three"></div>

    <div id="four"></div>
    <div id="five"></div>
    <div id="six"></div>

    <div id="seven"></div>
    <div id="eight"></div>
    <div id="nine"></div>
  </div>
</body>
```

Wouldn't it be nicer if we could operate on numbers instead of string ids?

But we can't have numeric IDs…

Is there some way to attach additional "data" to an element?

Data attributes

You can assign special data-* attributes to HTML elements to give associate additional data with the element.

`data-`***your-name***`="`***Your Value***`"`

```
<article
  id="electriccars"
  data-columns="3"
  data-index-number="12314"
  data-parent="cars">
...
</article>
```

Data attributes in JavaScript

You can access your custom-defined data attributes via the `dataset` object on the DOM object:

```
var article = document.getElementById('electriccars');

article.dataset.columns // "3"
article.dataset.indexNumber // "12314"
article.dataset.parent // "cars"
```

- Dash-separated words turn to camel case, e.g. `data-index-number` in HTML is `dataset.indexNumber` in JS
- **Aside:** Data attributes are returned as strings, but you can cast them to `Number` via `parseInt`

Data attributes in CSS

You can also style data attributes in CSS:

`[data-`***variable-name***`]` or
`[data-`***variable-name***`='`***value***`']` or
element`[data-`***variable-name***`]` etc

```
article[data-columns='3'] {
  width: 400px;
}
article[data-columns='4'] {
  width: 600px;
}
```

```html
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div data-index="0"></div>
    <div data-index="1"></div>
    <div data-index="2"></div>

    <div data-index="3"></div>
    <div data-index="4"></div>
    <div data-index="5"></div>

    <div data-index="6"></div>
    <div data-index="7"></div>
    <div data-index="8"></div>
  </div>
  <div id="results"></div>
</body>
```

[Final Solution CodePen]

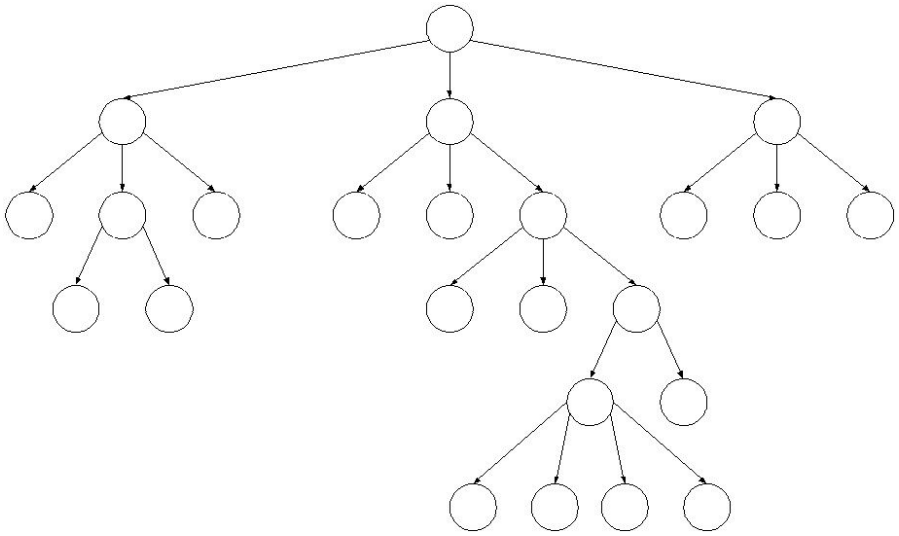
```
const index = parseInt(space.dataset.index);
takenBoxes[index] = owner;
```

Understanding the DOM

DOM Nodes

If the DOM is a tree composed of Nodes…

Q: Does that mean a Node in the DOM has child pointers like the trees we learned about in 106B?

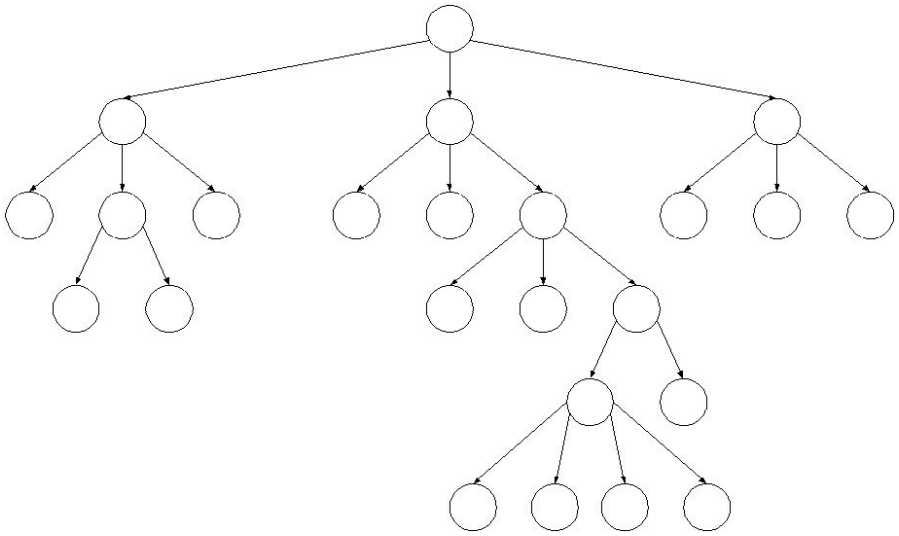


DOM Nodes

If the DOM is a tree composed of Nodes…

Q: Does that mean a Node in the DOM has child pointers like the trees we learned about in 106B?

A: Yes!



Node properties

Property	Description
textContent	The text content of a node and its descendants. (This property is writeable)
childNodes	An array of this node's children (empty if a leaf)
parentNode	A reference to this node's parent Node

