# Intro to Programming For the Web

## Session 8

jQuery/Javascript

# jQuery

- a Javascript library
- provides elegant way to manipulate the DOM + a few utilities
- it's just Javascript

# Javascript

- up to browser vendors to implement
  - all are dialects of a standard called ECMAScript
  - different implementations behave in slightly different ways
- it's evolving

Firefox

http://php-course.dev/scratch/index.php

HTTP

```
GET /index.php HTTP/1.1
Host: php-course.dev
```

apache

/Users/eric/Sites/php-course.dev/scratch/index.php

```
<!DOCTYPE html>
<html>
	<head>
	</head>
	<body>
		<h1>Madness??</h1>
		<strong>
			This is <?php echo 'PHP';?>!!!
		</strong>
	</body>
</html>
```

## php module

...This is <?php echo 'PHP';?>!!!

...This is PHP!!!

apache

HTTP

```
<!DOCTYPE html>
<html>
	<head>
	</head>
	<body>
		<h1>Madness??</h1>
		<strong>
			This is PHP!!!
		</strong>
	</body>
</html>
```

Firefox

http://php-intro.dev/

Most Visited Smart Bookmarks Getting Started Late

# Madness??

This is PHP!!!

Firefox

http://php-course.dev/scratch/index.php

HTTP

```
GET /index.php HTTP/1.1
Host: php-course.dev
```

apache

/Users/eric/Sites/php-course.dev/scratch/index.php

```
<!DOCTYPE html>
<html>
	<head>
	</head>
	<body>
		<h1>Madness??</h1>
		<strong>
			This is <?php echo 'PHP';?>!!!
		</strong>
		<script>alert('hi there');</script>
	</body>
</html>
```

## php module

...This is <?php echo 'PHP';?>!!!

...This is PHP!!!

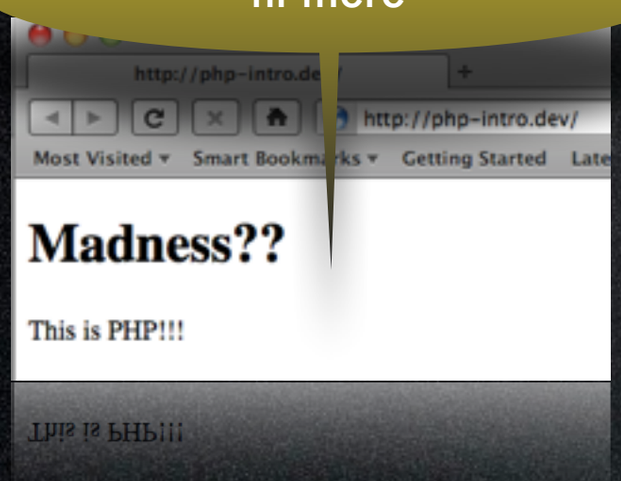
apache

HTTP

```
<!DOCTYPE html>
<html>
	<head>
	</head>
	<body>
		<h1>Madness??</h1>
		<strong>
			This is PHP!!!
		</strong>
		<script>alert('hi there');</script>
	</body>
</html>
```

Firefox

hi there



# Javascript in HTML

- inline within <script> tags
- or in a file, included with

  <script src="[path]"></script>

- inline scripts are interpreted synchronously as the page is parsed
- included scripts are fetched synchronously

# Javascript execution

- slow inline Javascript will delay load and display of page
- inline delay example

