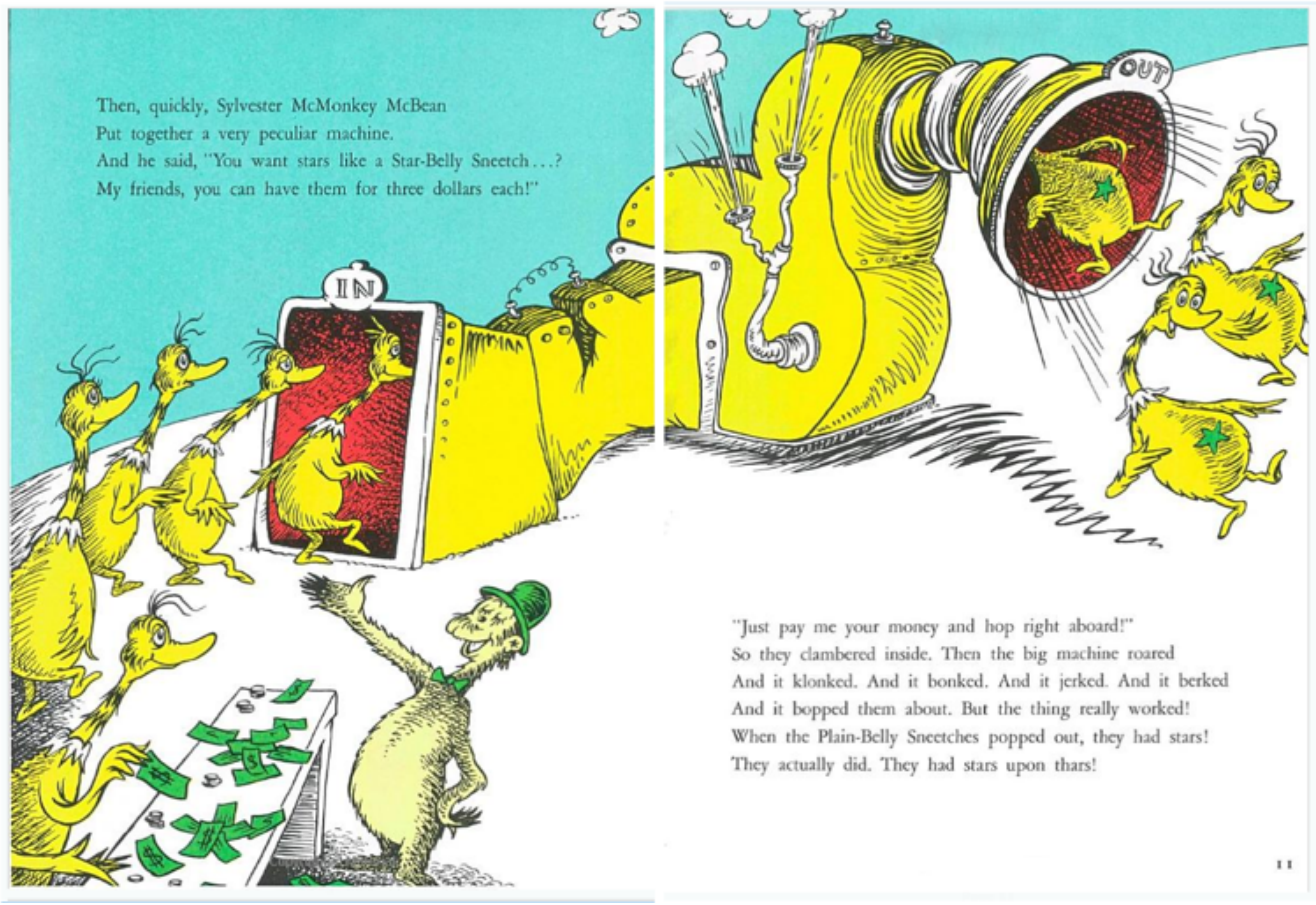
Lesson 8

more objects and DOM

Learning Objectives

- Identify likely objects, attributes, and methods in real-world scenarios
- Use event listeners to trigger events, using the DOM API

Exit ticket question: What is the nature of a function? Is the machine below a "does stuff" function, or a "returns stuff" function?



Then, quickly, Sylvester McMonkey McBean
Put together a very peculiar machine.
And he said, "You want stars like a Star-Belly Sneetch . . .?
My friends, you can have them for three dollars each!"