```
<body>
    <h1>My favorites</h1>
    <section>
        <p>Strawberries</p>
        <p>Chocolate</p>
    </section>
</body>
```

What's the **parentNode** of **<section>**?

parentNode

```
> section = document.querySelector('section');
<· ▶<section>…</section>
> section.parentNode
<· ▶<body>…</body>
```

```
<body>
    <h1>My favorites</h1>
    <section>
        <p>Strawberries</p>
        <p>Chocolate</p>
    </section>
</body>
```

The **parentNode** of **<section>** is **<body>**.

What are the **childNodes** of **<section>**?

childNodes

```
> section = document.querySelector('section');
<- ▶<section>…</section>
> section.childNodes
<- ▶[text, p, text, p, text]
> section.childNodes.length
<- 5
```

???

```
<body>
    <h1>My favorites</h1>
    <section>
        <p>Strawberries</p>
        <p>Chocolate</p>
    </section>
</body>
```

Why does `section` have 5 children, not 2?!

TextNode

In addition to `Element` nodes, the DOM also contains `Text` nodes. All text present in the HTML, **including whitespace**, is contained in a text node:

```
<body>
    <h1>My favorites</h1>
    <section>
        <p>Strawberries</p>
        <p>Chocolate</p>
    </section>
</body>
```

TextNode

All text present in the HTML, **including whitespace**, is contained in a `Text` node:

```
<body>
    <h1>My favorites</h1>
    <section>
        <p>Strawberries</p>
        <p>Chocolate</p>
    </section>
</body>
```

DOM and Text nodes

The DOM is composed of `Node`s, and there are several subtypes of `Node`.

- `Element`: HTML (or SVG) elements in the DOM
- `Text`: Text content in the DOM, including whitespace
  - `Text` nodes cannot contain children (are always leafs)
- `Comment`: HTML comments
- (more / DOM visualizer)

The type of a node is stored in the `nodeType` property

Traversing the DOM

Q: How would we print out all nodes in the DOM?

Traversing the DOM

Q: How would we print out all nodes in the DOM?

A: Recursively walk the DOM tree:

```
function walkTree(root, level) {
  if (root.nodeType == Node.TEXT_NODE) {
    console.log(level + 'text:' + root.textContent);
  } else {
    console.log(level + root.nodeName);
  }
  for (const child of root.childNodes) {
    walkTree(child, level + "    ");
  }
}
walkTree(document.querySelector('html'), "");
```

What's the point?

- If we have document.querySelector that lets us get elements in the DOM…
- And if we can change the HTML as necessary to add classes/ids/elements/etc to select the right things…

Q: When would we ever want to traverse the DOM?

What's the point?

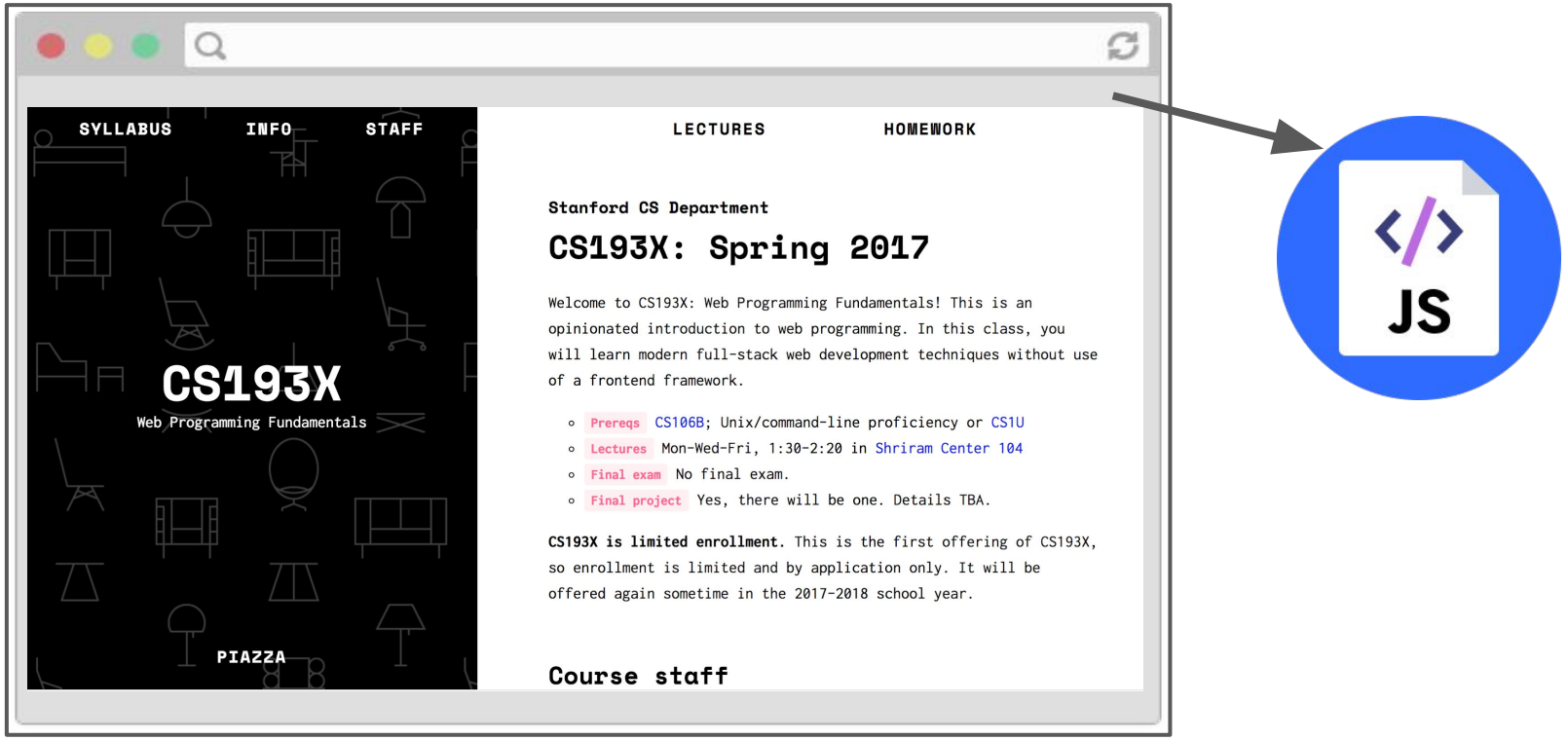
- If we have document.querySelector that lets us get elements in the DOM…
- And if we can change the HTML as necessary to add classes/ids/elements/etc to select the right things…

Q: When would we ever want to traverse the DOM?

A: Pretty much only in browser extensions or the Web Console (i.e. manipulating someone else's page)

Browser extensions

- Add-on that extends the functionality of the browser
- A piece of JavaScript that is injected into the webpage before or after it has loaded



Content scripts

- A type of extension that runs in the context of a web browser, meaning it can access the DOM of the page on which it's loaded
  - Concept is the same in Chrome, Firefox, probably other browsers

Usually composed of:

- `manifest.json`: Contains title, settings, etc for the script
- `page.js`: The extension file

We are using a **content script** in HW2, Part 2

Example `manifest.json`

