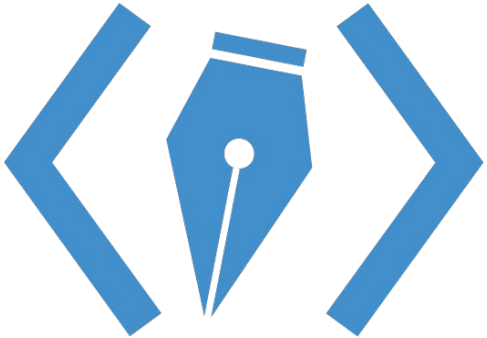
Week 10

Event Handlers and Conditionals



can someone who knows javascript help me please i'm desperate

1:56 AM · 8/12/19 · Twitter for iPhone

16k 485

BEST

u/Skizm • 2mo

```
if(goingToCrashIntoEachOther)
{ dont(); }
```

When the user clicks on [this button/link/image], then change [this HTML/CSS].

Review

```
let food =
  document.getElementById("food-list");
```

What does food return?

```
let food =
  document.querySelector("#food-list");
```

What does food return?

```
food.classList;
```

What does this return?

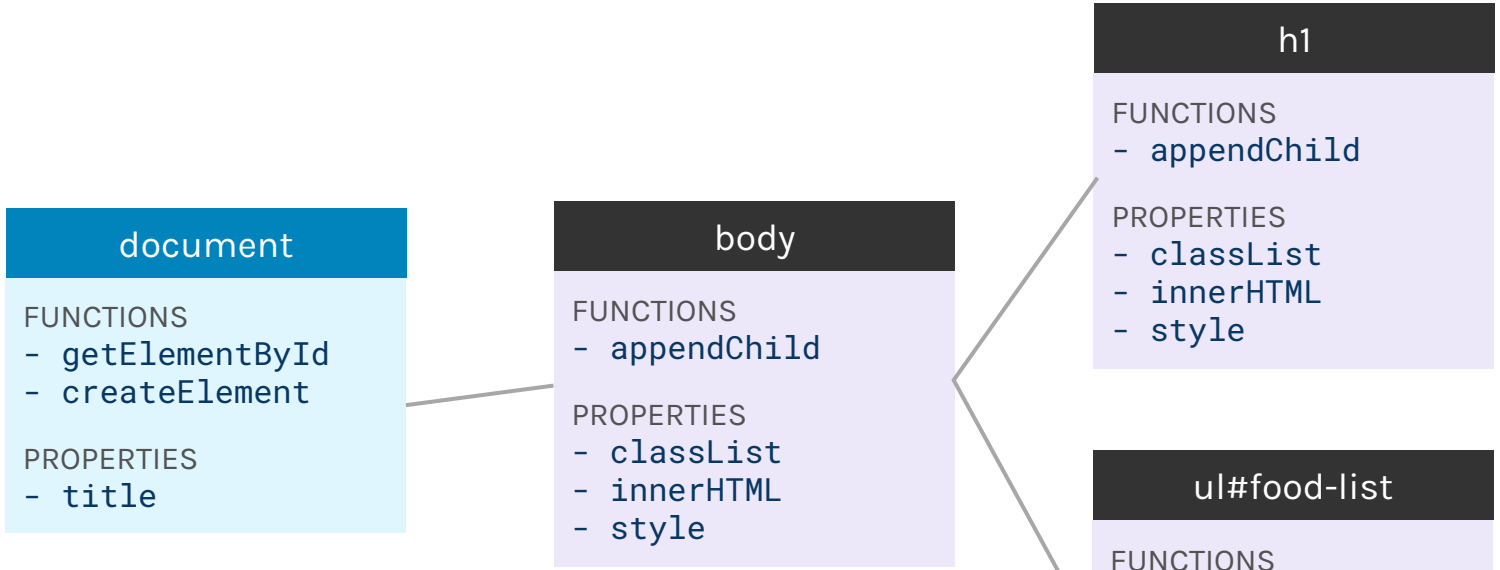
```
food.classList.add("foo");
```

What does this do?

```
food.classList.remove("bar");
```

What does this do?