"Just pay me your money and hop right aboard!"
So they clambered inside. Then the big machine roared
And it klonked. And it bonked. And it jerked. And it berked
And it bopped them about. But the thing really worked!
When the Plain-Belly Sneetches popped out, they had stars!
They actually did. They had stars upon thars!

11

Quick object exercise: Objects in the real world

- One of the most useful and powerful approaches for breaking down larger problems into smaller, simpler problems involves considering the world and our code model of it in terms of a collection of objects interacting with each other.

- If we consider things in terms of objects and their methods, we have a powerful tool for organizing our code and our thoughts.

- This is the heart of Object-Oriented Programming (OOP)

Let’s consider the following scenario:

- *A user, browsing on a shopping website, searches for size 12 sneakers, and examines several pairs before purchasing one.*
- Where are the nouns? What are their attributes? These nouns are good candidates for objects.
- Where are the verbs? Which verbs naturally go with which nouns? These are good candidates for methods.
- What other objects and methods might be implicit in this scenario?

In pairs or small groups, consider one of the following scenarios. Identify likely objects, attributes and methods:

- Reporting software analyzes the snow removal performance of each snow plow driver in the city.

- A simulation predicts the behavior of the MTA if ridership increases by 20%.

- A user is required to watch video training sessions and answer questions about them, as part of a recertification process.

- A user on a cooking website enters the number of dinner guests, and the cooking website adjusts all the recipes accordingly.

- A user who had reserved a Zipcar arrives to find it has not been returned yet, and customer service transfers her reservation to an available car.

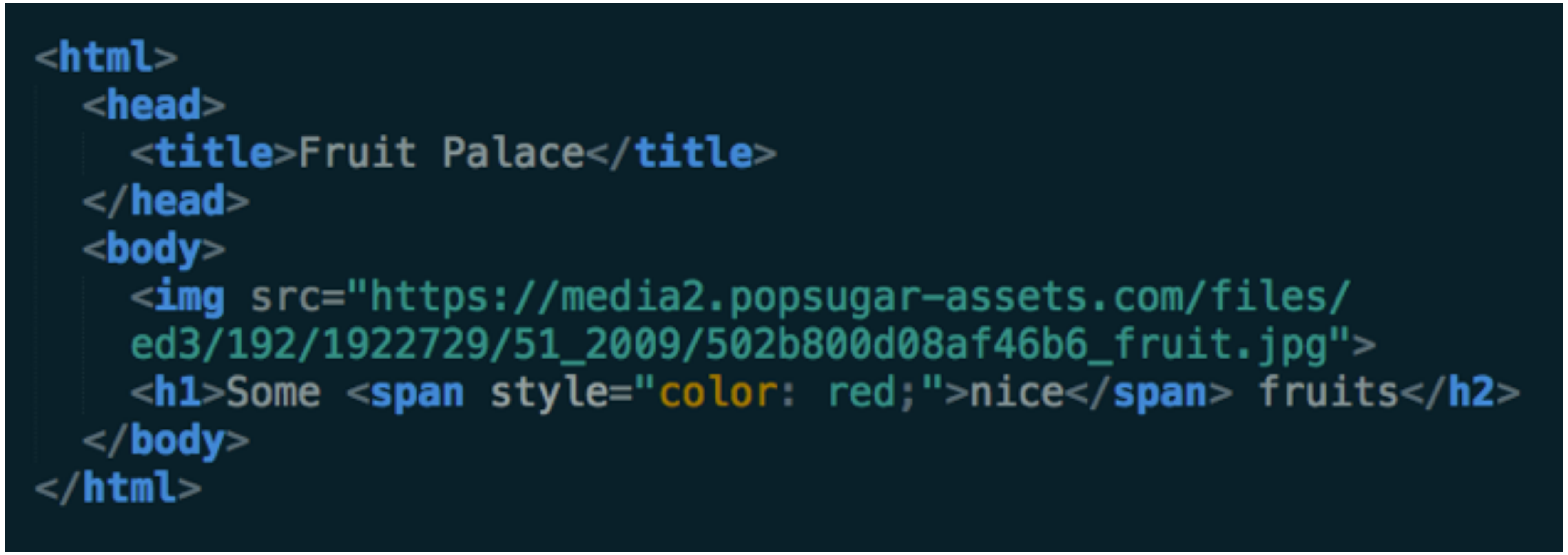
- A computer game allows the user to take the role of a unit commander or a general at Gettysburg and simulates the battle based on his or her commands.

- A user searches for her reservation on a hotel website, and changes the arrival date and room type.

