

<epam>

# JQuery

# AGENDA



jQuery  
overview



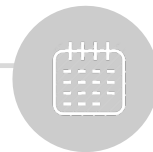
jQuery  
Selectors



jQuery  
Methods



JavaScript  
events



Event  
Handling



Custom  
Events



jQuery.AJAX



jQuery.deferred



jQuery plugin



SUMMARY

# JQUERY SELECTORS

## **SELECTING ELEMENTS**

# Basic Selectors

| \$ (selector)      | Description   | Example   |
|--------------------|---|---|
| *                  | Selects all elements  | <code>\$('*');</code>   |
| tag                | Selects all elements with given tag name                            | <code>\$('div');</code><br><br><i>Native:</i><br><code>document.getElementsByTagName('div')</code>                |
| #id                | Selects a <u>single first</u> * element with the given id attribute | <code>\$('#someDiv');</code><br><br><i>Native:</i><br><code>document.getElementById('#someDiv')</code>            |
| .class             | Selects all elements with the given class                           | <code>\$('.someClass');</code><br><br><i>Native:</i><br><code>document.getElementsByClassName('someClass')</code> |
| Multiple selectors | Selects all elements with the given tag name and class (union)      | <code>\$('div, span, .someClass');</code>   |

## Getting DOM element

```
let listItems = $( 'li' );  
  
let rawListItem = listItems[0];  
// or  
let rawListItem = listItems.get(0);  
  
let html = rawListItem.innerHTML;
```

5

CONFIDENTIAL

## Hierarchy Selectors

| \$ (selector)  | Description  | Example   |
|----------------|--|---|
| Parent > child | Selects all direct child elements matching “child” of “parents” elements   | <code>\$(‘ul.topnav &gt; li’);</code>   |
| Anc desc       | Selects all elements that are descendants of a given ancestor  | <code>\$(‘div span’);</code><br><br><i>Native:</i><br><code>document.getElementsByTagName('div')</code> |
| Prev + next    | Selects all next elements matching “next” that are immediately preceded by a sibling “prev”  | <code>\$(‘label + input’);</code>   |
| Prev ~ sibling | Selects all sibling elements that follow after the “prev” element, have the same parent, and match the filtering “siblings” selector | <code>\$(‘#prev ~ div ’);</code>  |

## REFINING & FILTERING SELECTIONS

- `$("div.foo").has("p")` – div.foo elements that contains <p> tags
- `$("h1").has(".bar")` – h1 elements that have a class of bar
- `$("h1").not(".bar")` – h1 elements that don't have a class of bar
- `$("ul li").filter(".current")` – unordered list items with class of current
- `$("ul li").first()` – just the first unordered list item
- `$("ul li").eq(5)` – the sixth list item

CONFIDENTIAL

Sometimes the selection contains more than what you're after. jQuery offers several methods for refining and filtering selections.

## SELECTING FORM ELEMENTS WITH SPECIAL PSEUDO-CLASS SELECTORS

|           |           |           |
|-----------|-----------|-----------|
| :button   | :file     | :reset    |
| :checkbox | :image    | :selected |
| :checked  | :input    | :submit   |
| :disabled | :password | :text     |
| :enabled  | :radio    | :focus    |

jQuery offers several pseudo-selectors that help find elements in forms. These are especially helpful because it can be difficult to distinguish between form elements based on their state or type using standard CSS selectors.



## 🕒 Ancestors:

```
$el.parent()
```

```
$el.parents( ".parent" )
```

```
$el.parentsUntil( ".gparent" )
```

```
$el.closest( "div" )
```

```
$el.offsetParent()
```

## 🕒 Children:

```
$el.children( "div" )
```

```
$el.find( "div" )
```

## 🕒 Siblings:

```
$el.next()
```

```
$el.nextUntil()
```

```
$el.prev()
```

```
$el.prevUntil()
```

```
$el.nextAll()
```

```
$el.siblings()
```

```
$el.prevAll()
```

Once you've made an initial selection with jQuery, you can traverse deeper into what was just selected. Traversing can be broken down into three basic parts: parents, children, and siblings.

jQuery has an abundance of easy-to-use methods for all these parts

# JQUERY METHODS

## GETTERS AND SETTERS

```
// The .html() method used as a setter  
$( "h1" ).html( "hello world" )
```

```
// The .html() method used as a getter;  
var headingHtml = $( "h1" ).html()
```

jQuery "overloads" its methods, so the method used to set a value generally has the same name as the method used to get a value. When a method is used to set a value, it's called a setter. When a method is used to get (or read) a value, it's called a getter. Setters affect all elements in a selection. Getters get the requested value only for the first element in the selection.

# GETTING AND SETTING INFORMATION ABOUT ELEMENTS

## GETTERS

`$el.html()` – get element's content  
`$el.text()` – get stripped HTML  
`$el.attr( attributeName )` – get HTML attribute  
`$el.width()` – get width  
`$el.height()` – get height  
`$el.val()` – get the value

## SETTERS

`$el.html( htmlString )` – set element's content  
`$el.text( text )` – set stripped HTML  
`$el.attr( attributeName, value )` – set HTML attribute  
`$el.width( value )` – set width  
`$el.height( value )` – set height  
`$el.val( value )` – set the value

There are many ways to change an existing element. Among the most common tasks is changing the inner HTML or attribute of an element. jQuery offers simple, cross-browser methods for these sorts of manipulations. You can also get information about elements using many of the same methods in their getter incarnations.

