# CLIENT SIDE WEB DEVELOPMENT TEMA6: JQUERY INTRODUCTION

CFGS - DAW 2

# Index

# 1. Performing tasks on page load

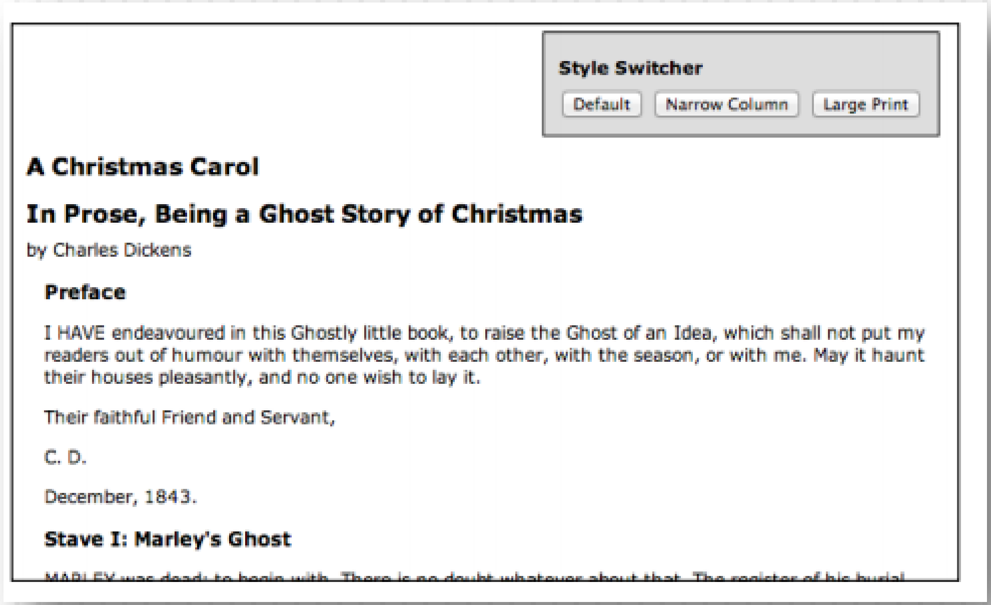
- We have already seen how to make jQuery react to the loading of a web page →$(document).ready()
- But …

¿Which is the difference between that one and the built-in window.onload? If you do not know it, find it.

# 2. Handling simple events

- Follow this example:

1. Download A63.rar and proceed as follows.
2. The first thing you will see looks like this:



Style Switcher

Default | Narrow Column | Large Print

**A Christmas Carol**

**In Prose, Being a Ghost Story of Christmas**

by Charles Dickens

**Preface**

I HAVE endeavoured in this Ghostly little book, to raise the Ghost of an Idea, which shall not put my readers out of humour with themselves, with each other, with the season, or with me. May it haunt their houses pleasantly, and no one wish to lay it.

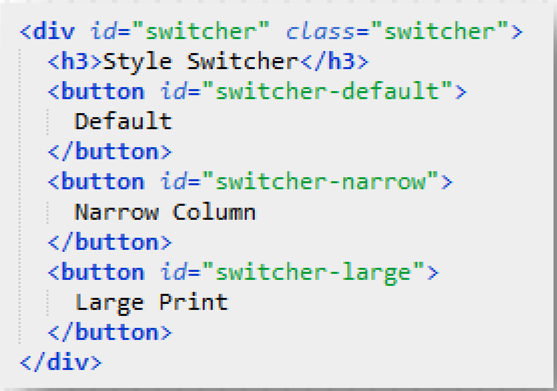
Their faithful Friend and Servant,

C. D.

December, 1843.

