girldevelopit

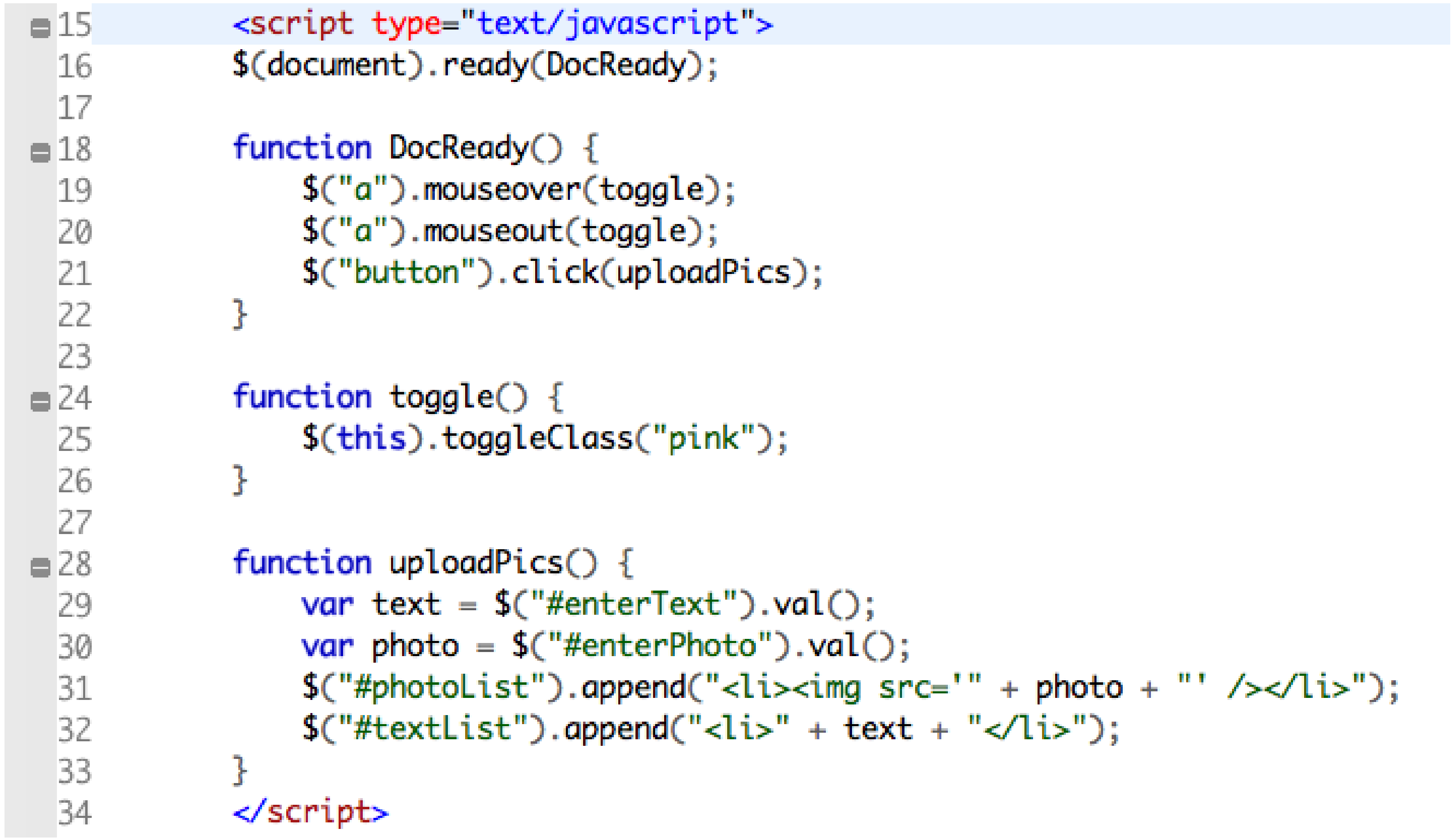
JavaScript Class 3: Element Manipulation

Alexis Goldstein, Sara Chipps
@alexisgoldstein, @sarajchipps

Agenda

- Walk through the code in the bakery.html file. Material covered:
  - document.ready()
  - Selectors, redux
  - “this”
  - Callbacks
  - Click()
  - Val(), append()
  - Anonymous functions
- Add code to a base file, using the concepts we just reviewed, to enrich another html file.