CONFIDENTIAL

# Miscellaneous Traversing

| .method()                 | Description                                 |
|---------------------------|---|
| .add( selector )          | Add elements to the set of matched elements |
| .add( DOMelements )       |   |
| .add( htmlStrin )         |   |
| .add( jqObject )          |   |
| .add( selector, context ) |   |

CONTIA

# Class Attribute

More than one class may be added or removed at a time, separated by a space, to the set of matched elements

```
$el.hasClass(className) boolean  
$el.toggleClass('class1 class2 ...')  
$el.addClass('class1 class2 ...', )  
$el.removeClass(['class1 class2 ...'] )
```

```
// add two classes to p element  
$( "p" ).addClass( "myClass yourClass" );  
  
// remove two classes from p element and add a new one  
$( "p" ).removeClass( "myClass noClass" ).addClass("yourClass");
```

# DOM Insertion

## , Inside

| .method()   | Description  |
|---|--|
| <code>\$el.append( content, [content])</code><br><code>\$el.append( function(i, oldHtml)</code>   | Insert content, specified by the parameter; <b>to the end of each element</b> in the set of matched elements       |
| <code>\$el.prepend( content, [content])</code><br><code>\$el.prepend( function(i, oldHtml)</code> | Insert content, specified by the parameter; <b>to the beginning of each element</b> in the set of matched elements |
| <code>\$el.appendTo( target )</code>  | Insert every element in the set of matched elements <b>to the end of the target.</b>                               |
| <code>\$el.prependTo( target )</code>   | Insert every element in the set of matched elements <b>to the beginning of the target.</b>                         |

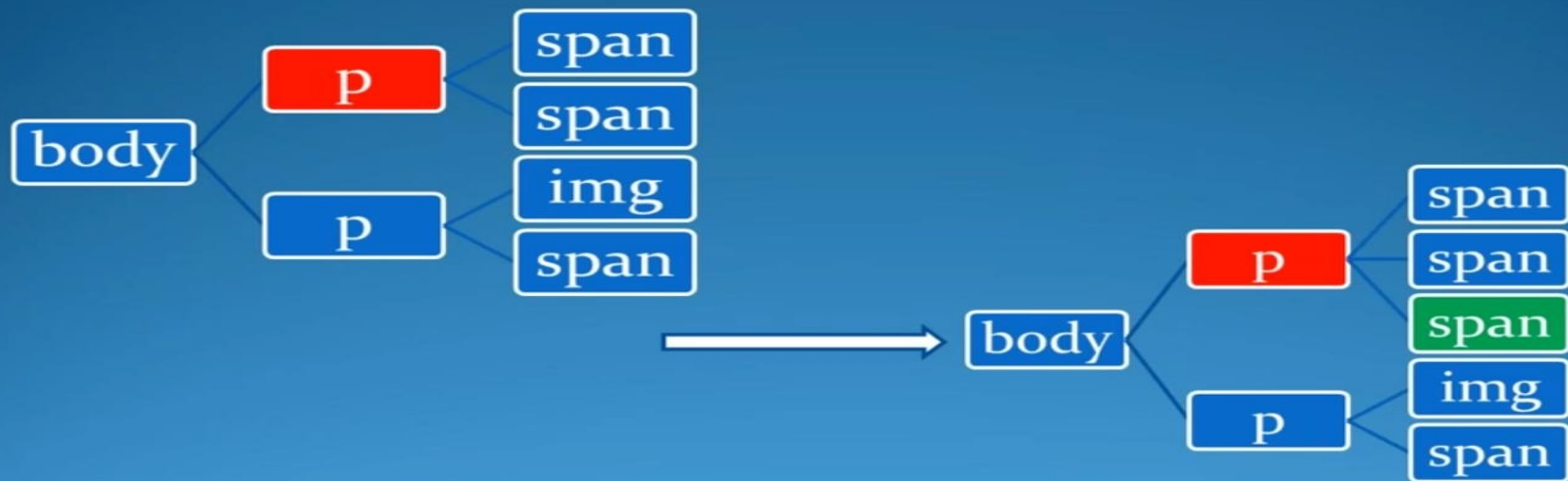
CONFIDENTIAL

15

- **Content** - DOM element, HTML string or jQuery object
- **Target** - selector, DOM element, HTML string or jQuery object