```
{
  "name": "Empty Chrome Extension",
  "version": "1.0",
  "description": "",
  "content_scripts": [
    {
      "matches": ["<all_urls>"],
      "js": ["page.js"]
    }
  ],
  "icons": {
    "16": "pizza.png",
    "48": "pizza.png",
    "128": "pizza.png"
  },
  "manifest_version": 2
}
```

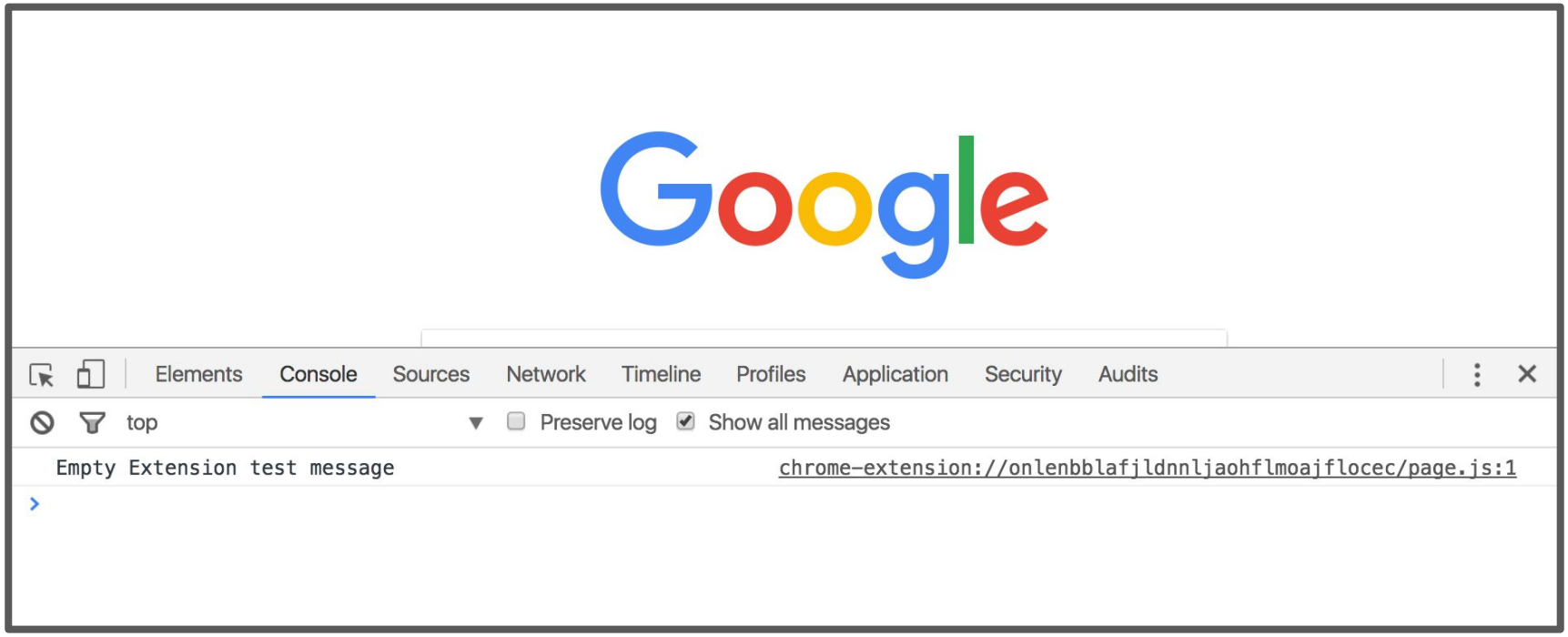
Example `page.js`

page.js