**Stave I: Marley's Ghost**

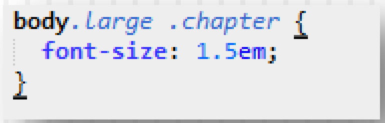
MARLEY was dead: to begin with. There is no doubt whatever about that. The register of his burial



```
<div id="switcher" class="switcher">
  <h3>Style Switcher</h3>
  <button id="switcher-default">
    Default
  </button>
  <button id="switcher-narrow">
    Narrow Column
  </button>
  <button id="switcher-large">
    Large Print
  </button>
</div>
```

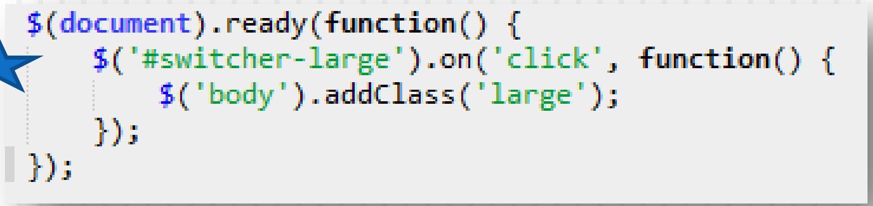
## 2. Handling simple events

3. To begin with, we'll make the Large Print button operate:
   a) We need a bit of CSS to implement our alternative view of the page as follows:



```
body.large .chapter {
  font-size: 1.5em;
}
```

   b) We want this to occur when the button is clicked, not when the page is loaded as we have seen so far. To do this, we'll introduce the ***.on()*** method.
      - This method allows us to specify any DOM event and to attach a behavior to it.
      - In this case, the event is called ***click***:



```
$(document).ready(function() {
    $('#switcher-large').on('click', function() {
        $('body').addClass('large');
    });
});
```

   c) Try it!

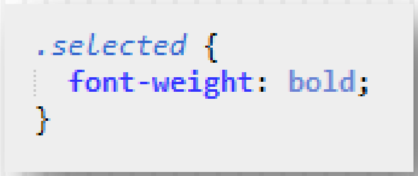
# 2. Handling simple events

A.6.3. Modify the example before in order it to apply similar handling to the other two buttons (Default and Narrow Column) to make them perform their tasks.
Hint: There is already a CSS rule called narrow defined, you only have to use it.

## 2. Handling simple events

- Our switcher is behaving correctly, but we are not giving the user any feedback about which button is currently active.
- Following with the same example as before:

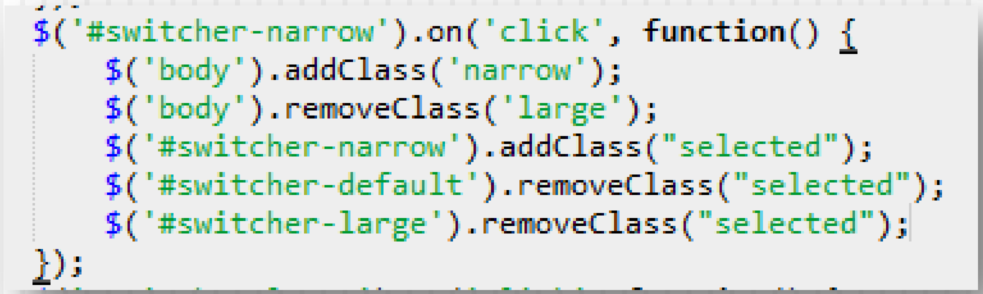
1. Our approach for handling this will be to apply the *selected* class to the button when it is clicked, and to remove this class from the other buttons.



```
.selected {
  font-weight: bold;
}
```