# Example

```
$(“p:first”).append("<span>Some Text</span>");
```



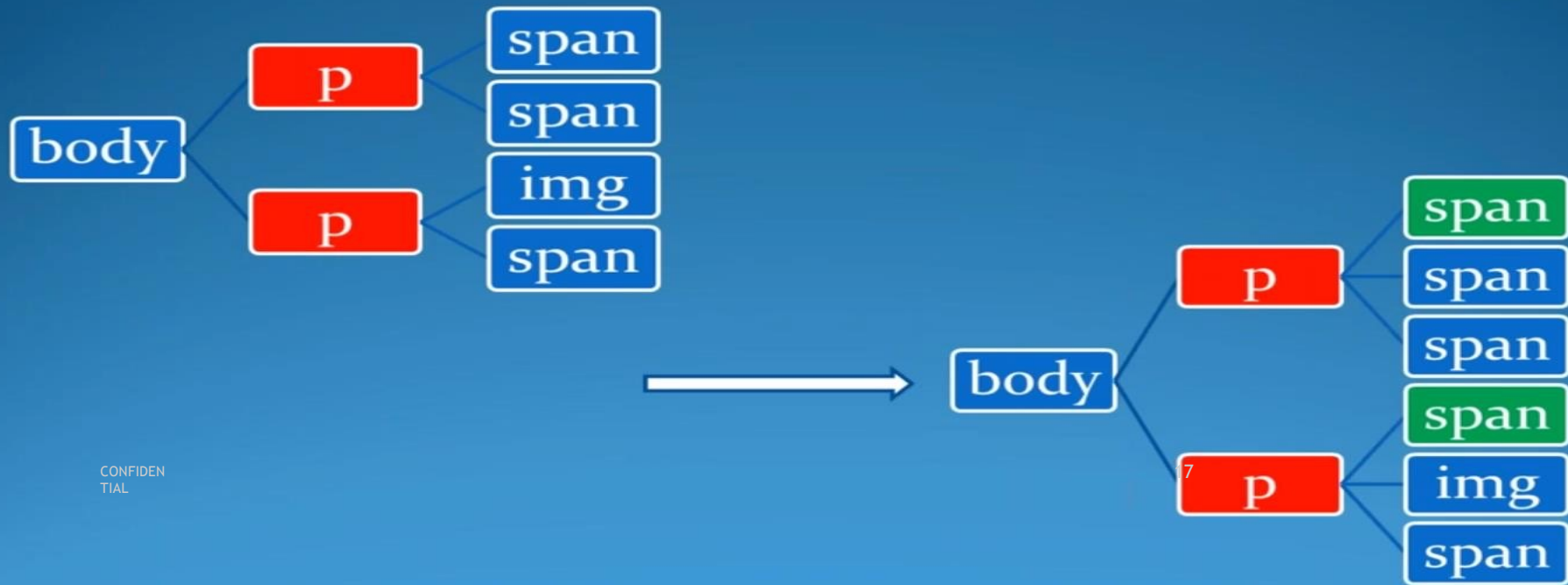
CONFIDENTIAL

16



# Example

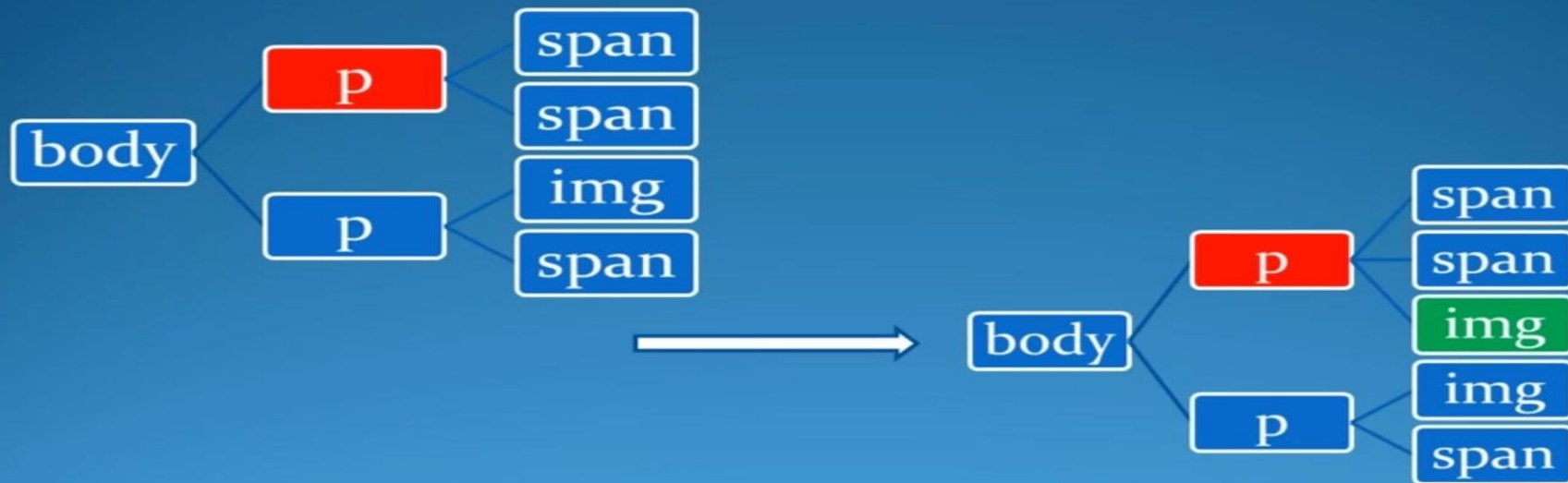
```
$(“p”).prepend(“<span>Some Text</span>”);
```



CONFIDENTIAL

# Example

```
$(("<img src='pic.jpg' />")).appendTo("p:first")
```



CONFIDENTIAL

18

# Example

```
$(("<img src='pic.jpg' />")).prependTo("p")
```



CONFIDENTIAL

## CREATING ELEMENTS FROM AN HTML STRING

```
$('<p>New paragraph</p>');  
  
$("<a/>", {  
  html: "This is a <strong>new</strong> link",  
  "class": "new",  
  href: "foo.html"  
});
```

jQuery offers a trivial and elegant way to create new elements using the same `$()` method used to make selections.