```
1  console.log('Empty Extension test message');
2
```

Result

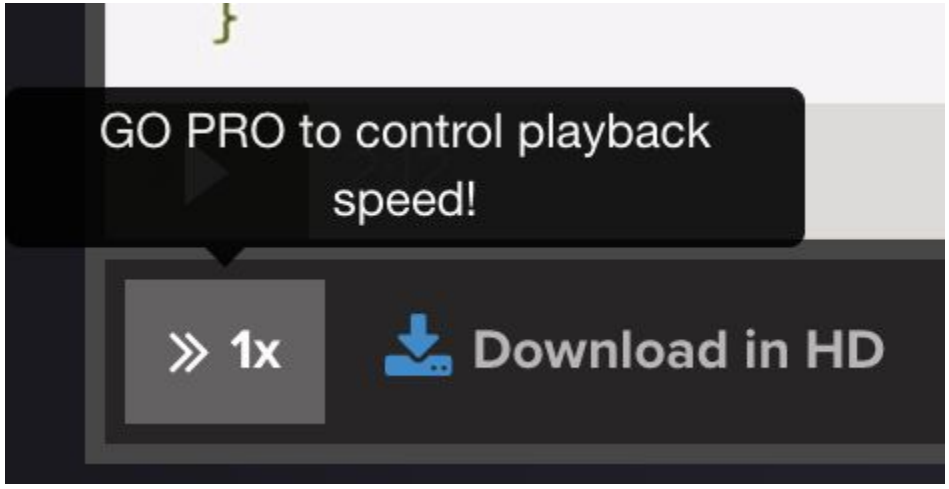
When this extension is loaded, the console message appears in the Web Inspector for every page:

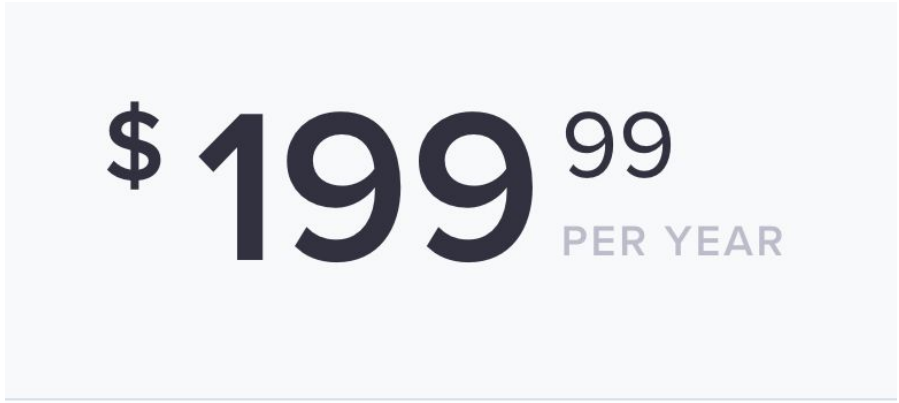


Hacks and Mischief

Example: Egghead

Dan Abromov's Redux videos





Example: Egghead

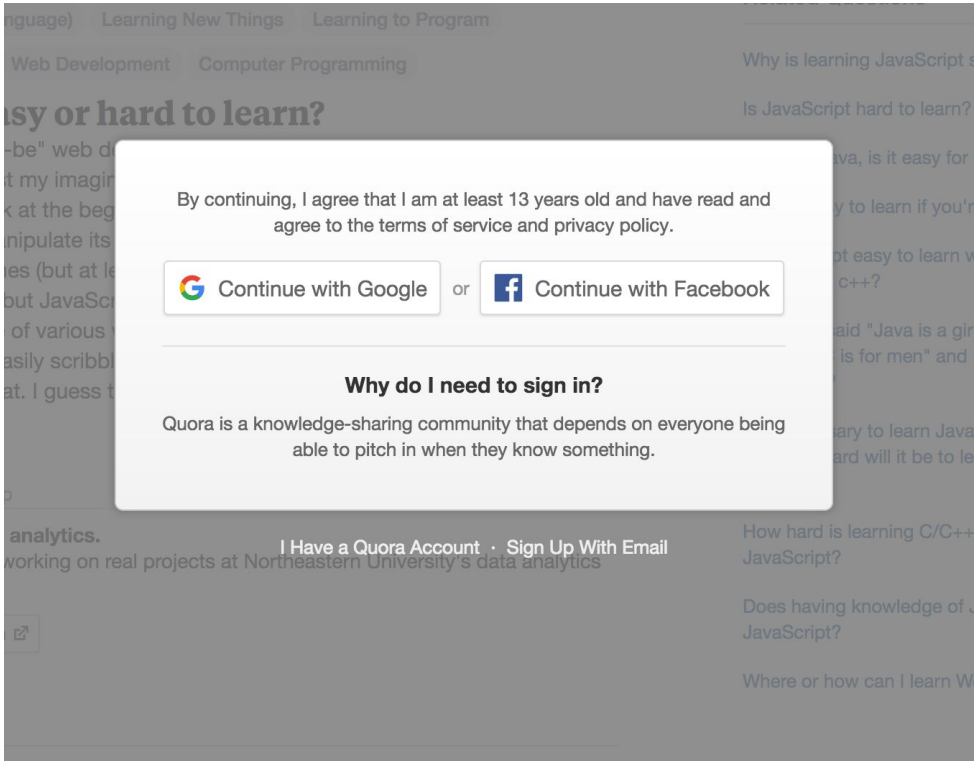
```
> document.querySelector('video')
<- ▶<video id="wistia_simple_video_25" crossorigin="anonymous" preload="me
   "background: transparent; display: block; height: 100%; position: static
> document.querySelector('video').playbackRate = 2
<- 2
> document.querySelector('video').playbackRate = 1.5
<- 1.5
>
```

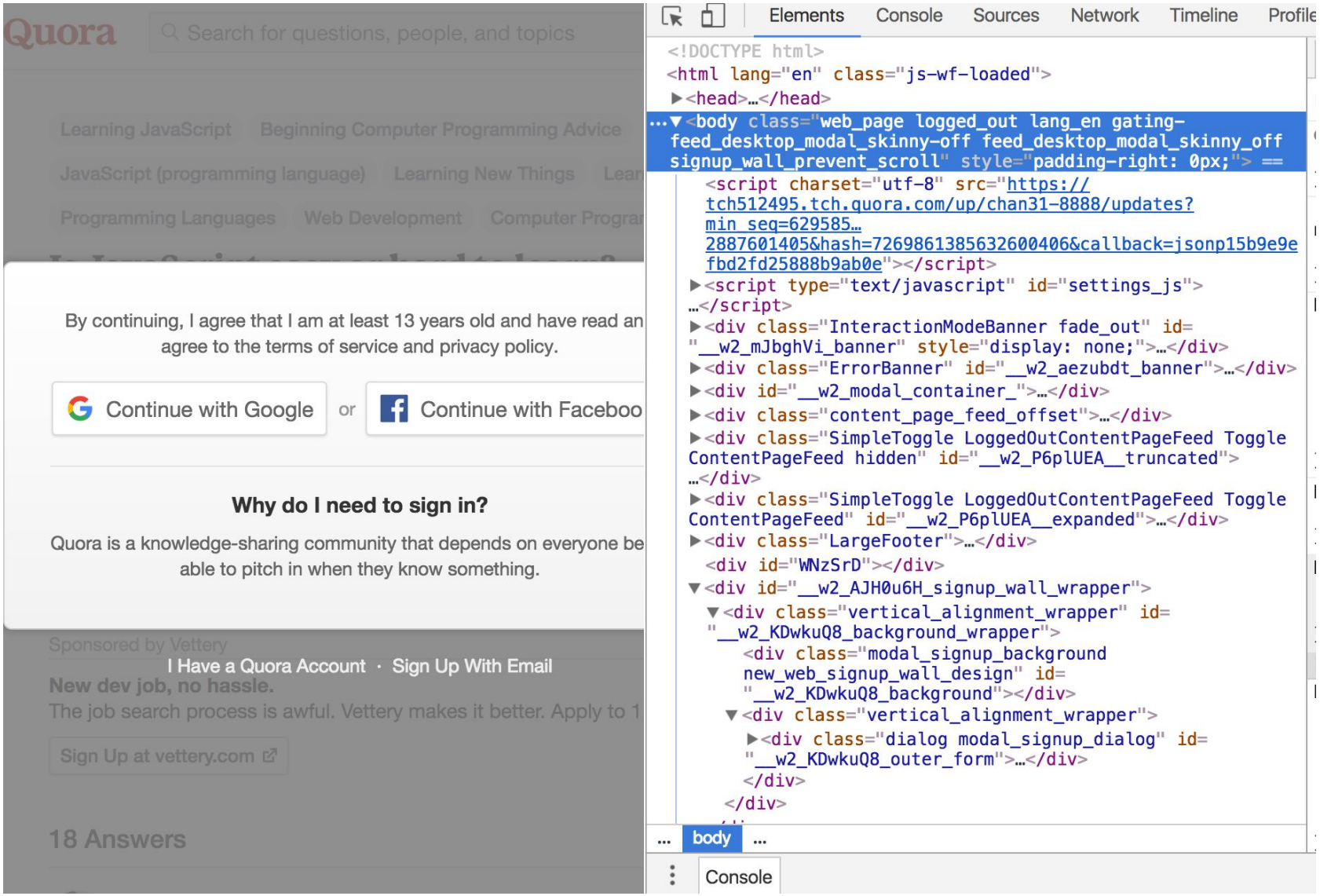
We can access the DOM via `querySelector` in the Web Inspector. Useful for debugging, and for speeding up videos.

(MDN for `playbackRate`)

Example: Quora signin wall

https://www.quora.com/Why-is-learning-JavaScript-so-hard




Quora
Search for questions, people, and topics
Learning JavaScript
Beginning Computer Programming Advice
JavaScript (programming language)
Learning New Things
Programming Languages
Web Development
By continuing, I agree that I am at least 13 years old and have read an
agree to the terms of service and privacy policy.
Continue with Google
or
Continue with Faceboo
Why do I need to sign in?
Quora is a knowledge-sharing community that depends on everyone be
able to pitch in when they know something.
Sponsored by Vettery
I Have a Quora Account · Sign Up With Email
New dev job, no hassle.
The job search process is awful. Vettery makes it better. Apply to 1
Sign Up at vettery.com
18 Answers
Elements
Console
Sources
Network
Timeline
Profile
<!DOCTYPE html>
<html lang="en" class="js-wf-loaded">
▶<head>…</head>
…▼<body class="web_page logged_out lang_en gating-
feed_desktop_modal_skinny-off feed_desktop_modal_skinny_off
signup_wall_prevent_scroll" style="padding-right: 0px;"> ==
<script charset="utf-8" src="https://
tch512495.tch.quora.com/up/chan31-8888/updates?
min_seq=629585…
2887601405&hash=726986138563260040 6&callback=jsonp15b9e9e
fbd2fd25888b9ab0e"></script>
▶<script type="text/javascript" id="settings_js">
…</script>
▶<div class="InteractionModeBanner fade_out" id=
"__w2_mJbghVi_banner" style="display: none;">…
▶ …
▶ …
▶ …
▶<div class="SimpleToggle LoggedOutContentPageFeed Toggle