## 2. Handling simple events

2. The direct way will be to refer to each button by ID and applying or removing classes as necessary:

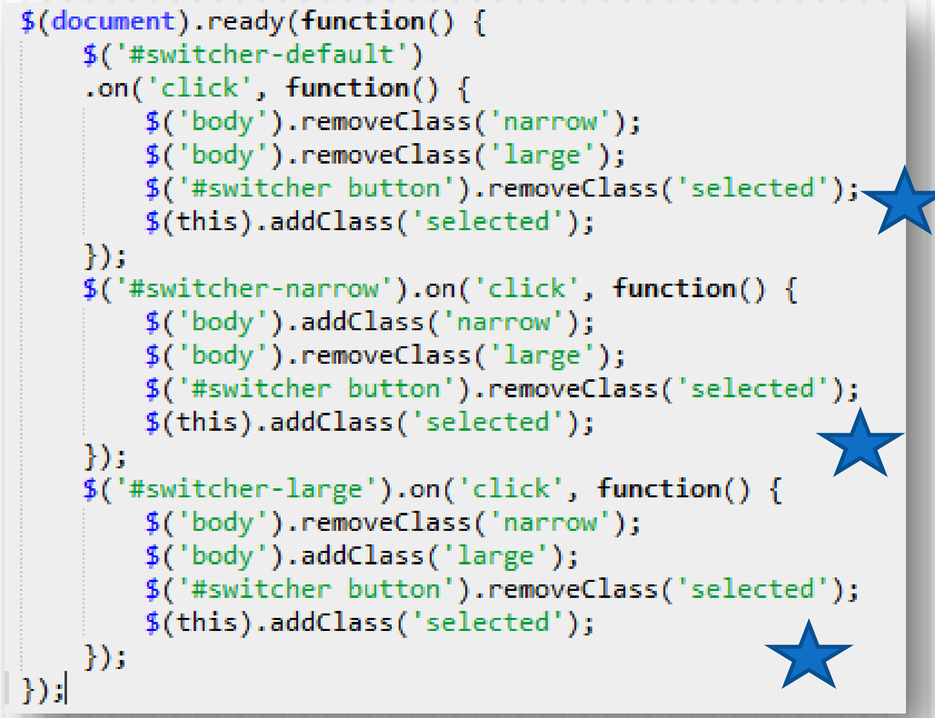


```
$('#switcher-narrow').on('click', function() {
    $('body').addClass('narrow');
    $('body').removeClass('large');
    $('#switcher-narrow').addClass("selected");
    $('#switcher-default').removeClass("selected");
    $('#switcher-large').removeClass("selected");
});
```

A.6.4. Do the same for all the buttons.

# 2. Handling simple events

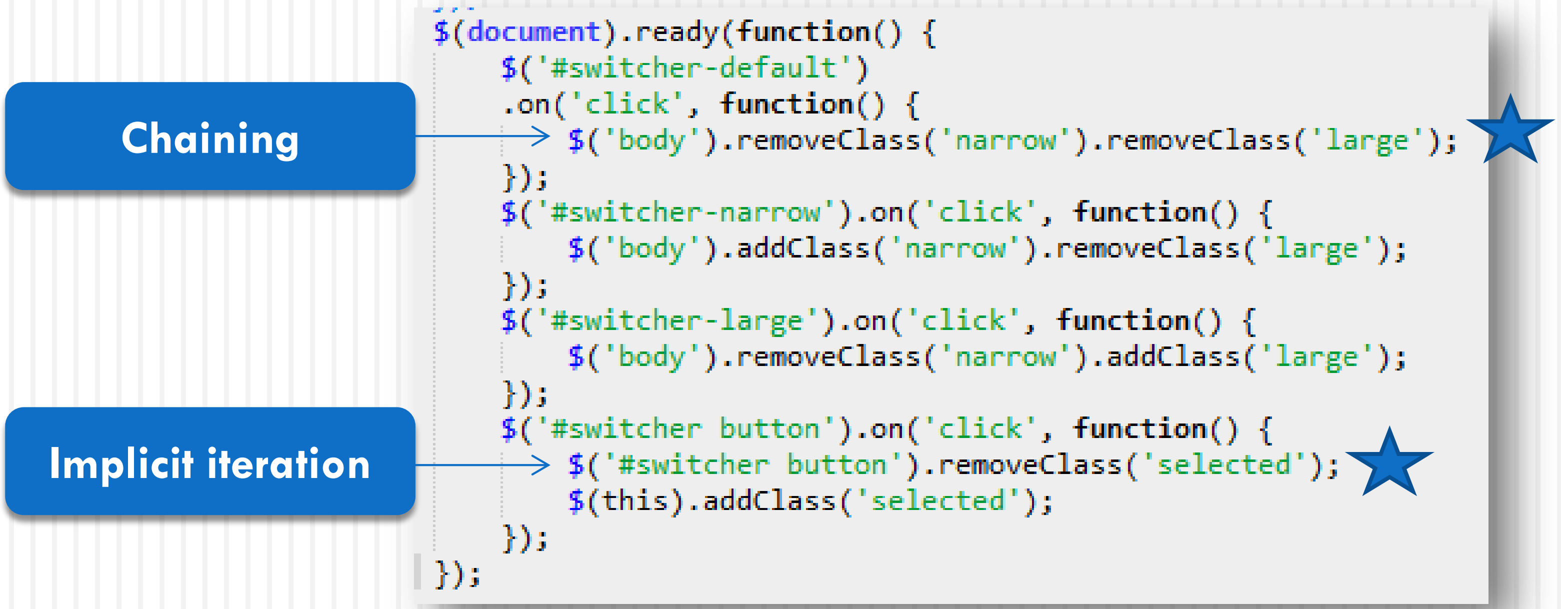
3. However, there is a more elegant way of doing this, by using two things:
   - A different selector in order to remove all the classes
   - And the context information that the events gives when it happens → using *$(this)*



```
$(document).ready(function() {
    $('#switcher-default')
    .on('click', function() {
        $('body').removeClass('narrow');
        $('body').removeClass('large');
        $('#switcher button').removeClass('selected');
        $(this).addClass('selected');
    });
    $('#switcher-narrow').on('click', function() {
        $('body').addClass('narrow');
        $('body').removeClass('large');
        $('#switcher button').removeClass('selected');
        $(this).addClass('selected');
    });
    $('#switcher-large').on('click', function() {
        $('body').removeClass('narrow');
        $('body').addClass('large');
        $('#switcher button').removeClass('selected');
        $(this).addClass('selected');
    });
});
```