## CHAINING

```
$( "#content" ) // Selection of #content
  .find( "h3" ) // Selection of to all h3s in #content
    .eq( 2 ) // Reduce set to element h3 with 2rd index
      .html( "new text for the third h3!" ) // Set new innerHTML
    .end() // Restores the selection to all h3s in #content
      .eq( 0 ) // Reduce set to element h3 with 0 index
        .html( "new text for the first h3!" ); // Set new innerHTML
```

If you call a method on a selection and that method returns a jQuery object, you can continue to call jQuery methods on the object without pausing for a semicolon. This practice is referred to as "chaining".

CONFIDENTIAL

# JAVASCRIPT EVENTS

# EVENT TYPES



## Browser events

- error
- load



## Mouse Events

- click
- mousemove



## Mobile events

- touch
- orientationchange



## Form Events

- submit
- change / select



## Keyboard

- keydown
- keypress



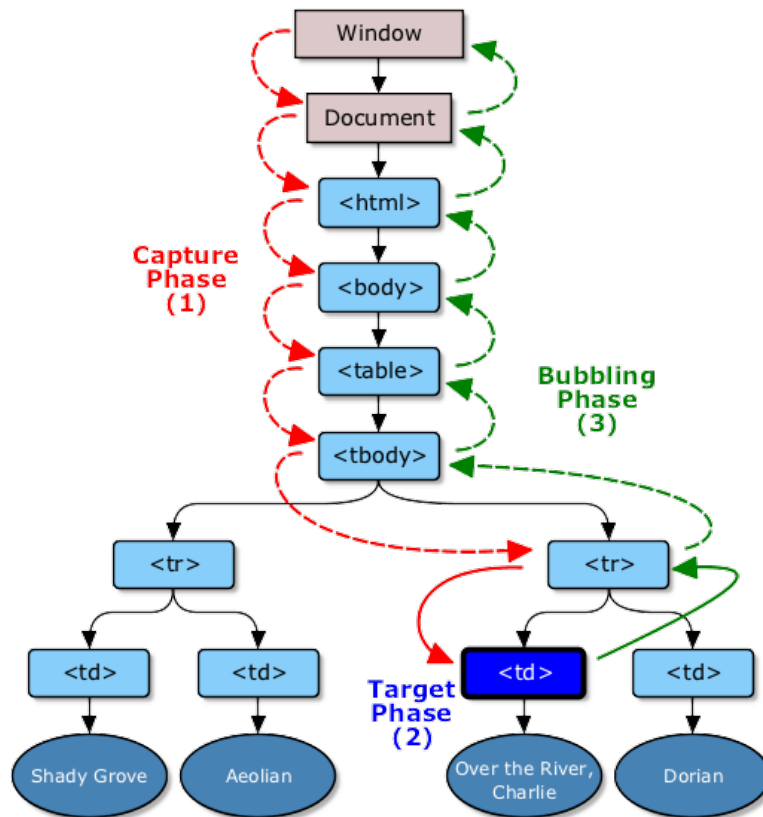
## Custom events

- makeMeASandwich
- awasomenessStart

23

CONFIDENTIAL

# EVENT FLOW



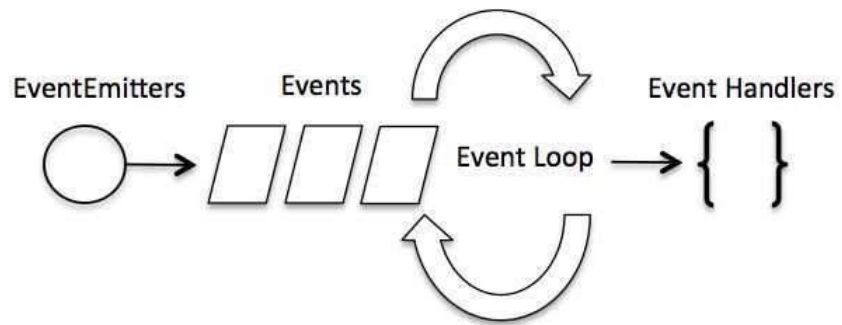
24

CONFIDENTIAL



# EVENT HANDLING

# EVENT BINDING



## JAVASCRIPT

```
var el = document.getElementById("outside");  
el.addEventListener("click", callback, false);
```



# CROSS BROWSER EVENT BINDING

## JAVASCRIPT

```
// add event cross browser
function addEvent(elem, event, fn) {
  if (elem.addEventListener) {
    elem.addEventListener(event, fn, false);