Today's first code sample: bakery.html



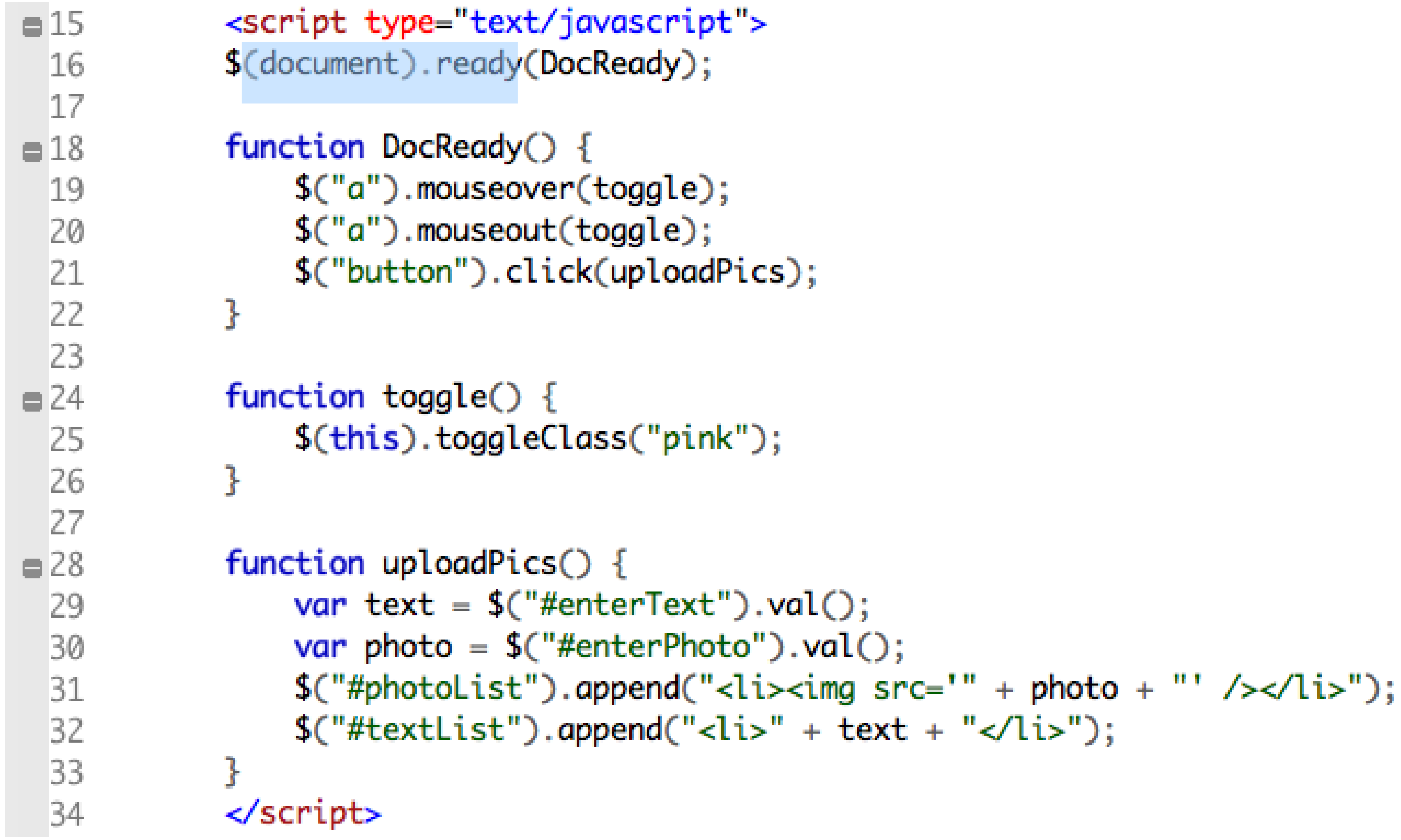
```
<script type="text/javascript">
$(document).ready(DocReady);

function DocReady() {
    $("a").mouseover(toggle);
    $("a").mouseout(toggle);
    $("button").click(uploadPics);
}

function toggle() {
    $(this).toggleClass("pink");
}

function uploadPics() {
    var text = $("#enterText").val();
    var photo = $("#enterPhoto").val();
    $("#photoList").append("<li><img src='" + photo + "' /></li>");
    $("#textList").append("<li>" + text + "</li>");
}
</script>
```

Today's first code sample: bakery.html

We will review the code line by line:

- To understand what each line is doing.
- To learn about each new action we're using.
- To explain basic conventions in jQuery.
  - Please download the sample code from: **http://bit.ly/jsCode3**

document.ready()



```
<script type="text/javascript">
$(document).ready(DocReady);

function DocReady() {
    $("a").mouseover(toggle);
    $("a").mouseout(toggle);
    $("button").click(uploadPics);
}

function toggle() {
    $(this).toggleClass("pink");
}

function uploadPics() {
    var text = $("#enterText").val();
    var photo = $("#enterPhoto").val();
    $("#photoList").append("<li><img src='" + photo + "' /></li>");
    $("#textList").append("<li>" + text + "</li>");
}
</script>
```

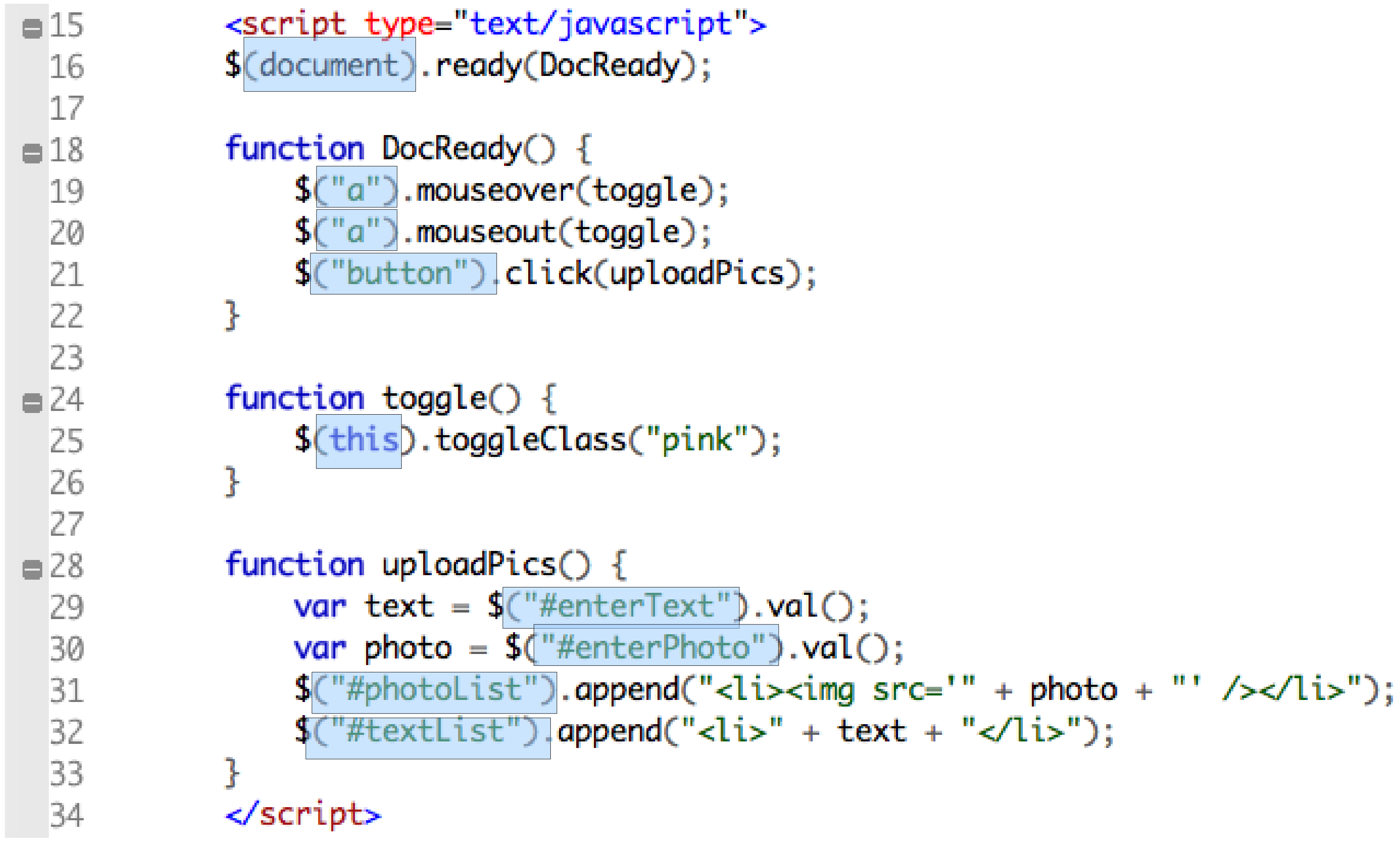
document.ready() – why do we use it?

