Web Engineering LAB

Lab # 08
jQuery

Instructor: Hurmat Hidayat

Course Code: SL3003

Semester Spring 2023

Department of Computer Science,
National University of Computer and Emerging Sciences FAST
Peshawar Campus

Content

jQuery Introduction

jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation. The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

jQuery doesn't do anything you cannot achieve with pure JavaScript. jQuery's motto is "Write less, do more," because it allows you to achieve the same goals but in fewer lines of code than you would need to write with plain JavaScript. Many of the biggest companies on the Web use jQuery, such as:

- Google
- Microsoft
- IBM
- Netflix

Download and Get started

There are several ways to start using jQuery on your web site. You can:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN, like Google

There are two versions of jQuery available for downloading:

- Production version - this is for your live website because it has been minified and compressed
- Development version - this is for testing and development (uncompressed and readable code)

The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag :

```
<script src="JS/jquery-3.6.0.min.js"></script>
```

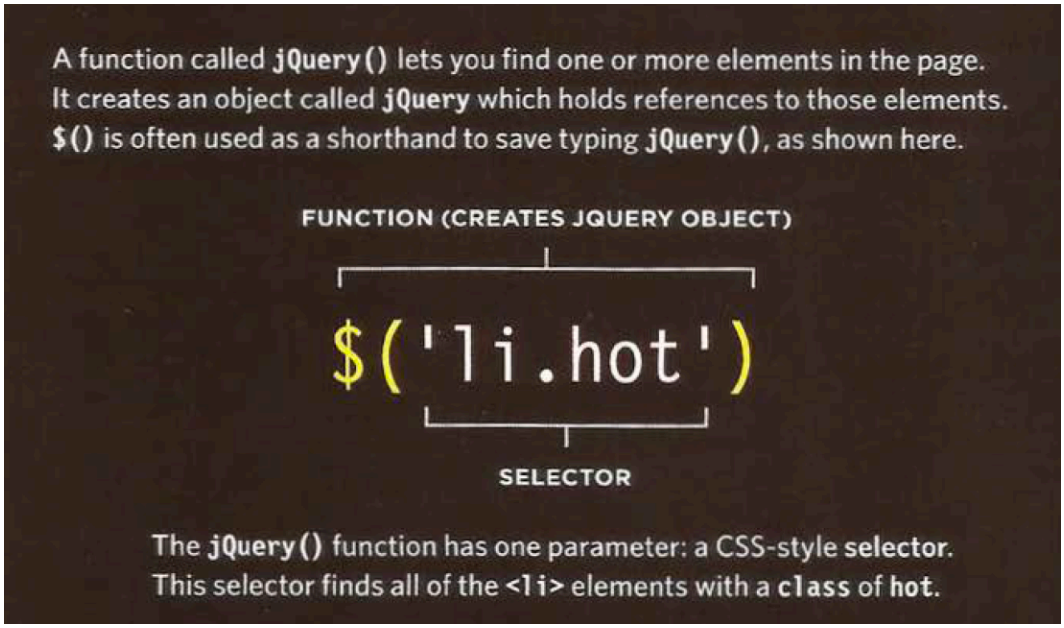
Syntax

The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).

Basic syntax is:

$(*selector*).*action*()

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)



Examples:

`$(this).hide()` - hides the current element.

`$("p").hide()` - hides all <p> elements.

`$(".test").hide()` - hides all elements with class="test".

`$("#test").hide()` - hides the element with id="test".

The Document Ready Event

It is good practice to wait for the document to be fully loaded and ready before working with it. This allows you to have your JavaScript code before the body of your document, in the head section. Here are some examples of actions that can fail if methods are run before the document is fully loaded:

- Trying to hide an element that is not created yet
- Trying to get the size of an image that is not loaded yet

```
<script src="JS/jquery-3.6.0.min.js"></script>
<script>
    $(document).ready(function () {

        // jQuery methods go here...

    });
</script>
```

Selectors

jQuery selectors allow you to select and manipulate HTML element(s). jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more.