ContentPageFeed hidden" id="__w2_P6plUEA__truncated">
…
▶<div class="SimpleToggle LoggedOutContentPageFeed Toggle
ContentPageFeed" id="__w2_P6plUEA__expanded">…
▶ …

▼
▼<div class="vertical_alignment_wrapper" id=
"__w2_KDwkuQ8_background_wrapper">
<div class="modal_signup_background
new_web_signup_wall_design" id=
"__w2_KDwkuQ8_background">
▼
▶<div class="dialog modal_signup_dialog" id=
"__w2_KDwkuQ8_outer_form">…

…
body
…
Console

Quora

Search for questions, people, and topics

Sign In

Learning JavaScript | Beginning Computer Programming Advice | JavaScript (programming language) | Learning New Things | Learning to Program | Programming Languages | Web Development | Computer Programming

Is JavaScript easy or hard to learn?

I'm a 15 year old "want-to-be" web developer and I have dreams of using a plethora of web languages to manifest my imagination onto the Internet. They're big dreams for me, but unfortunately I'm stuck at the beginning mulling over the semantic structure of JavaScript and how to manipulate its use for my whim. I've found it at sometimes frustrating and at sometimes (but at less times) absolutely magical. I've gotten over HTML quickly, and CSS as well; but JavaScript is by far obviously the hardest of all three. I have looked at the source code of various websites and become boggled at how developers for that specific website easily scribble down all that code. I tell myself sometimes that I will never be able to do that. I guess the essence of my question is: Is JavaScript easy or hard to learn?

18 Answers

Glen Murphy, JS Engineer@Google (no longer)

Related Questions

- Why is learning JavaScript so hard?
- Is JavaScript hard to learn?
- Is Javascript easy to learn when I already can program in c++?
- If I know Java, is it easy for me to learn JavaScript?
- Is PHP easy to learn if you're used to JavaScript?
- My friend said "Java is a girl's programming language and C++/C is for men" and I feel offended. How can I get over it?
- Is it necessary to learn JavaScript if I know Ruby? And how hard will it be to learn JavaScript if I know Ruby?
- How hard is learning C/C++ considering you know JavaScript?
- What is the best approach to learn Hybris?
- Does having knowledge of Java make it easy to learn JavaScript?

Quora Chrome Extension

You can make a Chrome extension to automate this:

manifest.json