  } else {
    elem.attachEvent("on" + event, function() {
      // set the this pointer same as addEventListener when fn is called

      return(fn.call(elem, window.event));
    });
  }
}

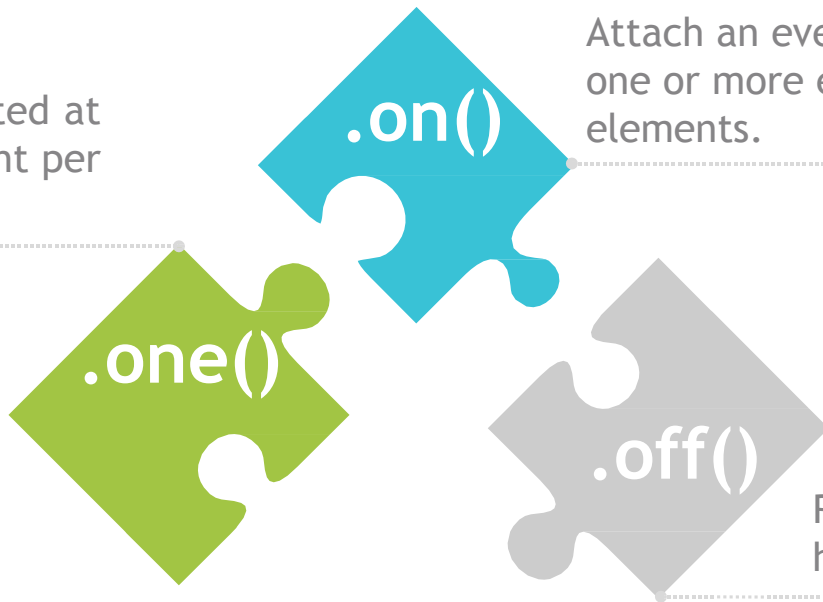
addEvent(document.getElementById('#element'), 'click', function() {
  console.log('Element clicked');
});
```

CONFIDENTIAL

27

# JQUERY EVENT BINDING

The handler is executed at most once per element per event type.



Attach an event handler function for one or more events to the selected elements.

Remove an event handler.

CONFIDENTIAL

# JQUERY EVENT BINDING

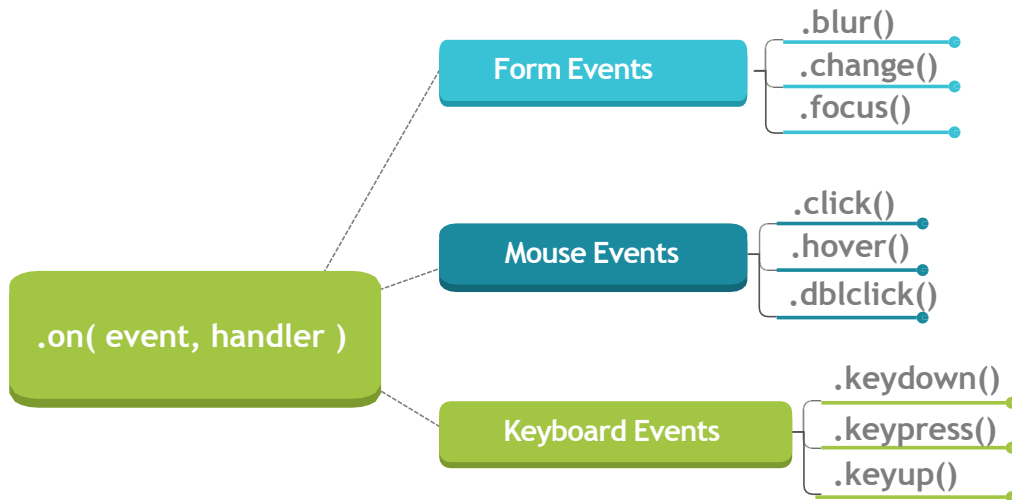
## JAVASCRIPT

```
$( "#element" )  
  .on( "click", function() {  
    console.log( $( this ).text() );  
  })  
  .one( "hover", function() {  
    console.log('This event fires only once');  
  })  
  .off("click"); //Remove click event handlers
```

CONFIDENTIAL

29

# SHORTCUT METHODS



CONFIDENTIAL

30

# JAVASCRIPT

```
$( "ul" ).on( "click", "h3", function( event ) {  
    console.log( event.target ); //<h3></h3>  
});
```

```
// Removing all delegated events without non-delegated
$( "ul" ).off( "click", "*" );
```

Takes up less memory

Fewer writers are better served when we encourage

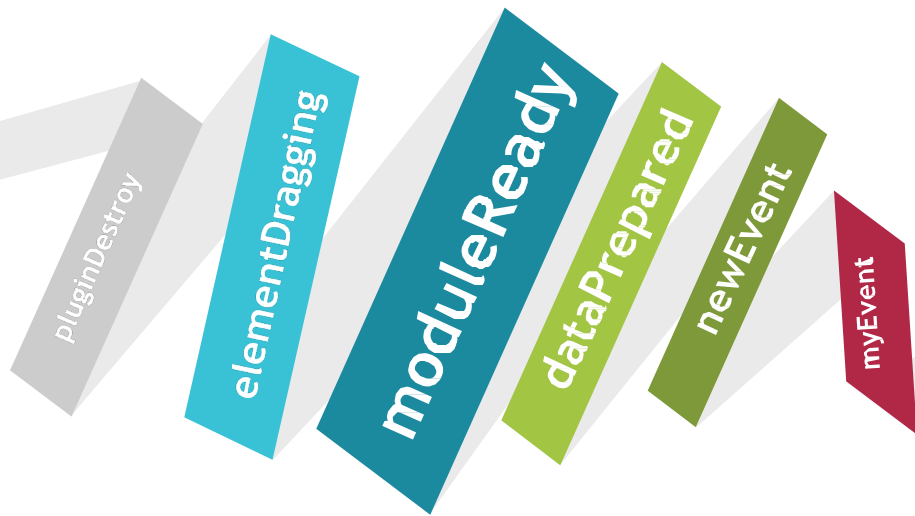
31

CUSTOM EVENTS



## CUSTOM EVENTS

Custom events open up a whole new world of event-driven programming.