Last week:
A page in 3 forms

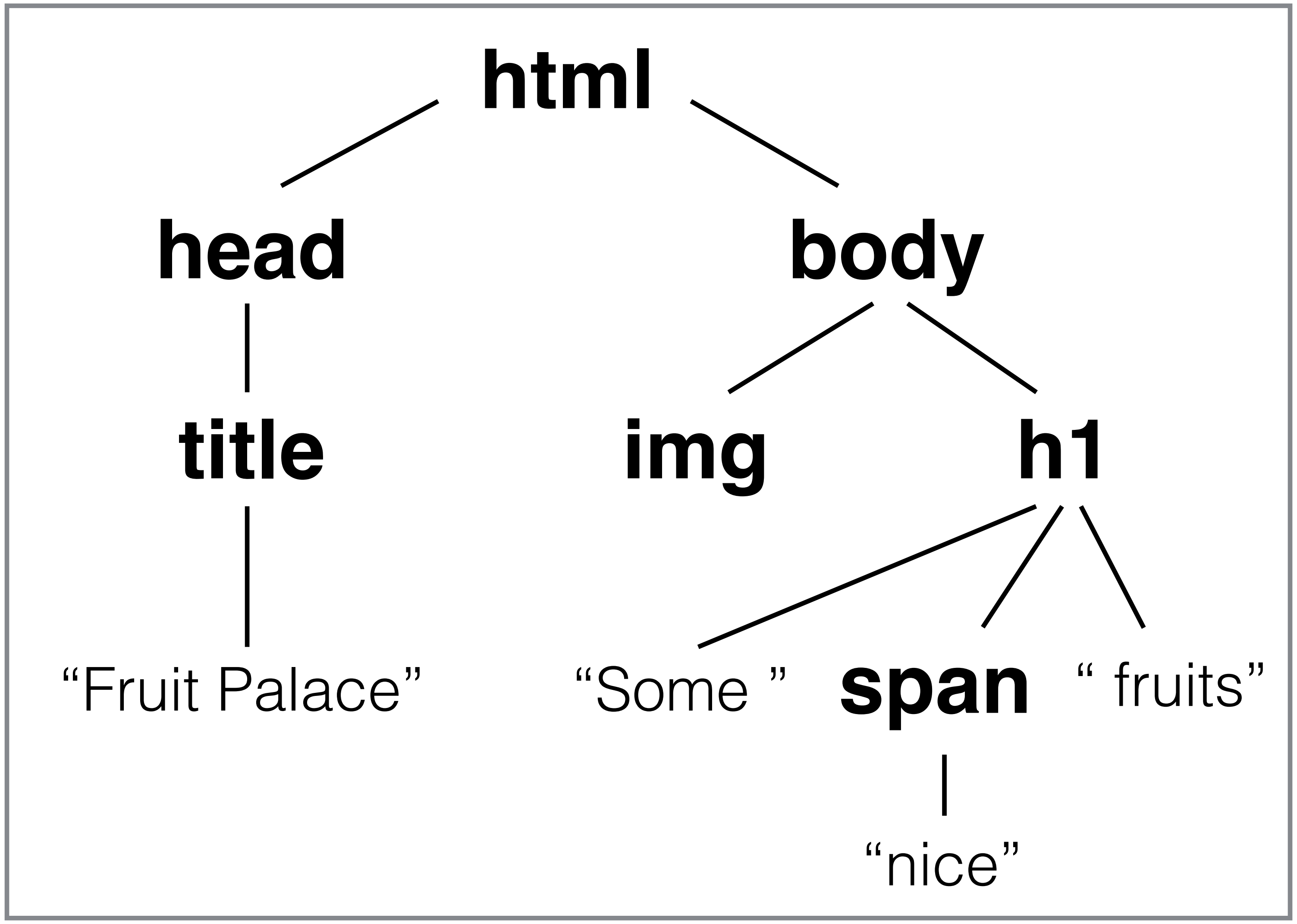
1. HTML text ->
2. DOM tree ->
3. rendered page