It's based on the existing CSS Selectors, and in addition, it has some own custom selectors. All selectors in jQuery start with the dollar sign and parentheses: $().

HTML:

```
<body>
    <p id="one"> 1. This is a paragraph. </p>
    <p id="two" class="test"> 2. This is a paragraph. </p>
    <p> 3. This is a paragraph. </p>
    <p> 4. This is a paragraph. </p>
    <p> 5. This is a paragraph. </p>
</body>
```

The element Selector

The jQuery element selector selects elements based on the element name. You can select all <p> elements on a page like this:

```
$("p")
```

Example:

```
<script>
    $(document).ready(function () {

        console.log("We are using jQuery");

        $('p').click(function () {

            console.log('you clicked on p', this);

        }); //do this when we click on p

    });
</script>
```

The #id Selector

The jQuery *#id* selector uses the id attribute of an HTML tag to find the specific element. An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element. To find an element with a specific id, write a hash character, followed by the id of the HTML;

```
$("#test")
```

Example:

```
<script>
    $(document).ready(function () {

        console.log("We are using jQuery");

        $("#one").click(function () {

            console.log('you clicked on p', this);

        }); //do this when we click on p

    });
</script>
```

The .class Selector

The jQuery *.class* selector finds elements with a specific class. To find elements with a specific class, write a period character, followed by the name of the class:

```
$(".test")
```

Example:

```
<script src="JS/jquery-3.6.0.min.js"></script>

<script>
    $(document).ready(function () {

        console.log("We are using jQuery");

        $(".test").click(function () {

            console.log('you clicked on p', this);

        }); //do this when we click on p

    });
</script>
```

Finding Elements

BASIC SELECTORS

`*`	All elements
`element`	All elements with that element name
`#id`	Elements whose **`id`** attribute has the value specified
`.class`	Elements whose **`class`** attribute has the value specified
`selector1, selector2`	Elements that match more than one selector (see also the **`.add()`** method, which is more efficient when combining selections)

HIERARCHY

`ancestor descendant`	An element that is a descendant of another element (e.g., **`li a`**)
`parent > child`	An element that is a direct child of another element (you can use * in the place of the child to select all child elements of the specified parent)
`previous + next`	Adjacent sibling selector only selects elements that are immediately followed by the previous element
`previous ~ siblings`	Sibling selector will select any elements that are a sibling of the previous element

BASIC FILTERS

`:not(`*`selector`***`)`**		All elements except the one in the selector (e.g., **`div:not('#summary')`**)
`:first`	jQ	The first element from the selection
`:last`	jQ	The last element from the selection
`:even`	jQ	Elements with an even index number in the selection
`:odd`	jQ	Elements with an odd index number in the selection
`:eq(`*`index`***`)`**	jQ	Elements with an index number equal to the one in the parameter
`:gt(`*`index`***`)`**	jQ	Elements with an index number greater than the parameter
`:lt(`*`index`***`)`**	jQ	Elements with an index number less than the parameter
`:header`	jQ	All **`<h1>`** - **`<h6>`** elements
`:animated`	jQ	Elements that are currently being animated
`:focus`		The element that currently has focus