DOM and Event Handlers



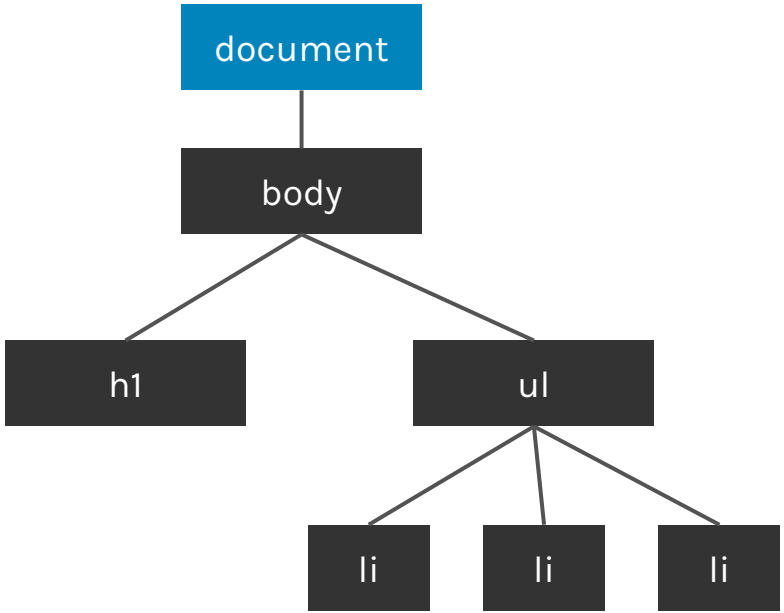
The DOM: Document Object Model

When the HTML page is loaded, JavaScript creates its own representation of the page, which is made up of objects.

Each HTML tag is an object.

```
document.getElementById("food-list")
```

What we're saying is that from our **document object model,** find the element that has the id **#food-list.**



JavaScript is event-driven.

Events -- actions that you take, as a user, when you're interacting with a website (hovering, clicking, typing, etc)

Examples -- `onmouseover`, `onclick`, `onkeypress`

Responsive ▼ 656 × 807 100% ▼ Online ▼

Applications are now closed. No late applications are being accepted at this time (not even if you email us!), but check back at the end of the semester!

Got it!

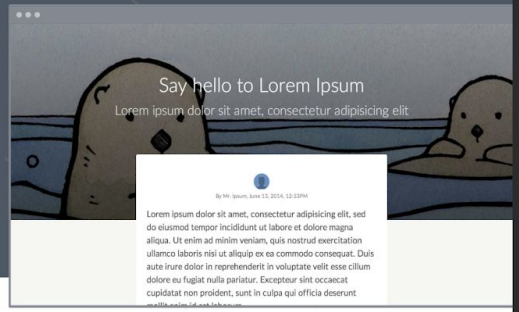
Web Design DeCal

A 2-unit course for anyone who wants to create a **beautiful** website.

About

Log in to WDD Portal | Learn More

Curriculum



Schedule

Elements | Console | Sources | Network | Performance | Memory | Application

```
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  ▶ <head>…</head>
  ▼ <body data-page="cover" data-window="front" data-front="cover" data-back="about">
    ▼ <div id="viewport" class="viewport-cover">
      ▶ <div id="banner">…</div>
      ▼ <div class="content">
        ▼ <div id="cover">
          ▶ <div id="cover-header">…</div>
          ▼ <div id="cover-footer">
              <div class="slide-overlay"></div> == $0
            </div>
          </div>
        ▶ <div id="about" class="slide">…</div>
        ▶ <div id="curriculum" class="slide">…
        </div>
        ▶ <div id="schedule">…</div>
        ▶ <div id="instructor" class="slide">…
        </div>
        ▶ <div id="faq" class="slide">…</div>
        ▶ <div id="apply" class="slide">…</div>
      </div>
    </div>
  ▶ <div class="window-container window-container-scaled" style="top: 243.1px;">…
  </div>
  ▶ <div id="apply-container">…</div>
  ▶ <div id="nav" class="nav-active-cover">…
  </div>
```

… div #cover #cover-footer div.slide-overlay

Styles | Computed | Event Listeners

Ancestors All ▼ Framework listeners

- DOMContentLoaded
- load
- mouseup
- readystatechange
- resize
- scroll

Console | What's New × | Search

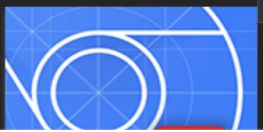
Highlights from the Chrome 78 update

Multi-client support in the Audits panel

Use the Audits panel in combination with other features, like Local Overrides or Request Blocking.