1. HTML

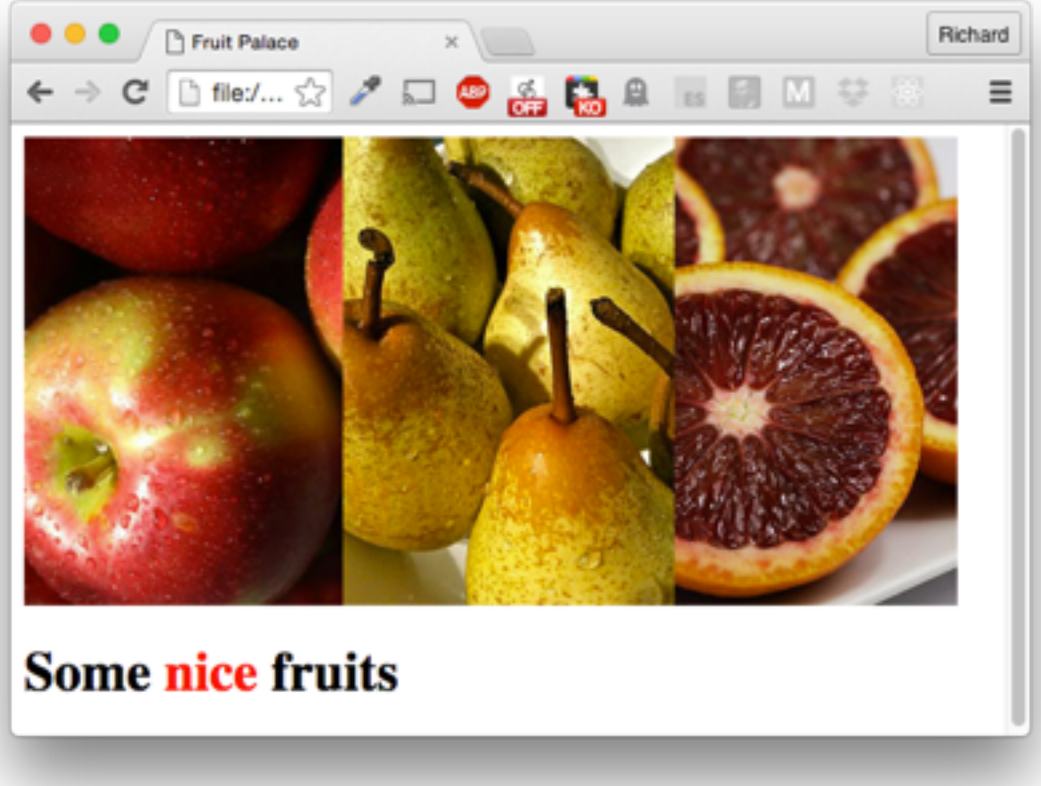


```
<html>
  <head>
    <title>Fruit Palace</title>
  </head>
  <body>
    <img src="https://media2.popsugar-assets.com/files/
    ed3/192/1922729/51_2009/502b800d08af46b6_fruit.jpg">
    <h1>Some <span style="color: red;">nice</span> fruits</h2>
  </body>
</html>
```

2. DOM tree



3. Voilà!



Some nice fruits

Codealong: quick review of creating and retrieving elements and text

- go to ~/GA-JS
- create a lesson8 folder
- create a file called dom.html
- create a file called dom.js

Practice on your own

- Using only JavaScript, add another **`<li>`** element to the end of the **`<ul>`** we just created. Make sure your new **`<li>`** element has some text in it.

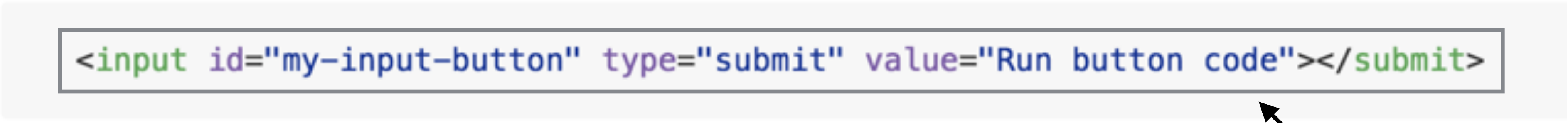
- *Bonus:* Make the last word of your new text green.

  - *Hint:* to make the text in a span called **`myElement`** green, do **`myElement.style.color = "green"`**