```
  "matches": ["*://www.quora.com/*"],
  "js": ["page.js"]
}
```

page.js

```
document.body.classList.remove('signup_wall_prevent_scroll');
const nagScreen = document.querySelector('.vertical_alignment_wrapper');
if (nagScreen) {
  nagScreen.style.display = 'none';
}
```

Aside: `style` attribute

Every `HTMLElement` also has a `style` attribute that lets you set a style directly on the element:

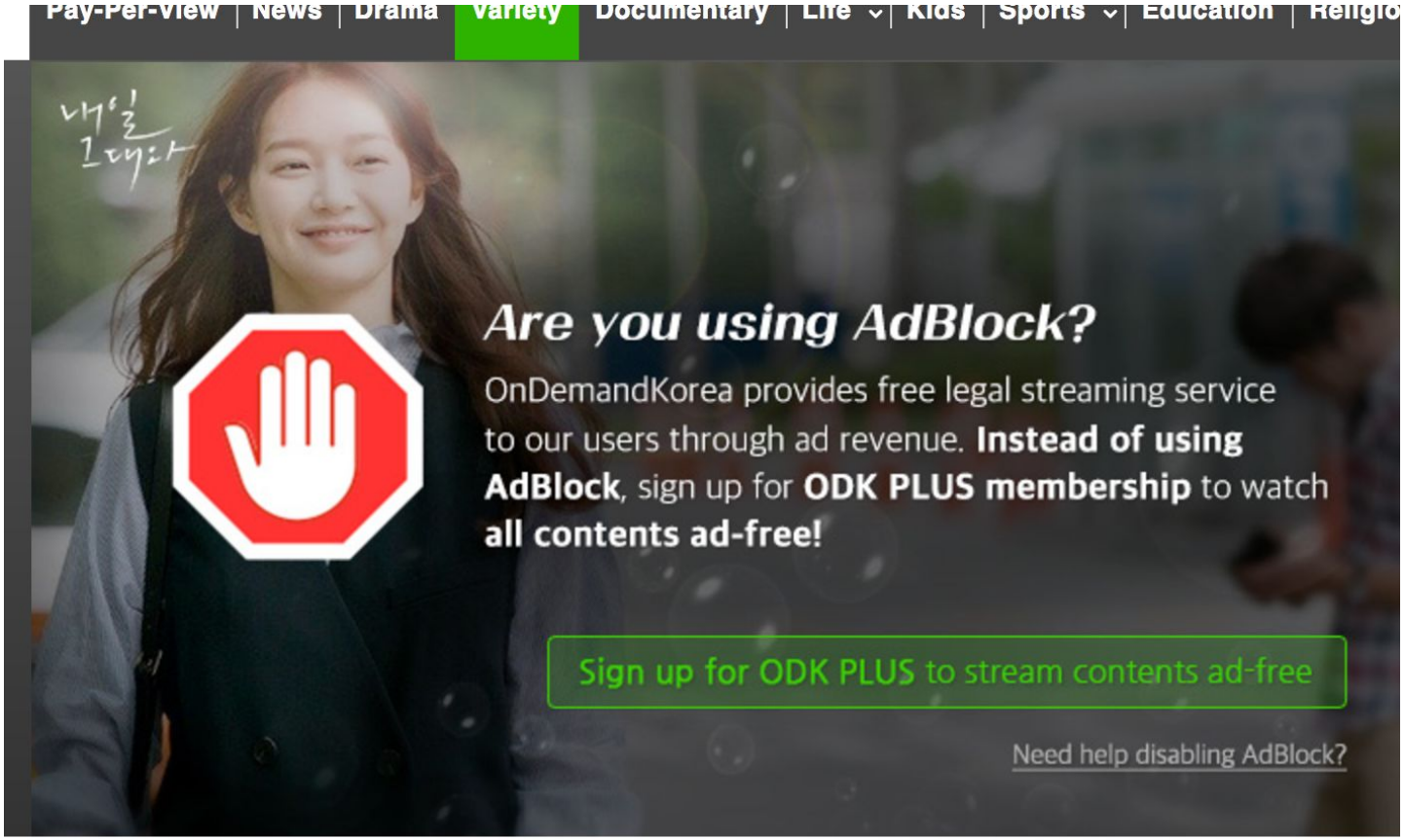
```
nagScreen.style.display = 'none';
```

Generally **you should not use this property**, as adding and removing classes via `classList` is a better way to change the style of an element via JavaScript

(But for extensions/hacking, `style` can be useful!)

Example: Adblock block

http://www.ondemandkorea.com/kpop-star-season-6-seoul-qualifier.html



View-source, search for "adblock"...

view-source:www.ondemandkorea.com/kpop-star-season-6-e20161120.html

adblock 2 of 10