CONTENT FILTERS

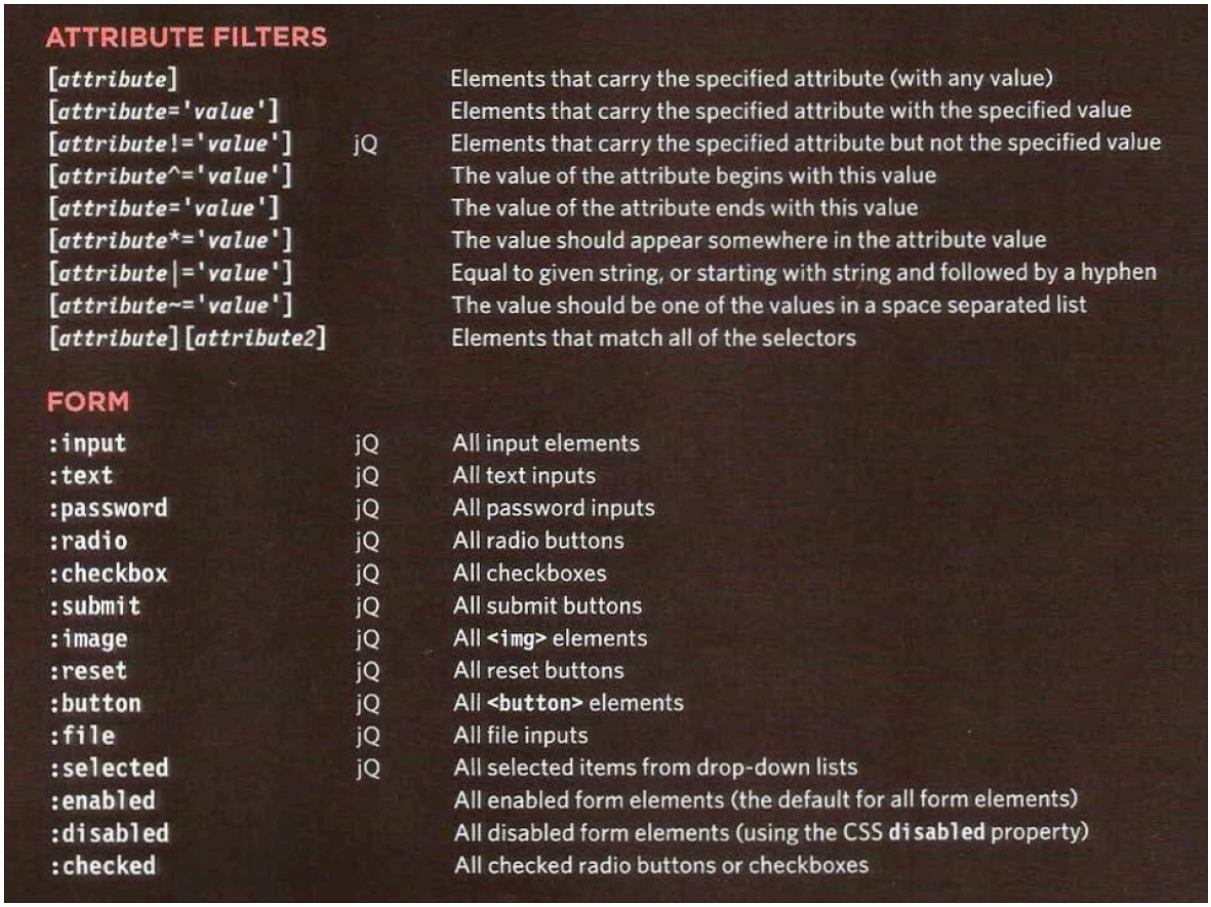
`:contains('`*`text`***`')`**		Elements that contain the specified text as a parameter
`:empty`		All elements that have no children
`:parent`	jQ	All elements that have a child node (can be text or element)
`:has(`*`selector`***`)`**	jQ	Elements that contain at least one element that matches the selector (e.g., **`div:has(p)`** matches all **`div`** elements that contain a **`<p>`** element)

VISIBILITY FILTERS

`:hidden`	jQ	All elements that are hidden
`:visible`	jQ	All elements that consume space in the layout of the page Not selected if: **`display: none`**; **`height`** / **`width`**: **`0`**; ancestor is hidden Selected if: **`visibility: hidden`**; **`opacity: 0`** because they would take up space in layout

CHILD FILTERS

`:nth-child(`*`expr`***`)`**	The value here is not zero-based e.g. **`ul li:nth-child(2)`**
`:first-child`	First child from the current selection
`:last-child`	Last child from the current selection
`:only-child`	When there is only one child of the element (**`div p:only-child`**)