Events

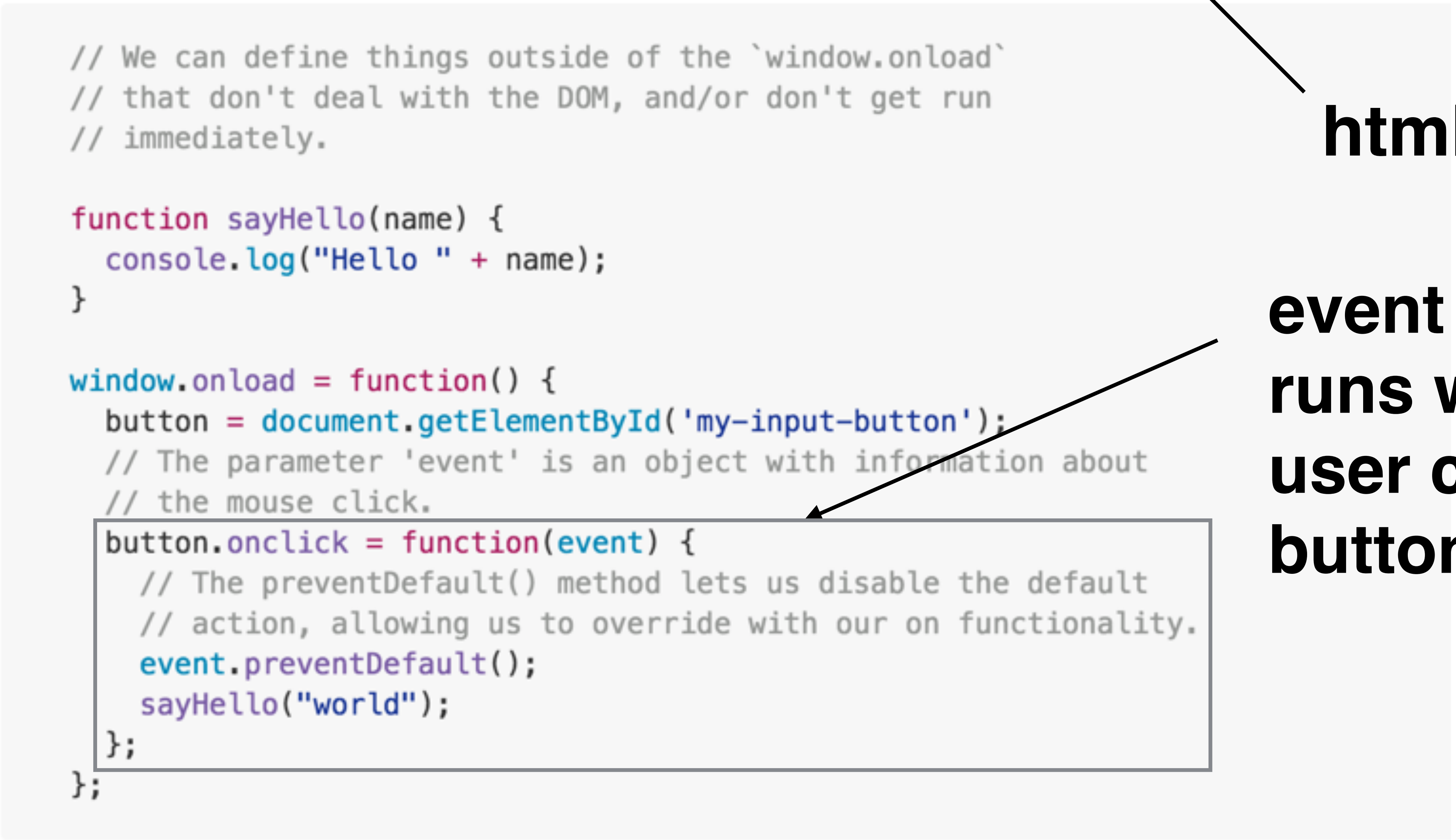
- We can use **event listeners** (also called “event handlers”) to designate certain code to run based on things that happen, called “events”.
- A user clicking a button is a common event
- The page finishing loading is another common event
  - we’ve already been using one event listener: **window.onload**

Setting an event listener



```html
<input id="my-input-button" type="submit" value="Run button code"></submit>
```

html element



```javascript
// We can define things outside of the `window.onload`
// that don't deal with the DOM, and/or don't get run
// immediately.

function sayHello(name) {
  console.log("Hello " + name);
}

window.onload = function() {
  button = document.getElementById('my-input-button');
  // The parameter 'event' is an object with information about
  // the mouse click.
  button.onclick = function(event) {
    // The preventDefault() method lets us disable the default
    // action, allowing us to override with our on functionality.
    event.preventDefault();
    sayHello("world");
  };
};
```

event listener: runs when the user clicks the button

Codealong

- create 2 new files in the lesson7 directory:
    - events.html
    - events.js

Lab: DOM manipulation

- http://bit.ly/jsdev2-lesson8-in-class-dom

- When the user clicks the submit button, take the value they've typed into the input box and add it to the list (remember: appendChild)

- Also, when a new item is added to the list, clear the input box.

    - *Hint:* The `value` property of the input box, before anyone types in it is the empty string.)