FEWD Week 7 • Class 12: Functions & Events

Quick Review

- Where do you put ***your*** scripts?
- What's wrong with this?

```
if( $('div').is(visible) {
  $('div').css({background-color: red})
}
```

- What's missing here?

```
$('div').click(function{
  $(this).html(<p>You clicked me!</p>)
});
```

What We'll Cover

- Writing functions to make our code DRY
- Understanding how events work in depth
- How to stop events
- Using event delegation

Functions

Objectives: Functions

- Write functions with the proper syntax
- Create functions that return values
- Execute functions
- Understand the difference between named and anonymous functions

What is a Function?

A function is a block of code within our overall script that performs some task. We use functions to make our code more readable, reuseable and DRY.

You Already Know Functions

We've been using a type of function called an ***anonymous function*** inside of our event listeners.

Function Types

- **Anonymous Functions**: Anonymous functions are most often run when triggered by a specific event.
- **Named Functions**: Named functions are executed when called by name.
- **IIFE**: Immediately Invoked Function Expressions are run the moment the javascript engine encounters them.

Function Syntax

```
function functionName(arg1, arg2) {

  /* Code block of stuff to do when this function is called. */

}
```

- Functions start with the keyword `function`.
- Named functions are given a name that follows the function keyword.
- The function keyword or name is followed by `( )`, which may or **may not** contain any arguments.
- The entire code block is then wrapped in `{ }`

Say Hello!

```
function sayHello() {
  console.log('Hello!');
}
```

Calling Named Functions

```
function sayHello() {
  console.log('Hello!');
}

sayHello();
```

- ***Calling a function*** is just a way of saying make the function run or execute.
- We call named functions with the name followed by `( )` wherever we want the function to execute in our overall script.

Writing and Calling Named Functions

What about Arguments?

```
function sayHello(name) {
  console.log('Hello, ' + name + '!');
}

sayHello('Jen');
```

- Arguments are sort of like variables.
- Whatever value you pass to the argument is accessible with the argument name inside the function.

Arguments Are Awesome!

```
function sayHello(fname, lname) {
  console.log('Hello, ' + fname +' '+ lname + '!');
}

var first = 'Jen';
var last = 'Meade';

sayHello(first, last);       /* output: "Hello, Jen Meade!" */
sayHello('James', 'Bond');   /* output: "Hello, James Bond!" */
```

Arguments

Functions that Give Back

```
function convertText(string, type) {
  if (type === 'uppercase') {
    return string.toUpperCase();
  } else if (type === 'lowercase') {
    return string.toLowerCase();
  }
}

var lowerJen = convertText('Jen', 'lowercase');
```

- Functions can return a value to the caller using the `return` keyword.
- Mostly used with variables and inside other functions.

Temperature Converter

Events

Objectives: Events

- Understand how to work with events in Javascript and jQuery

What are Events?

Events are the things that happen outside of our script. They most often result from the user taking some action, such as: scrolling, pressing a key, mousing over a target, clicking a button, swiping, resizing the screen, etc.

Responding to Events

jQuery has a whole host of event listeners you can use. The basic pattern is:

```
$('thing-receiving-the-event').eventName(function(){

  // A function that is called when the event happens

});
```

Event Syntax

Some Events are Handled by the Browser

Some events already have a default behavior that is invoked by the browser. For example, when clicked, an anchor tag will automatically jump the user to the page or place in the current page specified in the href attribute.