ATTRIBUTE FILTERS

[*attribute*]		Elements that carry the specified attribute (with any value)
[*attribute='value'*]		Elements that carry the specified attribute with the specified value
[*attribute!='value'*]	jQ	Elements that carry the specified attribute but not the specified value
[*attribute^='value'*]		The value of the attribute begins with this value
[*attribute='value'*]		The value of the attribute ends with this value
[*attribute**=*'value'*]		The value should appear somewhere in the attribute value
[*attribute*|=*'value'*]		Equal to given string, or starting with string and followed by a hyphen
[*attribute*~=*'value'*]		The value should be one of the values in a space separated list
[*attribute*] [*attribute2*]		Elements that match all of the selectors

FORM

:input	jQ	All input elements
:text	jQ	All text inputs
:password	jQ	All password inputs
:radio	jQ	All radio buttons
:checkbox	jQ	All checkboxes
:submit	jQ	All submit buttons
:image	jQ	All <img> elements
:reset	jQ	All reset buttons
:button	jQ	All <button> elements
:file	jQ	All file inputs
:selected	jQ	All selected items from drop-down lists
:enabled		All enabled form elements (the default for all form elements)
:disabled		All disabled form elements (using the CSS disabled property)
:checked		All checked radio buttons or checkboxes

Events

All the different visitors' actions that a web page can respond to are called events. An event represents the precise moment when something happens.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

The term "fires/fired" is often used with events. Example: "The keypress event is fired, the moment you press a key".
Here are some common DOM events:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

jQuery Syntax For Event Methods

In jQuery, most DOM events have an equivalent jQuery method. To assign a click event to all paragraphs on a page, you can do this:

```
<script>
    $(document).ready(function () {

        console.log("We are using jQuery");

        $("p").click(function () {
            console.log("p is clicked!!");
        });

    });
</script>
```

Commonly Used jQuery Event Methods

- The click() method attaches an event handler function to an HTML element. The function is executed when the user clicks on the HTML element.
- The mouseenter() method attaches an event handler function to an HTML element. The function is executed when the mouse pointer enters the HTML element.
- The mousedown() method attaches an event handler function to an HTML element. The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element.
- The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods. The first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves the HTML element.
- The blur() method attaches an event handler function to an HTML form field. The function is executed when the form field loses focus.
- The on() method attaches one or more event handlers for the selected elements.

Single:

```
$("p").on("click", function(){
    $(this).hide();
});
```

Multiple:

```
$("p").on({
  mouseenter: function(){
    $(this).css("background-color", "lightgray");
  },
  mouseleave: function(){
    $(this).css("background-color", "lightblue");
  },
  click: function(){
    $(this).css("background-color", "yellow");
```

```
    }
});
```

Example:

```
<script>
    $(document).ready(function () {

        console.log("We are using jQuery");

        $("p").click(function () {
            console.log("p is clicked!!");
        });

        $("#one").mouseenter(function () {
            alert("You entered paragraph one!");
        });

        $("#two").mousedown(function () {
            alert("Mouse down over p two!");
        });

        $("#three").hover(function () {
            console.log("You entered p3!");
        },
            function () {
                console.log("Bye! You now leave p3!");
            });
    });
</script>
```

JQuery Effects

When you start using jQuery, the effects methods can enhance your web page with transitions and movement. Here you can see some of the jQuery effects that show or hide elements and their content. You can animate them fading in and out, or slide them up and down.

Methods with toggle in their name will look at the current state of the element (whether it is visible or hidden) and will switch to the opposite state.