# Javascript execution

- slow included Javascript will delay load of page
- included delay example

# Javascript execution

- generally only run JS once the DOM is fully loaded anyway

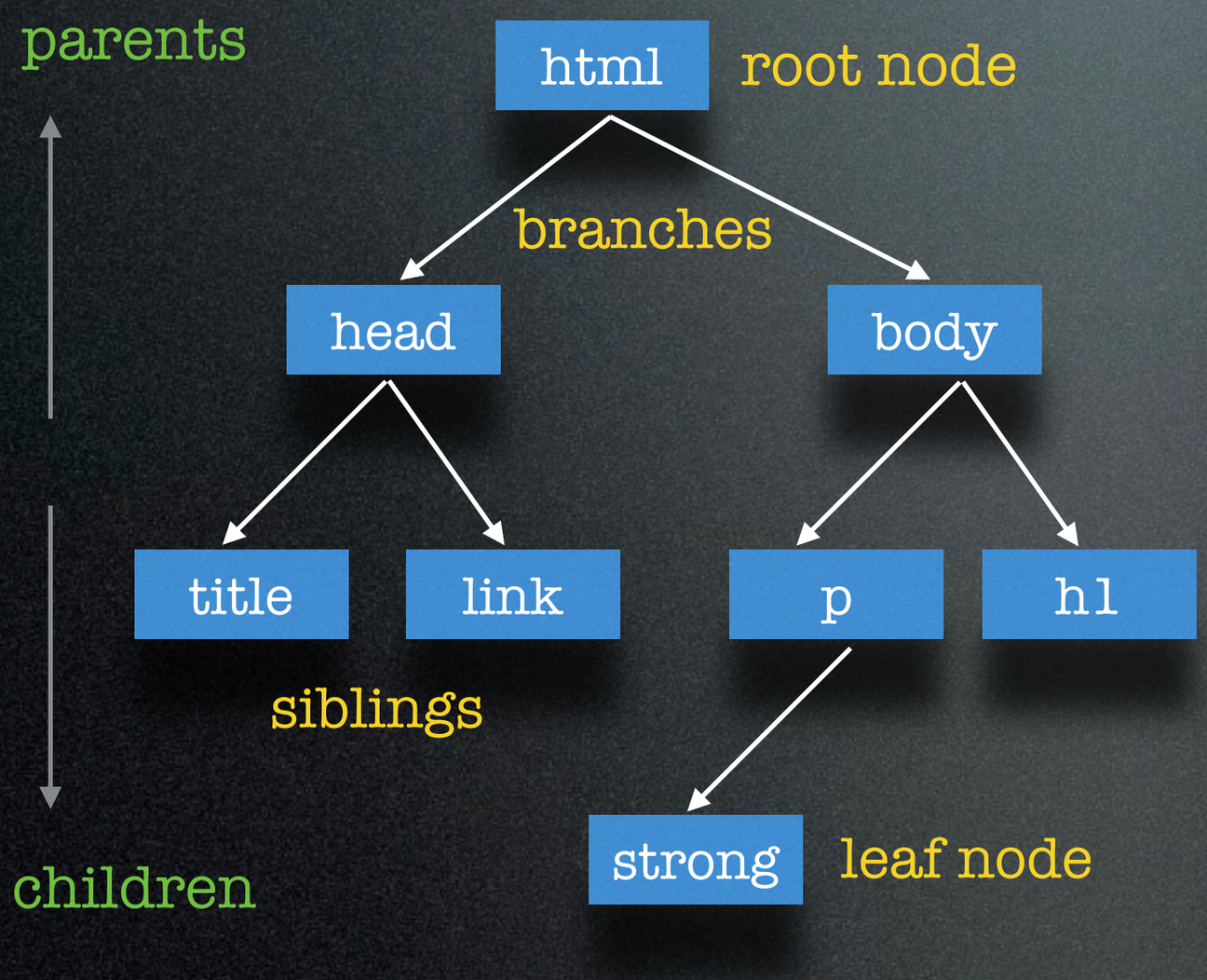
# Javascript

- best practice:
  - include files (rather than inline)
  - include just before </body> so page can be displayed while waiting
  - included at </body> example

# Javascript DOM manipulation

- DOM is a tree of Nodes
- Nodes have attributes
- some nodes are Elements
  - Elements have tagNames
- http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/

# HTML as Tree




```
<html>
    <head>
        <title>Pickles</title>
        <link href="style.css">
    </head>
    <body>
        <h1>Pickles, a Novel</h1>
        <p>
            Pickles are
            <strong>awesome</strong>.
        </p>
    </body>
</html>
```

# Javascript DOM manipulation

- browser exposes DOM and browser features (like current URL, alert dialog) via globals in Javascript:
  - document
  - window
  - location
  - alert
  - Math
  - etc...

# Javascript DOM manipulation

- document object
  - is root of DOM tree
  - contains factory methods for creating new nodes
  - and methods for finding nodes
    - getElementById
    - getElementsByTagName

# Javascript DOM manipulation

- appending hi there to the body

```
var pTag = document.createElement('p');
var pText = document.createTextNode(‘hi there’);
pTag.appendChild(pText);
var bodyTag = document.getElementsByTagName('body')[0];
bodyTag.appendChild(pTag);
```

# Javascript DOM manipulation

- turning all paragraphs red

```
var pElements = document.getElementsByTagName('p');
for (var i = 0; i < pElements.length; i++) {
    pElements[i].style.color = 'red';
}
```

# Javascript DOM manipulation

- task: find every element with class “hidden” and set its display property to none
- not hard if browser exposes getElementsByClassName, but...
- http://caniuse.com/getelementsbyclassname
- otherwise, it’s a lot of code.

# Javascript DOM manipulation

- using the DOM's exposed API directly requires a lot of coding
- also, different browsers will have slightly different implementations which must be accounted for

# Javascript DOM manipulation

- if only there were, i don't know ... a library of functions of some kind that abstracted all of those differences behind a powerful and expressive interface...

# jQuery

- based on the idea of “querying” the DOM for elements (as in a database query)
- uses CSS selector language

# jQuery

hey, jQuery

modify their style

to red

```
jQuery('p').css('color', 'red');
```

find all paragraphs

set their color

**vs**

```
var pElements = document.getElementsByTagName('p');
for (var i = 0; i < pElements.length; i++) {
    pElements[i].style.color = 'red';
}
```

# jQuery

hey, jQuery

and append to it ...

```
jQuery('body').append('<p>hi there</p>');
```

find the body element

this html

# jQuery

hey, jQuery

and hide them

```
jQuery('.hidden').hide();
```

find all elements
with class “hidden”

# jQuery

- jQuery(<argument>)
- returns an array-like object
  - which is a jQuery “result set”
  - with jQuery functions attached to it

# jQuery

- the result of a jQuery function is another jQuery result set
- on which another function can be invoked

# jQuery: chaining

hey, jQuery

find all <li>s in the #accordion, and ...

```
jQuery('#accordion li')
   .addClass('hidden')
   .filter('.visible-on-load')
      .removeClass('hidden')
;
```

add a "hidden" class

then, of those <li>s, find any with a class "visible-on-load"

and remove its "hidden" class

# jQuery and $

- “$” often used as shorthand for jQuery
- not always available
- best practice: rely on jQuery;
  put $ into scope yourself if you want it

