* Jump To … +  
    
  [<<< back to documentation](http://docs.google.com/index.html)  [base.js](http://docs.google.com/base.html)   [constraint.js](http://docs.google.com/constraint.html)   [defaults.js](http://docs.google.com/defaults.html)   [factory.js](http://docs.google.com/factory.html)   [field.js](http://docs.google.com/field.html)   [form.js](http://docs.google.com/form.html)   [main.js](http://docs.google.com/main.html)   [multiple.js](http://docs.google.com/multiple.html)   [pubsub.js](http://docs.google.com/pubsub.html)   [remote.js](http://docs.google.com/remote.html)   [ui.js](http://docs.google.com/ui.html)   [utils.js](http://docs.google.com/utils.html)   [validator.js](http://docs.google.com/validator.html)   [validator\_registry.js](http://docs.google.com/validator_registry.html)

# constraint.js

* [¶](#gjdgxs)  
  import Utils from './utils';  
  import Validator from './validator';  
    
  const Constraint = function(parsleyField, name, requirements, priority, isDomConstraint) {  
   const validatorSpec = window.Parsley.\_validatorRegistry.validators[name];  
   const validator = new Validator(validatorSpec);  
   priority = priority || parsleyField.options[name + 'Priority'] || validator.priority;  
   isDomConstraint = (true === isDomConstraint);  
    
   Object.assign(this, {  
   validator,  
   name,  
   requirements,  
   priority,  
   isDomConstraint  
   });  
   this.\_parseRequirements(parsleyField.options);  
  };  
    
  const capitalize = function(str) {  
   const cap = str[0].toUpperCase();  
   return cap + str.slice(1);  
  };  
    
  Constraint.prototype = {  
   validate: function(value, instance) {  
   return this.validator.validate(value, ...this.requirementList, instance);  
   },  
    
   \_parseRequirements: function(options) {  
   this.requirementList = this.validator.parseRequirements(this.requirements,  
   key => options[this.name + capitalize(key)]  
   );  
   }  
  };  
    
  export default Constraint;