- We use selectors in jQuery to select html elements, and then perform actions on them.
- What do you think would happen if we tried to perform an action on an element, like a large image, that hadn't loaded yet?
  - We can try this ourselves.
  - Preview the two files, **hideImageBeforeLoading.html** and **hideImageAfterLoads.html**

document.ready() – why do we use it?

- The document ready function prevents any jQuery code from running before the html document finishes loading.
- Other examples of actions that can fail if you do not wrap your code in **document.ready(…);**
  - Trying to get the size of an image that hasn't loaded.
  - Trying to hide an element that doesn't exist.

Selectors

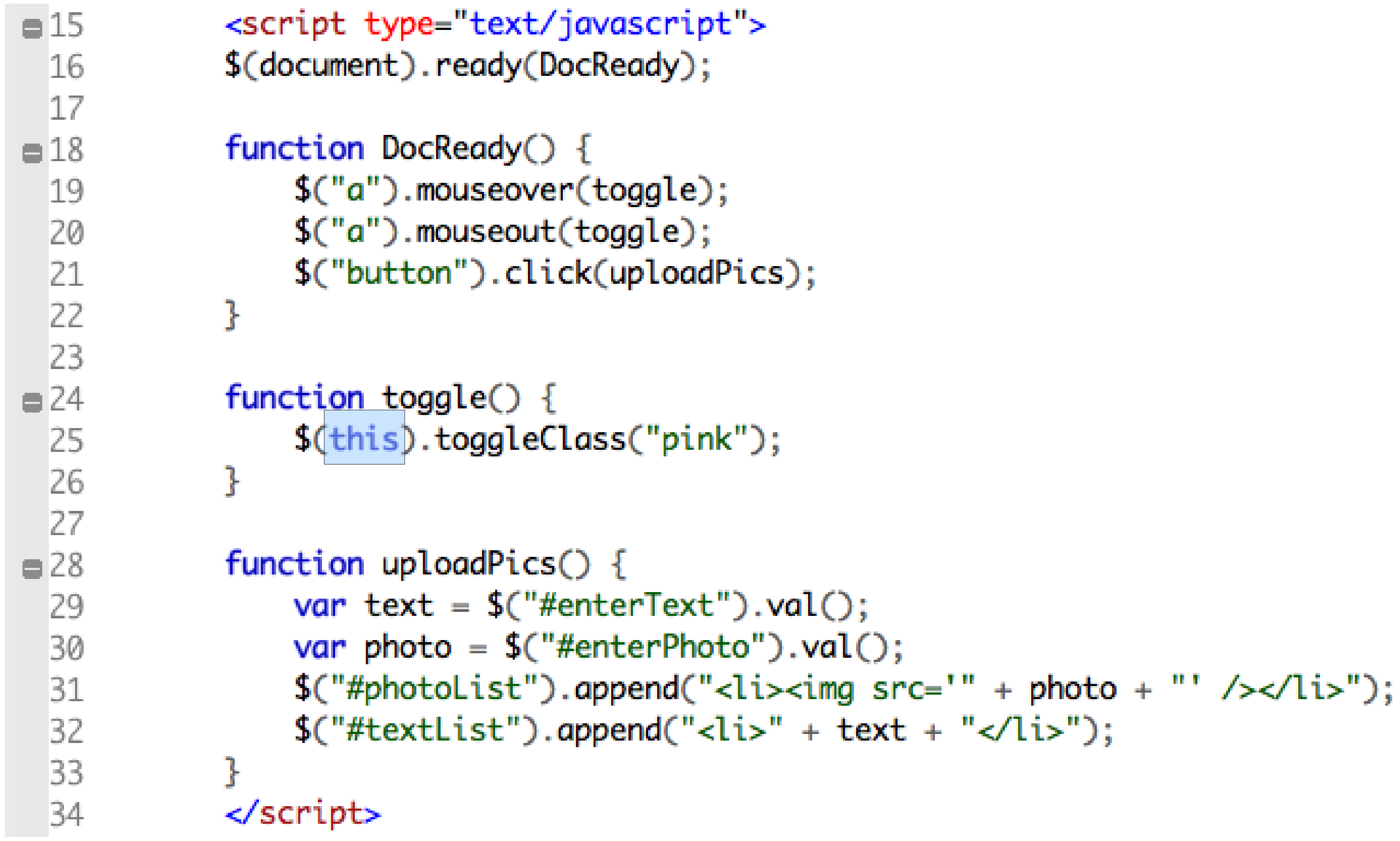


```
<script type="text/javascript">
$(document).ready(DocReady);

function DocReady() {
    $("a").mouseover(toggle);
    $("a").mouseout(toggle);
    $("button").click(uploadPics);
}

function toggle() {
    $(this).toggleClass("pink");
}

function uploadPics() {
    var text = $("#enterText").val();
    var photo = $("#enterPhoto").val();
    $("#photoList").append("<li><img src='" + photo + "' /></li>");
    $("#textList").append("<li>" + text + "</li>");
}
</script>
```

Selectors: a refresher

- Basic jQuery syntax is: $**(selector)**.action()
- jQuery selectors allow us to find html elements.
- jQuery selectors are much like CSS selectors.
  - Instead of finding to an html element to apply styles, you can find an html element to modify, remove or replace.
- jQuery selectors allow you to select HTML elements (or groups of elements) by element name, attribute name