CONFIDENTIAL

33

# CUSTOM EVENTS

```
$( element )  
  .on( "customEvent", function( event, val1, val2 ) {  
    console.log( val1 ); //foo  
    console.log( val2 ); //bar  
  })  
  // Triggering event and passing arbitrary data  
  .trigger( "customEvent", ['foo', 'bar'] );
```

## JS

```
//Alternative way to pass data to an event  
$( element ).trigger({  
  type: " customEvent",  
  message: "Hello World!",  
  time: new Date()  
});
```

34

JQUERY. AJAX



CONFIDENTIAL  
Perform an asynchronous HTTP (Ajax<sup>36</sup>) request

## High-level functions

---

- \$.get()
- \$.post()
- \$(elem).load()
- \$.getJSON()
- \$.getScript()

CONFIDENTIAL

## Low-level functions

---

- \$.ajax()
- \$.ajaxSetup()

# SIMPLE AJAX REQUEST EXAMPLE

## JAVASCRIPT

```
$.ajax({  
  method: 'POST',  
  url: '/link/to/host',  
  data: { id : menuId },  
  dataType: 'json'  
  success: function(){...},  
  error: function(){...},  
  complete: function(){...}  
});
```

CONFIDENTIAL

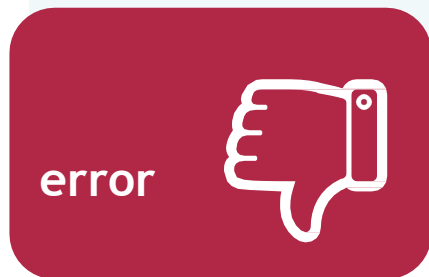
## List of data types

- ⦿ "xml"
- ⦿ "html"
- ⦿ "script"
- ⦿ "json"
- ⦿ "jsonp"
- ⦿ "text"

# JQUERY AJAX EVENTS



- data
- textStatus
- jqXHR



- jqXHR
- textStatus
- errorThrown



- jqXHR
- textStatus

CONFIDENTIAL

39



<http://api.jquery.com/jquery.ajax/>





JQUERY.DEFERRED



A factory function that returns a chainable utility object with methods to register multiple callbacks into callback queues, invoke callback queues, and relay the success or failure state of any synchronous or asynchronous function