BASIC EFFECTS

METHOD	DESCRIPTION
.show()	Displays selected elements
.hide()	Hides selected elements
.toggle()	Toggles between showing and hiding selected elements

FADING EFFECTS

METHOD	DESCRIPTION
.fadeIn()	Fades in selected elements making them opaque
.fadeOut()	Fades out selected elements making them transparent
.fadeTo()	Changes opacity of selected elements
.fadeToggle()	Hides or shows selected elements by changing their opacity (the opposite of their current state)

SLIDING EFFECTS

METHOD	DESCRIPTION
.slideUp()	Shows selected elements with a sliding motion
.slideDown()	Hides selected elements with a sliding motion
.slideToggle()	Hides or shows selected elements with a sliding motion (in the opposite direction to its current state)

CUSTOM EFFECTS

METHOD	DESCRIPTION
.delay()	Delays execution of subsequent items in queue
.stop()	Stops an animation if it is currently running
.animate()	Creates custom animations (see p334)

Hide /Show

With jQuery, you can hide and show HTML elements with the hide() and show() methods.

```
$("#hide").click(function(){
  $("p").hide();
});

$("#show").click(function(){
```

```
  $("p").show();
});
```

Syntax:

```
$(selector).hide(speed,callback);

$(selector).show(speed,callback);
```

The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds. The optional callback parameter is a function to be executed after the hide() or show() method completes .

```
$('#wiki').hide(2000, function () {
    console.log("hidden");
})
$('#wiki').show(2000, function () {
    console.log("show");
})
```

Fading

With jQuery you can fade an element in and out of visibility. jQuery has the following fade methods:

- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()

`fadeIn():` The jQuery fadeIn() method is used to fade in a hidden element.

```
$(selector).fadeIn(speed,callback);
```

```
$('#wiki').hide(2000, function () {
    console.log("hidden");
})

$('#but').click(function () {
    $('#wiki').fadeIn(1000);
})
```

fadeOut(): The jQuery fadeOut() method is used to fade out a visible element.
$(*selector*).fadeOut(*speed,callback*);

```
$('#wiki').show(2000, function () {
    console.log("show");
})
$('#but').click(function () {
    $('#wiki').fadeOut(1000);
})
```

fadeToggle(): The jQuery fadeToggle() method toggles between the fadeIn() and fadeOut() methods.

- If the elements are faded out, fadeToggle() will fade them in.
- If the elements are faded in, fadeToggle() will fade them out.

$(*selector*).fadeToggle(*speed,callback*);

```
$('#but').click(function () {
    $('#wiki').fadeToggle(1000);
})
```

fadeTo(): The jQuery fadeTo() method allows fading to a given opacity (value between 0 and 1).
$(*selector*).fadeTo(*speed,opacity,callback*);

```
$('#wiki').show(2000, function () {
    console.log("show");
})
$('#but').click(function () {
    $('#wiki').fadeTo(1000, 0.3);
})
```

Slide

With jQuery you can create a sliding effect on elements. jQuery has the following slide methods:

- slideDown()
- slideUp()
- slideToggle()

```
<body>

    <div id="flip">Click to slide down panel</div>
    <div id="panel">Hello world!</div>

</body>
```

slideDown(): The jQuery slideDown() method is used to slide down an element.

```
$(selector).slideDown(speed,callback);
```

```
<script src="JS/jquery-3.6.0.min.js"></script>
<script>
    $(document).ready(function () {
        $("#flip").click(function () {
            $("#panel").slideDown("slow");
        });
    });
</script>
```

slideUp(): The jQuery slideUp() method is used to slide up an element.

```
$(selector).slideUp(speed,callback);
```

```
<script>
    $(document).ready(function(){
      $("#flip").click(function(){
        $("#panel").slideUp("slow");
      });
    });
    </script>
```

slideToggle(): The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.