Payment Handler debugging

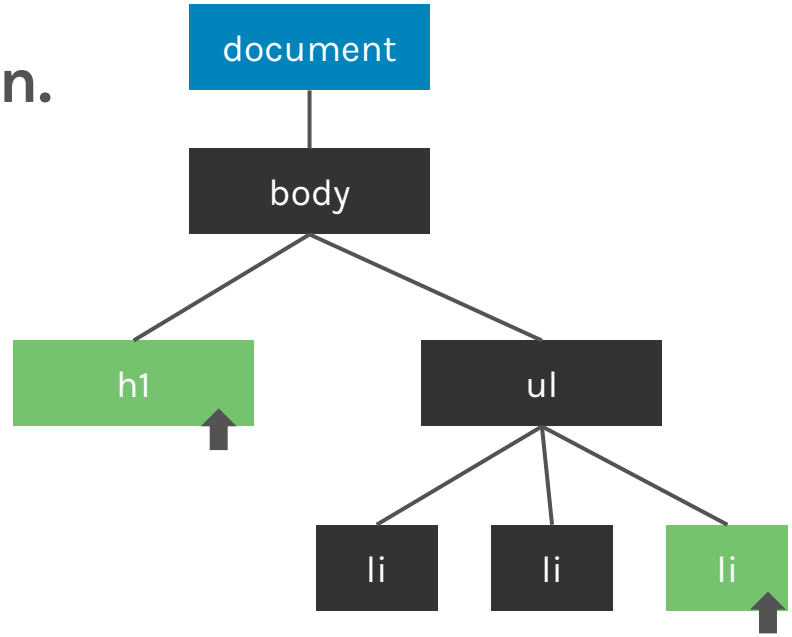
Record Payment Handler events for 3 days, even when DevTools is closed



With the DOM we can change different elements on our screen.

Previously with pseudo-selectors, we can only access the element we selected. In contrast, **when an event occurs,** we have access to the entire tree, meaning any element can be modified.

This is super useful for complex interactions, like buttons! **Example:** whenever we click on <li>, we can modify <h1>.



```
let buttonElement =
document.getElementById("popup-button");


function showGreeting() {

   alert("howdy!");

}

buttonElement.onclick = showGreeting;
```

Every DOM element object has an `onclick` function.

This is the function that will run when you click on the element, creating an onClick event.

```
let buttonElement =
document.getElementById("popup-button");


function showGreeting() {

   alert("howdy!");

}

buttonElement.onclick = showGreeting;
```

Note: `showGreeting` on the last line is without parentheses!

We don't want to call the function right now - we just want to tell the browser what function it should use later.

If we don't we'll call showGreeting and alert right away! Bad!

```
let buttonElement =
document.getElementById("popup-button");


function showGreeting() {

   alert("howdy!");

}

buttonElement.onmouseover = showGreeting;
```

Note: `showGreeting` can be called with different event handlers!

We can call the same function depending on when we want the user to see it -- **onmouseover, onkeypress, or onclick** are all valid use cases.

```
let buttonElement =
document.getElementById("popup-button");


buttonElement.onclick = function() {
   alert("howdy!");
};
```

Note: we can also assign functions directly.

These functions are without a name and are called **anonymous functions.**

Alternative Styling w/ JS

```
let buttonElement =
document.getElementById("button");

let boxElement =
document.getElementById("box”);


function changeColor(color) {

    boxElement.style.backgroundColor = color;

}

buttonElement.onclick = function() {

    changeColor("red");

}
```

We don’t have to add classes to modify CSS.

Note that we assign **style.property** to a string value -> our CSS reads text and so we use a string value to change the CSS property.

```
let buttonElement =
document.getElementById("button");

let boxElement =
document.getElementById("box”);


function changeColor(color) {

    boxElement.style.backgroundColor = color;

}

buttonElement.onclick = function() {

    changeColor("red");

}
```

We don’t have to add classes to modify CSS.

Note that we assign **style.property** to a string value -> our CSS reads text and so we use a string value to change the CSS property.

```
let buttonElement =
document.getElementById("button");

let boxElement =
document.getElementById("box”);


function changeColor(color) {

    boxElement.style.backgroundColor = color;

}

buttonElement.onclick = function() {

    changeColor("red");

}
```

We don’t have to add classes to modify CSS.

Here, we use anonymous functions to nest our action in our event handler. This is because we wait to change our color until we click!

Google: How do I change

[CSS property] with JavaScript?