The “**this**” Selector

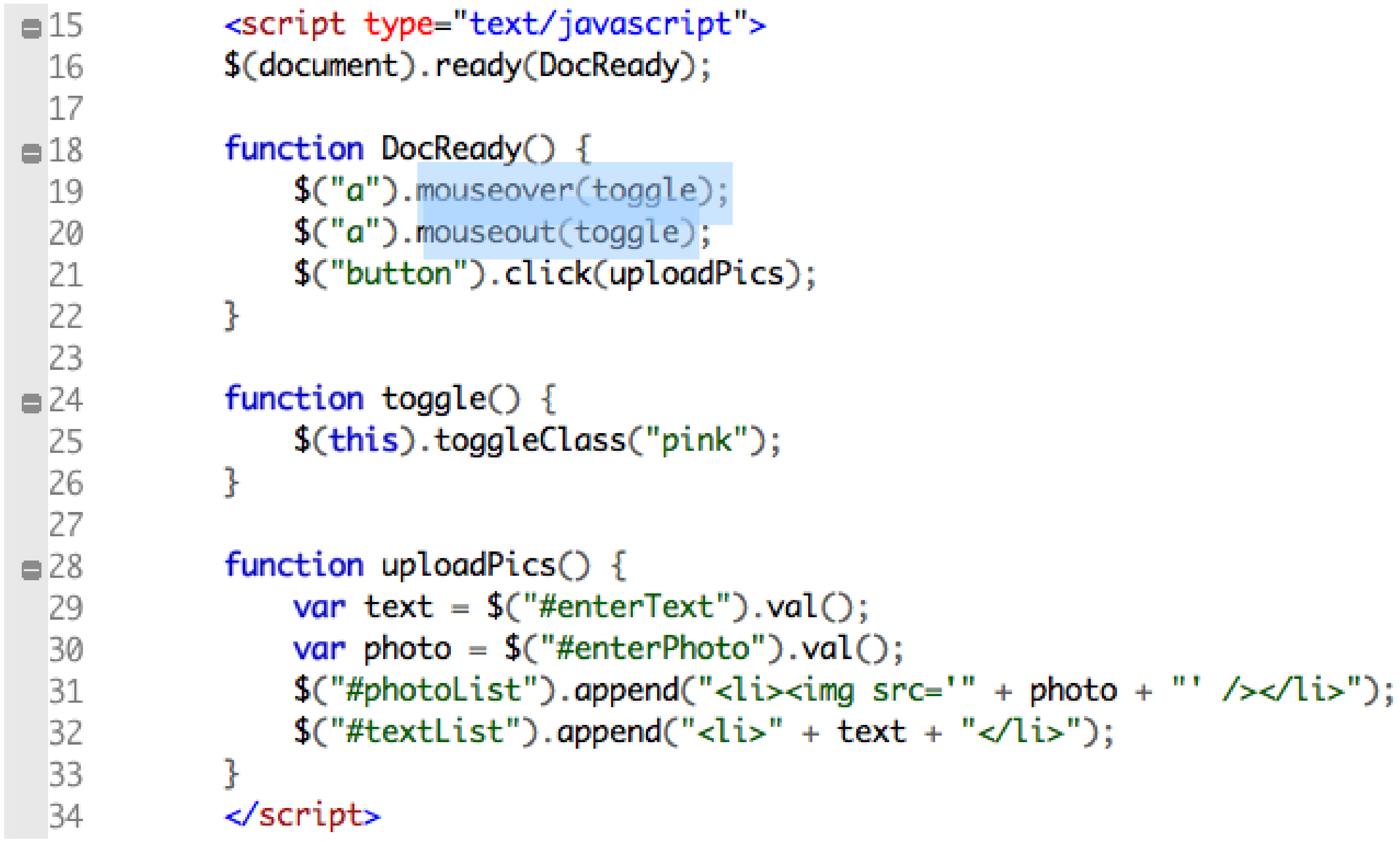


```
<script type="text/javascript">
$(document).ready(DocReady);

function DocReady() {
    $("a").mouseover(toggle);
    $("a").mouseout(toggle);
    $("button").click(uploadPics);
}

function toggle() {
    $(this).toggleClass("pink");
}

function uploadPics() {
    var text = $("#enterText").val();
    var photo = $("#enterPhoto").val();
    $("#photoList").append("<li><img src='" + photo + "' /></li>");
    $("#textList").append("<li>" + text + "</li>");
}
</script>
```

“this” selector

- The “this” selector in our toggle method is telling us to select the element that just called the function that is currently executing.
  - We called the toggle function after selecting a link (“a”).
  - So **this** in the toggle function will give us the link that we just interacted with.
- Read more: this demystified

Mouseover & mouseout



```
<script type="text/javascript">
$(document).ready(DocReady);

function DocReady() {
    $("a").mouseover(toggle);
    $("a").mouseout(toggle);
    $("button").click(uploadPics);
}

function toggle() {
    $(this).toggleClass("pink");
}

function uploadPics() {
    var text = $("#enterText").val();
    var photo = $("#enterPhoto").val();
    $("#photoList").append("<li><img src='" + photo + "' /></li>");
    $("#textList").append("<li>" + text + "</li>");
}
</script>
```

Mouseover & mouseout actions

- Mouseover is called when you place your mouse over an html element, like a link.
- Mouseout is called when your mouse leaves the element you had been on previously. i.e, you move your mouse off that link.
- Read more:
  - http://api.jquery.com/mouseout/
  - http://api.jquery.com/mouseover/

The toggle function & toggleClass

```
function toggle() {
   $(this).toggleClass("pink");
}
```

- Inside the toggle function, we call jQuery's toggleClass() function.
- ToggleClass() takes the name of the class you pass it, and:
  - If the element you've selected already **has** that class, it removes it.
  - It the element you've selected does NOT have that class, it adds it.

The toggle function & toggleClass

```
function toggle() {
   $(this).toggleClass("pink");
}
```

- If the the html was this before calling toggleClass:

  `<a href="cupcakes.html">Cupcakes</a>`

- Then after toggleClass, the html is:

  `<a class="pink"  href="cupcakes.html">`
`Cupcakes</a>`

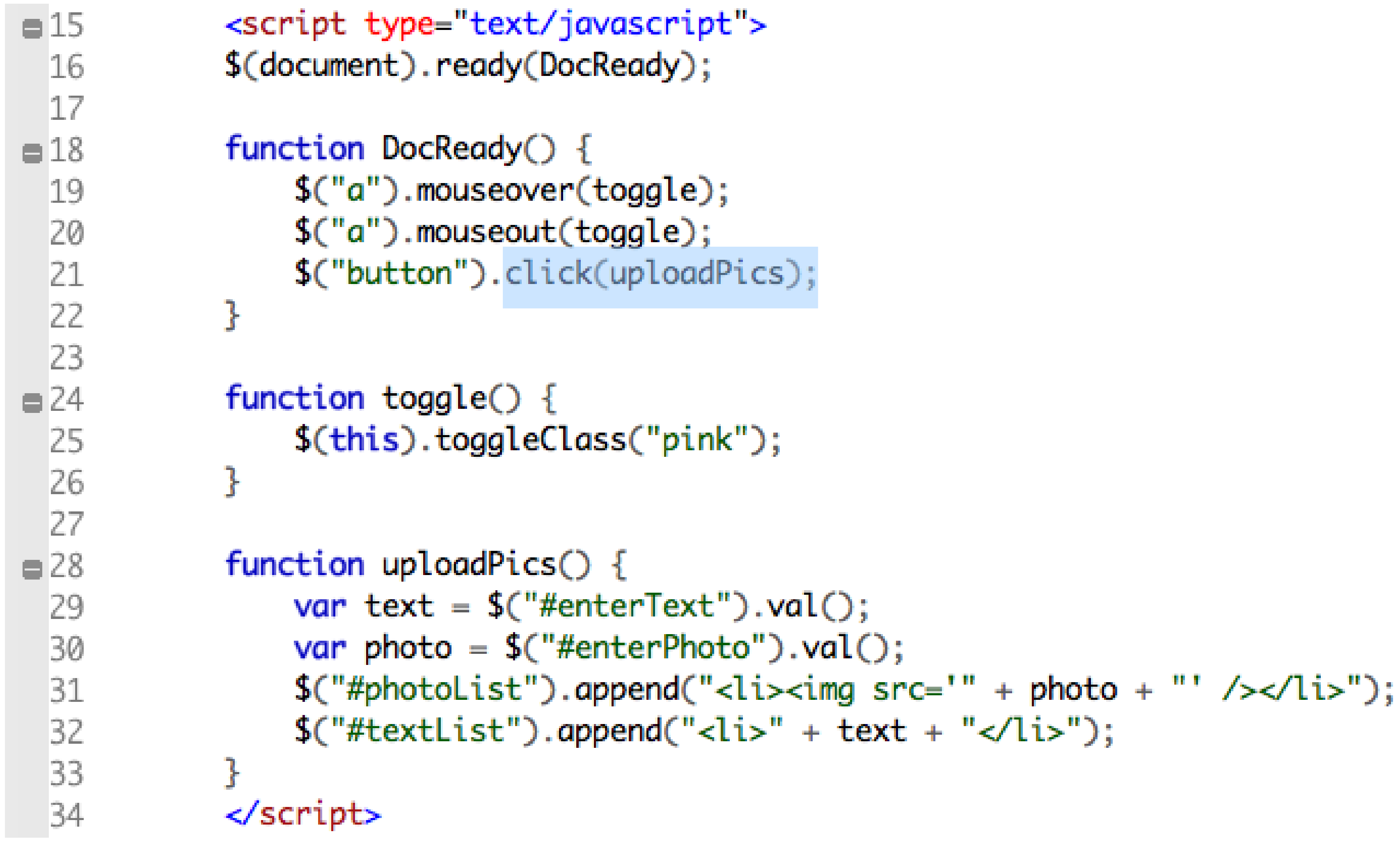
- Read more: `http://api.jquery.com/toggleClass/`

Important Jargon: Callbacks

- A callback is a function that is passed as an argument to another function.
- A callback is executed *after* the parent function is finished.
- We used our **toggle** function as a callback:

```
$("a").mouseout(toggle);
```

The click action



```
<script type="text/javascript">
$(document).ready(DocReady);

function DocReady() {
    $("a").mouseover(toggle);
    $("a").mouseout(toggle);
    $("button").click(uploadPics);
}

function toggle() {
    $(this).toggleClass("pink");
}

function uploadPics() {
    var text = $("#enterText").val();
    var photo = $("#enterPhoto").val();
    $("#photoList").append("<li><img src='" + photo + "' /></li>");
    $("#textList").append("<li>" + text + "</li>");
}
</script>
```

click action

- click is called when you place your click an html element, like a link or a button.
- Read more: http://api.jquery.com/click/

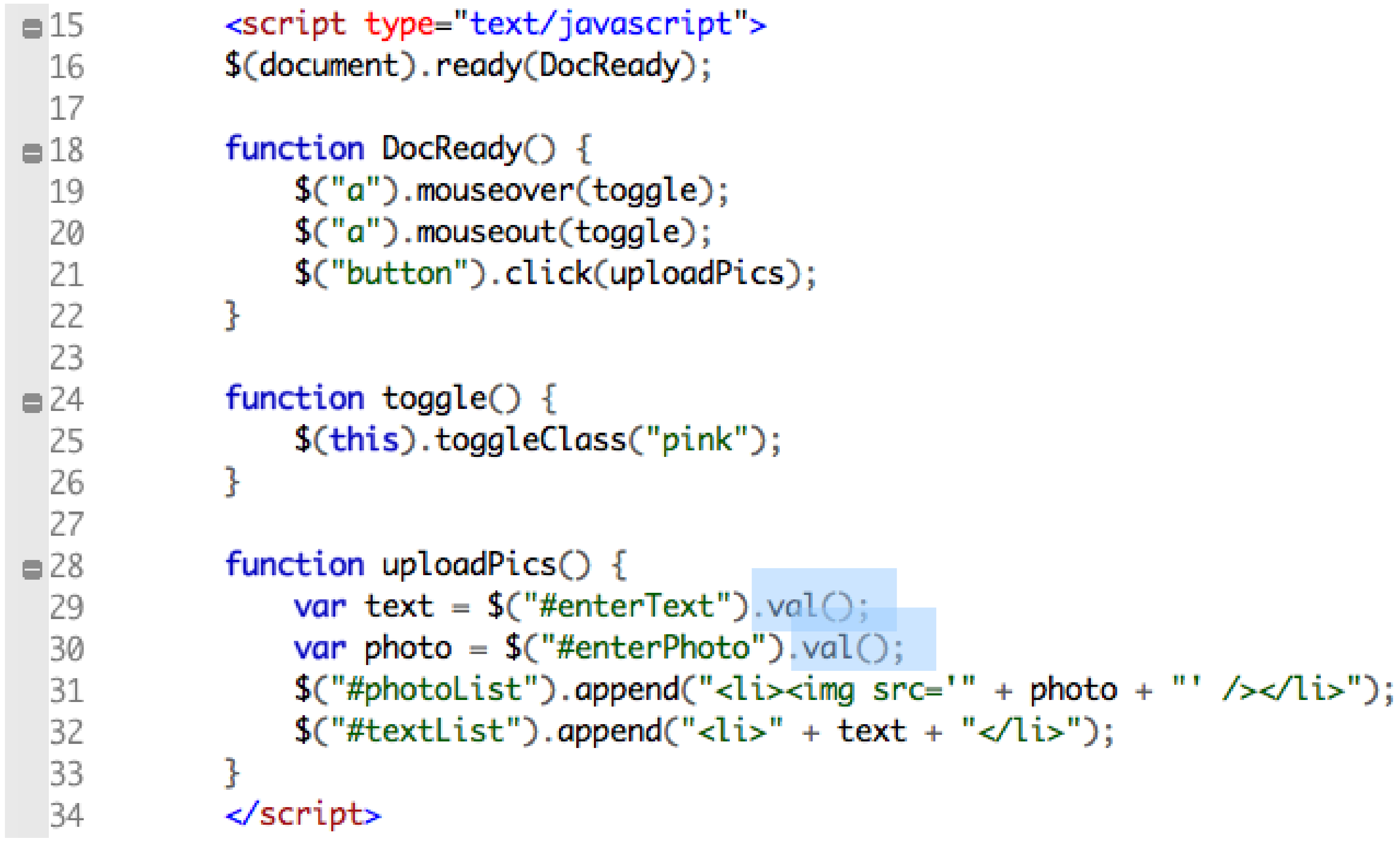
- In our code sample, we pass the uploadPics function as a *callback* to the click action.