# jQuery: $

```
<div id=”test” style=”display:none”></div>

<script>
    jQuery(‘#test’).fadeIn();
</script>
```

```
<div id=”test” style=”display:none”></div>

<script>
    $(‘#test’).fadeIn();
</script>
```

# jQuery: $

```
<div id="test" style="display:none"></div>

<script>
    var $ = jQuery;
    $('#test').fadeIn();
</script>
```

better

```
<div id="test" style="display:none"></div>

<script>
    (function($) {
        $('#test').fadeIn();
    })(jQuery);
</script>
```

best

anonymous function, invoked inline

# jQuery inclusion

- host locally or
- link directly to CDN-hosted: https://code.jquery.com
- WordPress comes with it
  - in your PHP theme/plugin: wp_enqueue_script('jquery');

# jQuery reference

- http://api.jquery.com

# jQuery exercises

- http://try.jquery.com
- http://jqexercise.droppages.com
- basic form:

```
$(‘css selector’).doSomething();
```

# Javascript Events

- click, hover, scroll, keypress, etc
- browsers expose these to Javascript
- we attach event “listeners” to elements
- JS method: addEventListener

# Javascript Events

```
<div id="container" style="background:red; height:400px;">
  <a id="change-bg" href="#">Change My Color</a>
</div>

<script>
    var link = document.getElementById('change-bg');
    link.addEventListener('click', function() {
        var container = document.getElementById('container');
        container.style.background = 'green';
    });
</script>
```

# jQuery Events

- again, use selector
- use on() method

# jQuery Events

```
<div id="container" style="background:red; height:400px;">
  <a id="change-bg" href="#">Change My Color</a>
</div>

<script>
    jQuery('#change-bg').on('click', function() {
        jQuery('#container').css('background, 'green');
    });
</script>
```

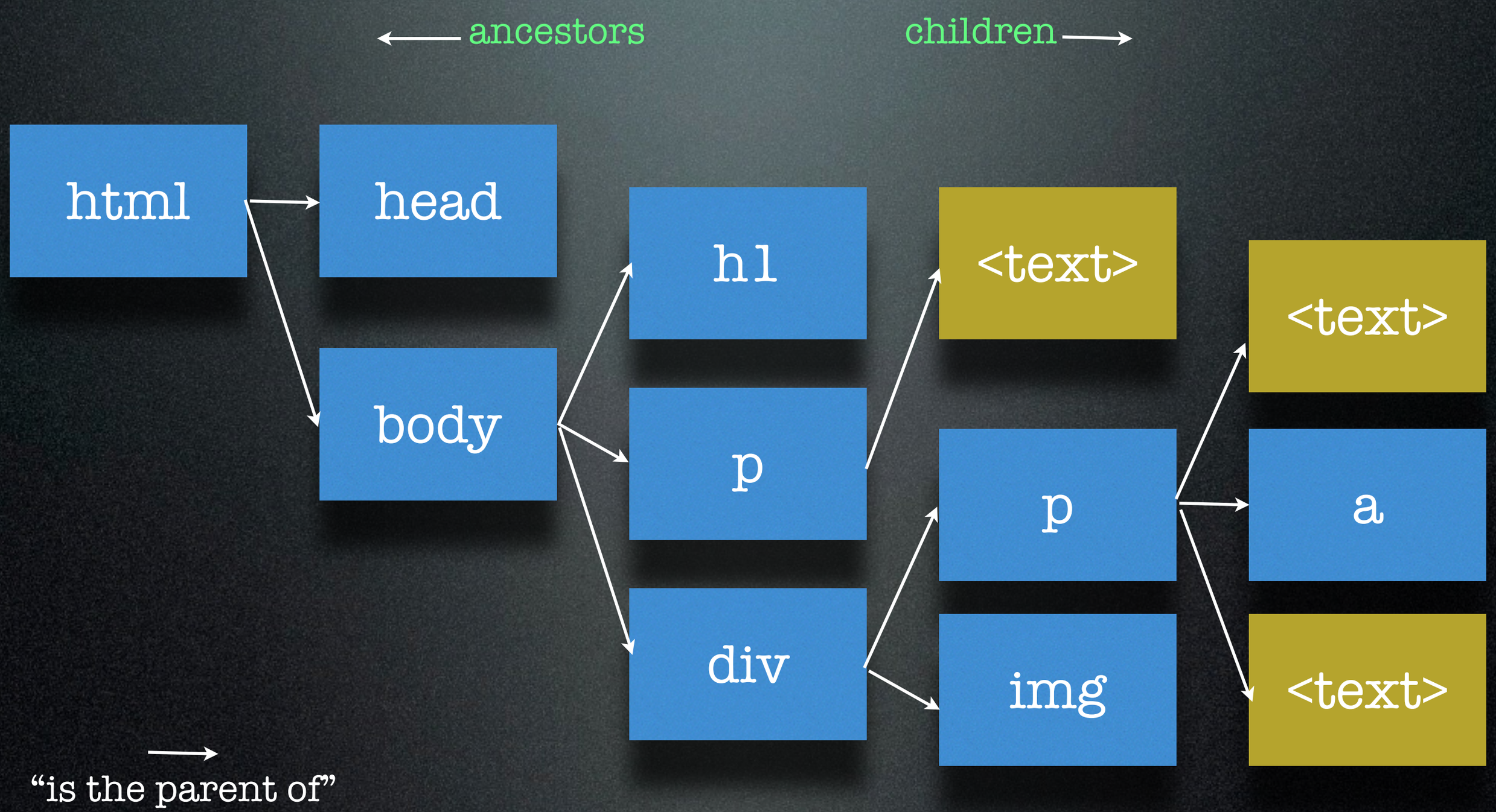
# Javascript Events

- events have two phases:
  - capture phase where they travel down the DOM tree to their target
  - bubble phase where they “bubble” up from their target
- can be handled anywhere along the ancestor path