# 2. Handling simple events

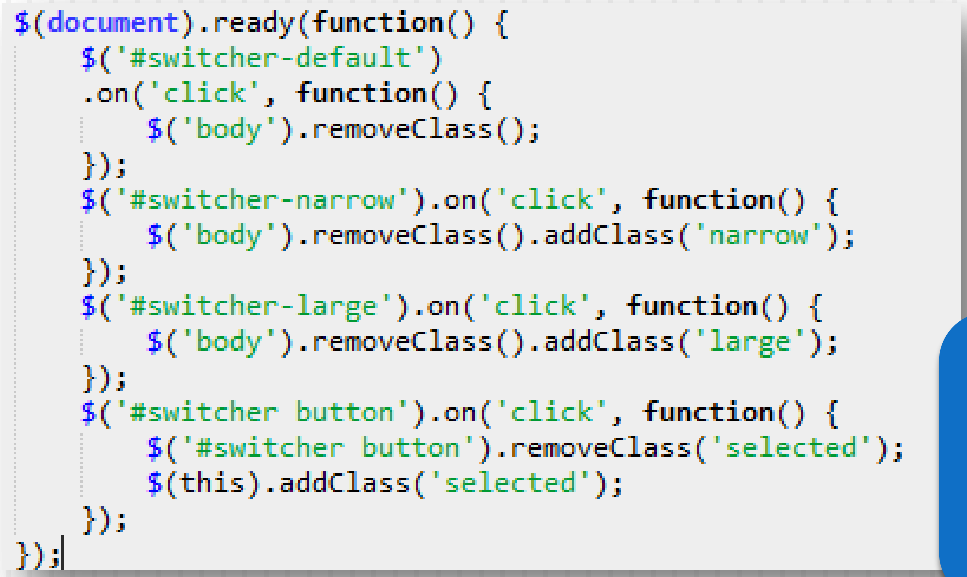
4. Another thing would be to generalize the code and refactor:



```
$(document).ready(function() {
    $('#switcher-default')
    .on('click', function() {
        $('body').removeClass('narrow').removeClass('large');
    });
    $('#switcher-narrow').on('click', function() {
        $('body').addClass('narrow').removeClass('large');
    });
    $('#switcher-large').on('click', function() {
        $('body').removeClass('narrow').addClass('large');
    });
    $('#switcher button').on('click', function() {
        $('#switcher button').removeClass('selected');
        $(this).addClass('selected');
    });
});
```

# 2. Handling simple events

5. Another refactor would be:
   - The .removeClass(**X**) method's parameter is optional; when omitted, it removes all classes from the element.