Boolean Expressions

A conditional statement allows you to do different things based on the truthiness of a condition.

"If x is true, then do y!"

Equality operator → `===`, `!==`

Expressions that compare if two variables are the same **type** and **value.**

Other relational operators → `>`, `<`, `>=`, `<=`

Equality operator → `===`, `!==`

`1 + 5 === 6` evaluates to `true`

`1 === 2` evaluates to `false`

`1 + 5 !== 2` evaluates to `true`

Expressions give two possible values: true, false

Logic Operators → `&&`, `||`, `!`

Symbols that combine or modify expressions together to create a new expression.

`&&` (and), `||` (or), `!` (not)

and → &&

`(true && false)` evaluates to `false`

`(false && false)` evaluates to `false`

`(true && true)` evaluates to `true`

`if x and y, then true`

or → ||

`(true || false)` evaluates to `true`

`(false || false)` evaluates to `false`

`(true || true)` evaluates to `true`

if x or y, then true

not → !

`!(true)` evaluates to `false`

`!(false)` evaluates to `true`

not returns the opposite value of the expression

Combined Expressions

```
((1 + 2 === 3) && !(5 * 6 === 31))
```

What does this return?

Conditional Statements


Conditional
statements
if { … }
else { … }

```
if (condition) {
  # do something
}
```

A **condition is an expression** that evaluates to true or false.
If the condition is true, **do the thing inside the curly braces.**

The if statement

```
let x = 5;    assigns a value x
function numberEqual(number) {
    if (number === 5) {    does something if true
        console.log("Number is equal to 5!");
    }
    if (number > 4) {    does something if true
        console.log("Number is greater than 4!");
    }
}
```

What does **numberEqual(5)** do?

What does **numberEqual(4)** do?

What does **numberEqual(3)** do?

The if statement

```
let x = 5;    assigns a value x
function numberEqual(number) {
    if (number === 5) {    does something if true
        console.log("Number is equal to 5!");
    } else {    otherwise, do this instead
        console.log("Number is not equal to 5");
    }
}
```

What does **numberEqual(5)** do?

What does **numberEqual(4)** do?

What does **numberEqual(3)** do?

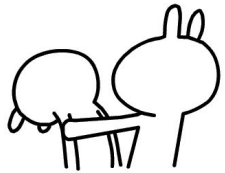
The if statement

```
let x = 5;    assigns a value x
function numberEqual(number) {
    if (number === 5) {    does something if true
        console.log("Number is equal to 5!");
    } else if (number > 4) {    otherwise do this
        console.log("Number is greater than 4!");
    } else {    if none of the above are true, do this
        console.log("Number is less than 5!");
    }
}
```

What does **numberEqual(6)** do?

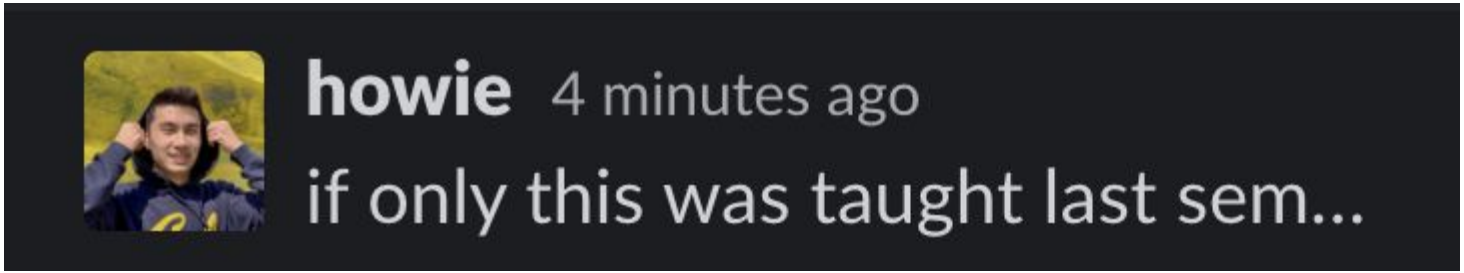
What does **numberEqual(5)** do?

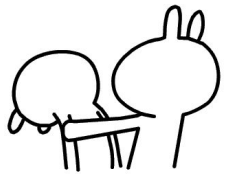
What does **numberEqual(4)** do?



wdd.io/go/js-event-demo

Solution: wdd.io/go/js-event-demo-sol


howie 4 minutes ago
if only this was taught last sem...



Questions?