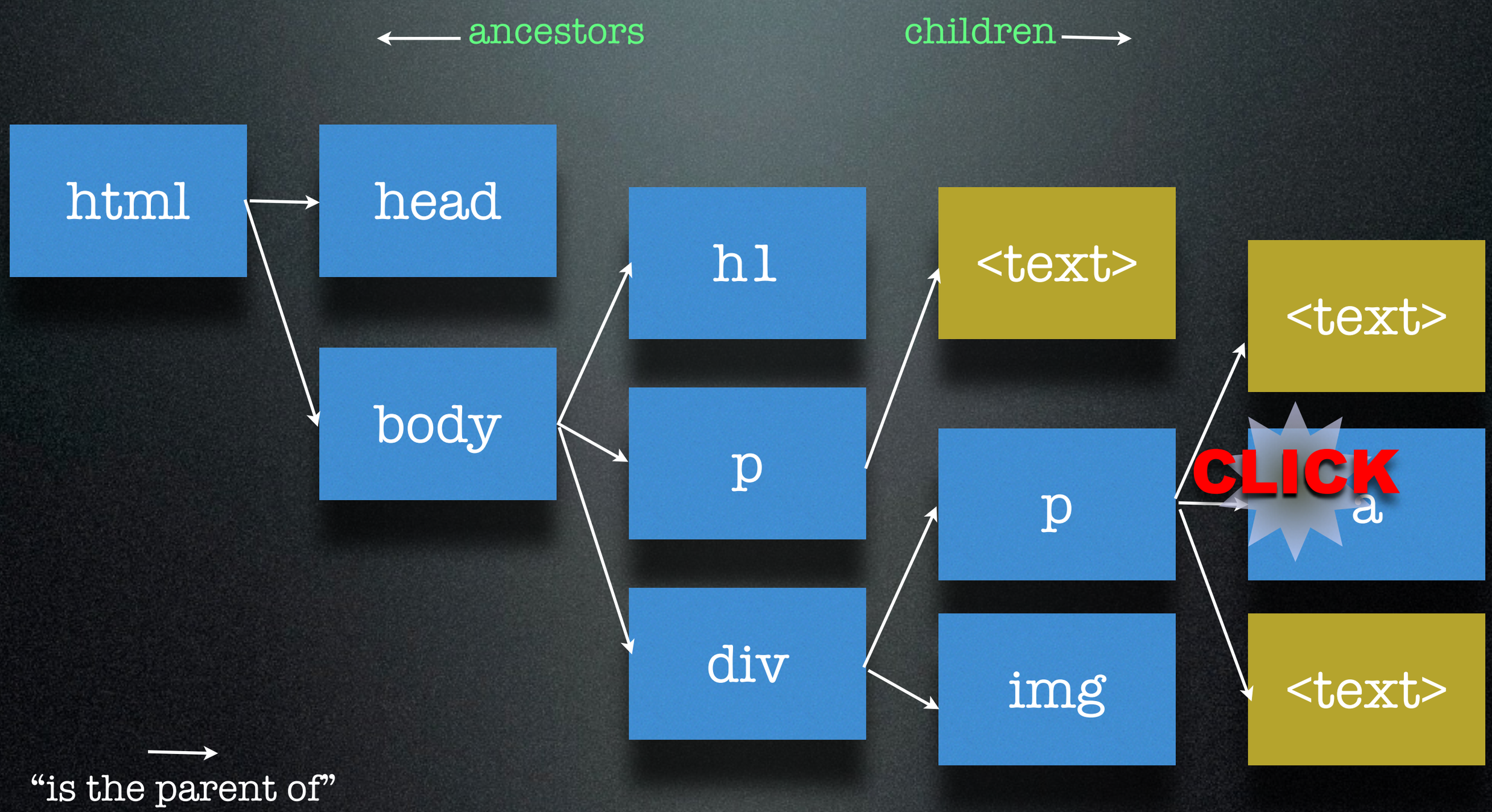
# Javascript Events

- capture phase not supported by all browsers, so rarely used in practice
- jQuery only deals with “bubble” phase