 **fulfilled (resolved)** - The action relating to the promise succeeded

 **rejected (failed)** - The action relating to the promise failed

 **pending** - Hasn't fulfilled or rejected yet

 **settled** - Has fulfilled or rejected

CONFIDENTIAL

④ **.then(doneHandler, failHandler)**

④ **.done(handler)**

④ **.fail(handler)**

④ **.always(alwaysHandler)**

## ADDITIONAL JQUERY.DEFERRED METHODS



 `.resolveWith()`

 `.reject()`

 `.rejectWith()`

 `.progress()`

 `.notify()`

 `.notifyWith()`

 `.isResolved()`

 `.isRejected()`

 `.state()`

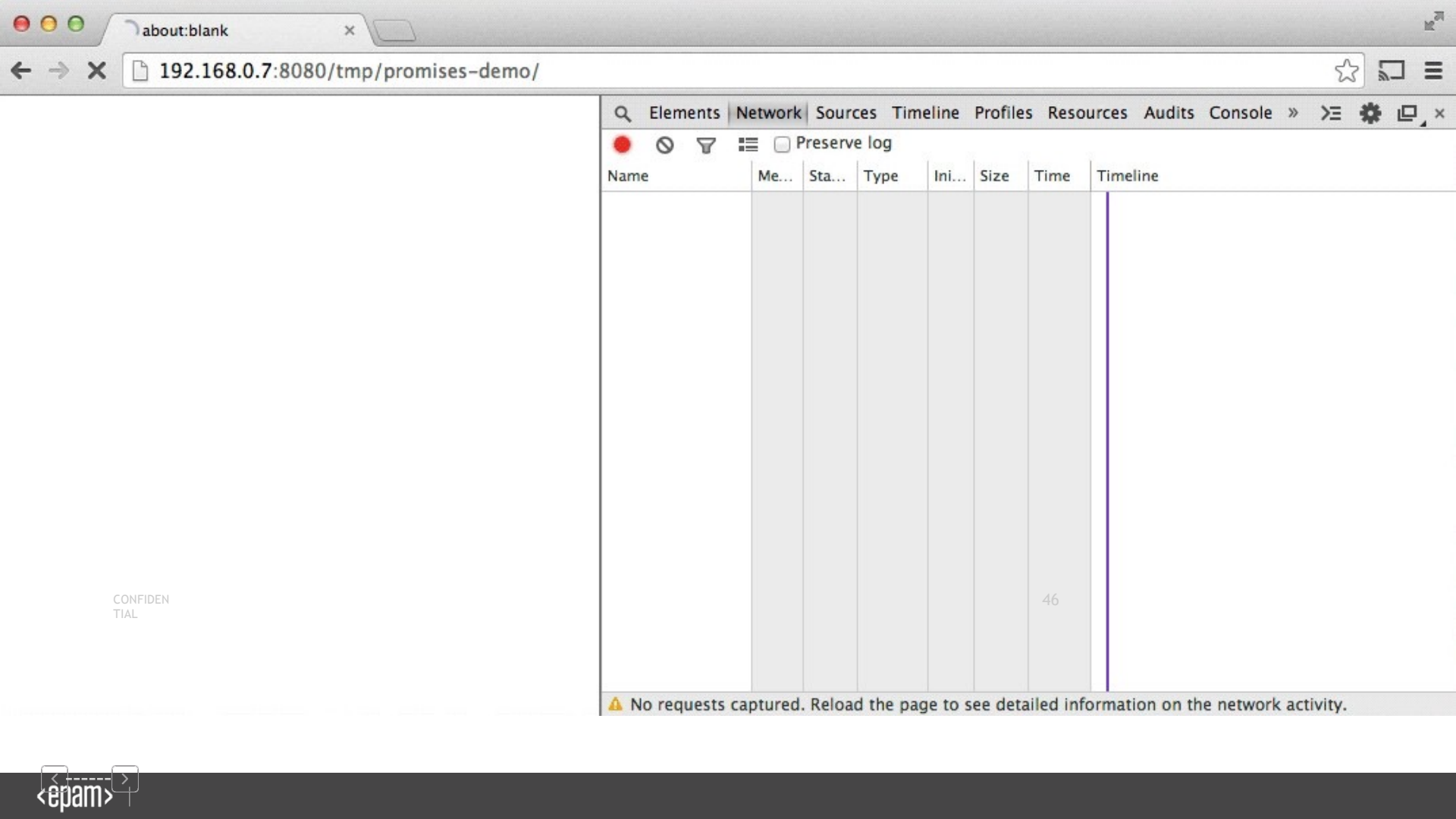
 `.promise()`

 `$(...).promise()`

 `$.when()`

45

CONFIDENTIAL



CONFIDENTIAL

## DEFERRED TO CONTROL STATE OF PROMISES

### JAVASCRIPT

```
// Creating a deferred  
var deferred = $.Deferred();  
  
if (condition) {  
    // resolving  
    deferred.resolve(value);  
} else {  
    // or rejecting  
    deferred.reject(error);  
}  
  
// Creating promise  
deferred.promise();
```

CONFIDENTIAL

47

## \$.when() WAITING ON MULTIPLE PROMISES TO FINISH WITH JQUERY

### JAVASCRIPT

```
var createPromise = function(multiple) {  
    var deferred = $.Deferred();  
    setTimeout(function() {  
        deferred.resolve(multiple);  
    }, multiple * 100);  
    return deferred;  
};  
  
var a = createPromise(10);  
var b = createPromise(11);  
var c = createPromise(12);  
var d = createPromise(13);  
  
$.when(a, b, c, d).then(function(v1, v2, v3, v4) {  
    console.log(v1, v2, v3, v4);  
});
```

CONFIDENTIAL

48



## \$.AJAX() JQUERY RETURNS PROMISES WITH ITS AJAX FUNCTION

### JAVASCRIPT

```
var ajaxPromise = $.ajax({  
  url: '/path/to/data',  
  dataType: 'json',  
  type: 'GET'  
});  
  
ajaxPromise.then(function(data) {  
  console.log(data);  
});
```

CONFIDENTIAL

49

# JQUERY.DEFERRED PROMISE

1

## JAVASCRIPT

```
function test() {  
    var d = $.Deferred();  
  
    setTimeout(function() {  
        someAction();  
        d.resolve();  
    }, 10000);  
  
    return d.promise();  
}
```

CONFIDENTIAL

2

## JAVASCRIPT

```
var defrr = test();  
  
defrr.done(function(){  
    alert('someAction');  
});  
  
if (defrr.isResolved()) {  
    alert('resolved');  
}
```

50

# JQUERY PLUGIN

# WHAT IS A PLUGIN

## Html

```
<ul class="carousel">
  <li class="item"><h4>1</h4></li>
  <li class="item"><h4>2</h4></li>
  <li class="item"><h4>3</h4></li>
  <li class="item"><h4>4</h4></li>
  <li class="item"><h4>5</h4></li>
  <li class="item"><h4>6</h4></li>
  <li class="item"><h4>7</h4></li>
  <li class="item"><h4>8</h4></li>
  <li class="item"><h4>9</h4></li>
</ul>
```

## JavaScript

```
$('.carousel').owlCarousel();
```

2

3

4

5

6

52

prev

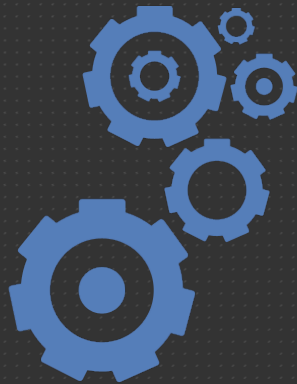
next



CONFIDENTIAL

# How JQuery works?

61



CONFIDENTIAL

53

`$('selector')` returns jquery object with all it's methods from `$.fn`

All we have to do is add a function called `greenify` to `$.fn`

JS

```
$.fn.greenify = function() {  
  this.css( "color", "green" );  
};  
  
$( "a" ).greenify(); // Makes all the links green.
```

Add a function to `$.fn`. It will be available like jQuery object method.