The uploadPics

```
function uploadPics() {
  var text = $("#enterText").val();
  var photo = $("#enterPhoto").val();
  $("#photoList").append("<li><img src='" +
photo + "' /></li>");
  $("#textList").append("<li>" + text +
"</li>");
}
```

- Inside the uploadPics function, we call the following jQuery actions: val(), append().

The val action



```
<script type="text/javascript">
$(document).ready(DocReady);

function DocReady() {
    $("a").mouseover(toggle);
    $("a").mouseout(toggle);
    $("button").click(uploadPics);
}

function toggle() {
    $(this).toggleClass("pink");
}

function uploadPics() {
    var text = $("#enterText").val();
    var photo = $("#enterPhoto").val();
    $("#photoList").append("<li><img src='" + photo + "' /></li>");
    $("#textList").append("<li>" + text + "</li>");
}
</script>
```

The val() action

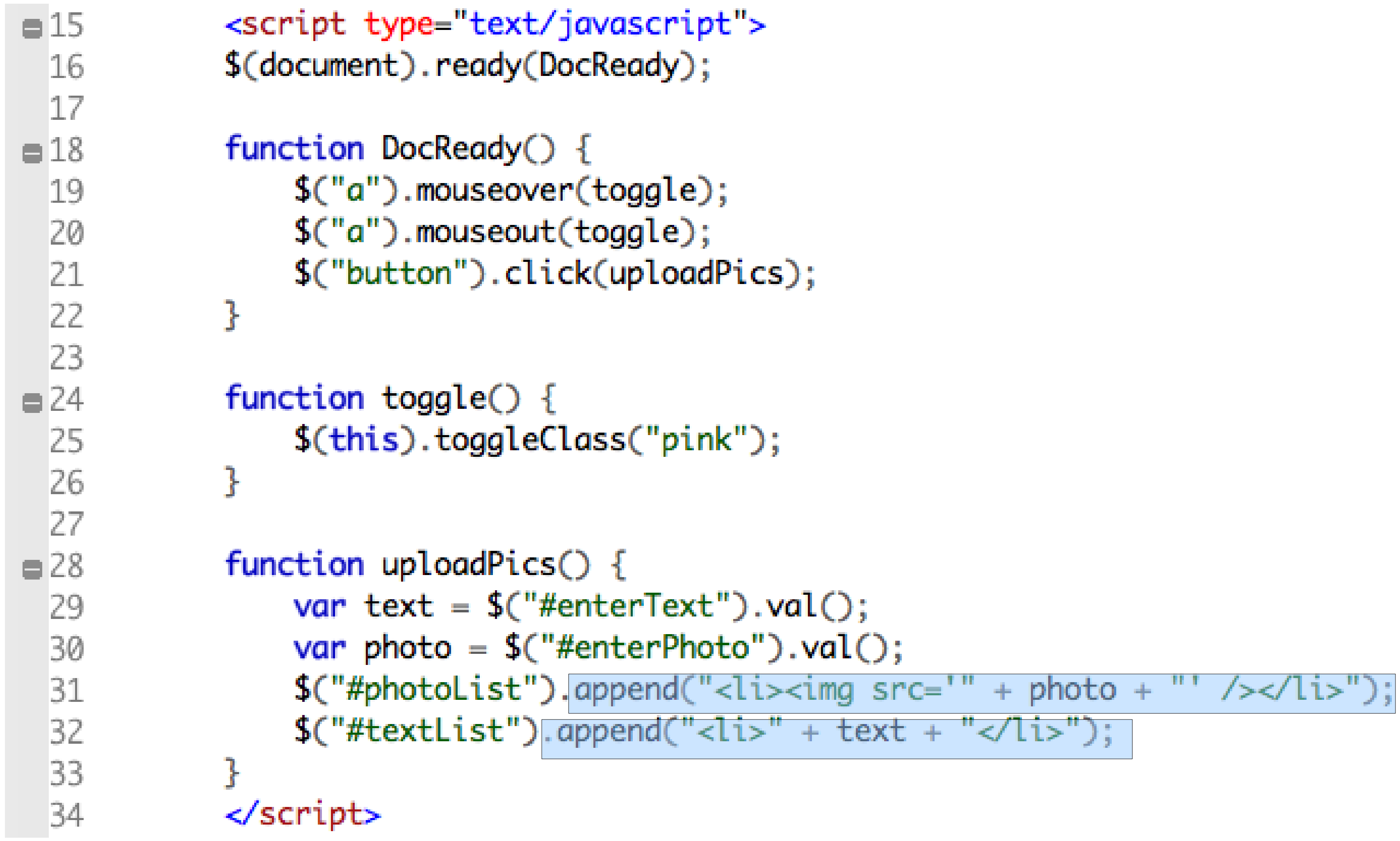
- Val() will get you the current value of the first element your selector matches.
  - If you selector matches more than one element, val() will only pass you back the current value of the **first** match.
- Val() is often used to grab the text out of a form element, like a text box.