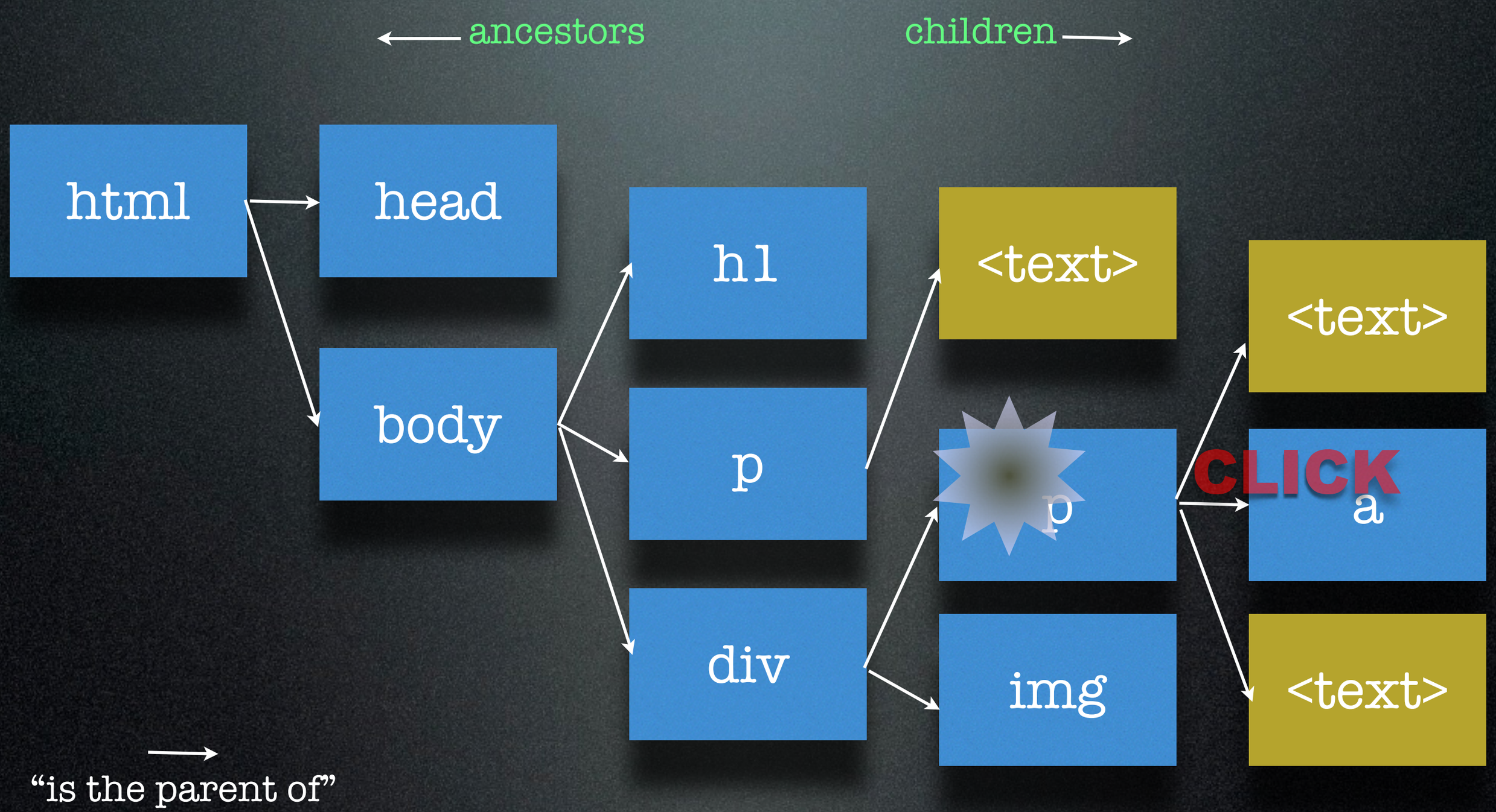
# Event Bubbling



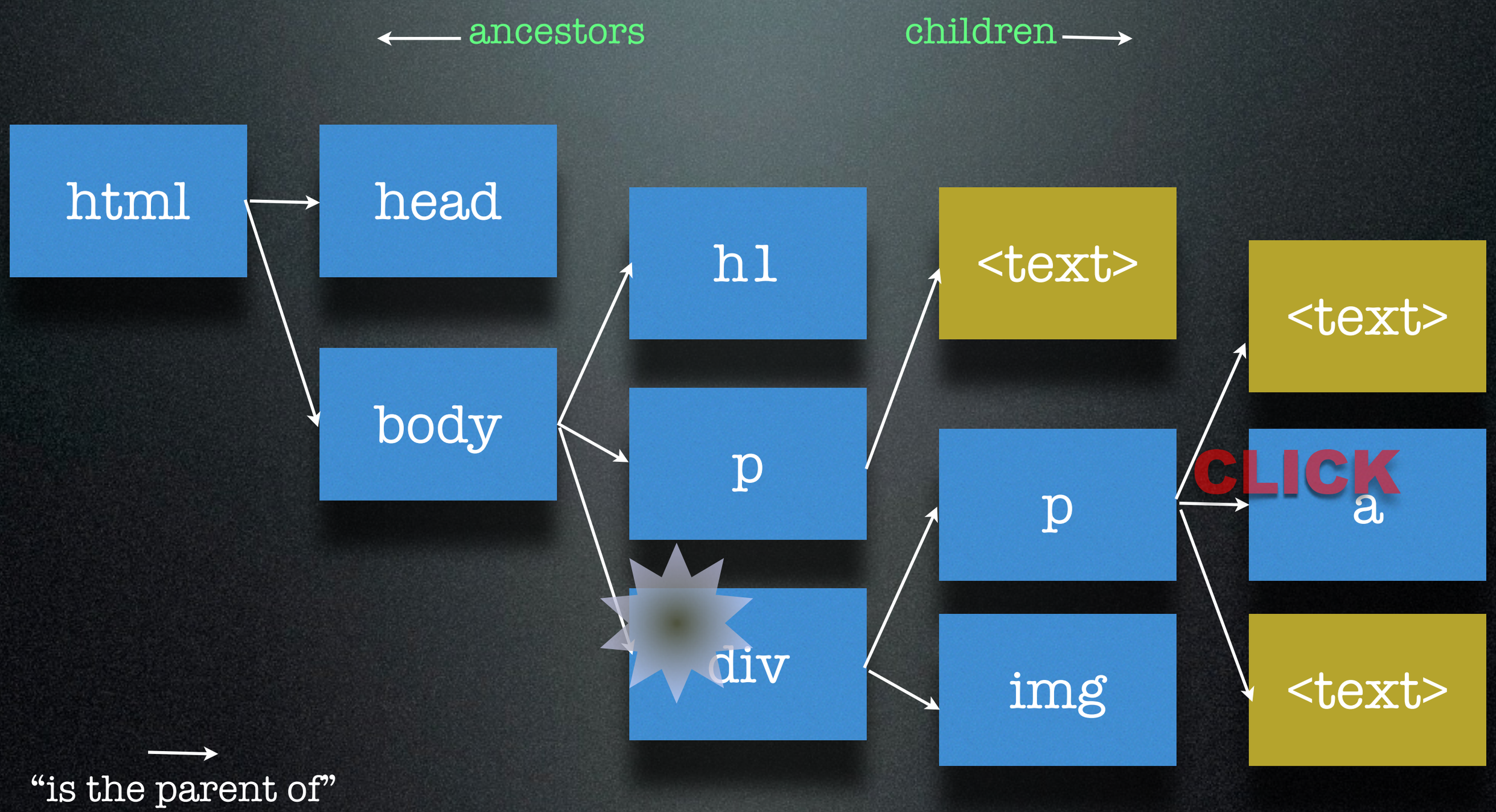
# Event Bubbling



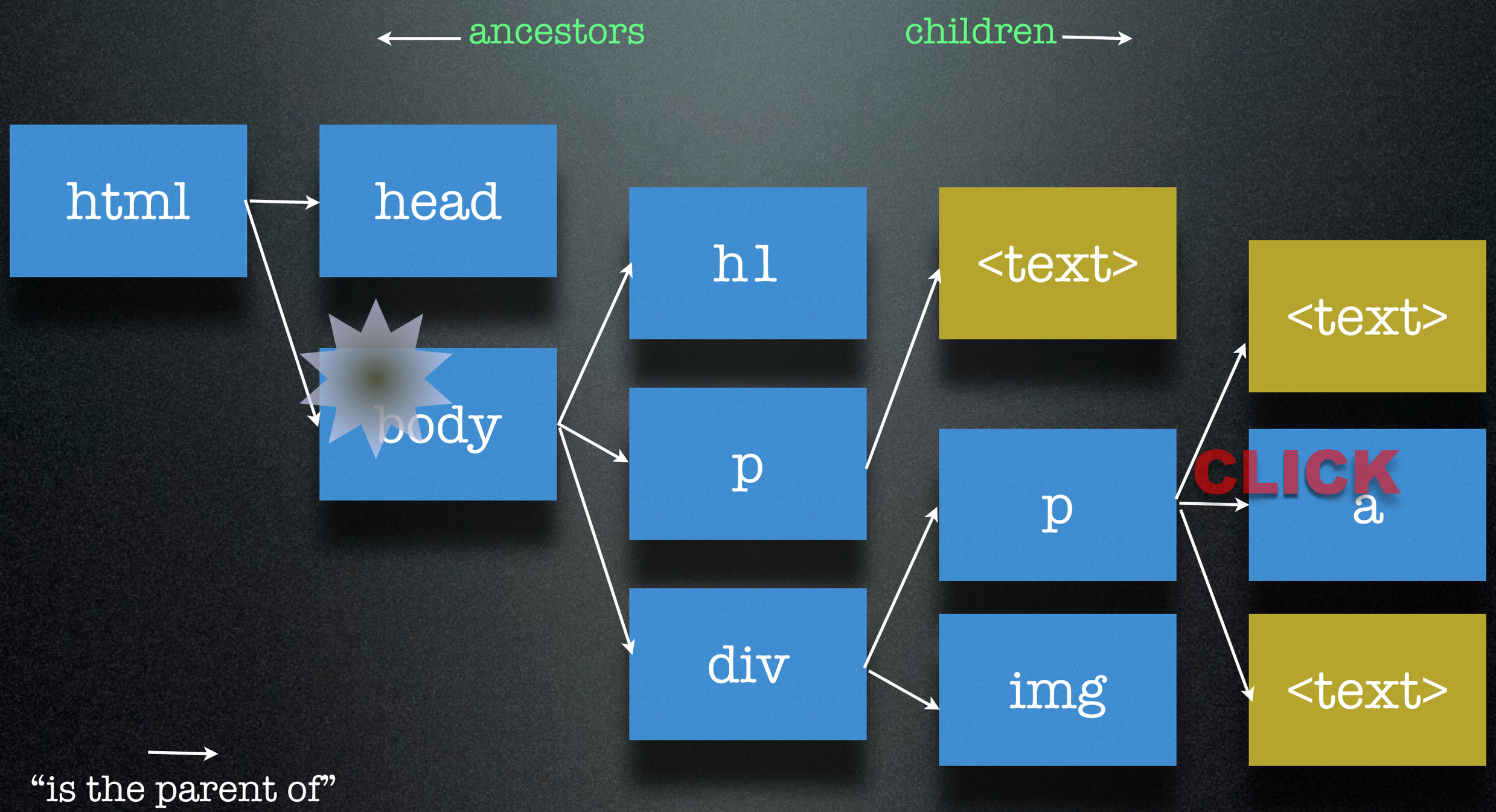
# Event Bubbling



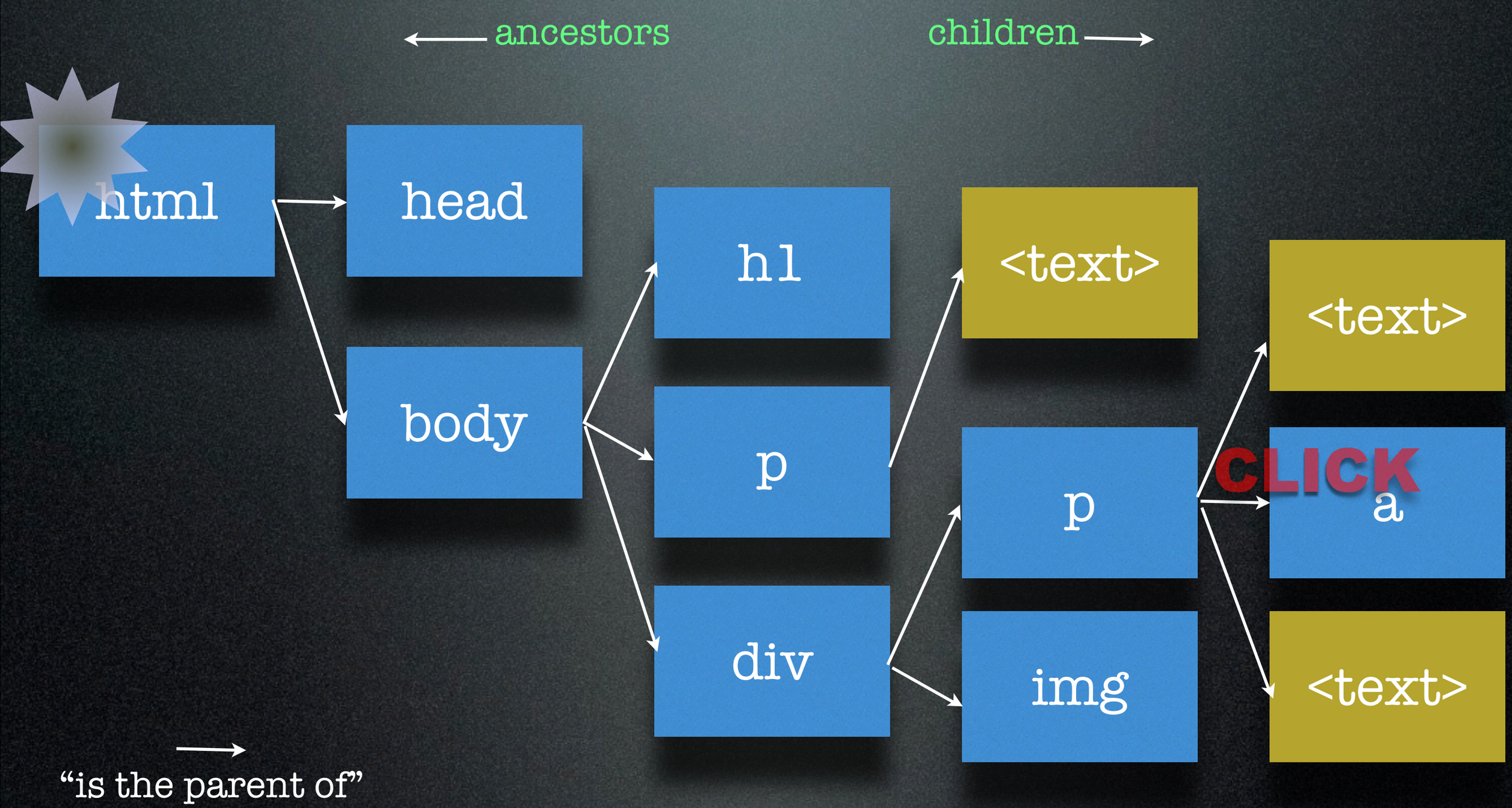
# Event Bubbling



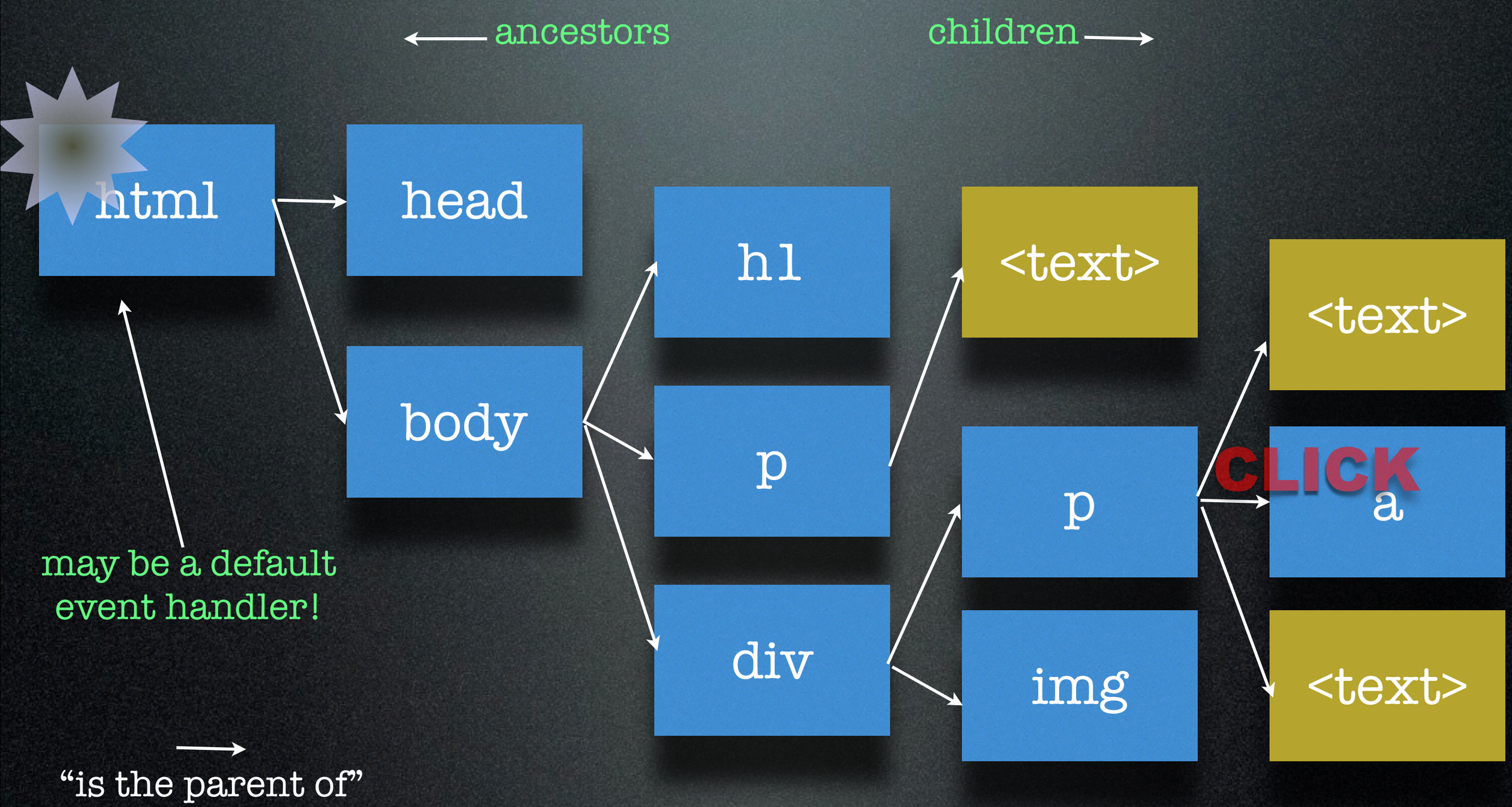
# Event Bubbling



# Event Bubbling



# Event Bubbling



# Events: default handler

- some events have default handlers
- anchor tags have a default handler which changes the browser location (URL)
- in jQuery you can return false from event handlers to stop the event from propagating

# jQuery

```
<div id="container" style="background:red; height:400px;">
  <a id="change-bg" href="#">Change My Color</a>
</div>

<script>