CONFIDENTIAL

54

all jQuery object methods **return** the original **jQuery object** again

JS

```
$.fn.greenify = function() {  
  this.css( "color", "green" );  
  return this;  
}  
  
$( "a" ).greenify().addClass( "greenified" );
```

return **this**

## PROTECTING THE \$ ALLIAS AND ADDING SCOPE

Put all of our code inside of an [Immediately Invoked Function Expression](#)

```
JS
(function ( $ ) {

    $.fn.greenify = function(){
        this.css("color", "green");
        return this;
    };

})(jQuery);
```

CONFIDENTIAL

Put plugin's code inside function  
and pass jQuery as argument \$



# ACCEPTING OPTIONS

It's a good idea to make your plugin **customizable** by accepting options

JS

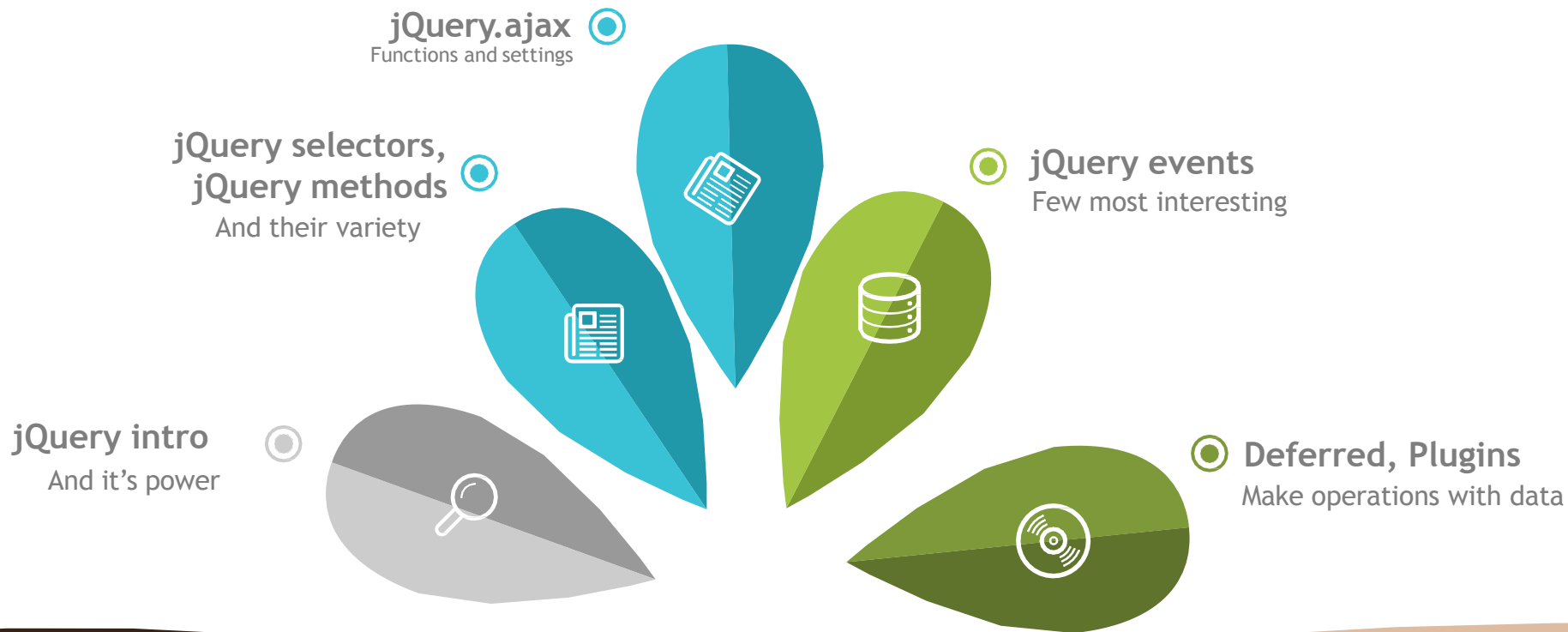
```
(function ( $ ) {  
  $.fn.greenify = function(options){  
    // This is the easiest way to have default options.  
    var settings = $.extend({  
      // These are the defaults.  
      color: "#556b2f",  
      backgroundColor: "white"  
    },options);  
    // Greenify the collection based on the settings variable.  
    return this.css({  
      color: settings.color,  
      backgroundColor: settings.backgroundColor  
    });  
  };  
})(jQuery);  
  
$("div").greenify({  
  color: "orange"  
});
```

CONFIDENTIAL

57

# CONCLUSION

# SUMMARY



## USEFUL LINKS

---

- <http://jquery.com/> - documentation
- <https://coursehunters.net/frontend/jquery>
- <http://anton.shevchuk.name/javascript/jquery-for-beginners>
- <https://github.com/jquery/jquery#how-to-build-your-own-jquery>
- <https://webref.ru/dev/jqfundamentals/events>

# FE Online UA Training Course Feedback

---

I hope that you will find this material useful.

If you find errors or inaccuracies in this material or know how to improve it, please report on to the electronic address:

Dmytro\_shakhov@epam.com

With the note [FE Online UA Training Course Feedback]

Thank you.

# Q&A



DRIVEN



CANDID



CREATIVE



ORIGINAL



INTELLIGENT



EXPERT

UA Frontend Online LAB