In our example, we grab the text entered into the `<input type="text" id="enterText"/>` textbox with this line:

```
var text = $("#enterText").val();
```

- Read more: http://api.jquery.com/val/

The append action



```
<script type="text/javascript">
$(document).ready(DocReady);

function DocReady() {
    $("a").mouseover(toggle);
    $("a").mouseout(toggle);
    $("button").click(uploadPics);
}

function toggle() {
    $(this).toggleClass("pink");
}

function uploadPics() {
    var text = $("#enterText").val();
    var photo = $("#enterPhoto").val();
    $("#photoList").append("<li><img src='" + photo + "' /></li>");
    $("#textList").append("<li>" + text + "</li>");
}
</script>
```

append action

- Append will insert the content you pass to it onto the end of the element you selected (but before that element's closing tag).

```
$("#textList").append("<li>" + text + "</li>");
```

First it finds the id=”textList” element (`<ul id="textList">`)

Then, it inserts `<li>the_text_you_entered</li>` right before the </ul> tag

- Read more: http://api.jquery.com/append/

The uploadPics function

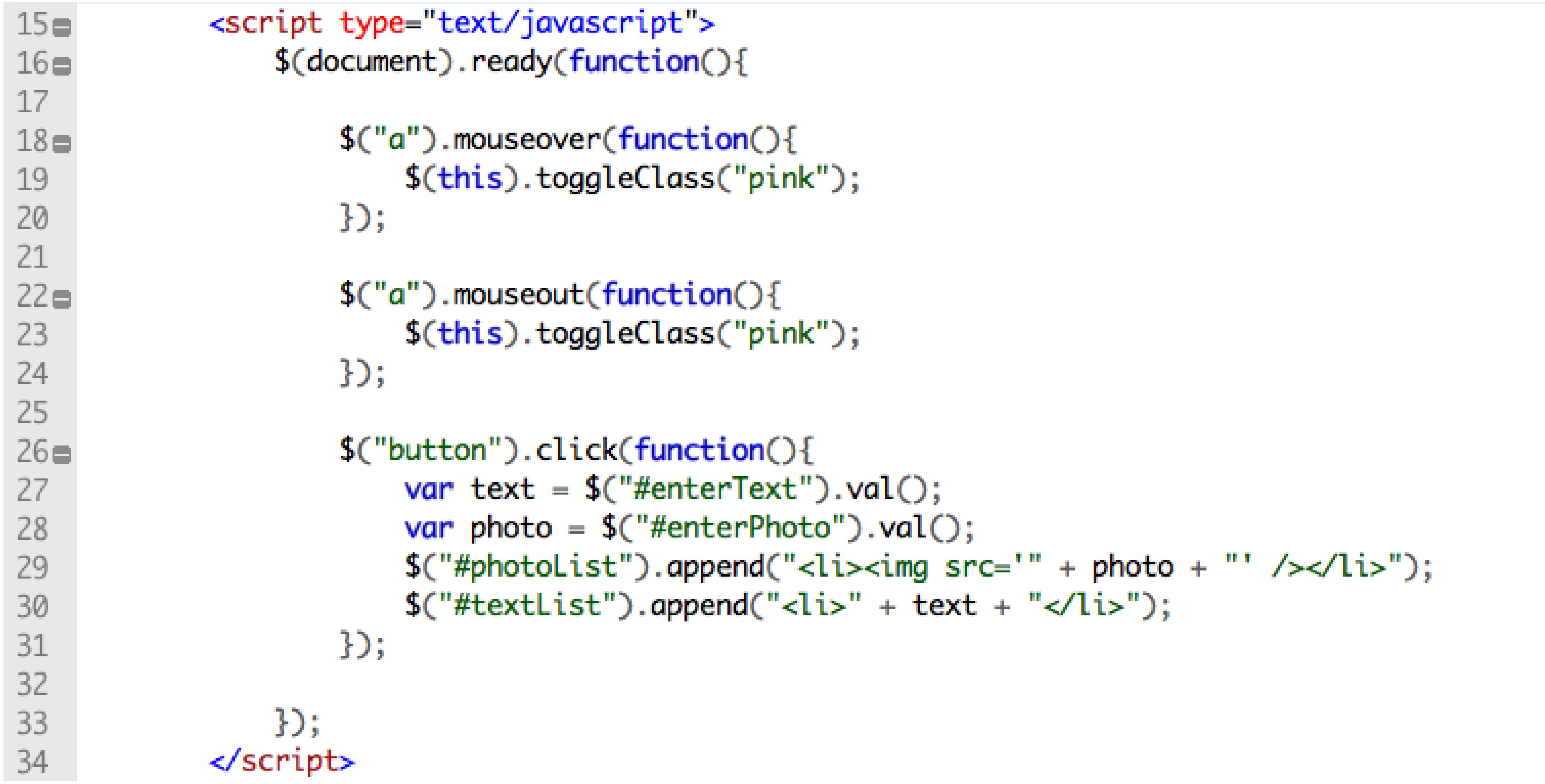
In summary, the uploadPics function:

- Grabs the html element with id=”enterText” (this is the first textbox), and requests the value of the text in the textbox.
- Does the same thing to the element with id=”enterPhoto” (the second textbox).
- Appends new html code, containing a new image nested inside a list item. The image contains the URL typed into the second textbox.
- Appends the text entered in the first textbox next to the new image.

There's more than one way to do it...