    jQuery('#change-bg').on('click', function() {
        jQuery('#container').css('background, 'green');
    });
</script>
```

the browser will load "#" after the event bubbles up to the document

# jQuery

```
<div id="container" style="background:red; height:400px;">
  <a id="change-bg" href="http://google.com">Change My Color</a>
</div>

<script>
    jQuery('#change-bg').on('click', function() {
        jQuery('#container').css('background, 'green');
    });
</script>
```

the browser will load google.com after the event bubbles up to the document

# jQuery

```
<div id="container" style="background:red; height:400px;">
  <a id="change-bg" href="http://google.com">Change My Color</a>
</div>

<script>
    jQuery('#change-bg').on('click', function() {
        jQuery('#container').css('background, 'green');
        return false;
    });
</script>
```

now it won't. the event stopped propagating so the default handler could not fire.

# JS/jQuery: “this”

- in Javascript, the this keyword refers to the context in which a function is running
- for now it’s enough to know that this in a jQuery event handler refers to the element on which an event was triggered

# JS/jQuery: “this”

```
<div id=”container” style=”background:red; height:400px;”>
  <a id=”change-bg” href=”#”>Change My Color</a>
</div>

<script>
    jQuery(‘#change-bg’).on(‘click’, function() {
        jQuery(‘#container’).css(‘background, ‘green’);
        return false;
    });
</script>
```

can be simplifed. we know we’re “inside” the anchor tag when this is running, so ...

# JS/jQuery: “this”

```
<div id=”container” style=”background:red; height:400px;”>
  <a id=”change-bg” href=”#”>Change My Color</a>
</div>

<script>
    jQuery(‘#change-bg’).on(‘click’, function() {
        jQuery(this).parent().css(‘background, ‘green’);
        return false;
    });
</script>
```

# jQuery exercise

- make an accordion

```
<ul id="accordion">
  <li>
    <h2>Heading 1</h2>
    <div class="content">Content 1</div>
  </li>
  <li class="visible-on-load">
    <h2>Heading 2</h2>
    <div class="content">Content 2</div>
  </li>
  <!-- etc, etc... -->
</div>
```