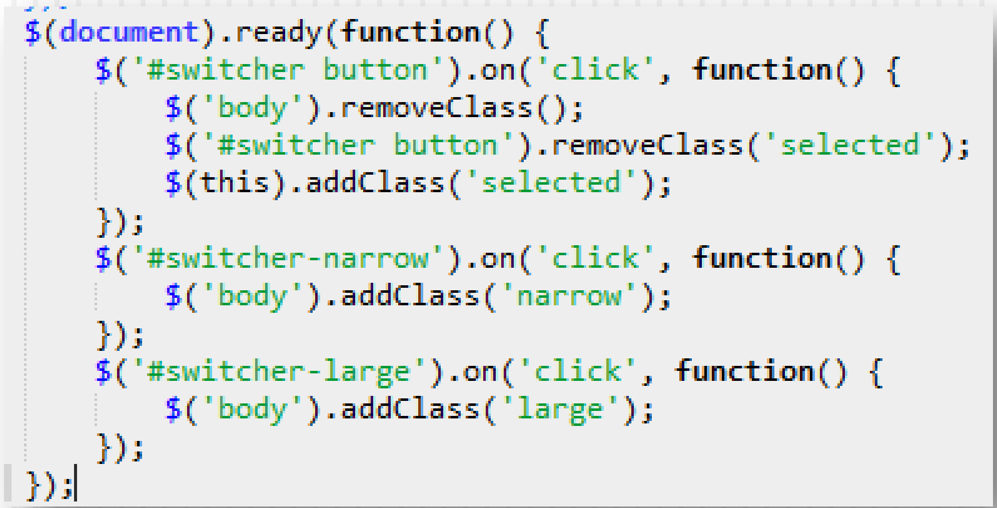


```
$(document).ready(function() {
    $('#switcher-default')
    .on('click', function() {
        $('body').removeClass();
    });
    $('#switcher-narrow').on('click', function() {
        $('body').removeClass().addClass('narrow');
    });
    $('#switcher-large').on('click', function() {
        $('body').removeClass().addClass('large');
    });
    $('#switcher button').on('click', function() {
        $('#switcher button').removeClass('selected');
        $(this).addClass('selected');
    });
});
```

Same code in each of the buttons' handlers…

# 2. Handling simple events

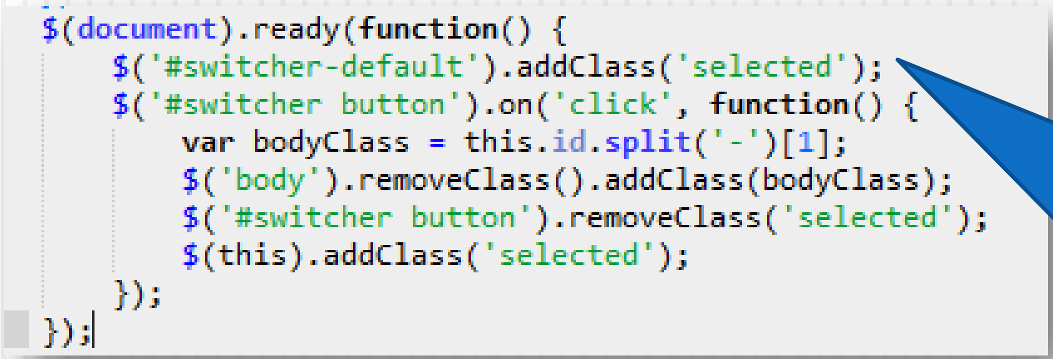
6. This can be easily factored out into our general button click handler:



```
$(document).ready(function() {
    $('#switcher button').on('click', function() {
        $('body').removeClass();
        $('#switcher button').removeClass('selected');
        $(this).addClass('selected');
    });
    $('#switcher-narrow').on('click', function() {
        $('body').addClass('narrow');
    });
    $('#switcher-large').on('click', function() {
        $('body').addClass('large');
    });
});
```