We Can Prevent the Default

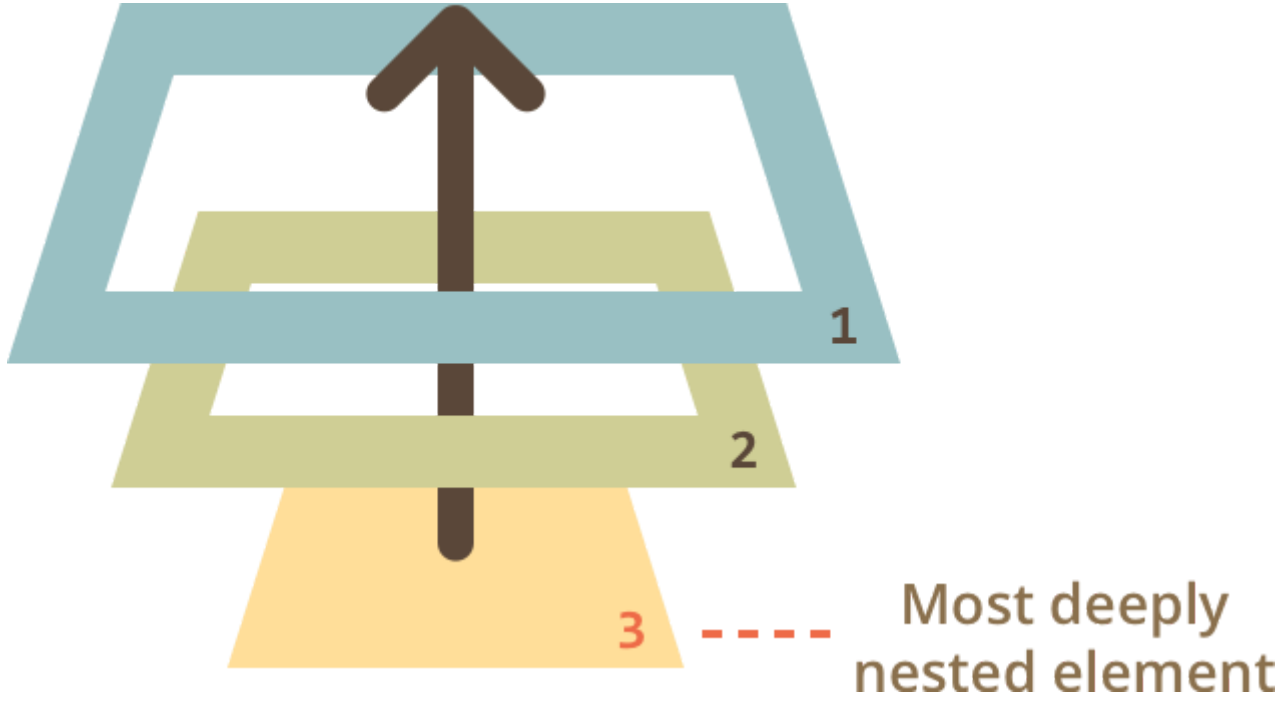
```
$('a').click(function(event){

  event.preventDefault(); /* Prevents the default link navigation */
  console.log('Link navigation prevented!');

});
```

Event Bubbling

Events in Javascript ***bubble***. When an event happens on an element, it first runs the handlers on it, then on its parent, then all the way up on other ancestors.



Event Bubbling

Stopping Bubbling

We can also prevent this behavior if we need to by passing in the event to the handler and using the stopPropagation() method.

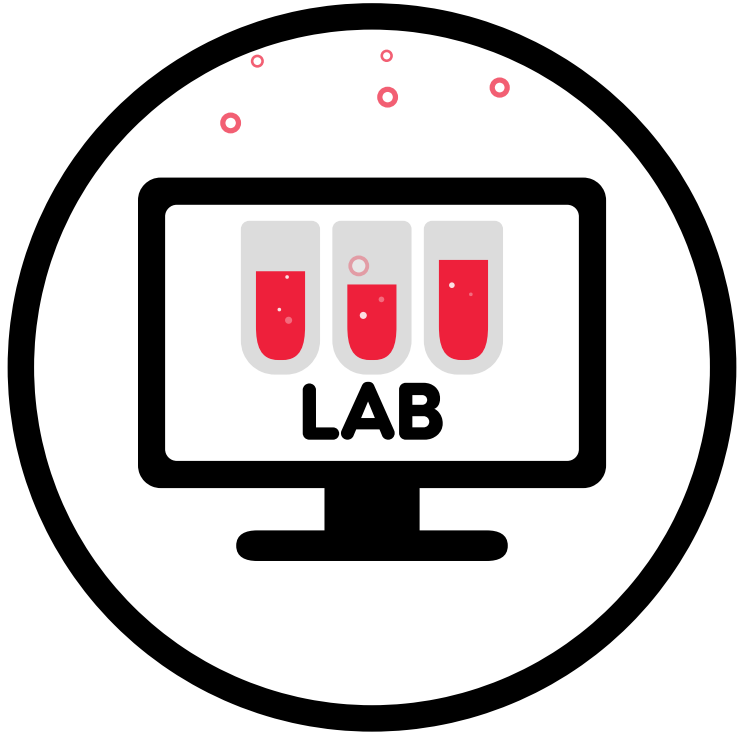
```
$('div').click(function(event) {
  event.stopPropagation();
  // Do some stuff here...
});
```

Event Delegation

Event delegation takes advantage of bubbling and allows us to use one click handler for multiple elements. This is much more efficient and it works with injected elements!

```
$('#div').on('click', 'button', function(event) {
    event.preventDefault();
    console.log($(this).text());
});
```

Event Delegation



Hello Badge

Go Build Awesome Things!