```
        });

        var view_seconds = 0;
        var last_watched_position = 0;
        var view_timer = setInterval(function() {
            if(jwplayer().getState() == 'playing')
            {
                if(jwplayer().getPosition() != last_watched_position)
                {
                    view_seconds += 1;
                    if(view_seconds > 0 && view_seconds%300 == 0)
                    {
                        $p("send","watched_5_min");
                        ga('set', 'dimension5', 'sbs_sub');
                        ga('send', 'event', 'Video/variety', 'ViewhourLog', 'kpop-star-season-6-e20161120', 300);
                    }
                }
            }
            last_watched_position = jwplayer().getPosition();
        }, 1000);

                            jwplayer().on('adBlock',function(event) {
                        jwplayer().remove();
                        $("#odk_player").html('<a id="adblock_signup_in_player" href="https://www.ondemandkorea.com/odk-plus-benefits" target="_blank"
style="position: absolute;margin-top: 291px;margin-left: 262px;width: 406px;height: 38px;"></a><a href="http://www.ondemandkorea.com/blog/?p=2085"
target="_blank" style="position: absolute;margin-top: 341px;margin-left: 497px;width: 172px;height: 20px;"></a><img
src="http://www.ondemandkorea.com/images/player_adblock_0_en.jpg">');
                    });
                            if(offset != false)
        {
            jwplayer().on('firstFrame',function(){
                jwplayer().seek(offset);
            });
        }
                        var overlay_shown = true;
                var overlay_showing_now = false;
                var ad_playing = false;

                jwplayer("odk_player").on('time',function(e){
                    if(overlay_shown == false && ad_playing == false)
                    {
                        if(overlay_showing_now == false && (e.position > 0.5 && e.position < 4.5))
                        {
```

Lol global scope

```
jwplayer().on('adBlock', function(event) {
   jwplayer().remove();
   $("#odk_player").html('<a id="adblock_signup_in_player"
href="https://www.ondemandkorea.com/odk-plus-benefits" target="_blank"
style="position: absolute;margin-top: 291px;margin-left: 262px;width:
406px;height: 38px;"></a><a
href="http://www.ondemandkorea.com/blog/?p=2085" target="_blank"
style="position: absolute;margin-top: 341px;margin-left: 497px;width:
172px;height: 20px;"></a><img
src="http://www.ondemandkorea.com/images/player_adblock_0_en.jpg">');
});
```

Note that this is using the jQuery library, which I **do not** recommend you use

jQuery on?

Again, I **do not** recommend you use jQuery. But if we're trying to understand someone else's jQuery code...

.on()

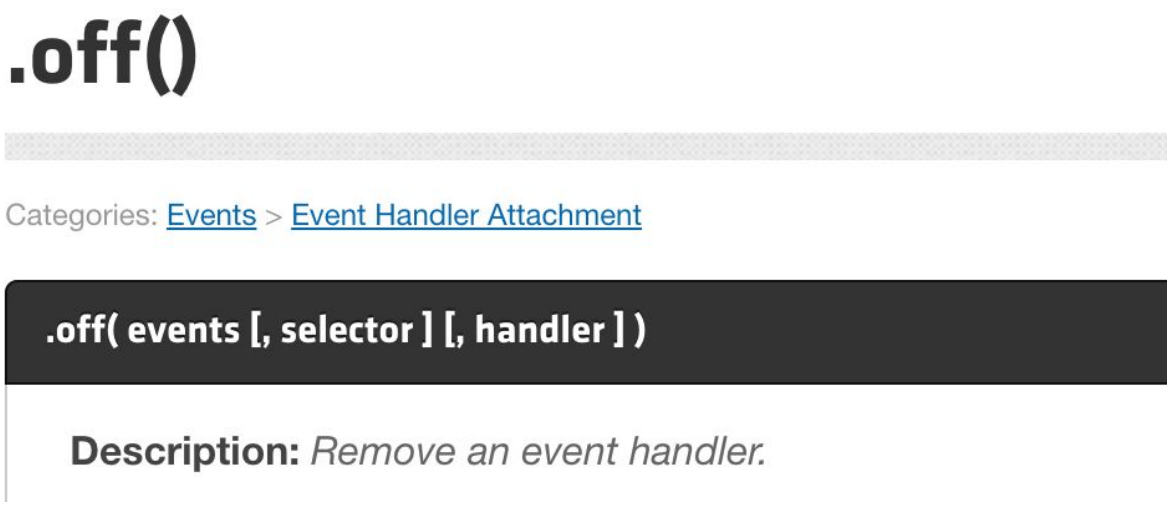
Categories: Events > Event Handler Attachment

.on(events [, selector] [, data], handler)

Description: *Attach an event handler function for one or more events to the selected elements.*

We see that `.on()` is essentially `addEventListener`

jQuery off



And `.off()` is essentially `removeEventListener`

Lol global scope

What if we try this...

page.js