## 2. Handling simple events

7. Finally, since the context keyword ***this*** gives us a DOM element rather than a jQuery object, we can use native DOM properties to determine the ID of the element that was clicked:

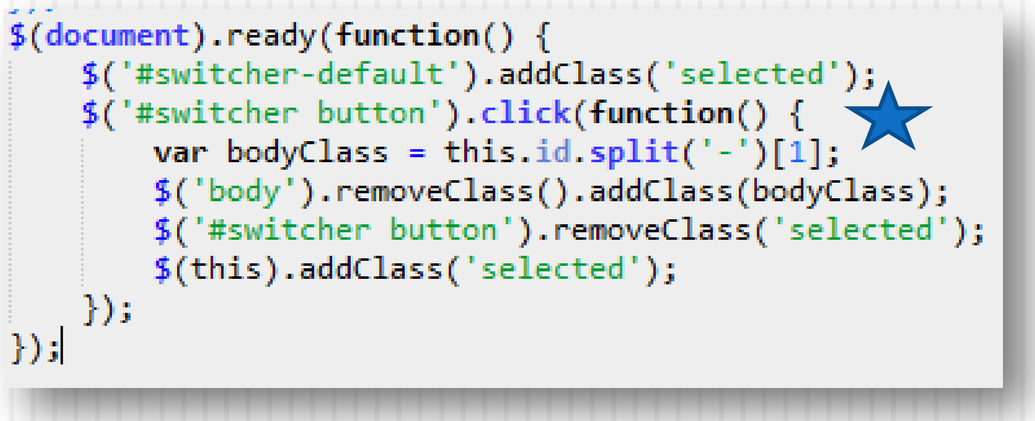


```
$(document).ready(function() {
    $('#switcher-default').addClass('selected');
    $('#switcher button').on('click', function() {
        var bodyClass = this.id.split('-')[1];
        $('body').removeClass().addClass(bodyClass);
        $('#switcher button').removeClass('selected');
        $(this).addClass('selected');
    });
});
```

The first line is added in order *default* to be initially selected

# 2. Handling simple events

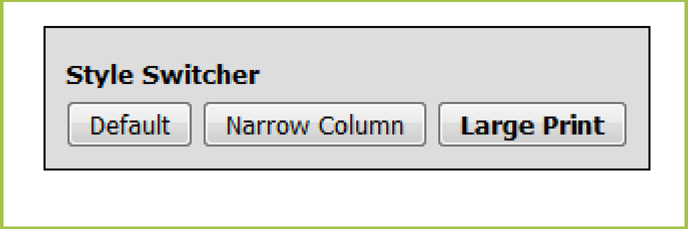
- Binding a handler for an event (such as a *click* event) is such a common task that jQuery provides → <u>shorthand event methods:</u>

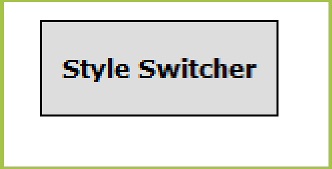


```
$(document).ready(function() {
    $('#switcher-default').addClass('selected');
    $('#switcher button').click(function() {
        var bodyClass = this.id.split('-')[1];
        $('body').removeClass().addClass(bodyClass);
        $('#switcher button').removeClass('selected');
        $(this).addClass('selected');
    });
});
```

# 2. Handling simple events

A.6.5. Suppose that we now want to be able to hide our style switcher when it is not needed. There is a CSS rule with a class called *hidden.*
This time do not use *removeClass* and *addClasss*, use ***toggleClass*** instead (Find how does it works!)





# 2. Handling simple events

- Different events:
  - Keyboard Events 'keydown' 'keypress' 'keyup'
  - Mouse Events 'click' 'mousedown' 'mouseup' 'mousemove'
  - Form Events 'change' 'focus' 'blur'