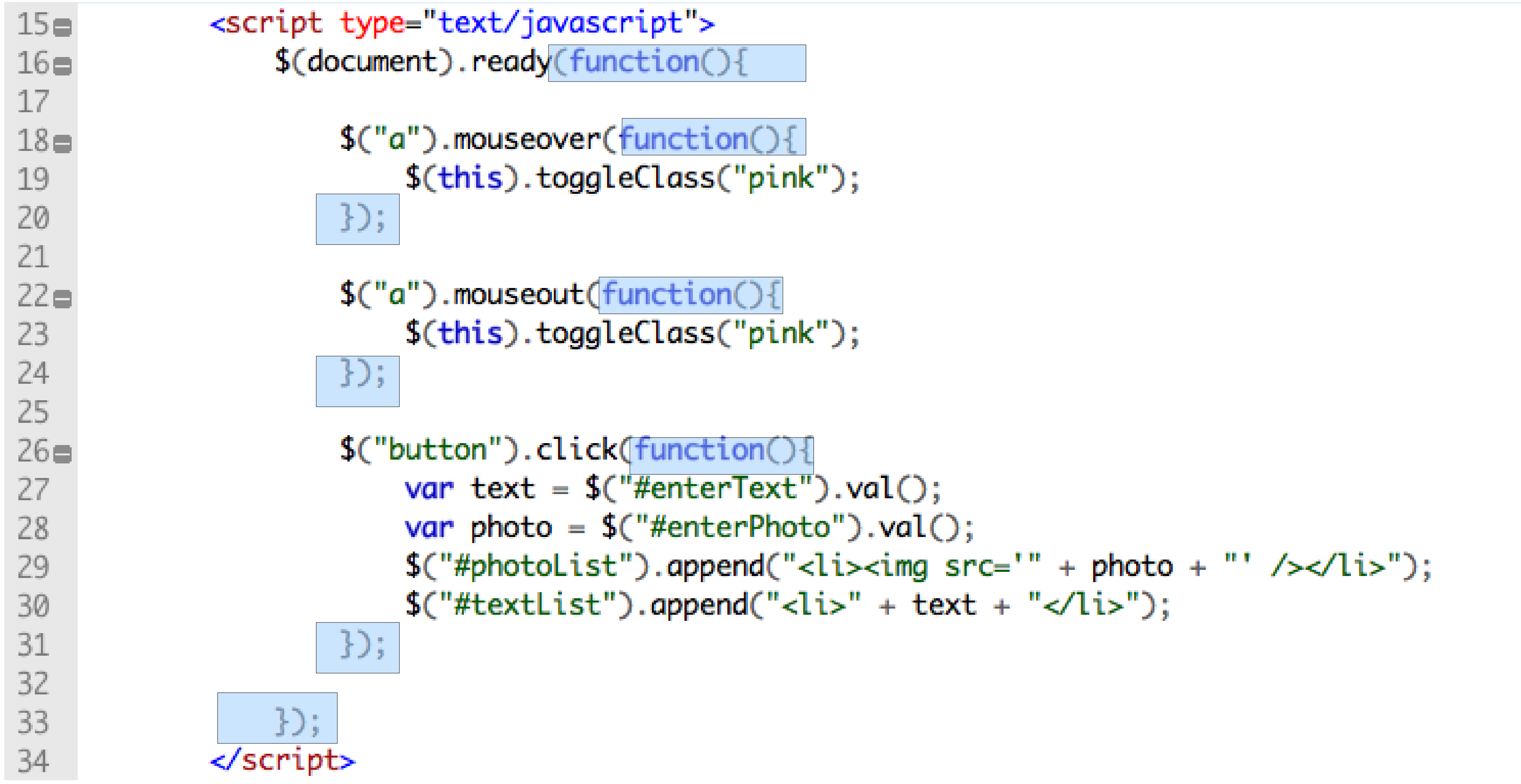
- There is another way we could have written the same code, in a slightly more condensed way, than the way we just reviewed.
- This other way leverages something called an **Anonymous Function**.

There's more than one way to do it...



```
<script type="text/javascript">
    $(document).ready(function(){

        $("a").mouseover(function(){
            $(this).toggleClass("pink");
        });

        $("a").mouseout(function(){
            $(this).toggleClass("pink");
        });

        $("button").click(function(){
            var text = $("#enterText").val();
            var photo = $("#enterPhoto").val();
            $("#photoList").append("<li><img src='" + photo + "' /></li>");
            $("#textList").append("<li>" + text + "</li>");
        });

    });
</script>
```

Anonymous Functions



```
<script type="text/javascript">
    $(document).ready(function(){

        $("a").mouseover(function(){
            $(this).toggleClass("pink");
        });

        $("a").mouseout(function(){
            $(this).toggleClass("pink");
        });

        $("button").click(function(){
            var text = $("#enterText").val();
            var photo = $("#enterPhoto").val();
            $("#photoList").append("<li><img src='" + photo + "' /></li>");
            $("#textList").append("<li>" + text + "</li>");
        });

    });
</script>
```

Anonymous Functions—what are they?

- An anonymous function is all those **function() { /*some code */ });** blocks you see passed to events.
  - Example: $(“p”).hide(function(){
    alert(“the paragraph is now hidden!”);
    });
- Anonymous functions don't have names.
- This is NOT an anonymous function: function passTheSalt() { … }

Anonymous Functions—placement matters

- When you see an anonymous function inside an event, you are **binding** the anonymous function to that event.

  -
    ```
    $(“p”).hide(function(){
        alert(“the paragraph is now hidden!”);
    });
    ```
  - In the above example, we are binding the anonymous function whose code displays an alert to the **hide** event.

Anonymous Functions—when they are **required**

- If a function takes a parameter, and you want to call that function from within an event, you **must** use Anonymous Functions.
- Let's review this in the sample code file, anonFxnExample.html
  - This file is contained within the zip file at: **http://bit.ly/jsCode3**

Anonymous Functions—when they are **required**

- Let's say we have a function that takes one parameter:

```
function fadeLinkAway(aLink) {
  $(aLink).fadeOut();
}
```

- The function accepts a link, and then fades that link out.
- If you try and call it like this, it will **NOT** work:

```
  $("a").mouseover(fadeLinkAway2(this));
```

Anonymous Functions—when they are **required**

```
function fadeLinkAway(aLink) {
  $(aLink).fadeOut();
}
```

- Because you are passing a parameter, you must use an anonymous function:

```
  $("a").mouseover(function(){
    fadeLinkAway(this);
  });
```

Anonymous Functions—when they are **required**

- To read more, see the "Callback with arguments" section of "How jQuery Works" by John Resig (creator of jQuery).

Extra Topics: Validation plugins and Regular Expressions

- Time permitting, we will discuss the jQuery validation plugin, which allows us to validate form data: http://docs.jquery.com/Plugins/validation
  - See the example here: http://docs.jquery.com/Plugins/validation#Example
- The plugin uses Regular Expresions to do the validation. To read more about regular expressions: http://en.wikipedia.org/wiki/Regular_expression

Lab Time!

- Next up, we're going to do a new example that is similar to the bakery example, and re-uses many of the same jQuery actions.
- We are going to use this a base file:
  - LINK to be given in class
- We are going to modify this file to create a LOLCAT generator!

Homework

- Reading:
  - http://api.jquery.com/append/
  - http://docs.jquery.com/Val
  - http://api.jquery.com/toggleClass/
  - http://docs.jquery.com/Tutorials:Getting_Started_with_jQuer
  - http://docs.jquery.com/Plugins/validation
- Lab:
  - To be announced in class