```
1  const child = document.createElement('script');
2  child.textContent += 'jwplayer().off(\'adBlock\');';
3  document.body.appendChild(child);
4
```

Aside: inline `<script>`

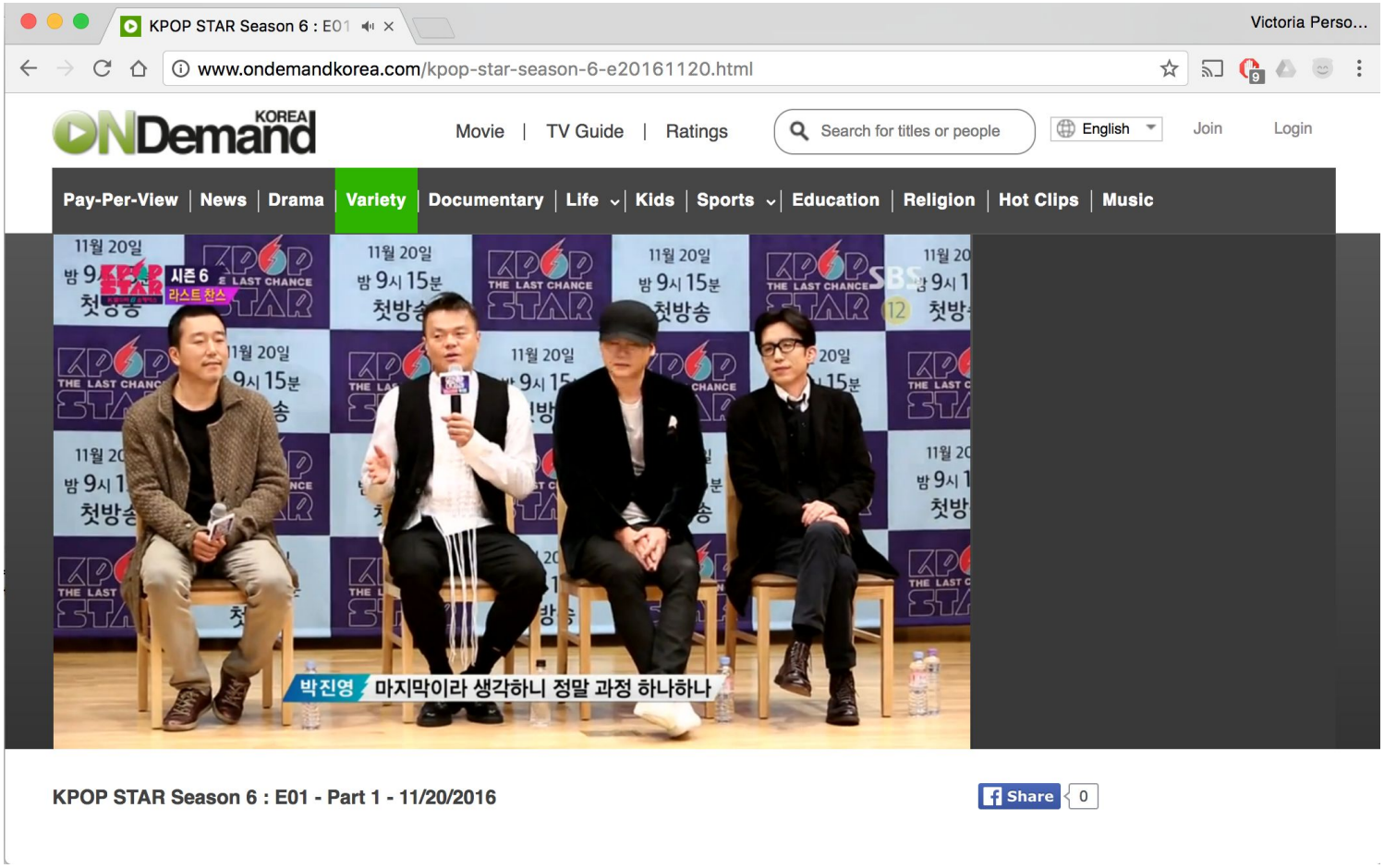
You can put JavaScript directly in your HTML by adding it to the body of a <script> tag:

```
<script>console.log('hello');</script>
```

Generally **you should not use do this**, as it's a violation of Separation of Concerns, mixing behavior (JS) in content (HTML)

(But for extensions/hacking, adding JavaScript directly in a `<script>` tag can be useful!)

Success!


KPOP STAR Season 6 : E01
Victoria Perso...
www.ondemandkorea.com/kpop-star-season-6-e20161120.html
ONDemand KOREA
Movie | TV Guide | Ratings
Search for titles or people
English
Join
Login
Pay-Per-View | News | Drama | Variety | Documentary | Life | Kids | Sports | Education | Religion | Hot Clips | Music
박진영 마지막이라 생각하니 정말 과정 하나하나
KPOP STAR Season 6 : E01 - Part 1 - 11/20/2016
Share 0