- If the elements have been slid down, slideToggle() will slide them up.
- If the elements have been slid up, slideToggle() will slide them down.

```
$(selector).slideToggle(speed,callback);
```

```
<script>
    $(document).ready(function () {
        $("#flip").click(function () {
            $("#panel").slideToggle("slow");
        });
    });
</script>
```

Animate

The jQuery animate() method is used to create custom animations.

```
$(selector).animate({params},speed,callback);
```

The required params parameter defines the CSS properties to be animated. The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the animation completes.
The following example demonstrates a simple use of the animate() method; it moves a element to the right, until it has reached a left property of 250px:

```
<script>
    $(document).ready(function () {
        $("button").click(function () {
            $("div").animate({ left: '250px' });
        });
    });
</script>
```

Animate- Manipulate Multiple Properties

Multiple properties can be animated at the same time:

```
<script>
    $(document).ready(function () {
        $("button").click(function () {
            $("div").animate({
                left: '250px',
                opacity: '0.5',
                height: '150px',
                width: '150px'
            });
        });
    });
</script>
```

stop()

The jQuery stop() method is used to stop an animation or effect before it is finished.
The stop() method works for all jQuery effect functions, including sliding, fading and custom animations. The optional stopAll parameter specifies whether also the animation queue should be cleared or not. Default is false, which means that only the active animation will be stopped, allowing any queued animations to be performed afterwards.
The optional goToEnd parameter specifies whether or not to complete the current animation immediately. Default is false.

So, by default, the stop() method kills the current animation being performed on the selected element.

The following example demonstrates the stop() method, with no parameters:

```
$("#stop").click(function(){
  $("#panel").stop();
});
```

Callback

JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors. To prevent this, you can create a callback function. A callback function is executed after the current effect is finished.

Typical syntax:

$(*selector*).hide(*speed,callback*);

The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

```
$("button").click(function(){
  $("p").hide("slow", function(){
    alert("The paragraph is now hidden");
  });
});
```

The example below has no callback parameter, and the alert box will be displayed before the hide effect is completed:

```
$("button").click(function(){
  $("p").hide(1000);
  alert("The paragraph is now hidden");
});
```

jQuery DOM Manipulation

One very important part of jQuery is the possibility to manipulate the DOM.
jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes.

Get and Set Content

Three simple, but useful, jQuery methods for DOM manipulation are:

- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields

```
$("#btn1").click(function(){
  alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
  alert("HTML: " + $("#test").html());
});
```

The following example demonstrates how to get the value of an input field with the jQuery val() method:

```
$("#btn1").click(function(){
  alert("Value: " + $("#test").val());
});
```

The following example demonstrates how to set content with the jQuery text(), html(), and val() methods:

```
$("#btn1").click(function(){
  $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
  $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
  $("#test3").val("Dolly Duck");
});
```

Note: Read book for more details.

References

1. Duckett J. JavaScript & jQuery. Wiley VCH; 2015.
2. https://www.w3schools.com/jquery/default.asp

Lab Tasks:

1. Click Event:

   a. On click of a button, welcome message should display.
   b. There are five input textboxes. On click of the input box , the background colour should change to yellow. (Hint : use *this*)
   c. Remove and add different style to the element by using JQuery. Create two style classes and by default apply first one to one element. On click of a button change the elements associated property to other style.
   d. Toggle the style properties of the element on click

2. Write code that can be linked to an HTML page in a script tag so that when the page loads, every table in the page will become "zebra striped." Zebra striping in this case means that every odd row of the table will be colored to have a black background and white text. (The page's CSS file has a class named zebrastripe that you can attach to the odd rows if you like.) For example, the following page should have the following appearance after your code runs:

before

Name	Bounces
Snuggles	7
Horseface	13
Cornelius	3
Tigger	1000

Name	Favorite Pasta
Snuggles	Fettucini Alfredo
Horseface	Spaghetti Pomodoro
Cornelius	Chicken Penne

after

Name	Bounces
Snuggles	7
Horseface	13
Cornelius	3
Tigger	1000

Name	Favorite Pasta
Snuggles	Fettucini Alfredo
Horseface	Spaghetti Pomodoro
Cornelius	Chicken Penne

Note that each table has the same counting for zebra striping, i.e., the first, third, fifth, ... rows are colored. The number of rows in one table should not affect the striping for another table.